5. Matrično-produktni nastavki: TEBD

Miha Srdinšek

Ta teden sem potem nadaljeval z delom, ki sem ga začel v prejšnji nalogi. V prejšnji nalogi sem predstavil, kako lahko stanje verige $|\Lambda|$ spinov, poljubnih velikosti (d nivojev) opišemo v bazi

$$\{|s_1, s_2, \ldots, s_{|\Lambda|}\rangle\}_{s_n \in 1,\ldots,d \,;\, \forall n \in 1,\ldots,|\Lambda|}, \tag{1}$$

enostavno s skupino $|\Lambda|$ tenzorjev $A^{(j)}_{s_j}$, kjer $j \in 1, 2, ..., |\Lambda|$. Na tak način potem lako reproduciramo stanje v tej bazi, tako da izračunamo koeficiente kot

$$\psi_{s_1,s_2,\ldots,s_{|\Lambda|}} = \langle s_1, s_2, \ldots, s_{|\Lambda|}|\psi\rangle = A^{(1)}_{s_1} A^{(2)}_{s_2} \ldots A^{(|\Lambda|)}_{s_{|\Lambda|}}. \tag{2}$$

V tej nalogi sem moral večinoma uporabljati drugo notacijo. Ta notacija ima nekaj prednosti, kar spremenimo, pa je to, da definiramo

$$B^{(1)}_{s_1} := A^{(1)}_{s_1} \qquad B^{(j)}_{s_j} = \left(\lambda^{(j-1)}\right)^{-1} A^{(j)}_{s_j}, \tag{3}$$

pri čemer so $\lambda^{(j)}$ diagonalne matrike Schmidtovih koeficientov, ki jih pridelamo med razcepom začetnega stanja na A matrike. Če bi želeli zdaj pridi nazaj do začetnega stanja bi morali koeficiente izračunati kot

$$\psi_{s_1,s_2,\ldots,s_{|\Lambda|}} = B^{(1)}_{s_1} \lambda^{(1)} \ldots \lambda^{(|\Lambda|-1)} B^{(|\Lambda|)}_{s_{|\Lambda|}}. \tag{4}$$

Temu bom od zdaj naprej pravil algoritem 1. S tem torej mislim razcep začetnega stanja na matrike A ali B in λ, pa tudi vračanje v začetno stanje iz teh matrik.

Algoritem je v resnici relativno preprost, amapk je dolg in hitro se izgubiš v njem. Pri tem uporabimo veliko tega kar smo se naučili pri prejšnjih nalogah. Algoritem TEBD je namreč algoritem s katerim propagiramo neko začetno stanje v času, lahko pa tudi v imaginarnem času β. Tako kot smo to že počeli pri ostalih metodah, se bomo tudi tu poslužili Suzuki-Trotterjeve sheme, kjer propagator enostavno razbijemo na dva dela, ki sama s sabo komutirata, med seboj pa ne, torej v našem primeru, ker obravnavamo Heisenbergov model

$$U(z) = \left(U(z/m)\right)^m \approx \left[\prod_{k=1}^{|\Lambda|/2} U_{2k-1,sk}(z/m) \prod_{j=1}^{(|\Lambda|-1)/2} U_{2k,2k+1}(z/m)\right]^m, \tag{5}$$

z je čas do katerega propagiramo z m koraki dolžine z/m. Za ta razcep lahko uporabimo katerokoli shemo, ki so jo spoznali, jaz sem recimo ves čas uporabljal shemo četrtega reda. Zdaj se moramo samo še naučiti narediti posamične korake, torej operacijo

$$U_{i,i+1}(z/m)|\psi\rangle, \tag{6}$$

saj imamo vedno samo interakcijo med sosednima paroma. Na enem paru potem delujemo enostavno tako, da zapišemo

$$B^{(j,j+1)}_{s_j,s_{j+1}} = \sum_{s_j,s'_{j+1}} U'_{(s_j,s_{j+1}),(s'_j,s'_{j+1})} B^{(j)}_{s'_j} \lambda^{(j)} B^{(j+1)}_{s'_{j+1}}, \tag{7}$$

kjer za naš primer, tako kot že pri neki prejšnji nalogi uporabimo

$$U'(z) = e^{z\vec{\sigma}_1\cdot\vec{\sigma}_2} = e^{-z}\begin{pmatrix} e^{2z} & 0 & 0 & 0 \\ 0 & ch(2z) & sh(2z) & 0 \\ 0 & sh(2z) & ch(2z) & 0 \\ 0 & 0 & 0 & e^{2z} \end{pmatrix}. \tag{8}$$

kar se da narediti pravzaprav precej lepo. Samo ne smemo pozabiti na bomo na koncu zopet dobili neko matriko. V tem primeru pravzaprav matrike množimi z nekimi koeficienti $U'_{i,j}$ in jih seštevamo. A naslednji kroak algoritma je še bolj nenavaden. Zapišemo namreč

$$Q_{(k_{j-1},s_j),(k_{j+1},s_{j+1})} := \lambda^{(j-1)}_{k_{j-1}} \left(B^{(j,j+1)}_{(s_j,s_{j+1})}\right)_{k_{j-1},k_{j+1}} \lambda^{(j+1)}_{k_{j+1}}. \tag{9}$$

Profesor pravi, da to sicer ni nujen korak, je pa nujen za dobro natančnost. Pri konstruiranju te matrike Q moramo biti zelo natančni. Jaz sem se tega lotil na relativno preprost način, tako da sem indekse Q obravnaval kot binarna števila. Torej sem jo definiral tako, da sem šel z zanko skozi vse elemnte in določal s s prvo števko v binarnem zapisu in k z številom levo od prve števke. Zdaj je treba dati to matriko Q v algoritem za SVD razcep in na koncu definirati nove B kot

$$(B'^{(j)}_{s_j})_{k_{j-1},k_j} := \left(\lambda^{(j-1)}_{k_{j-1}}\right)^{-1} U_{(j_{j-1},s_j),k_j} \tag{10}$$

$$\lambda'^{(j)}_{k_j} := D_{k_j,k_j} \tag{11}$$

$$(B'^{(j+1)}_{s_{j+1}})_{k_j,k_{j+1}} := V^{\dagger}_{k_{j'},(k_{j+1},s_{j+1})} \left(\lambda^{(j+1)}_{k_{j+1}}\right)^{-1} \tag{12}$$

kjer so U,D,$V^{\dagger}$ matrike, ki jih dobimo s SVD razcepom. To je zdaj en korak. V enem koraku algoritme, naredimo vse take korake lihe ali sode pare in to počnemo po eni izbed ST shem. Temu bom rekel algoritem 2.

Druga pomembna stvar je, kako izračunati kotrakcijo med dvema stanjema s pomočjo algoritma 1, saj si s tem lahko močno pohitrimo algoritem. To storimo tako, da stanji $|\psi\rangle$ in $|\phi\rangle$ razcepimo na matrike A. V prektični situaciji imaš sicer matrike B in jih povem sproti pretvarjaš v matrike A. Potem pa ustvariš nove matrike.

$$T^{(j)} = \sum_{s=0}^{d-1} (A^{\psi}_s)^{(j)} \otimes (A^{\phi}_s)^{(j)}, \tag{13}$$

pri čemer paziš,da prvo definiraš brez lamde v priemru da delaš z B. Na koncu le še zmnožiš v se te matrike kot

$$\langle\psi|\phi\rangle = T^{(1)}T^{(2)}\dots T^{(n)}, \tag{14}$$

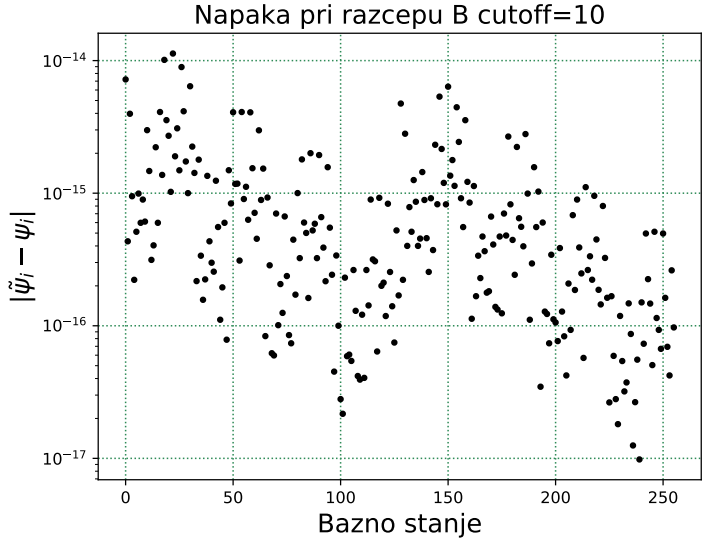
in s tem dobiš kontrakcijo. Na isti način bi potem lahko merili tudi opazljivke, tako, da enostavno na mestu, kjer opazljivke deluje spremenimo definicijo T, kot

$$T^{(j)} = \sum_{s,s'=0}^{d-1} O_{s,s'} (A_s^{\psi})^{(j)} \otimes (A_{s'}^{\phi})^{(j)}, \tag{15}$$

I. Algoritem

Najprej sem napisal celotni algoritem in to precej direktno. Večinoma sem po definiciji prepisoval in pri tem ves čas vse definiral s kompleksnimi števili. Ves čas sem imel vse komponente A in B določene velikosti (cutoff). Že od začetka sem vse definiral s seznami napolnjenimi z ničlami in enakih dimenzij, potem pa sem jih samo polnil. Ko sem definiral nov A, sem ga vstavil v moj seznam, tako da sem ga dal v zgorji levi kot matrike, če je bil manjši od mojih matrik, če je bil večji, pa sem ga obrazal, tako da je od njega ostal samo zgornji levi del. Na tak način se je vse čisto lepo izšlo.

Zopet sem preveril, če racep na B ohrani začetno stanj isto, in ga je ohranilo 1. Poleg tega sem moral pri konstrukciji B tudi paziti, da nisem delil z $\lambda = 0$. Tega sem se ognil tako, da sem delil z while stavkom, ki se je ustavil, ko je prišla prva λ enaka nič, ali manjša od določene meje, ki sem jo postavil. Večinoma sem kar vstrajal pri meji 0. Začetna stanja sem za majhne N na začetku konstruiral tako, da sem izžrebal $2x2^N$ normalnih naključin števil z deviacijo 1 in $\mu = 0$. To se je hitro izkazalo za nemogoče pri N nad 25, zato sem kasneje zapisal algortime tako, da sem izžrebal zgolj $2xN$ normalnih naključnih števil in jih postavil v levi zgornji kot matrik B, ostalo pa pustil prazno. Za λ sem storil podobno in sicer zgolj pri vsaki lambdi postavil na vrh 1 in spodaj ničle. Dokaz da je tak način ustvaril lepo stanje, ki se ohranja z rezcepom vidimo spet na sliki 1.



Slika 1: *Slika prikazuje za koliko se razlikujejo vrednosti koeficientov za bazna stanja pred in po razcepu. Hkrati gre za bazna stanja dobljena iz enostavnih matrik B in koeficientov λ.*

Algoritem za iskanje osnovnega stanja in predvsem nejgove energije, sem potem zapisal tako, da sem začel z nekim naključnim kompleksnim normiranim stanjem. To verjetno ni nujno, lahko bi si vzel kakšno bolj enostavno stanje. Potem pa sem ponavljal in ponavljal korake in na

koncu računal

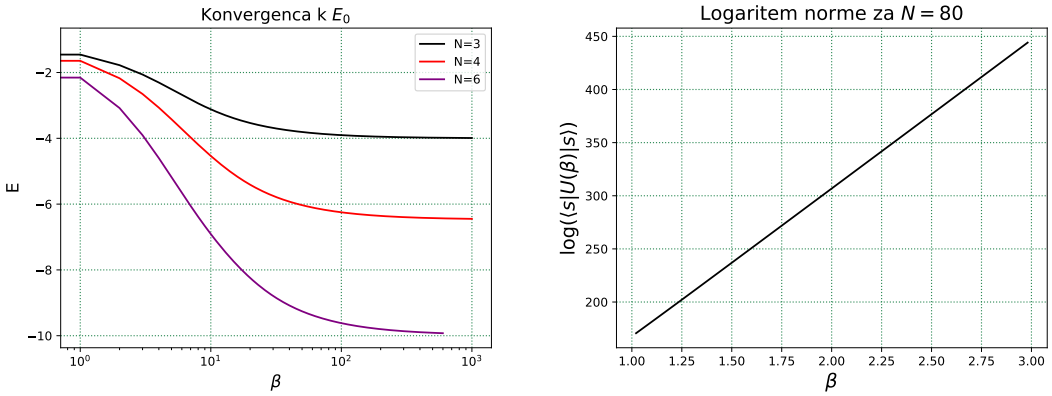
$$\beta E_0 = -\log\left(\langle s|e^{-\beta H}|s\rangle\right), \tag{16}$$

kjer je H hamiltonjan, s začetno stanje in E_0 energija osnovnega stanja. Po nekem β sem dobil lepo premco, iz katere sem potem izluščil naklon E_0, kar vidimo na sliki 2 desno. Normo sem računal z metodo, ki smo se jo naučili na predavanjih, ker med seboj pravilno zmnožiš matrike A (14).

Preden se zares lotim računanja energije s prileganjem premice na logaritem norme, bi rad predstavil alternativni način, po enačbi

$$E = -\lim_{\beta\to\infty}\frac{1}{\beta}\log\frac{\langle s|e^{-\beta H}|s\rangle}{\langle s|s\rangle}. \tag{17}$$

V tem primeru sem torej risal le zgornjo količino pri vsakem β in opazoval kam konvergira. To vidimo na sliki 2 levo. S takim načinom sem lahko določil energije do kakšne tretje morda četrte decimalke natančno, a še zdaleč ne tako dobro kot nam ta algoritem dopušča. Problem je tudi v tem, da na neki točki norma divergira in tam račun več ni mogoč in višji kot je N prej divergira.



Slika 2: *Slika levo: prikazuje kako zgornji izraz res počasi konvergira k energiji osnovnega stanja, ko povečujemo β, za nekaj $N \in 3, 4, 6$, kjer je N število spinov $1/2$. Slika desno: Dokaz, da norma res eksponentno narašča. To je za primer $N = 80$.*

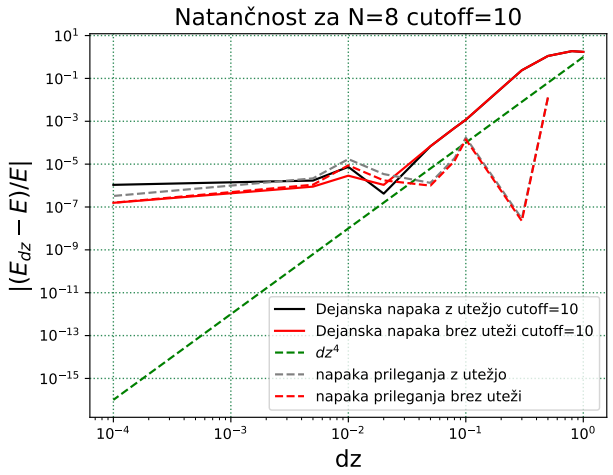
Pri prileganju premice na eksponent pa je norma zelo hitro prešla v eksponentno odvisnost (večinoma že kar pri $\beta = 1$), tako da niti nisem potreboval dolgih "časov". Na tak način sem izračunal kar nekaj energij in to pri različnih natančnostih zgolj zato, da lahko človek ugotovi do katere decimalke je algoritem dejansko konvergiral 1. Pri manjših N sem poleg tega lahko tudi izračunal dejansko napako, saj poznam eksatno energijo iz diagonalizacije hamiltnjana.

cutoff	N	z	meja	E_0	Eksaktno	razlika
10	3	0,01	0	-4.0000000529	-4	-5.3 10^{-8}
10	3	0,0001	0	-4.00000000000125	-4	-1.2 10^{-12}
10	4	0,01	0	-6.464102178021265	-6.4641016151377535	-5.6 10^{-7}
5	4	0,0001	0	-6.4641016151256245	-6.4641016151377535	1.2 10^{-11}
10	4	0,0001	0	-6.464101615137336	-6.4641016151377535	4.2 10^{-13}
10	5	0,01	0	-7.71154552704	-7.7115450132719765	-5.1 10^{-7}
10	6	0,001	0	-9.97430853551	-9.974308535551715	4.1 10^{-11}
10	7	0,001	0	-11.34495872283	-11.344958722746599	-8.3 10^{-11}
20	8	0,01	0	-13.49973180338	-13.49973039475156	-1.4 10^{-6}
30	10	0,01	0	-17.032141948123	-17.032140829131514	-1.1 10^{-6}
50	10	0,01	0	-17.0321389840	-17.032140829131514	1.8 10^{-6}
100	13	0,1	0	-22.12677932	-22.10128838833474	-0.025
10	13	0,01	0	-22.1012621545	-22.10128838833474	2.6 10^{-5}
70	13	0,01	0	-22.10129051407	-22.10128838833474	-2.1 10^{-6}
10	13	0,001	0	-22.101248592810702	-22.10128838833474	4 10^{-5}
50	13	0,001	0	-22.10128796236374	-22.10128838833474	4.3 10^{-7}
10	15	0,1	0	-25.6972361761		
10	16	0,01	0	-27.646646513813195		
10	20	0,1	0	-34.77793578629		
10	20	0,01	0	-34.72904908860		
10	20	0,001	0	-34.729081019191554		
10	21	0,01	0	-36.34475334		
10	21	0,001	0	-36.34436506729		
10	22	0,001	0	-38.27556258950117		
10	25	0,001	0	-43.451329795132025		
10	50	0.001	0	-87.86480955977707		
10	100	0.01	0	-173.87849263488994		
15	100	0.01	0	-174.495282945616		
20	100	0.01	0	-175.1409563221842		
30	100	0.01	0	-174.87244327330467		
50	100	0.01	0	-175.78730300382287		

Tabela 1: *V tabeli so prikazane izračunane energije za različne N. Pri izbranem N sem včasih pogledal še kako se rezultat spremeni s cutoffom ali pa z dz. V glavnem velja, da se dosti bolj splača zmanjšati dz kot povečati cutoff. Pri manjših N sem izračunal še eksaktno energijo z diagonalizacijo hamiltonjana in tako ocenil absolutno napako.*

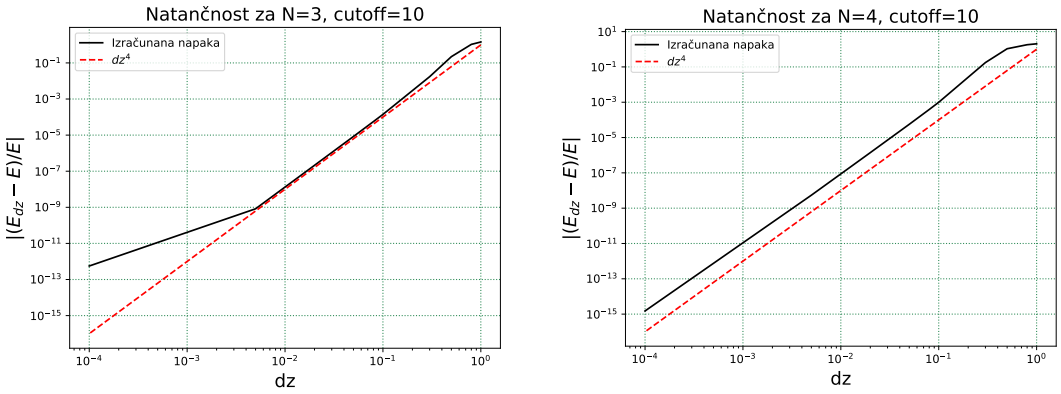
II. Napaka

Po tem ko sem na opisani način zapisal delujoč algoritem, ki je dajal primerne rezultate, sem se odločil da moram malo bolje oceniti napako, ki jo produciram. Predvsem me je zanimalo kako je napaka odvisna od "časovnega"koraka dz in od rezanja oziroma $cutoffa$. Za prvo pričakujemo da bo sorazmerna dz^4, za drugo pa nimamo tako natančne ocen, zgolj numerično. Tekom algoritma lahko seštevamo kvadrate odrezanih λ. Sam sem to izvedel tako, da sem pred seštevanjem kvadratov še normiral lambde, kar verjetno ni potrebno. S tem sem dobil oceno za absolutno napako pričakovene vrednosti. Pri prileganju eksponenta na normo v odvisnosti od β sem potem upošteval to napako kot utež in na koncu dobil oceno za napako prileganega koeficienta. V končni fazi sem torej napako lahko uporabil zgolj pri prileganju in s tem rahlo spremenil oceno napake pri energiji, vendar je v tem primeru tudi energija malo drugačna. Ta razmislek vidimo na sliki 3.



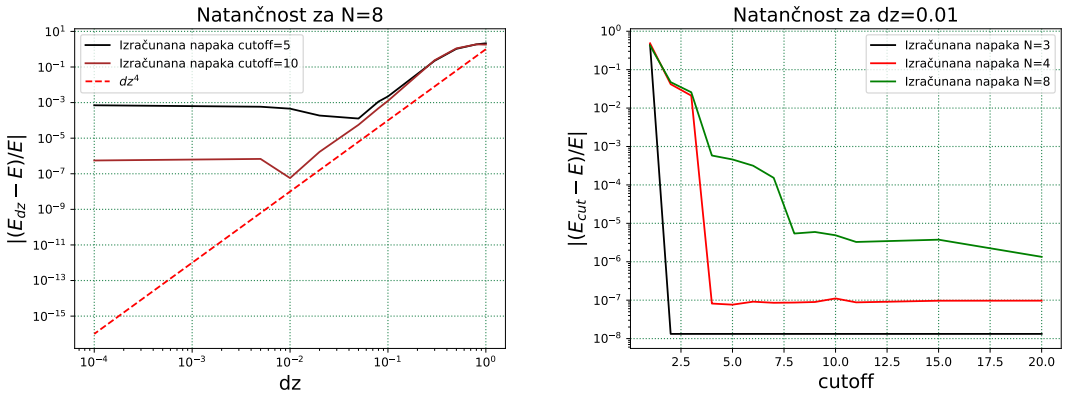
Slika 3: *Na sliki sem prikazal kako se dejanska napaka izračunane energije ujema z napovedjo* ($N = 8$, *cutoff*$= 10$) *v odvisnosti od dz. S polno črno črto sem prikazal relativno napako, če upoštevamo kot utež pri prileganju eksponenta, ocenjeno napako od odrezanih λ. S polno rdečo črto pa relativno napako če uteži ne upoštevamo pri prileganju. S črtkanima črtama (rdeča in siva) sem prikazal kolikšna je ocena napake od prileganja. Z zeleno črto pa sem prikazal krivuljo dz^4, kar je ocena za napako mojega integratorja.*

Ker me je presenetilo tako lepo ujemanje, sem si žele pogledati kaj se zgodi, kadar je cutoff dovolj velik, da nebi smel vplivati na napako. To vidimo na sliki 4, kjer je dejasnko edini prispevek samo dz^4. Napak zaradi prileganja potem nisem več risal, saj so zelo nenatančne in se ne ujemajo vedno z rezultatom (uporabljal sem dve rutini za prileganje premic in tista, ki je vrnila oceno napake je dajala tudi manj natančne rezultate saj je rezultat določala numerično). Tako ali tako je bil cel namen risanja te napake, poskus določitve prispevka od rezanja λ k koeficientu, kar pa smo videli na sliki 3 da nam ni spelo določiti, saj smo pridelali le napako od prileganja, ki bo očitno vedno kar precej velika.



Slika 4: *Na slikah sem prikazal s polno črto dejansko napako in s črtkano črto dz^4. S tem sem potrdil, da je dejansko napaka od dz odvisna kot dz^4.*

Na slikah 5 pa sem pogledal kakšen je potem dejanski vpliv cutoffa na napako pri izbranem dz. Na sliki levo vidimo, da obstaja očitna razlika v napaki pri različnih cutoffih, na sliki desno pa sem to prikazal bolj sistematično a hkrati boj abstraktno. Lepo se opazi, kako pri izbranem dz obstaja minimalna napaka, ki se ji približamo z večanjem cutoffa.

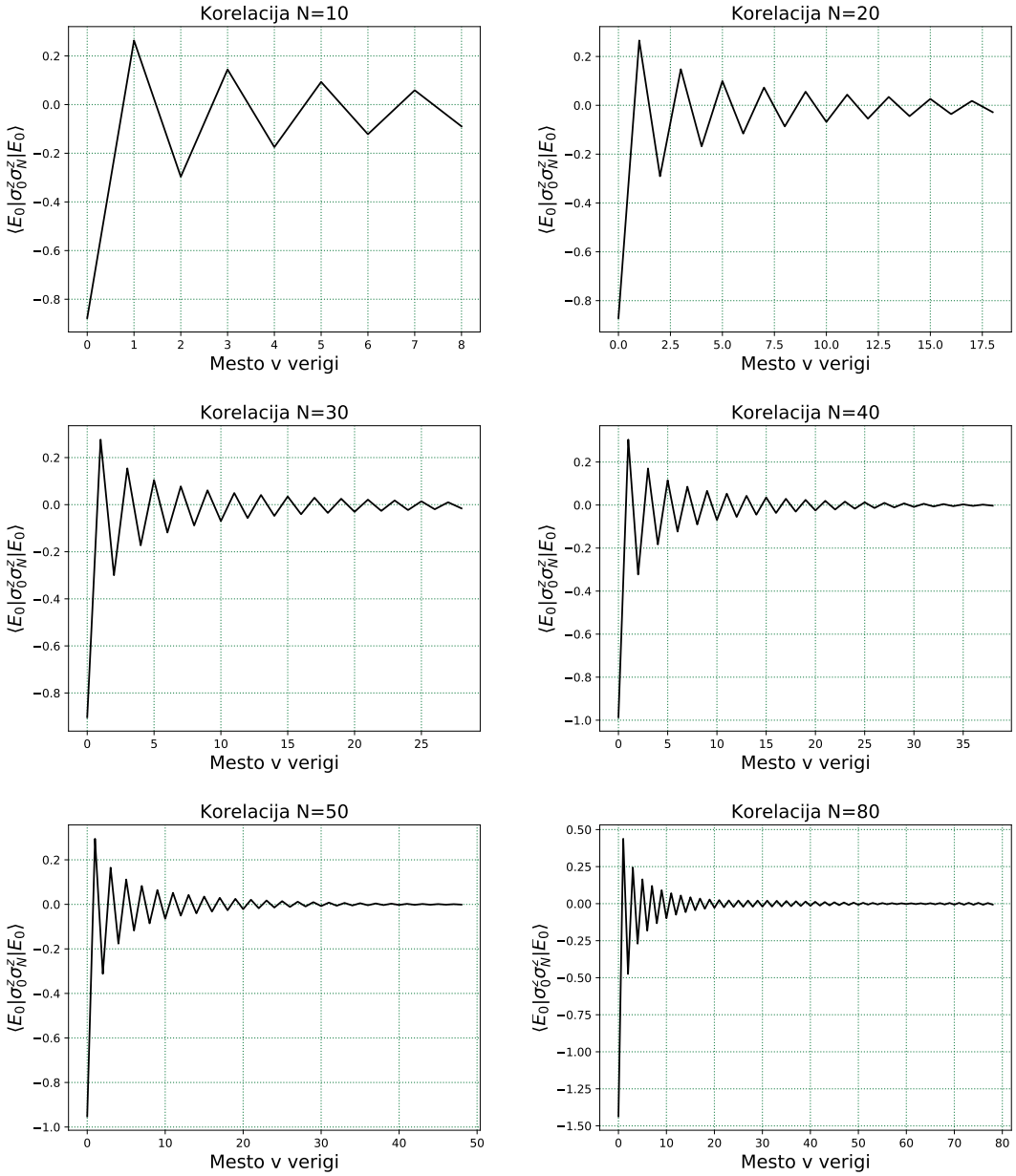


Slika 5: *Na slikah sem prikazal kako je dejanska napaka odvisna od cutoffa. La levi imamo napako od dz za $N = 8$ pri različnih cutoffih (5,10), na desni pa enostavno napako v odvisnosti cutoffa pri fiksnem $dz = 0.01$ in različnih $N \in 3, 4, 9$.*

Potril sem torej odvisnsot napake od dz^4 in opazil nenavadno napako od $cutoffa$, a v praksi bi pač vedno izračunal energijo pri različnih cutoffih in videl do katerega mesta je energija skonvergirala. Druga možnost bi bila, da bi za različne N poskusil določiti konstanto pred dz^4 in potem poskusil ekstrapolirati odvisnost na ostale N. S tem bi potem dobil dobro odvisnost. Ker se mi to ne zdi namen te naloge, se potem nisem zagnal v to.

III. Korelacijska funkcija

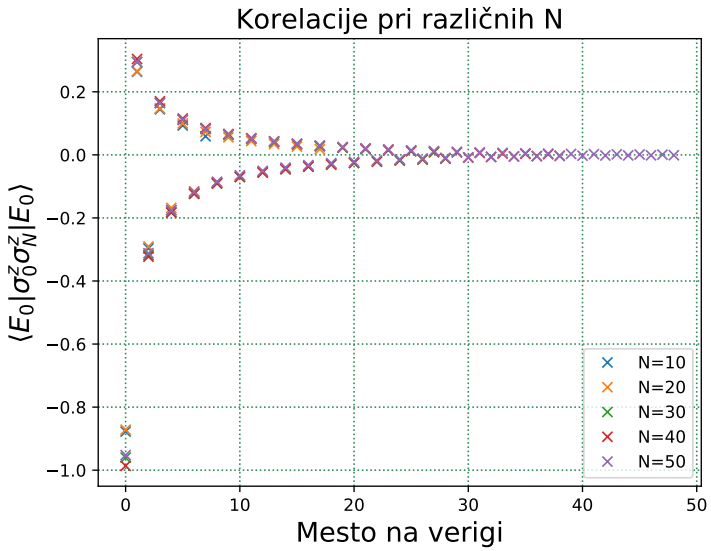
Korelacijsko funckijo sem izračunal po postopku opisanim v uvodu. Najprej sem pustil algoritem, za izračun energije da skonvergira in vzel tisti paket matrik B in λ. S tem sem izračunal korelacijo in jo normiral. Na tak način sem sproduciral rezultate prikazane na slikah 6



Slika 6: *Slike prikazujejo korelacijske funkcije v odvisnosti od N. Korelacijske funckije so oblike $\sigma_0^z\sigma_k^z$. Čas relaksacije in cutoff sem prilagajal za vsak primer posebej. Pri velikih N je to še posebej zahtevno.*

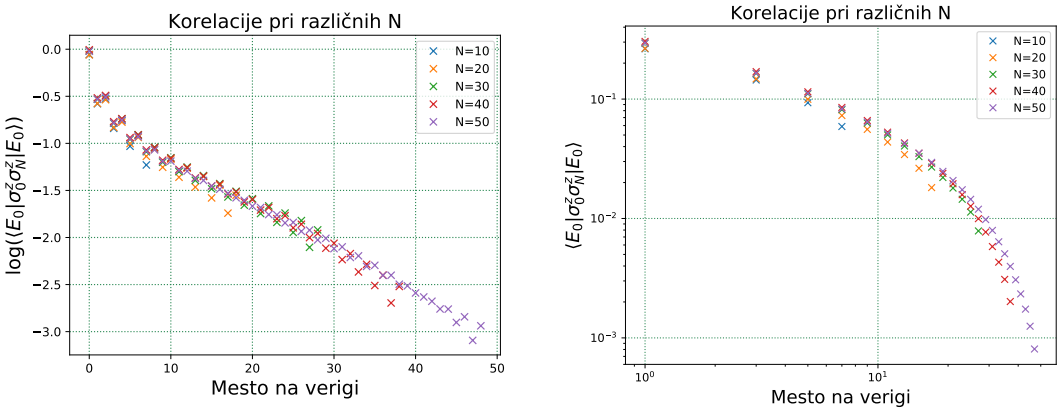
Na slikah vidimo pričakovan rezultat. Pri prejšnji oddaji sem risal absolutne vrednosti teh korelacij. Ker se mi je zdelo da so taki rezultati bolj pregledni sem zamenjal način. Moje korelacije so sicer kompleksna števila, vendar so kompleksni prispevki okoli dvajset redov velikosti manjši od realnih, zato se s tem nisem obremenjeval. Pri velikem N, recimo $N = 80$

sem imel že težave s konvergenco in računanje korelacij je postalo bolj časovno zahtevno. Zanimivo se mi je zdelo, da se korelacije pravzaprav sploh ne spreminjajo, temveč le nadaljujejo, kar vidimo na sliki 7.



Slika 7: *Na sliki vidimo izračunano spin spin korelacijo pri različnem številu delcev. Kar je zanimivo, je to, da ima korelacija enako odvisnsot, kot funkcija mesta v verigi, le konča se ko zmanjka mest. Če si rezultat natačno pogledamo lahko opazimo, da se negativni prispevki bolj ujemajo med sabo kot pozitivni.*

Zanimivo bi bilo preveriti tudi kakšna je odvisnost korelacijske funkcije od mesta v verigi. Preveriti bi se splačalo ali je odvinost eksponentna, potenčna, ali kaj drugega.



Slika 8: *Slike prikazujejo rezultate 7 v različnih skalah.*

Iz slik se lepo vidi, da mora biti na nekem vmesnem delu odvisnost eksponentna, a na začetku eksponentna zagotovo ni. Po drugi strani je res očitno da potenčna ni, razen morda čisto na začetku. Iz tega razmilseka, z nekaj divjega ugibanja in na podlagi namigov iz literature sem ugotovil, da je funkcija

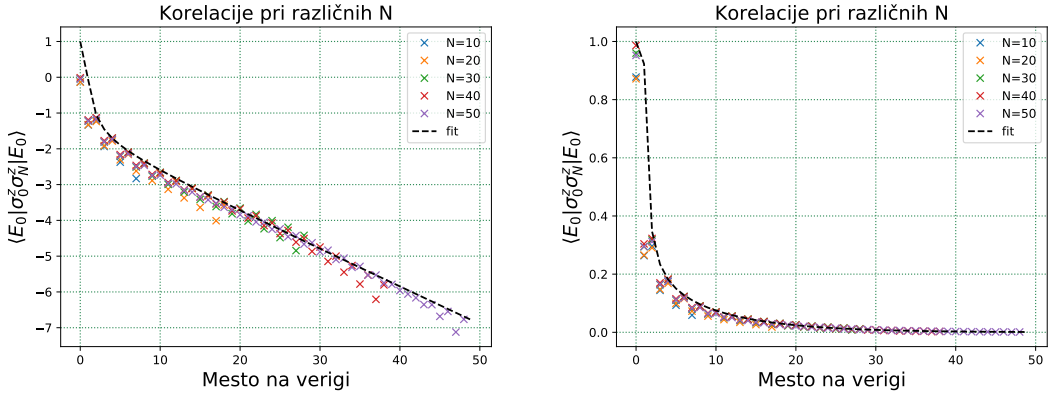
$$f(x) = e^{c-a/x-bx}, \tag{18}$$

zelo dober približek odvisnsoti ki jo vidimo. Pri tem sem dodatno opazil, da najverjetneje velja $a = b$. Če tako funkcijo res prilegamo na naše rezultate dobimo zelo dobro ujemanje,

kar vidimo na sliki 9. Mislim, da res lahko zaključimo $a = b$ in s tem dobimo neko oceno za korelacijsko dolžino

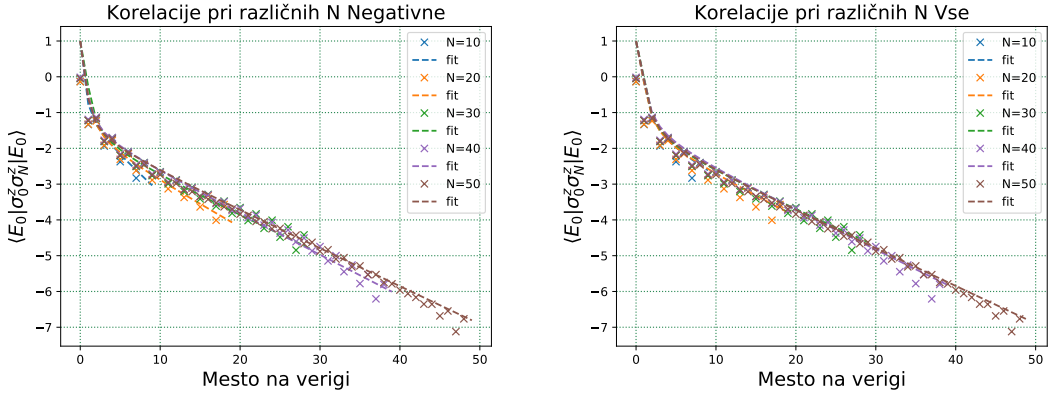
$$\xi \approx 1/1.73 = 0.58. \tag{19}$$

Ta rezultat je povsem neintuitiven, saj res ne izgleda tako, zato bi morali malo redefinirati konstante in izbrati nekaj drugega za korelacijsko dolžino. Pri tem moram poudariti, da nisem upošteval zelo pomembnega detajla - da je rešitev alternirajoča, in pozitivni in negativni del ne sledita isti krivulji. Slika absolutne vrednosti je rahlo zigzagasta. Ocenjujem da gre tukaj za minorne popravke, sem pa na vseh slikah 8 in 9 risal absolutne vrednsoti, da sem se ognil kompleksnim številom in negativnim vrednostim v logaritemski skali.



Slika 9: *Slike prikazujejo kako pri pravilnem izboru parametrov, funkcija 18 res dobro opiše obnašanje rezultatov prikazanih na 7. Parametri so* $c = 0.10412597$, $a = -1.74780376$, $b = -1.72400997$, *ki jih dobimo če prilegamo krivuljo na podatke za* $N = 50$. *Zanimivo lahko opazimo, da morda velja* $a = b$. *Če uporabim dobljeni konstanti in predpostavim* $a = b$ *dobim identični rezultat, je pares, da za metoda za prileganje odpove, če privzamem* $a = b$ *in ne najde tako dobrega prileganja.*

Na isti način lahko potem prilegamo mojo funkcijo 18 na podatke za ostale N in s tem dobimo več ocen za ξ, lahko pa tudi prilegamo funkcijo zgolj na lihe člene (tiste z negativno vrednostjo). V zadnjem primeru dobimo kanček nižjo vrednost konstant a. Lepo se vidi tudi to, da konstante konvergirajo proti vrednosti pri $N = 50$. Za dokaz, da je temu res tako, sem to prikazal na slikah 10.



Slika 10: *Sliki prikazujeta isto kot na 9, le da prilegam funkcijo 18 na na podatke za vse* N. *Levo samo na tiste člene, ki imajo negaitvno vrednost, desno pa na vse. Kot smo upali so slike precej nepregledne, kar pomeni da funkcija res vedno ustreza podatkom.*

Kot smo upali so slike precej nepregledne, kar pomeni da funkcija res vedno ustreza podatkom. Kadar prilegam na vse točke, so vrednosti a na intervalu $a \in (1.63, 1.86)$, kadar pa prilegam le negativne člene, so vrednsoti a na intervalu $a \in (0.7, 1.15)$ in $b \in (1.3, 1.8)$. Očitno torej za lihe člene ne velja $a = b$, ampak saj to sploh ni tisto kar nas zanima, to omenjam samo kot zanimivost.