5. Matrično-produktni nastavki

Miha Srdinšek

I. Uvod

V tej nalogi sem zapisal algoritem, ki razbije mnogodelčno kvantno stanje na produkt matrik.

Najprej bom definiral osnovne pojme. Opravka bomo imeli s kvantno mrežo točk Λ dolžine $|\Lambda|$. V našem primeru si bomo to lahko predstavljali kot verigo spinov. $|\Lambda|$ je v tem primeru število spinov, med tem ko ima vsak spin d nivojev (recimo $d = 2$, plus in minus, za $s = 10/2$). Hilbertov prostor celotne kvantne mreže je torej

$$\mathcal{H}_\Lambda = \bigotimes_{x\in\Lambda} \mathcal{H}_x, . \tag{1}$$

Stanje lahko torej razvijemo po bazi

$$\{|s_1, s_2, \ldots, s_{|\Lambda|}\rangle\}_{s_n\in 1,\ldots,d\ ;\ \forall n\in 1,\ldots,|\Lambda|} \tag{2}$$

in tako dobimo, mnogodelčno valovno funkcijo kot

$$|\psi\rangle = \sum_{All\ combinations\ of\ s} \psi_{s_1,s_2,\ldots,s_{|\Lambda|}} |s_1, s_2, \ldots, s_{|\Lambda|}\rangle. \tag{3}$$

Tu uvedemo algoritem ki smo ga spoznali na predavanjih. Temelji na SVD razcepu, s pomočjo katerega valovno funkcijo izrazimo kot

$$\psi_{s_1,s_2,\ldots,s_{|\Lambda|}} = \langle s_1, s_2, \ldots, s_{|\Lambda|}|\psi\rangle = \langle L|A^{(1)}_{s_1} A^{(2)}_{s_2} \ldots A^{(|\Lambda|)}_{s_{|\Lambda|}}|R\rangle, \tag{4}$$

kjer so A matrike, ki jih bom še definiral. Za ta postopek je potreben razcep SVD pri akterem matriko razcepimo na tri matrike. O tem postopku smo se učili e na prvi stopnji in smo ga uporabljali pri modelski analizi. Iz tega razloga, se vanj nisem poglabljal še enkrat, ampak sem zgolj klical funkcijo, ki naredi SVD razcep, iz pythonove knjižnice *numpy.linalg*.

Postopek sem izvedel tako, kot nam je to profesor predstavil na predavnjih. Začel sem z nekim stanjem , ki ga lahko zapišem z valovno funkcijo

$$\psi_{s_1,s_2,\ldots,s_{|\Lambda|}}. \tag{5}$$

To valovno funkcijo sem potem preoblikoval na poseben način, tako da sem lahko izvedel želeni SVD razcep. Če si $(s_1, s_2, \ldots, s_{|\Lambda|})$ predstavljam kot število v $d-$tiškem zapisu, potem

si lahko valovno funkcijo $\psi_{s_1,s_2,\ldots,s_{|\Lambda|}}$ namesto kot funkcijo vseh kombinacij predstavljamo kot funkcijo parov $\psi_{s_1,(s_2,\ldots,s_{|\Lambda|})}$. To pomeni, da se začetno stanje verige treh spinov $1/2$

$$\psi = (0,1,0,0,1,0,0,0) \tag{6}$$

lahko zapiše kot

$$\psi_0 = (0,1,0,0) \quad in \quad \psi_1 = (1,0,0,0), \tag{7}$$

oziroma kot matriko $\psi_{s_1,(s_2,\ldots,s_{|\Lambda|})}$. Na taki matriki potem izvedemo prvi SVD razcep in dobimo

$$\psi_{s_1,(s_2,\ldots,s_{|\Lambda|})} = \sum_{k_1=1}^{M_1} U^{(1)}_{s_1,k_1} \lambda^{(1)}_{k_1} V^{(1)\dagger}_{k_1,(s_2,\ldots,s_{|\Lambda|})}, \tag{8}$$

in definiramo matriki

$$(A^{(1)}_{s_1})_{k_1} := U^{(1)}_{s_1,k_1}, \qquad \psi^{(2)}_{k_1,s_2,\ldots,s_{|\Lambda|}} := \lambda^{(1)}_{k_1} V^{(1)\dagger}_{k_1,s_2,\ldots,s_{|\Lambda|}}. \tag{9}$$

Potem pravzraprav ponavljamo isti postopek še in še s predpisom za $j-$ti razcep

$$\psi^{(j)}_{(k_{j-1},s_j),(s_{j+1},\ldots,s_{|\Lambda|})} = \sum_{k_j=1}^{M_j} U^{(j)}_{(k_{j-1},s_j),k_j} \lambda^{(j)}_{k_j} V^{(j)\dagger}_{k_j,(s_{j+1},\ldots,s_{|\Lambda|})}, \tag{10}$$

kjer iz tega razcepa počrpamo definiciji

$$(A^{(j)}_{s_j})_{k_{j-1},k_j} := U^{(j)}_{(k_{j-1},s_j),k_j} \qquad \psi^{(j+1)\dagger}_{k_j,(s_{j+1},\ldots,s_{|\Lambda|})} := \lambda^{(j)}_{k_j} V^{(j)\dagger}_{k_j,(s_{j+1},\ldots,s_{|\Lambda|})}. \tag{11}$$

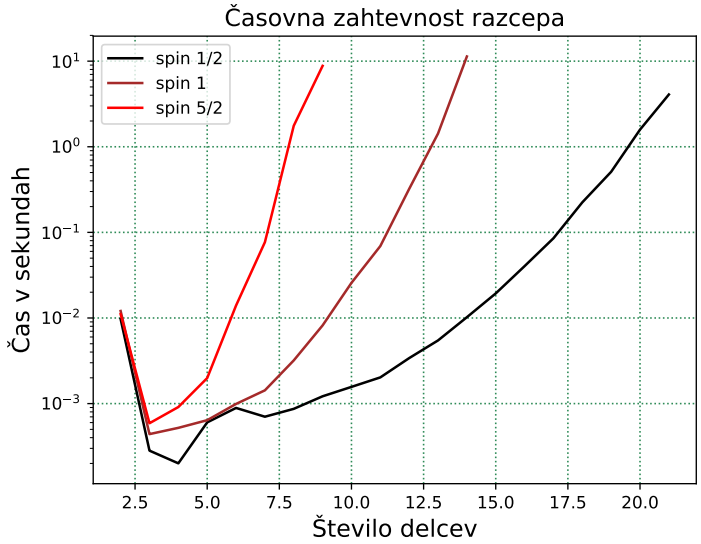
Algoritem zaključimo s tem da v zadnjem koraku $|\Lambda|$ priredimo

$$(A^{(|\Lambda|)}_{s_{|\Lambda|}})_{k_{|\Lambda|-1}} := \psi^{(|\Lambda|)}_{k_{|\Lambda|-1},s_{|\Lambda|}}. \tag{12}$$

To je načeloma to kar je bilo za narediti pri tej nalogi. Ko sem enkrat napisal ta razcep sem preeril, če je pravilen, tako da sem pravilno med seboj zmnožil matrike A, tako da sem dobil ven začetno stanje. Vso to manevriranje in preoblikovanje matrik niti ni tako zelo zahtevno, saj vedno samo razrežemo matrike, in ne rabimo biti preveč pozorni. Ko pa na koncu matrike zmnožimo skupaj da dobimo začetno stanje, moramo biti bolj previdni. Tudi tu sem si jaz pomagal enostavno z $d-$narnimi števili.

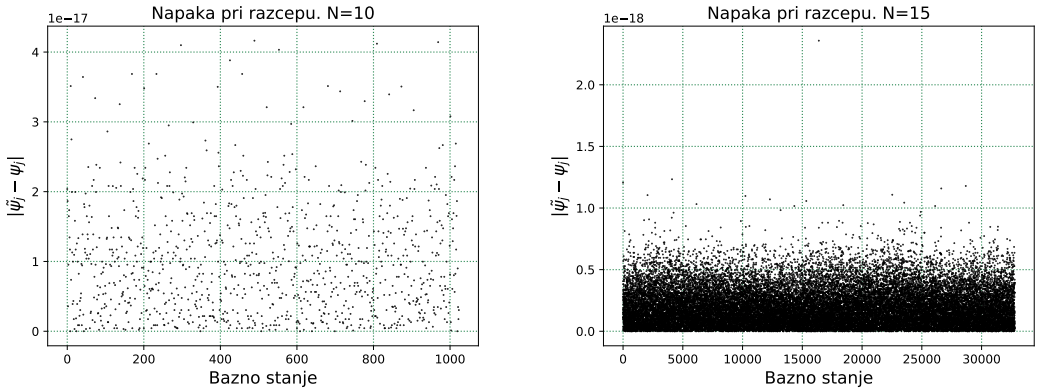
II. Postavitev algoritma

Najprej sem zapisal algoritem, kar se je izkazalo za relativno preprost postopek v pythonu, saj sem lahko uporabil fukcijo *numpy.reshape*, ki je preoblikovanje matrik poenostavila. Postopek sem že kar od začetka zastavil za poljubni spin, čeprav kasneje drugih spinov nisem potreboval. S spinom 1/2 so kasnejši triki res poenostavljeni z binarnimi števili, med tem ko se pri višjih spinih zaplete, saj $d - narni$ načini niso tako dobro zastopani. Najprej sem zase preveril časovno zahtevnost, da sem vedel kako velike sisteme si lahko privošim in izvedel 1.



Slika 1: *Časovna zahtevnost razcepa naključnega začetnega stanja v odvisnosti od števila delcev in spina teh delcev.*

Preveril sme tudi kako natančen je ta algoritem in navdušen sem bil nad izjemno, strojno, natančnostjo. Preizkušal sem da nad normiranem naključnim stanjem. Žal sem pozabil na možnost, da bi zapisal kompleksno začetno stanje, tako da je tole realno.



Slika 2: *Razlika med začetnim stanjem in stanjem, ki ga izračunamo iz naših matrik A. Y os je v logaritemski skali.*

III. Heisenbergov model

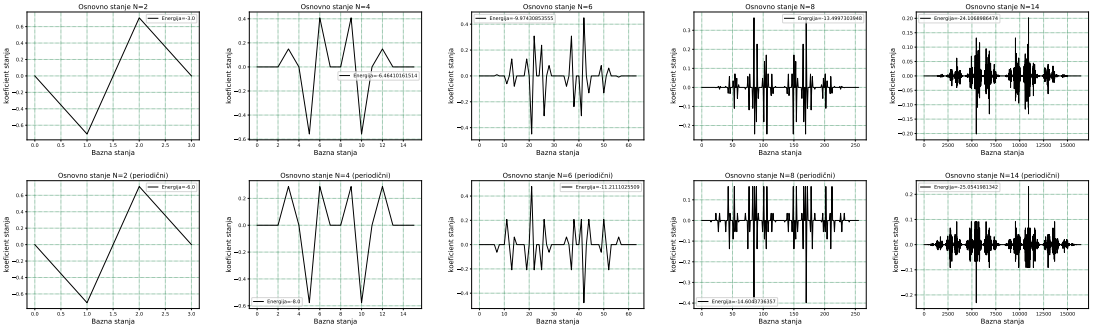
Tako se je začel meni osebno najzahtevneši del naloge. Moral sem zapisati hamiltonjanko

$$H = \sum_i \vec{\sigma}_j \vec{\sigma_{j+1}}, \tag{13}$$

kjer je $\vec{\sigma}_j$ vektor matrik, ki deluje na $j-$tem mestu na stanje tistega delca s paulijevo matriko. V bazi tistega delca smo že pri prejšnji nalogi ugotovili da element izgleda tako:

$$\vec{\sigma}_j \vec{\sigma_{j+1}} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & -1 & 2 & 0 \\ 0 & 2 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}, \tag{14}$$

a ga je potem ptreba tenzorsko pomnožiti z identiteto, na mestih na katerih ne deluje, tako, da je na koncu matrika dimenzije $d^{|\Lambda|}$. Tako sem najrpej poskušal tenzorsko množiti matrike med sabo, nakar sem se v potu svojega obraza spomnil, da sem hamiltonko za heisenbergov model napisal že v prejšnji nalogi. Tako sem samo vzel tisti algoritem in nenadoma je vse delalao brezhibno. V algoritem sem dajal bazne vektorje in rezultatate definiral kot vrstice matrike H, saj je H simetrična matrika. Vseeno je bil ta algoritem izredno počasen. Komaj sem izračunal hamiltonjan za prvih 8 delcev. Zato sem se malo potrudil in ga prepisal v malo hitrejšo obliko, tako da sem ga pohitril za faktor 800. Po tej hudi pohitritvi število delcev več ni bil problem, kvečjemu velikost pomnilnika. Hamiltonjan sem diagonaliziral tako, da sem ga shranil kot redko matriko in na njem uporabil diagonalizacijo iz knjižnice, ki izračuna zgolj osnovno stanje in njegovo energijo. Osnovna stanja so prikazana na slikah 3 za različne robne pogoje in različno število delcev.



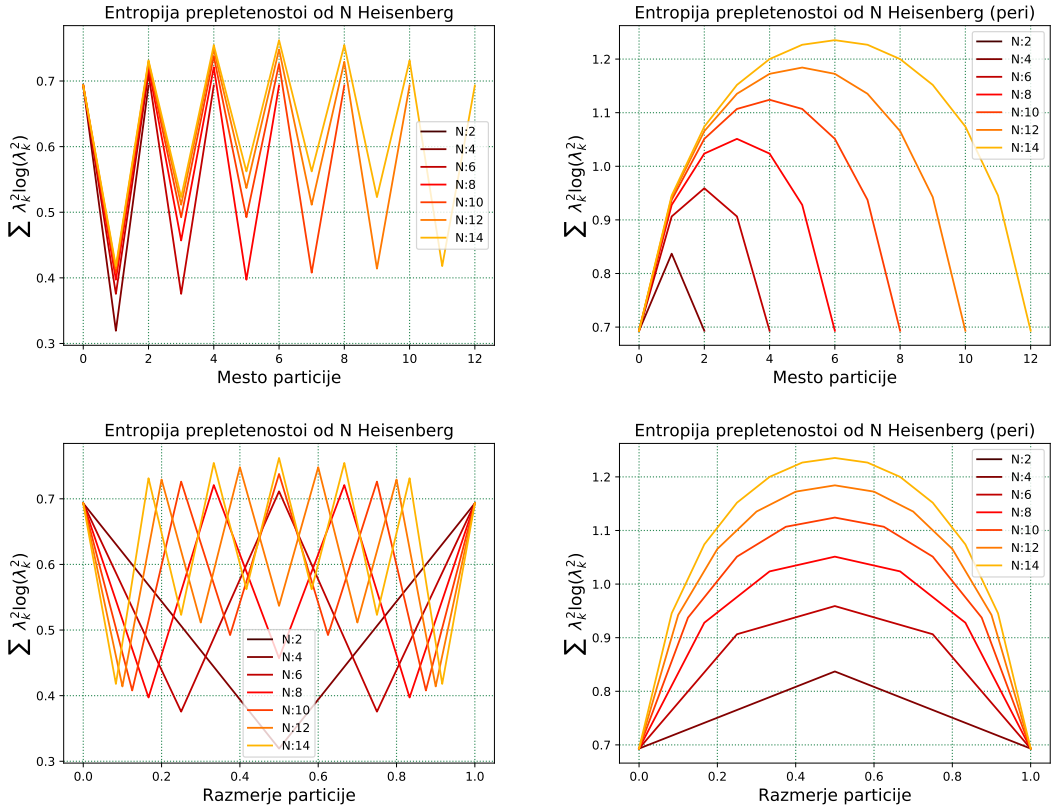
Slika 3: *Osnovna stanja Heisenbergovega modela pri različnem številu delcev (2,4,6,8,14). Zgoraj nimamo periodičnih, spodaj pa imamo periodične robne pogoje. Na slikah je izrisana vrednost valovne funkcije pri nekem stanju.*

Cel namen tega pridelovanja osnovnega stanje je bil v tem, da lahko izračunamo takoimenovano entropijo prepletenosti med dvema blokoma. Če namreč začetno stanje razdelimo na dva bloka s SVD algortimom, nam diagonalna matrika predstavlja lastne vrednosti s katerimi lahko izračunamo entropijo prepletenosti z enačbo

$$S_j = -\sum_{k=1}^{M-j} |\lambda_k^{(j)}|^2 \log |\lambda_k^{(j)}|^2, \tag{15}$$

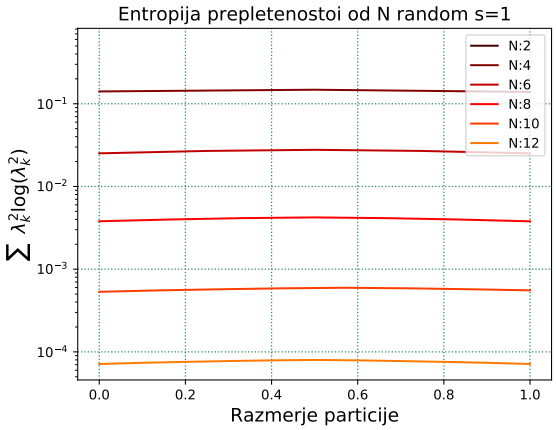
kjer indeks k teče po vseh diagonalnih elementih diagonalne matrike. Torej smo z našim razcepom že izračunali tudi entropijo prepletenosti med bloki (en delec, vsi desno od njega), (dva sosednja delca, vsi delci desno od niju),. . . ,(vsi delci pred zadnjim, zadni delec). Na ta

način lahko zgolj izrišemo naše rezultate. Najpej si bomo pogledali kako izgleda entropija prepletoenosti v odvisnosti od tega koliko delcev je v levem bloku v absolutnem in relativnem smislu na slikah 4.



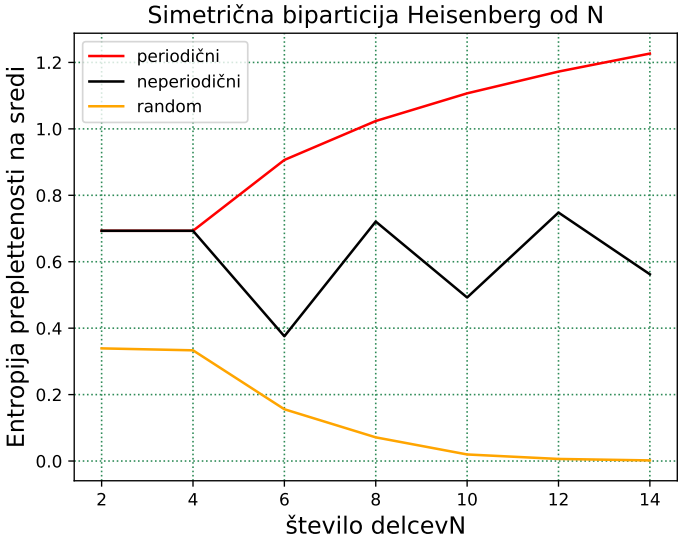
Slika 4: *Entropija prepletenosti v odvisnosti od velikosti levega bloka, če naredimo biparticijo na levi in desni blok. Zgoraj imamo absolutne vrednsoti, med tem ko imamo spodaj relativno razmerej med velikostima blokov.*

Tipično in pričakovano je da je entropija največja kadar sta bloka enako velika, to velja tudi če izvedemo isti postopek an naključnem normiranem začetnem stanju. Zanimivo pa je, da se z velikostjo sistema v tem primeru magnituda močno spremeni kar lahko vidimo na slikah 5.



Slika 5: *Isto kot na slikah 3, le za naključno normirano stanje.*

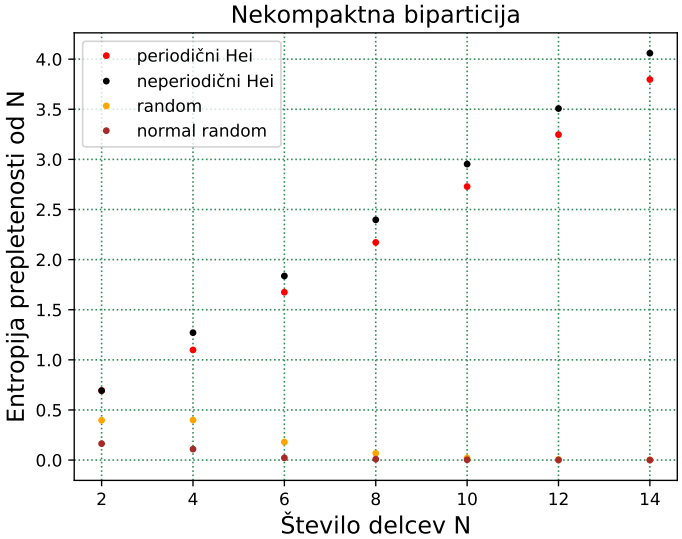
Iz tega se lepo vidi, da je zelo naravna izbira to, da pogledamo kako se entropija prepletenosti spreminja pri simetrični biparticiji v odvisnosti od velikosti sistema. Zato si pogledamo sliko 6. Na njej je pravzaprav to kar smo že videli indirekto na prejšnjih grafih. Opozoril bi, da je krivulja za naključno stanje precej spremenljiva in ni vedno enaka. Se pa vsakič zgodi, da zelo hitro, tako kot v našem primeru, limitira k 0, ko povečujemo N, ne gelde na to kje se krivulja začne.



Slika 6: *Entropija prepletenosti pri simetrini biparticiji v odvisnosti od števlia delcev N.*

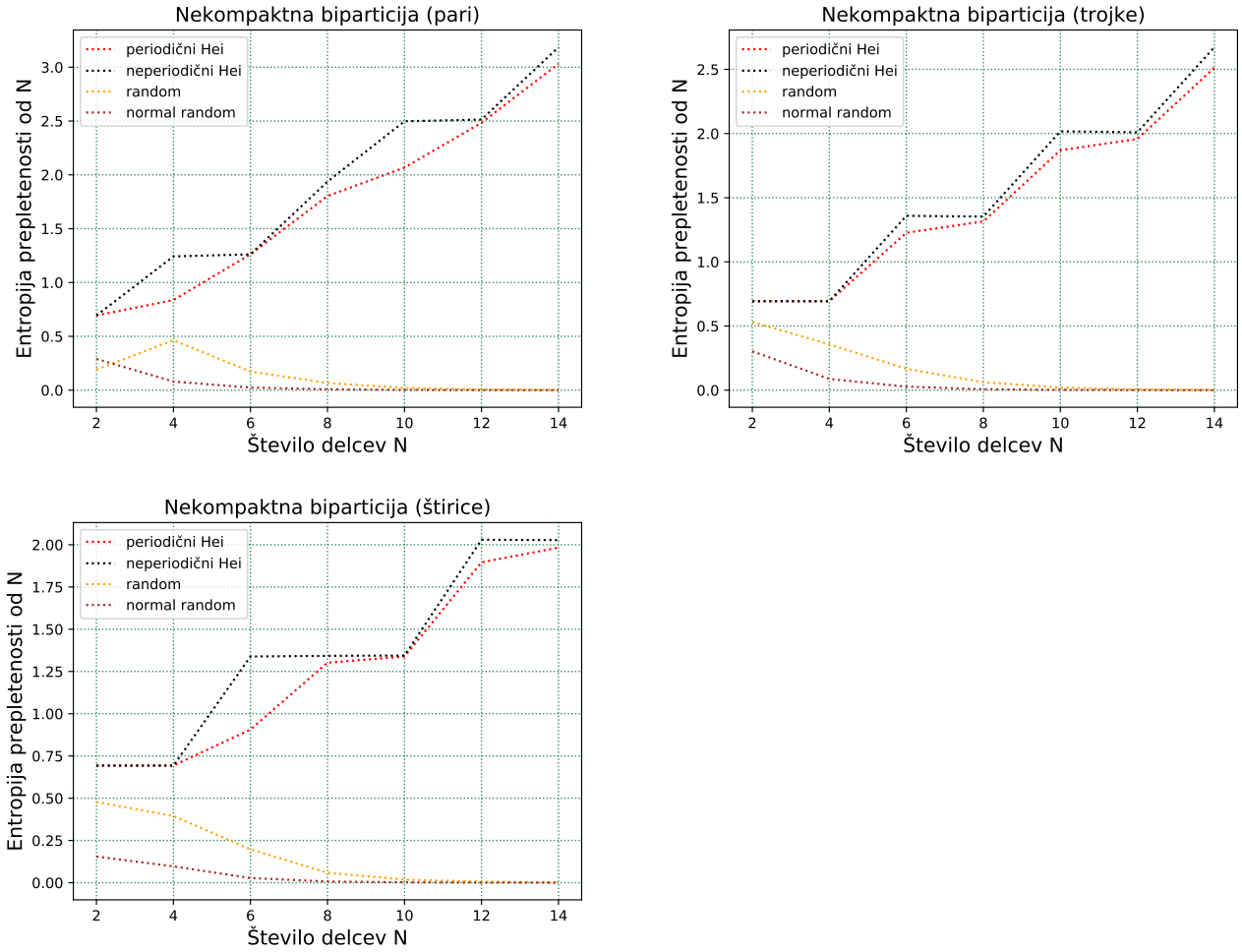
I. Nekompaktna bi-particija

Kar se je zanimivo vprašati je kako izgleda takšna entropija prepletenosti, če biparticija ni med bloki, ki so med seboj povezni. Torej med bloki, ki ga sestavljajo delci, ki so med seboj sosedi. Kaj če vzamemo bi-particijo tako, da damo vsak drugi delec v svoj blok in vse ostale v drugi. Tehnično to ni tako zelo zahtevno. Zopet si pomagamo tako, da si začetno stanje predstavljamo kot funkcijo $d-narnih$ števil. Sedaj zgolj funkcijo razbijemo na funkcijo dveh spremenljivk. Prva spremenljivka je vsaka druga števka iz prvega števila zlepljena skupaj v novo število in druga števila ravno obratno. Na tako zapisanem stanju, v obliki matrike, izvedemo SVD razcep in tako dobimo entropijo prepletenosti.



Slika 7: *Entropija prepletenosti pri nekompaktnem razcepu v odvisnosti od velikosti verige. Normal random je primer, ko naključno stanje žrebamo kompleksno, vsako komponento kot gaussovo število z deviacijo* 1 *in mediano* 0*, potem pa normiramo.*

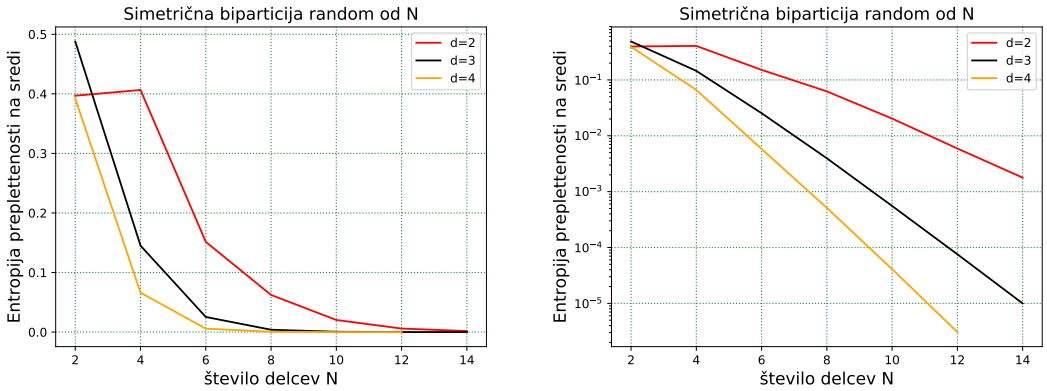
Zanimivo je da pri naključnem stanju tudi tokrat hitro pademo k ničli, kar je spet nekaj kar bi pričakovali, med tem ko pri Heisenbergovemu modelu dobimo linearno naraščanje entropije prepletenosti. To pa je zelo zanimivo. Zdaj smo vzeli dvak drugi spin, kaj pa če pustimo majhne gručice po par spinov, ki jih ločujejo posamezni spini? V tem primeru dobimo odvisnost prikazano na slikah 8. ENtropijo zanimivo poskakuje, kar je pričakovano, glede na to, da skupki niso vedno mogoči in imamo lahko precej nenavadno biparticijo, kot recimo v primeru trojk na $N = 6$ spinih. V tem primeru imamo trojček treh spinov iz prve množice, nato en spin iz druge množice in nato zgolj par spinov iz prve množice. V primeru periodičnih robnih pogojev torej govorimo o kompaktni skupini petih spinov proti enemu spinu, v primeru neperiodičnih pa o skupku (3,2) proti enemu spinu. Stvar je torej delno nesmiselna, ampak zanimiva.



Slika 8: *Entropija prepletenosti pri bi-particiji kjer vzamemo pare, trojke, četrojčke spinov, ki jih med seboj ločujejo posamezni spini.*

IV. Drugi spini

V tem poglavju sem se odločil pogledati eksotično opcijo, verige spinov, ki niso 1/2. ALgoritem sem namreč zapišal za splošen d in sem želel to izkoristiti. Zdelo se mi je preveč naporno zapisati Hamiltonjan za spin 1, zato sem raje preučeval samo naključno stanje. Naivno bi pričakovali, da bomo pri večjem d opazili hitrejšo konvergenco k ničli. To tudi opazimo na slikah 9.



Slika 9: *Entropija prepletenosti pri simetrični biparticiji v odvisnosti od števila delcev s številom prostostnih stopenj d.*