UNIVERZITET U NOVOM SADU
PRIRODNO-MATEMATI ČKI
FAKULTET
DEPARTMAN ZA FIZIKU

Određivanje termodinamičkih osobina Izingovog modela na petougaonoj rešetki primenom Monte Karlo simulacija

-diplomski rad-

Mentor: dr Petar Mali

Kandidat: Stefan Velja

Novi Sad, 2019

Želeo bih iskazati svoju veliku zahvalnost profesoru Petru Malom na tome što me je motivisao i pomogao da pronađem oblast koja će me zainteresovati, kao i na zalaganju, strpljenju i razumevanju koje je pokazao kao moj mentor. Zahvalan sam i svim ostalim profesorima koji su doprineli mojoj ljubavi prema nauci, porodici i devojci koji su mi svojom podrškom omogućili da se posvetim pisanju ovog rada i pomagali oko njegovog sadržaja, Dimitriju na čitanju rada iznova i iznova dajući mi sugestije i ispravljajući greške i svim drugim dragim ljudima u mom životu, koji su mi pomogli na ovaj ili onaj način.

Sadržaj

Glava 1

Uvod

Cilj ovog rada je određivanje termodinamičkih svojstava magnetika predstavljenog Izingovim modelom na petougaonoj rešetki putem Monte Karlo Markovljevih lanaca i ocena ovog pristupa poređenjem sa analitičkim rešenjem.

Postoji veliki broj problema koji za cilj imaju određivanje verovatnoće odigravanja nekog događaja ili pronalaženja broja stanja koja ispunjavaju određene uslove. Neke od tih problema je vrlo lako rešiti kombinatorno ili čak nabrajajući sve moguće događaje ili kombinacije koje mogu postojati, a potom od njih izabrati one tražene. Na primer, računanje verovatnoće da tri bačena novčića pokažu pismo spada u elementarne kombinatorne probleme verovatnoće. Sa druge strane, većinu savremenih problema sretanih kako u nauci tako i u svakodnevnom životu je izuzetno teško i nepraktično rešavati na ovaj način. Jedan konkretan takav problem, računanje verovatnoće da se iz standardnog špila od 52 karte podeli rešiva partija pasijansa, inspirisao je nastanak klase empirijskih algoritama danas poznatih kao Monte Karlo algoritmi.

Monte Karlo algoritam podrazumeva nasumično generisanje velikog broja događaja do kojih može doći, a potom odabir traženih događaja među njima. Razlika između ovakvog i matematički egzaktnog postupka leži u tome da Monte Karlo algoritam ne uzima u obzir *sve* događaje, nego samo one koje generiše. Na taj način se egzaktno rešenje pretvara u statističko, koje, imajući u vidu mogućnosti savremenih računara i brzinu obrade podataka, retko značajno odstupa od egzaktnog. Vreme rešavanje problema se ubrzava mnogostruko – uglavnom izražavano u redovima veličine, zadržavajući zadovoljavajuću tačnost. Naravno, sa složenošću problema, rastu i vremenska i memorijska zahtevnost algoritma, što predstavlja na neki način ograničenje Monte Karlo algoritama.

Jednostavan primer ovog postupka predstavlja izračunavanje površine kruga poznatog poluprečnika. Krug se nacrta, recimo, u pesku na plaži, potom se oko njega opiše kvadrat, nakon čega se nasumično unutar kvadrata bacaju kamenčići. Nakon dovoljno velikog broja bacanja, zbog njegove nasumičnosti, raspodela kamenčića unutar kvadrata može se smatrati uniformnom. Zbog toga je odnos brojeva kamenčića unutar kruga i unutar kvadrata jednak odnosu njihovih površina, odakle se površina kruga može izračunati na osnovu poznate površine kvadrata. Drugi primer bi bio računanje broja π bacanjem velikog broja šibica dužine l po papiru na kojem su povučene paralelne prave linije na rastojanju l. Udeo šibica koje se padnu preko jedne od linija iznosi $\frac{\pi}{4}$, [Griffiths & Schroeter, 2018] odakle sledi i vrednost broja π.

Markovljev lanac je model za opisivanje niza događaja ili stanja posmatranog sistema, pri čemu verovatnoća nekog događaja zavisi samo od stanja koje mu je prethodilo. Konkretna implementacija Markovljevog lanca nad nekim sistemom naziva se Markovljevim procesom. Sva-

kodnevan, a donekle kombinatorno složen primer ovakvog modela je igra "Ne ljuti se, čoveče", gde, kada su figure zauzele neko mesto na tabli, pri određivanju verovatnoće za neki idući raspored figura ulogu igraju samo *trenutna* mesta na tabli i bacanje kocke, a ne dešavanja koja su tome prethodila.

Na kraju, Izingov model je matematički model koji za cilj ima, između ostalog, da opiše vrednosti spina atoma u magnetiku, koje mogu biti +1 ili -1, što predstavlja spinove gore i dole, redom. Neretko upotrebljavani termini za ove vrednosti spina su i "up" i "down" spin. Atomi su najčešće raspoređeni u magnetnu ili kristalnu rešetku, apstrahovanu grafom, čijim čvorovima su predstavljeni upravo atomi, a granama interakcija između njih. Česta je upotreba aproksimacije da interakcija postoji samo između najbližih suseda, shodno potrebama u datom slučaju.

Konkretna, uža oblast u kojoj je Izingov model pronašao primenu su magnetni izolatori, za čije se modelovanje koristi. Ekscitacije takvih jedinjenja, poput $CoCs_3$, $DyPO_4$, $CoRb_3Cl_5$ [Nolting & Ramakanth, 2009] i drugih, mogu se adekvatno opisati Izingovim modelom [Cowley & Buyers, 1972]. U druge oblasti, kako u polju fizike, tako i drugih nauka, za čiji opis se ovaj model može koristiti spadaju [Nolting & Ramakanth, 2009]:

- Model binarnih legura – u leguri postoje dve različite vrste atoma, te se uzima da spin +1 odgovara atomu A, a spin -1 atomu B.

- Model rešetkastog gasa – prostor u kojem se gas nalazi biva izdeljen na elementarne zapremine, koje su reprezentovane čvorovima u Izingovom modelu [Lee & Yang, 1952], tako da vrednost +1 označava prisustvo atoma u posmatranoj zapremini, a -1 njegovo odsustvo.

- Neurologija – mreža nervnih ćelija predstavljena je Izingovim modelom [Schneidman et al., 2006], tako da su aktivni neuroni označeni sa +1, a neaktivni sa -1.

- Ekonofizika – prilikom modelovanja potencijalne utaje poreza od strane velike populacije [Pickhardt & Seibold, 2014], licima koja vrše utaju poreza dodeljen je čvor sa vrednošću +1, a onima koji porez uredno plaćaju -1.

Na kraju, zbog svoje jednostavnosti u poređenju sa drugim modelima, Izingov model se vrlo često koristi pri nastavi, kao generalni demostracioni model u statističkoj fizici.

Već postojeće mogućnosti primene Izingovog modela čine istraživanje njegovih osobina i tehnika simulacija za njihovo određivanje veoma zanimljivim. Ovome dodatno ide u prilog to da u nauci do raznih otkrića dolazi neočekivano ili čak potpuno slučajno. Može se desiti da ispitivanje Izingovog modela dovede do novih saznanja ili da simulacione tehnike korišćene u ovom radu doprinesu objašnjenju nekih sasvim drugih pojava, što predstavlja i glavnu motivaciju za ovakvo istraživanje.

Glava 2

Monte Karlo Markovljevi lanci

U ovoj glavi će biti demonstrirano funkcionisanje Markovljevog lanca kojim se pomoću Monte Karlo algoritma simulira ponašanje nekog sistema sa vise mogućih stanja. Smatramo da je, prema modelu kojim se sistem opisuje, u vremenu definisano proizvoljno mnogo diskretnih trenutaka, između kojih ne dolazi do promena u sistemu. U svakom pomenutom trenutku, sistem može preći iz trenutnog stanja u neko drugo. Ovaj događaj - upit da li će sistem preći u drugo stanje i, ako da, u koje, naziva se Monte Karlo korakom.

Neka je n broj mogućih stanja sistema i u idućem koraku, sistem sa određenom verovatnoćom može biti u svakom od n stanja. Verovatnoća prelaza iz stanja i u stanje j data je kao T_{ji}, gde se T naziva tranzicionom matricom. Verovatnoća nalaženja sistema sadržana je u vektoru stanja, S. Jedan Monte Karlo korak u ovom modelu vrši se delovanjem tranzicione matrice na vektor stanja:

$$S(t_{i+1}) = TS(t_i),$$

pri čemu će $S(t_i)$ skraćeno biti pisano kao S^i. Dakle, gornji indeks iznad vektora stanja definiše vreme u kojem je vektor posmatran, dok donji indeks označava konkretno stanje čija je verovatnoća sadržana u vektoru. [1] Takođe, radi jednostavnosti, odabrani vremenski trenuci posmatrani su kao ekvidistantni, sa razlikom od 1, bez upotrebe fizičkih jedinica za vreme.

2.1 Osobine tranzicione matrice i vektora stanja

Budući da je zbir verovatnoća za nalaženje sistema u nekom od stanja jedak jedinici, tako važi:

$$\sum_{i=1}^{n} S_i = 1.$$

Takođe, kako svaka kolona unutar T označava verovatnoću nalaženja sistema po stanjima nakon sledećeg Monte Karlo koraka, zbir takvih verovatnoća za svako trenutno stanje i jednak je 1, te pišemo:

$$\sum_{j=1}^{n} T_{ji} = 1, \quad \forall i,$$

[1] U literaturi se može naći i drugačiji opis tranzicione matrice, koji odgovara transponovanoj matrici T, definisanoj u ovom radu. Tada su S vektori vrste, a Monte Karlo korak vrši se kao $S^{i+1} = S^i T$.

odnosno,

$$\sum_{i=1}^{n}\sum_{j=1}^{n} T_{ji} = n.$$

Kako bi ovakav metod simuliranja bio adekvatan za opisivanje Izingovog modela, sistem, odnosno, tranziciona matrica koja ga opisuje, mora ispunjavati određene uslove, tako da Markovljev lanac nad sistemom mora biti ergodički. Ovo podrazumeva da nakon beskonačno mnogo koraka, sistem mora posetiti svako stanje barem jednom. Da bi bio ergodički, Markovljev lanac mora biti:

- Ireducibilan - iz svakog stanja sistem mora biti u mogućnosti dospeti, nakon dovoljno velikog broja koraka, u svako drugo stanje sistema.
- Aperiodičan - ne sme postojati broj $t_0 \in \mathbb{N}$, takav da ako je sistem u trenutku t bio u stanju k, u nekom drugom trenutku t' važi:
  - $S_k = 0$, za $t' - t \neq t_0, 2t_0, 3t_0, \ldots$
  - $S_k \neq 0$, za $t' - t = t_0, 2t_0, 3t_0, \ldots$

Kao posledica ireducibilnosti, T ne sme biti blok-matrica ni za jednu permutaciju indeksa kojima su stanja obeležavana. Uz to, da bi ireducibilan sistem bio aperiodičan, dovoljno je da sva stanja ne čine jedan veliki ciklus, odnosno, da je uslov aperiodičnosti ispunjen za $t_0 = n$, dok je za ostale vrednosti t_0 aperiodičnost garantovana ireducibilnošću.

Primer jednog reducibilnog lanca, koji samim tim nije ergodički, predstavljen je sledećom tranzicionom matricom:

$$S = \begin{bmatrix} 0.3 & 0.8 & 0 & 0 \\ 0.7 & 0.2 & 0 & 0 \\ 0 & 0 & 0.5 & 1 \\ 0 & 0 & 0.5 & 0 \end{bmatrix}.$$

Može se primetiti da sistem, ukoliko se jednom nađe u stanjima 1 ili 2, nikada neće dospeti u stanja 3 i 4, jer je, prema ovako definisanoj tranzicionoj matrici, verovatnoća odigravanja takvog prelaza jednaka nuli. Važi i obrnuto, za slučaj da se sistem nađe u stanjima 3 ili 4. Ova osobina direktno krši uslov ergodičnosti. Matrica S se u ovom slučaju može napisati i kao blok-matrica, što smo naveli kao zabranjeno:

$$S = \begin{bmatrix} T_1 & 0 \\ 0 & T_2 \end{bmatrix},$$

gde je

$$T_1 = \begin{bmatrix} 0.3 & 0.8 \\ 0.7 & 0.2 \end{bmatrix} \text{ i } T_2 = \begin{bmatrix} 0.5 & 1 \\ 0.5 & 0 \end{bmatrix}.$$

Jednostavan ireducibilan, ali periodičan lanac može imati iduću tranzicionu matricu:

$$T = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}.$$

Ovaj sistem svakim korakom prelazi u stanje sa indeksom većim za 1 (pri čemu iz stanja 4 prelazi u stanje 1). Odavde se jasno vidi periodičnost, sa korakom od $t_0 = 4$.

2.2 Konvergencija Markovljevog procesa

U skladu sa definicijom tranzicione matrice i vektora stanja sistema, jedan Monte Karlo korak predstavljen je delovanjem T na S. Ukoliko je potrebno razmatrati stanje sistema nakon dva Monte Karlo koraka, zaključujemo:

$$S^{i+2} = TS^{i+1} = TTS^i = T^2S^i.$$

Poslednji znak jednakosti posledica je asocijativnosti množenja matrica. Analogno opisanom, možemo zaključiti da se N Monte Karlovih koraka predstavlja kao delovanjem T^N na S:

$$S^{i+N} = T^N S^i.$$

Vrlo često je bitno praviti simulacije sistema sa velikim brojem Monte Karlo koraka, budući da je svaki korak efektivno još jedno merenje u sistemu, čime se povećava preciznost simulacije. Konkretno, za primene Monte Karlo algoritma gde računamo srednju vrednost neke veličine x, standardna devijacija izražava se formulom:

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

odakle dalje sledi:

$$\sigma \propto \frac{1}{\sqrt{N}}. \tag{2.1}$$

Zbog toga je značajno ispitati ponašanje sistema nakon velikog broja koraka, odnosno, izračunati vrednost stepena tranzicione matrice datog limesom:

$$\lim_{N\to\infty} T^N.$$

Konvergencija stepena tranzicione matrice najlakše se računa putem njene dijagonalizacije. Matrica T je dijagonalizabilna ukoliko postoji invertibilna matrica P, tako da se T može predstaviti kao proizvod PDP^{-1}, gde je D dijagonalna matrica [Radošević & Mali, 2017].

Elementi D su svojstvene vrednosti matrice T, a vektori kolone unutar P su svojstveni vektori iste matrice. Svojstveni vektori u P su poređani tako da i-ta kolona u P odgovara svojstvenoj vrednosti D_{ii}. Kako je D dijagonalna matrica, to za njen N-ti stepen važi:

$$D^N = [D_{ij}^N].$$

Odatle za N-ti stepen matrice T sledi:

$$T^N = (PDP^{-1})(PDP^{-1})...(PDP^{-1}) = PD^N P^{-1}.$$

Ovde je takođe upotrebljena osobina asocijativnosti množenja matrica prilikom skraćivanja proizvoda $P \cdot P^{-1}$. Iz jednakosti limesa

$$\lim_{N\to\infty} T^N = P[\lim_{N\to\infty} D_{ii}^N]P^{-1},$$

lako se izračunava tražena vrednost matrice T^N.

Dodatna olakšavajuća okolnost proizilazi iz osobina matrice T; tačno jedna njena svojstvena vrednost je jednaka 1, dok su njene ostale svojstvene vrednosti po modulu manje od 1 [Gallager, 2011]. To znači da se $\lim_{N\to\infty} D_{ii}^N$ svodi na 1, za $D_{ii} = \lambda_i = 1$, odnosno na 0, za $D_{ii} = \lambda_i = 0$.

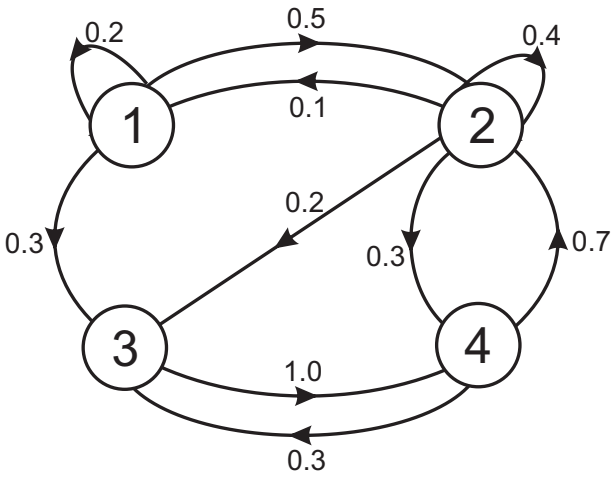
2.3 Primer Markovljevog procesa

Posmatrajmo sistem opisan tranzicionom matricom T:

$$T = \begin{bmatrix} 0.2 & 0.1 & 0 & 0 \\ 0.5 & 0.4 & 0 & 0.7 \\ 0.3 & 0.2 & 0 & 0.3 \\ 0 & 0.3 & 1 & 0 \end{bmatrix}. \tag{2.2}$$

Sistem je ilustrovan na sl. 2.1. Stanja su obeležena brojevima od 1 do 4 i upisana u krugove, a linije između stanja opisuju verovatnoće prelaska date tranzicionom matricom. Neka su verovatnoće nalaženja sistema u početnom trenutku $t = 0$ date sledećim vektorom stanja:

$$S^0 = \begin{bmatrix} 0.5 \\ 0 \\ 0.3 \\ 0.2 \end{bmatrix}$$



Slika 2.1: Ilustracija Markovljevog procesa

I sa slike i iz vrednosti tranzicione matrice se vidi da sistem ispunjava uslove ergodičnosti. Sistem iz svakog od četiri stanja može doći u svako drugo stanje i ne postoji ciklus u kojem može biti zarobljen. Metodom opisanom u 2.2, iz jednačine (2.2) se računa (približna) granična vrednost stepena tranzicione matrice:

$$\lim_{N\to\infty} T^N = \lim_{N\to\infty} PD^N P^{-1} = \begin{bmatrix} 0.0530 & 0.0530 & 0.0530 & 0.0530 \\ 0.4236 & 0.4236 & 0.4236 & 0.4236 \\ 0.1982 & 0.1982 & 0.1982 & 0.1982 \\ 0.3253 & 0.3253 & 0.3253 & 0.3253 \end{bmatrix}.^{2}$$

Sve kolone matrice $\lim_{N\to\infty} T^N$ su iste i njenim delovanjem na ma koji vektor stanja dobija se vektor jednak kolonama ove matrice, što se može proveriti delovanjem na S^0. Štaviše, ovo

[2]Napominjemo da je u konkretnom primeru odstupanje zbira članova unutar kolona od jedinice posledica zaokruživanja.

važi za svaku tranzicionu matricu takvu da je Markovljev proces koji joj odgovara ergodički. Ovo zapažanje je vrlo znacajno pošto pokazuje da nakon velikog broja Monte Karlo koraka postaje svejedno u kom se početnom stanju sistem nalazio. Napominjemo da je u konkretnom primeru odstupanje zbira članova unutar kolona od jedinice posledica zaokruživanja.

Glava 3

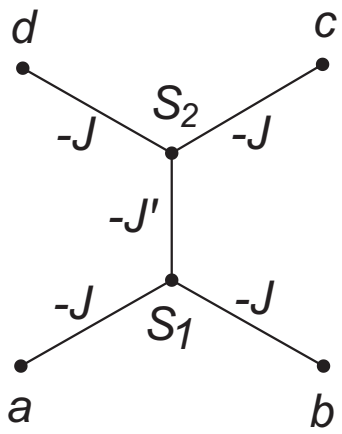
Simulacija

3.1 Model

Posmatrani model sastoji se od čvorova spina ± 1 unutar rešetke u ravni, čije ivice formiraju stranice podudarnih petougaonika. Postoje dve vrste interakcije između čvorova: interakcija između čvora sa četiri i čvora sa tri suseda u svom doprinosu hamiltonijanu čitave rešetke ima koeficijent $-J$, dok interakcija između dva čvora sa tri suseda ima koeficijent $-J'$. Ukupan doprinos jedne elementarne ćelije hamiltonijanu može se izračunati preko formule:

$$\widetilde{H} = -J(s_1(a+b) + s_2(c+d)) - J's_1s_2, \tag{3.1}$$

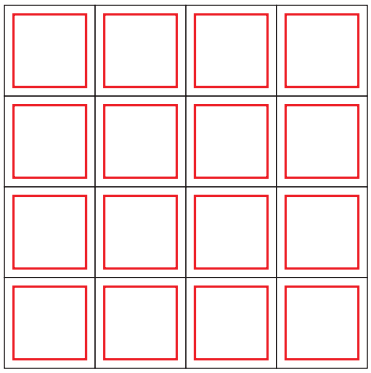
gde su s_1, s_2, a, b, c i d vrednosti spina odgovarajućih cvorova, kao što je prikazano na sl. 3.1.



Slika 3.1: Elementarna ćelija koja se ponavlja kroz rešetku

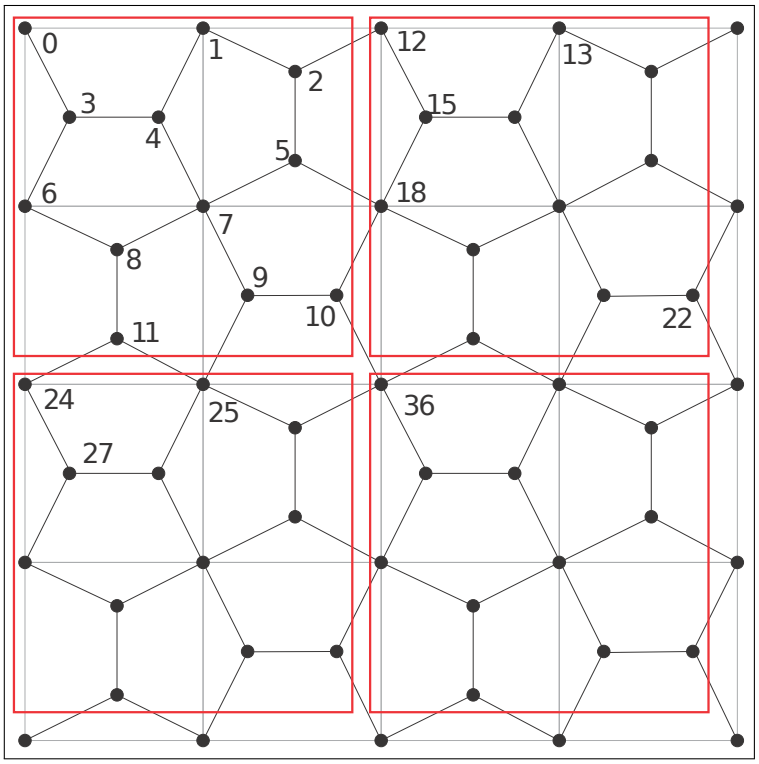
Kako bi se simulacija pojednostavila i ubrzala, graf rešetke nije implementiran kao generički graf, koristeći niz pokazivača na čvorove. Umesto toga, svakom čvoru je pripisan niz indeksa njegovih suseda, prema pravilu koje će uskoro biti navedeno.

Budući da je rešetka periodična, može se podeliti na osnovne segmente koji se ponavljaju, od kojih svaki sadrži 12 čvorova. Segmenti su poređani kao na sl. 3.2.



Slika 3.2: Podela rešetke na segmente

Označimo širinu rešetke izraženu u osnovnim segmentima sa L, broj segmenata sa $n = L^2$ i ukupan broj čvorova u rešetki sa $N = 12L^2$. Posmatrajmo malu rešetku, sa $L = 2$ (videti sl. 3.3). Čvorovi unutar segmenta označeni su brojevima od 0 do 11, a redni broj samog segmenta pomnožen je sa 12 i dodat indeksu čvora, kao što je prikazano.



Slika 3.3: Primer indeksiranja čvorova

Svi čvorovi sa istim ostatkom pri deljenju sa 12 posmatrani su na isti način po pitanju rasporeda grana između njih. Ovim metodom indeksiranja, graf rešetke može se konstruisati prilično lako, koristeći pravilo za određivanje suseda čvora indeksa i:

- $i \mod 12 = 0$: $i + 3$ (čvorovi sa indeksima i i $i + 3$ su susedi)
- $i \mod 12 = 1$: $i + 1$, $i + 3$
- $i \mod 12 = 2$: $i + 3$, $f(i, 10)$
- $i \mod 12 = 3$: $i + 1$, $i + 3$
- $i \mod 12 = 4$: $i + 3$
- $i \mod 12 = 5$: $i + 2$, $f(i, 13)$
- $i \mod 12 = 6$: $i + 2$
- $i \mod 12 = 7$: $i + 1$, $i + 2$
- $i \mod 12 = 8$: $i + 3$
- $i \mod 12 = 9$: $i + 1$, $g(i, 8)$
- $i \mod 12 = 10$: $f(i, 8)$, $f(g(i, 8), 10)$
- $i \mod 12 = 11$: $g(i, 11)$, $g(i, 10)$

Ovde su f i g funkcije koje povezuju čvorove iz različitih segmenata. $f(i, j)$ daje indeks čvora sa ostatkom j pri deljenju sa 12, iz segmenta desno od segmenta čvora i. $g(i, j)$ radi analogno, za segment ispod segmenta čvora i. Funkcije su definisane na idući način:

- $f(i, j) = i - i \mod 12L + (i + j) \mod 12L$
- $g(i, j) = (i + 12L - j) \mod N$

Treba napomenuti da je graf neusmeren. Stoga, ukoliko, na primer, algoritam pronađe čvor 15 kao suseda čvora 12, takođe će označiti i čvor 12 kao sused čvora 15.

Kako bismo oslikali ovo pravilo, demonstriraćemo ga određivanjem suseda čvora sa indeksom $i = 10$ i proveriti na sl 3.3:

- $f(i, 8) = f(10, 8) = 10 - 10 + 18 = 18$, što jeste sused čvora 10
- $f(g(i, 8), 10) = f(10 + 24 - 8, 10) = f(26, 10) = 26 - 2 + 12 = 36$, što je takođe ispravno
- prilikom određivanja suseda čvora sa indeksom 9, algoritam zaključuje da je čvor 10 njegov sused, te odmah beleži susedstvo i u obrnutom smeru

Dakle, čvor 10 ima tri suseda sa indeksima 9, 18 i 36, što se poklapa sa sl. 3.3.

3.2 Metropolisov Monte Karlo algoritam

Prilikom nasumičnog generisanja događaja ili stanja u Monte Karlo algoritmima, neki uzorci imaju veći uticaj na određivanje srednje vrednosti ispitivane veličine od drugih. Uzorkovanje takvih elemenata (događaja ili stanja) smanjuje varijaciju srednje vrednosti i time čini simulaciju preciznijom. Posmatrano sa druge strane, ukoliko želimo fiksirati maksimalnu dozvoljenu varijaciju, korišćenje uticajnijih uzoraka može učiniti algoritam mnogostruko efikasnijim. Stoga uvodimo tehniku poznatu kao uzorkovanje po značaju (Importance sampling, eng) [Srinivasan, 2013].

Primera radi, pretpostavimo da treba izračunati integral neke raspodele na intervalu $[a, b]$. Monte Karlo algoritam uzima nasumično tačke iz intervala $[a, b]$, na osnovu kojih će integral kasnije biti izračunat numerički. Izborom dobre distribucije verovatnoće izbora tačaka $P(x)$ smanjuje se varijacija konačne vrednosti integrala. Pored toga, što su veće granice integracije, to je potrebna i bolja distribucija verovatnoće. Za postizanje tog cilja, distribucija verovatnoće uzorkovanja tačaka treba da je što sličnija posmatranoj raspodeli [Kroese et al., 2013].

Neka je unapred zadata distribucija $g(x|y)$, koja opisuje gustinu verovatnoće izbora tačke x ukoliko joj je prethodila tačka y. $g(x|y)$ mora biti simetrična, odnosno, $g(x|y) = g(y|x)$ i često se uzima kao Gausova raspodela [Yildirim, 2012]. Tada Metropolis-Hastings algoritam vrši uzorkovanje po značaju na sledeći način:

- Izabere se proizvoljna početna tačka x_0, u skladu sa $P(x)$

Za sve naredne korake:

- Nasumično se odabere kandidat za sledeću tačku x' na osnovu $g(x'|x_i)$
- Izračuna se verovatnoća prihvatanja kandidata $A(x_i, x') = \min(1, \frac{P(x')}{P(x_i)})$
- Uniformno se generiše nasumičan broj $u \in [0, 1]$:
  - Za $u \leq A(x_i, x')$, prihvata se kandidat i postavlja se $x_{i+1} = x'$
  - Inače, kandidat se odbija i zadržava se trenutna tačka: $x_{i+1} = x_i$

Stohastički metod korišćen u ovom radu za određivanje osobina rešetke je takozvani Metropolis-Hastings algoritam sa pojedinačnim obrtanjem (single-flip, eng) [Metropolis et al., 1953]. Pod stanjem Izingovog modela podrazumevamo uređeni skup spinova svih čvorova. Inicijalizacija Metropolis-Hastings algoritma vrši postavljanjem svakog čvora nasumično na +1 ili -1, nakon čega se N puta odabira po jedan čvor koji će algoritam posetiti, takođe nasumično. Ovaj način obilaženja znači da se može desiti da svaki čvor bude obiđen po jednom, ali i da jedan čvor bude obiđen N puta. Pri posećivanju čvora s_{ij}, kandidat za novo stanje se uzima kao trenutno stanje, uz obrnutu vrednost posmatranog čvora:

$$s_{ij} \to -s_{ij} \leftrightarrow x \to x'.$$

Za funkciju $P(T)$ uzima se Bolcmanova distribucija energije:

$$P(s, T) = \mathrm{e}^{-\frac{H(s)}{k_\mathrm{B}T}},$$

gde je sa s označeno stanje modela, a sa $H(s)$ hamiltonijan sistema koji se dobija kao zbir hamiltonijana elementarnih ćelija $\widetilde{H}$ definisanih u jednakosti 3.1. Odavde se za verovatnoću prihvatanja novog stanja dobija:

$$A(s, s', T) = \min(1, \frac{\exp(-\frac{H(s')}{k_\mathrm{B}T})}{\exp(-\frac{H(s)}{k_\mathrm{B}T})}) = \min(1, \exp(-\frac{\Delta E}{k_\mathrm{B}T})),$$

gde je

$$\Delta E = H(s') - H(s).$$

Ovime se završava postupak inicijalizacije, posle kog algoritam prolazi kroz N_{MC} Monte Karlo koraka. Unutar svakog koraka se, kao i prilikom inicijalizacije, N puta odabira po jedan posećivani čvor. Nakon obilaska svih čvorova se tekući prosek za svaku praćenu termodinamičku veličinu ažurira, što i razlikuje ovaj postupak od onog korišcenog tokom inicijalizacije. Monte Karlo algoritam se ponavlja za različite temperature i rešetke, u skladu sa ciljem ispitivanja.

Cilj simulacije je provera da li se Monte Karlo algoritam može koristiti za određivanje kritične temperature za petougaonu rešetku, kao u modelu Viktora Urumova [Urumov, 2002]. Ovo se postiže poređenjem kritičnih temperatura sistema računatih putem simulacije i analitički. Kritična temperatura T_C definiše se kao temperatura iznad koje model prestaje ispoljavati feromagnetne osobine i ponaša se kao paramagnetik. Stoga, za beskonačne rešetke, na temperaturi $T < T_C$, materijal je strogo uređen i srednja magnetizacija iznosi ± 1, dok je za $T > T_C$ njena vrednost jednaka nuli. Za konačne rešetke, magnetizacija nije egzaktno jednaka nuli za $T > T_C$, ali joj jeste vrlo bliska.

U praksi, T_C je prvo određena okvirno, kao temperatura na kojoj toplotni kapacitet i magnetna susceptibilnost supstance imaju maksimalne vrednosti. Ovo je posebno korisno, budući da za ograničene rešetke, drugi metodi ne pokazuju približnu vrednost T_C toliko jasno. Potom se opseg temperatura simulacije suzi oko nađenog pika, a broj koraka se znatno poveća. Binderov parametar, kumulant četvrtog reda, računa se kao:

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

gde je s parametar uređenosti [Heermann & Binder, 2010], što u našem slučaju predstavlja srednju magnetizaciju po čvoru. Simulacije se ponavljaju za više rešetki različitih veličina i, u teoriji, temperatura u kojoj se seku Binderovi parametri za sve rešetke je T_C.

Vrednost T_C izračunata je analitički za dve vrednosti $\frac{J'}{\sqrt{J^2+J'^2}}$ [Urumov, 2002]:

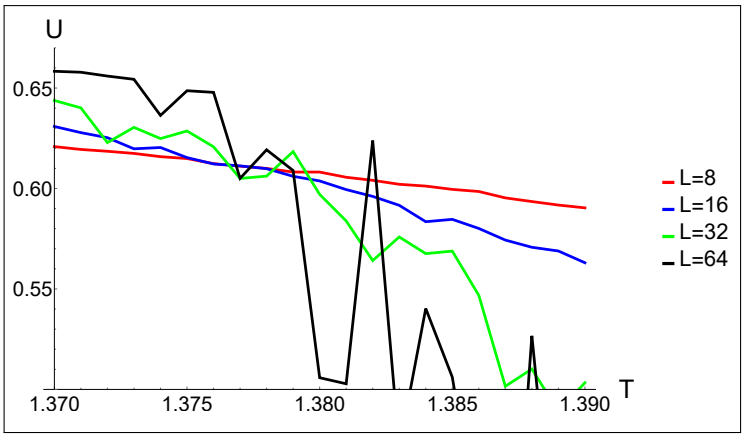
- Za $\frac{J'}{\sqrt{J^2+J'^2}} = 0.1$, $T_C = 1.377454026\ldots$
- Za $\frac{J'}{\sqrt{J^2+J'^2}} = \frac{1}{\sqrt{2}}$, $T_C = 1.799166700\ldots$

Tokom proračuna, vrednosti k_B i J se uzimaju kao jednake 1. U našim simulacijama, za oba pomenuta para (J, J'), četiri rešetke su simulirane, sa $L = 8, 16, 32$ i 64.

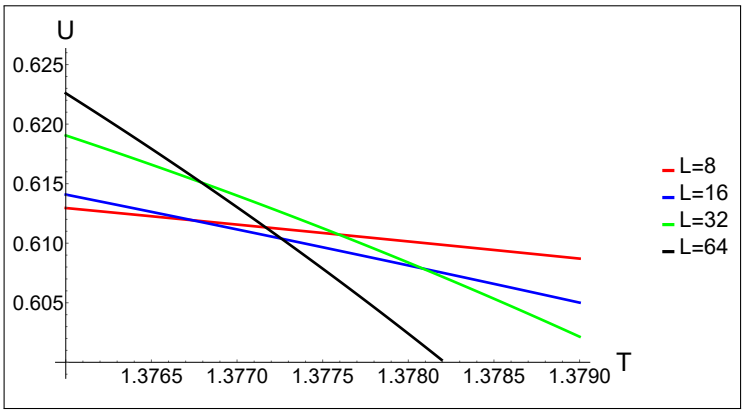
3.3 Rezultati

Prvi skup rezultata odgovara postavci sa $\frac{J'}{\sqrt{J^2+J'^2}} = 0.1$. Za sve četiri rešetke, temperatura je varirana oko teorijske T_C, to jest, od 1.37 do 1.39, sa korakom od 0.001. Binderovi parametri računati su za svaku od ovih temperatura i fitovani polinomima trećeg stepena. Preseci su izračunati kao temperature u kojima ovi polinomi imaju jednake vrednosti. Srednja vrednost temperatura svih šest preseka (po jedan za svaki par rešetki) uzeta je kao T_C. Za svaki par polinoma postoji tri rešenja zbog njihovog stepena, ali samo realna rešenja unutar posmatranog opsega su uzeta u obzir.

Binderovi parametri su prikazani na sl. 3.4, dok sl. 3.5 pokazuje kako su fitovani i gde se presecaju.



Slika 3.4: Vrednosti Binderovih parametara za različite rešetke, $J' \approx 0.1005$



Slika 3.5: Fit funkcije Binderovih parametara, uveličano u okolini T_C, $J' \approx 0.1005$

Fitujući polinomi navedeni su u tabeli 3.1, a tabela 3.2 prikazuje prikazana rešenja jednačina preseka polinoma. Realna resenja unutar opsega $[1.37, 1.39]$ posebno su obeležena.

L	$P_L(T)$			
8	$-567.4\,T^3$	$+2332.0\,T^2$	$-3196.1\,T$	$+1461.4$
16	$-725.6\,T^3$	$+2948.9\,T^2$	$-3996.8\,T$	$+1807.2$
32	$-4493.3\,T^3$	$+18275.3\,T^2$	$-24775.1\,T$	$+11196.0$
64	$-33342.0\,T^3$	$+137209.0\,T^2$	$-188221.0\,T$	$+86070.4$

Tabela 3.1: Polinomi kojima su fitovani Binderovi parametri, $J' \approx 0.1005$

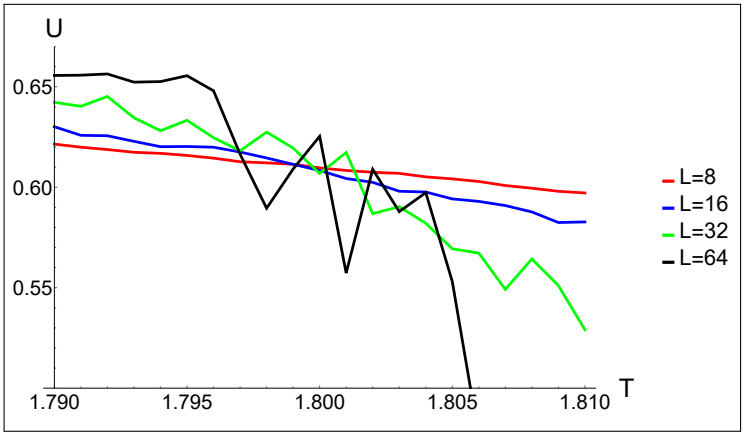
L_1	L_2	T_1	T_2	T_3
8	16	1.20289	1.32027	**1.37674**
8	32	1.32742	1.35596	**1.37759**
8	64	$1.3691 - 0.014i$	$1.3691 + 0.014i$	**1.37717**
16	32	1.32641	1.36325	**1.37808**
16	64	$1.3695 - 0.013i$	$1.3695 + 0.013i$	**1.37726**
32	64	$1.3729 - 0.0122i$	$1.3729 + 0.012i$	**1.37679**

Tabela 3.2: Tačke preseka za različite veličine rešetki, $J' \approx 0.1005$

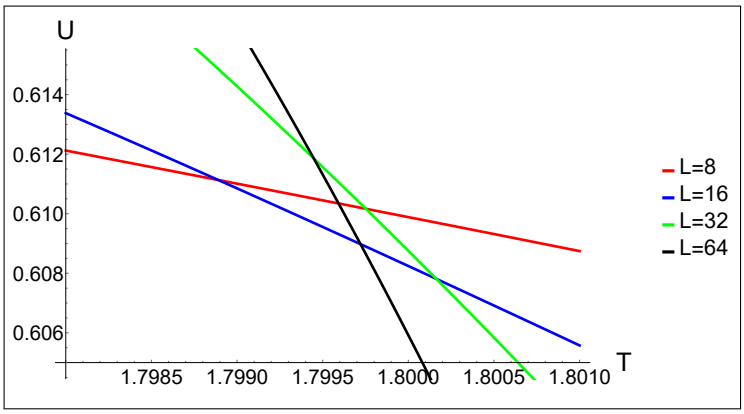
Vrednost T_C dobijeno putem simulacije izračunata je kao $T_{Csim} = 1.37727$. Njena relativna greška je prilično mala:

$$\delta_{T_C} = \frac{|T_{Ctheor} - T_{Csim}|}{T_{Ctheor}} = 1.3157 \cdot 10^{-4}$$

Drugi skup rezultata odgovara slučaju kada je $\frac{J'}{\sqrt{J^2+J'^2}} = \frac{1}{\sqrt{2}}$. Pikovi susceptibilnosti i toplotnog kapaciteta uočeni su na $T \approx 1.7995$, nakon čega je simulacija sužena na interval $[1.79, 1.81]$. Fitujući polinomi su analogno računati, a samo oni unutar ovog opsega su uzimani u obzir. Binderovi parametri i odgovarajući fitovi prikazani su na sl 3.6 i sl. 3.7.



Slika 3.6: Vrednosti Binderovih parametara za različite rešetke, $J' = 1$



Slika 3.7: Fit funkcije Binderovih parametara, uveličano u okolini T_C, $J' = 1$

Tabela 3.3 sadrži funkcije nacrtane na sl. 3.7, dok se vrednosti T_C mogu naći u tabeli 3.4.

L	$P_L(T)$			
8	$-960.0\,T^3$	$+5175.4\,T^2$	$-9301.2\,T$	$+5573.3$
16	$+3346.5\,T^3$	$-18101.3\,T^2$	$+32634.0\,T$	-19609.2
32	$+4199.3\,T^3$	$-22880.0\,T^2$	$+41545.2\,T$	-25139.8
64	$-15980.2\,T^3$	$+85523.9\,T^2$	$-152569.3\,T$	$+90724.8$

Tabela 3.3: Polinomi kojima su fitovani Binderovi parametri, $J' = 1$

L_1	L_2	T_1	T_2	T_3
8	16	1.78426	**1.79889**	1.82184
8	32	1.78376	**1.79975**	1.85432
8	64	$1.7749 - 0.004$i	$1.7749 + 0.004$i	**1.79959**
16	32	1.78338	**1.80016**	2.0201
16	64	$1.7810 - 0.008$i	$1.7810 + 0.008$i	**1.79972**
32	64	$1.7863 - 0.008$i	$1.7863 + 0.008$i	**1.79945**

Tabela 3.4: Tačke preseka za različite veličine rešetki, $J' = 1$

U ovom slučaju, vrednost T_C iznosi $T_{Csim} = 1.79959$. Njena relativna greška je takođe vrlo mala:

$$\delta_{T_C} = \frac{|T_{Ctheor} - T_{Csim}|}{T_{Ctheor}} = 2.37938 \cdot 10^{-4}$$

Zanimljivo je uporediti osobine kritične temperature dvodimenzionalnog Izingovog modela na kvadratnoj i na petougaonoj rešetki. Za kvadratnu rešetku, analitičko rešenje daje nam zavisnost kritične temperatura od integrala izmene kao u [Nolting & Ramakanth, 2009]:

$$T_C = \frac{2J}{k_\mathrm{B} \ln(1 + \sqrt{2})},$$

to jest

$$T_C \propto J.$$

Na osnovu ove formule vidi se da ukoliko se simulacijom dobije jedna vrednost T_C u sistemu jedinica $k_\mathrm{B} = J = 1$, vrednosti T_C za druge vrednosti J mogu se dobiti jednostavnim skaliranjem. Na petougaonoj rešetki je zavisnost T_C od parametara J i J' mnogo složenija [Urumov, 2002], te se pri promeni J i J' ovakvo skaliranje ne može izvršiti.

Glava 4

Zaključak

Dobijena greška prilikom poređenja teorijske vrednosti za kritičnu temperaturu Izingovog modela na petougaonoj rešetki [Urumov, 2002] i simulacijom dobijene vrednosti iste veličine je zadovoljavajuće mala. Simulacija je pokretana za različite veličine rešetke paralelno i složenost algoritma srazmerna je broju čvorova rešetke. Stoga je vremenski najzahtevnija rešetka bila ona sa $n = 64^2$ segmenata i trajanje simulacije nad njom bilo je glavno vremensko ograničenje. Prilikom pokretanja simulacija za svaku od ovih rešetaka, broj Monte Karlo koraka podešavan je tako da bude obrnuto srazmeran broju čvorova u njoj. Ovako su manje rešetke bile simulirane približno jednako dugo kao i najveća, što je omogućilo njihovo preciznije merenje za vreme koje bi svakako bilo utrošeno na najzahtevniju rešetku. Posledice ovoga su vidljive na sl. 3.5, gde vrednosti Binderovih parametara više osciluju sa porastom veličine rešetke. Ova odstupanja kod velikih rešetaka se mogu smanjiti povećanjem broja Monte Karlo koraka, što bi u našem slučaju bilo kako previše vremenski zahtevno, tako i nepotrebno za poređenje kritičnih temperatura dobijenih na dva načina. Slaganje analitičkog rešenja i onog dobijenog simulacijom ukazuje na mogućnost ispitivanja osobina Izingovog modela na petougaonoj rešetki i za negativne vrednosti parametara interakcije.

Glava 5

Dodatak

Ovaj dodatak sadrži C++ kod korišćen pri simulacijama i "ran1" generator nasumičnih brojeva, definisan u [Press et al., 1992].

```
1  #include <conio.h>
2  #include <cstdio>
3  #include <fstream>
4  #include <iomanip>
5  #include <iostream>
6  #include <math.h>
7  #include <vector>
8  #include <sstream>
9  #include <stdlib.h>
10 #include <string>
11
12 #include "ran1.h"
13
14 #pragma warning (disable : 4996)
15
16 using namespace std;
17
18 std::ostringstream streamObj;
19
20 int L;
21 int N;
22 long long MCS;
23 long long PrepSteps;
24 long long BaseStepsLimit;
25 long long LastSteps;
26 long long NORM;
27 double T0;
28 double TMax;
29 double dT;
30 double J = 1;
31 double JJ = 1; //J'
32
33 const int NT = 10000;
34 long seed = -1000000007;
35
36 double T;
37 int Ti;
```

```
38
39
40 vector<int> lattice; // spinovi
41 vector<int> neighbours[5]; // susedi
42 vector<int> neighbourCount; // broj suseda, corner ima 4, necorner 3
43
44 double GetNodeEnergy(int index)
45 {
46     double e = 0;
47     for (int i = 0; i < neighbourCount[index]; i++)
48     {
49         if (neighbourCount[index] == 3 && neighbourCount[neighbours[i][index
               ]] == 3)
50         {
51             e -= JJ * lattice[index] * lattice[neighbours[i][index]];
52         }
53         else
54         {
55             e -= J * lattice[index] * lattice[neighbours[i][index]];
56         }
57     }
58     return e;
59 }
60
61 bool WillFlip(int index, double &energyDiff)
62 {
63     energyDiff = -2 * GetNodeEnergy(index);
64     return energyDiff < 0 || ran1(&seed) < exp(-energyDiff / T);
65 }
66
67 bool WillFlip(int index)
68 {
69     double energyDiff;
70     return WillFlip(index, energyDiff);
71 }
72
73 void Flip(int index)
74 {
75     lattice[index] *= -1;
76 }
77
78 void Join(int i1, int i2)
79 {
80     neighbours[neighbourCount[i1]][i1] = i2;
81     neighbourCount[i1]++;
82
83     neighbours[neighbourCount[i2]][i2] = i1;
84     neighbourCount[i2]++;
85 }
86
87 int f(int i, int x)
88 {
89     return i - (i % (12 * L)) + ((i + x) % (12 * L));
90 }
91
```

```
92  int g(int i, int x)
93  {
94      return (i + 12 * L - x) % (N);
95  }
96
97  int GetRandomNode()
98  {
99      return int (ran1(&seed)*N);
100 }
101
102 void PreProcess()
103 {
104     for (int i = 0; i < PrepSteps; i++)
105     {
106         if (i % (PrepSteps / 20) == 0)
107         {
108             printf("#");
109         }
110         for (int j = 0; j < N; j++)
111         {
112             int randNode = GetRandomNode();
113             if (WillFlip(randNode)) Flip(randNode);
114         }
115     }
116     cout << endl;
117 }
118
119 double GetTotalMagnetization(int filter)
120 {
121     double magnetization = 0;
122     for (int i = 0; i < N; i++)
123     {
124         if ((filter == 1 && neighbourCount[i] == 4) || (filter == 2 &&
                neighbourCount[i] == 3))
125         {
126             continue;
127         }
128         magnetization += lattice[i];
129     }
130     return magnetization;
131 }
132
133 double GetTotalEnergy(int filter)
134 {
135     double energy = 0;
136     for (int i = 0; i < N; i++)
137     {
138         if ((filter == 1 && neighbourCount[i] == 4) || (filter == 2 &&
                neighbourCount[i] == 3))
139         {
140             continue;
141         }
142         energy += GetNodeEnergy(i);
143     }
144     return energy;
```

```
145 }
146
147 class Results
148 {
149 public:
150     double averageEnergy;
151     double averageSquareEnergy;
152     double averageMagnetization;
153     double averageSquareMagnetization;
154     double averageAbsoluteMagnetization;
155     double averageQuadMagnetization;
156     double magneticSusceptibility;
157     double thermalCapacity;
158     double BinderCumulant;
159     vector<int> spins;
160 };
161
162 Results allRes[NT];
163
164 double sqr(double x)
165 {
166     return x * x;
167 }
168
169 void InitSpins()
170 {
171     for (int i = 0; i < N; i++)
172         lattice[i] = 1 - 2 * int(ran1(&seed) * 2);
173 }
174
175 void Initialize()
176 {
177     N = L * L * 12;
178     int tsteps = (int) ((TMax - T0 + dT / 100) / dT) + 1;
179     MCS = (long long) BaseStepsLimit * 8 * 8 * 15 / (L * L * tsteps);
180     PrepSteps = MCS / 10;
181     NORM = MCS * N;
182
183     J = 1;
184     T = T0;
185
186     lattice.resize(N);
187     neighbourCount.resize(N);
188     for (int i = 0; i < 5; i++)
189         neighbours[i].resize(N);
190
191     double tmpT = T0;
192     int tmpi = 0;
193     while (tmpT < TMax + dT / 100)
194     {
195         allRes[tmpi].spins.resize(N);
196         for (int i = 0; i < N; i++)
197             allRes[tmpi].spins[i] = 0;
198         tmpi++;
199         tmpT += dT;
```

```
200     }
201
202     for (int i = 0; i < N; i++)
203     {
204         if (i % 12 == 0)
205         {
206             Join(i, i + 3);
207         }
208         else if (i % 12 == 1)
209         {
210             Join(i, i + 3);
211             Join(i, i + 1);
212         }
213         else if (i % 12 == 2)
214         {
215             Join(i, i + 3);
216             Join(i, f(i, 10));
217         }
218         else if (i % 12 == 3)
219         {
220             Join(i, i + 1);
221             Join(i, i + 3);
222         }
223         else if (i % 12 == 4)
224         {
225             Join(i, i + 3);
226         }
227         else if (i % 12 == 5)
228         {
229             Join(i, i + 2);
230             Join(i, f(i, 13));
231         }
232         else if (i % 12 == 6)
233         {
234             Join(i, i + 2);
235         }
236         else if (i % 12 == 7)
237         {
238             Join(i, i + 1);
239             Join(i, i + 2);
240         }
241         else if (i % 12 == 8)
242         {
243             Join(i, i + 3);
244         }
245         else if (i % 12 == 9)
246         {
247             Join(i, i + 1);
248             Join(i, g(i, 8));
249         }
250         else if (i % 12 == 10)
251         {
252             Join(i, f(i, 8));
253             Join(i, f(g(i, 8), 10));
254         }
```

```
255         else if (i % 12 == 11)
256         {
257             Join(i, g(i, 10));
258             Join(i, g(i, 11));
259         }
260     }
261 }
262 
263 
264 int main(int argc, const char* argv[])
265 {
266     JJ = stod(argv[1]);
267     L = stoi(argv[2]);
268     T0 = stod(argv[3]);
269     TMax = stod(argv[4]);
270     dT = stod(argv[5]);
271     BaseStepsLimit = stoll(argv[6]);
272     LastSteps = stoll(argv[8]);
273 
274     Initialize();
275     for (; T <= TMax + dT / 100; T += dT)
276     {
277         printf("Temperature: %f\n", T);
278         InitSpins();
279         printf("Initialized.\n");
280         PreProcess();
281         printf("Preprocessing done.\n");
282         double tmpMagn = GetTotalMagnetization(0);
283         double tmpAbsMagn = abs(tmpMagn);
284         double tmpE = GetTotalEnergy(0);
285         double dE = 0;
286         int iii = 0;
287 
288         for (long long mcs = 0; mcs < MCS; mcs++)
289         {
290             if (MCS>20 && mcs % (MCS / 20) == 0)
291             {
292                 printf("#"); // Progress meter
293             }
294             for (long long metro = 0; metro < N; metro++)
295             {
296                 int randNode = GetRandomNode();
297                 if (WillFlip(randNode, dE))
298                 {
299                     lattice[randNode] *= -1;
300                     tmpE += 2 * dE;
301                     tmpMagn += 2 * lattice[randNode];
302                     tmpAbsMagn += abs(lattice[randNode]);
303                 }
304             }
305             allRes[Ti].averageMagnetization += GetTotalMagnetization(0);
306             allRes[Ti].averageAbsoluteMagnetization += abs(tmpMagn);
307             allRes[Ti].averageSquareMagnetization += sqr(tmpMagn);
308             allRes[Ti].averageQuadMagnetization += sqr(sqr(tmpMagn));
309             allRes[Ti].averageEnergy += tmpE / 2;
```

```
310             allRes[Ti].averageSquareEnergy += (tmpE / 2)*(tmpE / 2);
311
312             if (mcs >= MCS - LastSteps)
313             {
314                 for (long long i = 0; i < N; i++)
315                     allRes[Ti].spins[i] += lattice[i];
316             }
317         }
318         printf("\nMonte-Carlo done.\n");
319         allRes[Ti].averageMagnetization /= NORM;
320         allRes[Ti].averageAbsoluteMagnetization /= NORM;
321         allRes[Ti].averageSquareMagnetization = allRes[Ti].
               averageSquareMagnetization / (NORM*N);
322         allRes[Ti].averageQuadMagnetization = allRes[Ti].
               averageQuadMagnetization / (NORM*N*N*N);
323         allRes[Ti].averageEnergy /= NORM;
324         allRes[Ti].averageSquareEnergy = allRes[Ti].averageSquareEnergy / (
               NORM*N);
325         allRes[Ti].magneticSusceptibility = N * (allRes[Ti].
               averageSquareMagnetization - sqr(allRes[Ti].
               averageAbsoluteMagnetization)) / T;
326         allRes[Ti].thermalCapacity = N * (allRes[Ti].averageSquareEnergy -
               sqr(allRes[Ti].averageEnergy)) / sqr(T);
327         allRes[Ti].BinderCumulant = 1 - allRes[Ti].averageQuadMagnetization
               / (3 * sqr(allRes[Ti].averageSquareMagnetization));
328
329         Ti++;
330     }
331
332     return 0;
333 }
```

Kod za "ran1" generator:

```
#pragma once
#define IA 16807
#define IM 2147483647
#define AM (1.0/IM)
#define IQ 127773
#define IR 2836
#define NTAB 32
#define NDIV (1+(IM-1)/NTAB)
#define EPS 1.2e-7
#define RNMX (1.0-EPS)

double ran1(long *idum)
{
    int j;
    long k;
    static long iy = 0;
    static long iv[NTAB];
    double temp;

    if (*idum <= 0 || !iy) {
        if (-(*idum) < 1) *idum = 1;
        else *idum = -(*idum);
        for (j = NTAB + 7; j >= 0; j--) {
            k = (*idum) / IQ;
            *idum = IA*(*idum - k*IQ) - IR*k;
            if (*idum < 0) *idum += IM;
            if (j < NTAB) iv[j] = *idum;
        }
        iy = iv[0];
    }
    k = (*idum) / IQ;
    *idum = IA*(*idum - k*IQ) - IR*k;
    if (*idum < 0) *idum += IM;
    j = iy / NDIV;
    iy = iv[j];
    iv[j] = *idum;
    if ((temp = AM*iy) > RNMX) return RNMX;
    else return temp;
}
#undef IA
#undef IM
#undef AM
#undef IQ
#undef IR
#undef NTAB
#undef NDIV
#undef EPS
#undef RNMX
```

Bibliografija

[Cowley & Buyers, 1972] Cowley, R. & Buyers, W. (1972). The Properties of Defects in Magnetic Insulators. *Reviews of Modern Physics*, 44(2), 406.

[Gallager, 2011] Gallager, R. (2011). Lecture 8: Markov Eigenvalues and Eigenvectors.

[Griffiths & Schroeter, 2018] Griffiths, D. J. & Schroeter, D. F. (2018). *Introduction to Quantum Mechanics*. Cambridge University Press.

[Heermann & Binder, 2010] Heermann, D. W. & Binder, K. (2010). *Monte Carlo Simulation in Statistical Physics*. Springer-Verlag Berlin Heidelberg.

[Kroese et al., 2013] Kroese, D. P., Taimre, T., & Botev, Z. I. (2013). *Handbook of Monte Carlo Methods*, volume 706. John Wiley & Sons.

[Lee & Yang, 1952] Lee, T.-D. & Yang, C.-N. (1952). Statistical Theory of Equations of State and Phase Transitions. II. Lattice Gas and Ising Model. *Physical Review*, 87(3), 410.

[Metropolis et al., 1953] Metropolis, N., Rosenbluth, A. W., Rosenbluth, M. N., Teller, A. H., & Teller, E. (1953). Equation of State Calculations by Fast Computing Machines. *The Journal of Chemical Physics*, 21(6), 1087–1092.

[Nolting & Ramakanth, 2009] Nolting, W. & Ramakanth, A. (2009). *Quantum Theory of Magnetism*. Springer Science & Business Media.

[Pickhardt & Seibold, 2014] Pickhardt, M. & Seibold, G. (2014). Income Tax Evasion Dynamics: Evidence from an Agent-Based Econophysics Model. *Journal of Economic Psychology*, 40, 147–160.

[Press et al., 1992] Press, W. H., Flannery, B. P., Teukolsky, S. A., & Vetterling, W. T. (1992). *Numerical Recipes in C: The Art of Scientific Computing, Second Edition*. Cambridge University Press.

[Radošević & Mali, 2017] Radošević, S. & Mali, P. (2017). *Zbirka zadataka iz matematičke fizike*. Prirodno-matematički fakultet u Novom Sadu, Departman za fiziku.

[Schneidman et al., 2006] Schneidman, E., Berry II, M. J., Segev, R., & Bialek, W. (2006). Weak Pairwise Correlations Imply Strongly Correlated Network States in a Neural Population. *Nature*, 440(7087), 1007.

[Srinivasan, 2013] Srinivasan, R. (2013). *Importance Sampling: Applications in Communications and Detection*. Springer Science & Business Media.

[Urumov, 2002] Urumov, V. (2002). Exact Solution of the Ising Model on a Pentagonal Lattice. *Journal of Physics A: Mathematical and General*, 35(34), 7317.

[Yildirim, 2012] Yildirim, I. (2012). Bayesian Inference: Metropolis-Hastings Sampling. *Dept. of Brain and Cognitive Sciences, Univ. of Rochester, Rochester, NY.*

Biografija

Stefan Velja rođen je 13.8.1996. u Novom Sadu. Osnovnu školu "Jovan Popović", a potom i specijalni smer za obdarene učenike u gimnaziji "Jovan Jovanović Zmaj" završava u Novom Sadu. Godine 2015. je upisao Prirodno-matematički fakultet, smer diplomirani fizičar.

31.7.2019. Stefan Velja

UNIVERZITET U NOVOM SADU
PRIRODNO-MATEMATIČKI FAKULTET

KLJUČNA DOKUMENTACIJSKA INFORMACIJA

Redni broj: **RBR**	
Identifikacioni broj: **IBR**	
Tip dokumentacije: **TD**	Monografska dokumentacija
Tip zapisa: **TZ**	Tekstualni štampani materijal
Vrsta rada: **VR**	Diplomski rad
Autor: **AU**	Velja Stefan, (405/15)
Mentor: **MN**	Dr Petar Mali
Naslov rada: **NR**	Određivanje termodinamičkih svojstava Izingovog modela na petougaonoj rešetki primenom Monte Karlo simulacija
Jezik publikacije: **JP**	srpski (latinica)
Jezik izvoda: **JI**	srpski/engleski
Zemlja publikovanja: **ZP**	Srbija
Uže geografsko područje: **UGP**	Vojvodina
Godina: **GO**	2019
Izdavač: **IZ**	Autorski reprint
Mesto i adresa: **MA**	Prirodno-matematički fakultet, Trg Dositeja Obradovića 4, Novi Sad
Fizički opis rada: **FO**	5 poglavlja/32 stranice/15 referenci
Naučna oblast: **NO**	Fizika
Naučna disciplina: **ND**	Teorijska fizika kondenzovanog stanja
Predmetna odrednica/ ključne reči: **PO**	Izingov model, Monte Karlo simulacija, Metropolisov algoritam
UDK	
Čuva se: **ČU**	Biblioteka departmana za fiziku, PMF-a u Novom Sadu
Važna napomena: **VN**	nema
Izvod: **IZ**	Izingov model na petougaonoj rešetki proučavan je metodom Monte Karlo simulacija uz korišćenje Metropolis-Hastings algoritma. Na osnovu simulacija, izračunata je približna vrednost kritične temperature modela i upoređena je sa teorijskom vrednošću.
Datum prihvatanja teme od NN veća: **DP**	
Datum odbrane: **DO**	
Članovi komisije: **KO**	
Predsednik:	Dr Miodrag Krmar, redovni profesor
član:	Dr Petar Mali, docent
član:	Dr Slobodan Radošević, vanredni profesor

UNIVERSITY OF NOVI SAD
FACULTY OF SCIENCE AND MATHEMATICS

KEY WORDS DOCUMENTATION

Accession number: **ANO**	
Identification number: **INO**	
Document type: **DT**	Monograph publication
Type of record: **TR**	Textual printed material
Content code: **CC**	Final paper
Author: **AU**	Stefan Velja (405/15)
Mentor/comentor: **MN**	Dr Petar Mali
Title: **TI**	Determination of Thermodynamic Properties of the Ising Model on a Pentagonal Lattice via Monte Carlo Simulations
Language of text: **LT**	Serbian (Latin)
Language of abstract: **LA**	English
Country of publication: **CP**	Serbia
Locality of publication: **LP**	Vojvodina
Publication year: **PY**	2019
Publisher: **PU**	Author's reprint
Publication place: **PP**	Faculty of Science and Mathematics, Trg Dositeja Obradovića 4, Novi Sad
Physical description: ***PD***	5 chapters/32 pages/15 references
Scientific field: **SF**	Physics
Scientific discipline: **SD**	Solid state theory
Subject/ Key words: **SKW**	Ising model, Monte Carlo simulation, Metropolis algorithm
UC	
Holding data: **HD**	Library of Department of Physics, Trg Dositeja Obradovića 4
Note: **N**	none
Abstract: **AB**	The Ising model on a pentagonal lattice was studied via Monte Carlo simulations, using the Metropolis-Hastings algorithm. The critical temperature of the model was estimated based on the simulations and compared to the theoretical value.
Accepted by the Scientific Board: **ASB**	
Defended on: **DE**	
Thesis defend board: **DB**	
President:	Miodrag Krmar, full professor
Member:	Petar Mali, assistant professor
Member:	Slobodan Radošević, associate professor