Naslov

Rok Mlinar Vahtar

2024

1 Uvod

V sledečem dokumentu bom raziskoval uporabnost dodajanja merizev spina v večih smereh, v proceduro za iskanje osnovnih stanj spinskih hamiltonianov, ki mi jo je predal Gregor Humar.

2 Procedura

Procedura za iskanje osnovnega stanja hamiltoniana, bazira na večkratnem sklapljanju sistema, ki ga raziskujemo, z kopeljo, ki je v nizkoentropijskem stanju. Raziskovani hamiltonian in hamiltonian kopeli sta lahko poljubna, pomembno je, da je vsak spin znotraj kopeli, povezan z enim spinom v raziskomanem sistemu z hamiltonianom oblike

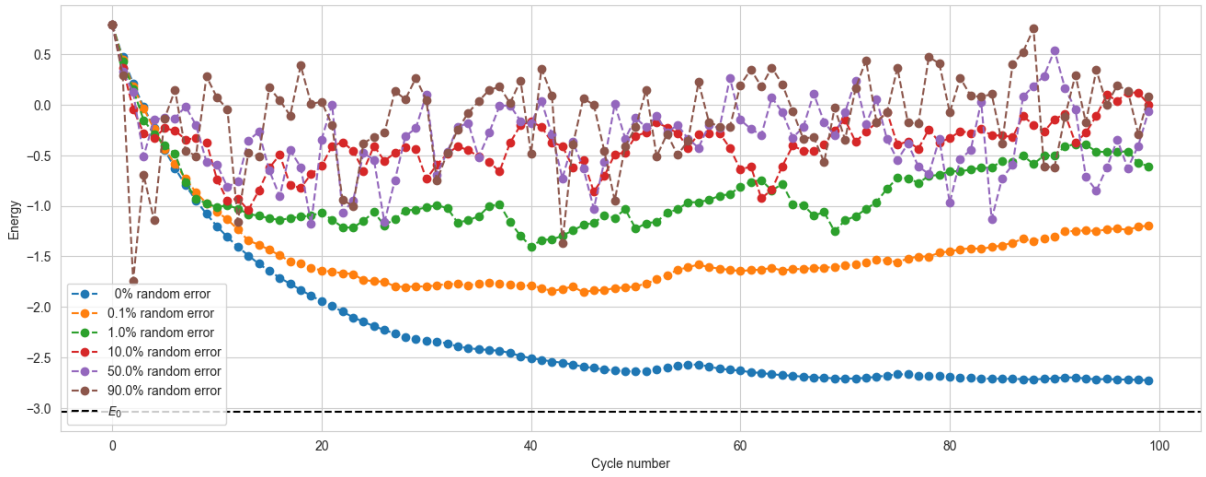
$$J_c(t) \cdot s^z_{sistem} s^z_{kopel}.$$

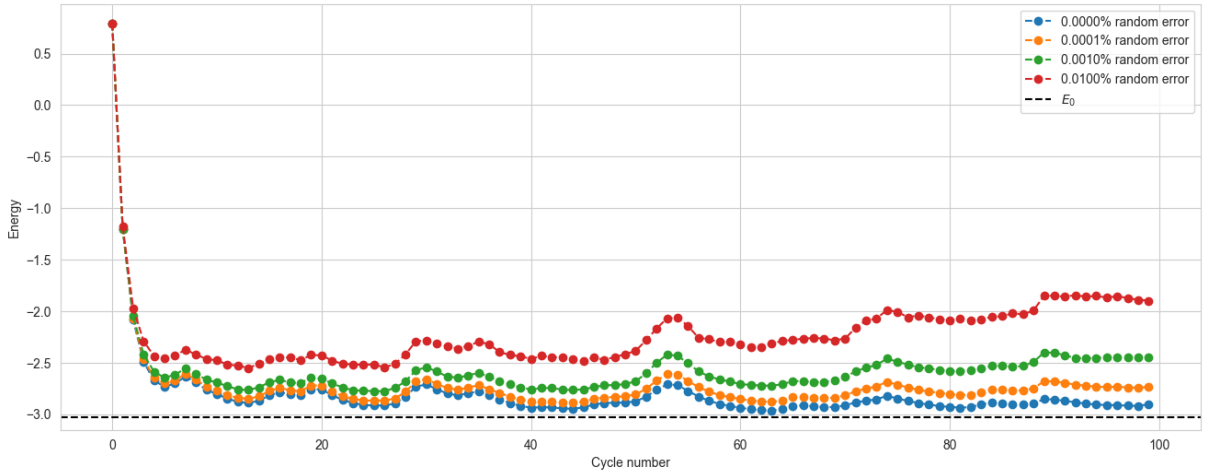
Sistem, ki je tako povezan z kopeljo tako pustimo nekaj časa, nato pa kopel odcepimo in zamenjamo z novo, pri tem pa postopoma zmanjšujemo moč sklopitve J_c.

3 Kako hudo napako?

V idealnem primeru bi pri menjavi kopeli obdržali vso informacijo celotne gostotne matrike sistema, a to ni praktično izvedljivo, saj ta kolapsira v eno izmed lastnih stanj pri meritvi. Obstaja pa veliko načinov, na katere lahko vseeno dobimo približek stanja, ki ga prenašamo v naslednji cikel. Ker vemo, da v idealnem primeru ta procedura doseže osnovno stanje, se porodi vprašanje, kako hudo lahko pokvarimo stanje, preden procedura neha delovati. To sem preveril tako, da sem med cikli stanjem prištel nek delež naključnega stanje

$$|\psi_{\text{Novo}}\rangle = \sqrt{1-\alpha}|\psi_{\text{Staro}}\rangle + \sqrt{\alpha}|\psi_{\text{Naključno}}\rangle.$$





4 Rekonstrukcija stanja

Zdaj ko vemo, da je napaka, ki si jo lahko privoščimo relativno majhna, se lahko osredotočimo na to, kako bomo prenašali stanje iz cikla v cikel tako, da ta nebi postala prevelika.

Najbolj trivialnen način rekonstrukcije stanja je ta, da ga kar pomerimo, recimo v z smeri. To pomeni, da bo na začetku vsakega cikla stanje spet obrnjeno le v z smer, kar zagotovo ni dobro, če iskano osnovno stanje deloma leži v xy ravnini. To pa lahko poskusimo popraviti tako, da v rekonstruirano stanje dodamo informacijo še o x in y smeri. V nadaljevanju bom uporabil dve različni metodi. Prva je ta, da sestavimo produktno stanje, katerega pričakovani vrednosti s_x in s_z se točno ujemata z začetnim stanjem, druga pa, da sestavimo produktno stanje, katerega pričakovane vrednosti s_x, s_y in s_z se približno ujemajo začetnemu stanju po metodi najmanjših kvadratov.

4.1 Rekonstrukcija stanja iz $\langle S_i^z \rangle$ in $\langle S_i^x \rangle$

Najenostavnejša rekonstrukcija, ki si jo lahko zamislimo je to, da ustvarimo produktno stanje posameznih spinov, katerega pričakovane vrednosti se ujemajo z originalnim stanjem. Ker ima eno spin-$\frac{1}{2}$ stanje le dve prostostni stopnji, lahko z posameznim spinom zadostimo le dvema pričakovanima vrednostima. V tem primeru izberemo $\langle S_i^z \rangle$ in $\langle S_i^x \rangle$. Splošno stanje enega spina zapišemo kot

$$|\psi\rangle = \cos(\theta/2)|\uparrow\rangle + e^{i\varphi}\sin(\theta/2)|\downarrow\rangle.$$

Za takšno stanje velja

$$\langle S_z \rangle = \cos(\theta)$$
$$\langle S_x \rangle = \sin(\theta)\cos(\varphi)$$
$$\langle S_y \rangle = \sin(\theta)\sin(\varphi).$$

Kot vidimo, lahko nastavimo vrednosti θ in φ tako, da se $\langle S_i^z \rangle$ in $\langle S_i^x \rangle$ ujemata z izmerjenimi, zmanjka pa nam še en prosti parameter, s katerim bi nastavili še $\langle S_i^y \rangle$.

Produktno stanje zdaj sestavimo tako, da vsakemu posameznemu spinu določimo $\langle S_i^z \rangle$ in $\langle S_i^x \rangle$, nato pa enospinska stanja tenzorsko pomnožimo.

4.2 Rekonstrukcija stanja iz $\langle S_i^z \rangle$, $\langle S_i^x \rangle$ in $\langle S_i^y \rangle$

Kot smo videli, tega, kar obljublja naslov, ne moremo storiti na isti način kot prej, saj imamo predoločen sistem enačb, ki v splošnem nima rešitve. Ko se soočamo z predoločenimi sistemi, pa pogosto pridemo do metode najmanjših kvadratov. V tem primeru to storimo tako, da najdemo izbiro θ in φ, ki minimizira kvadratno formo

$$2J = \left(\langle S_i^z \rangle - \cos(\theta)\right)^2 + \left(\langle S_i^x \rangle - \sin(\theta)\cos(\varphi)\right)^2 + \left(\langle S_i^y \rangle - \sin(\theta)\sin(\varphi)\right)^2.$$

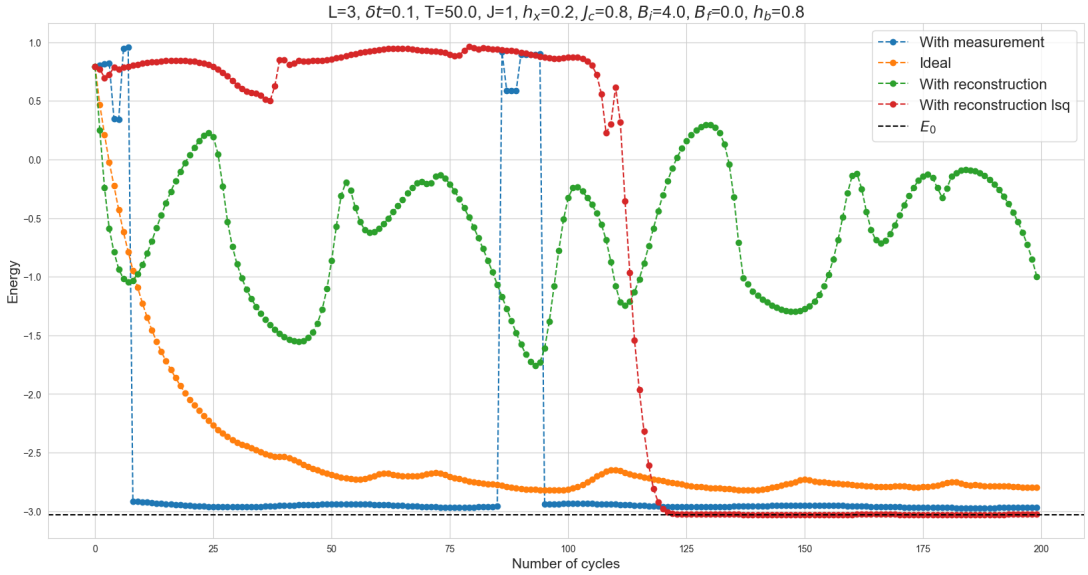
Ko to izbiro enkrat imamo, nadaljujemo kot prej.

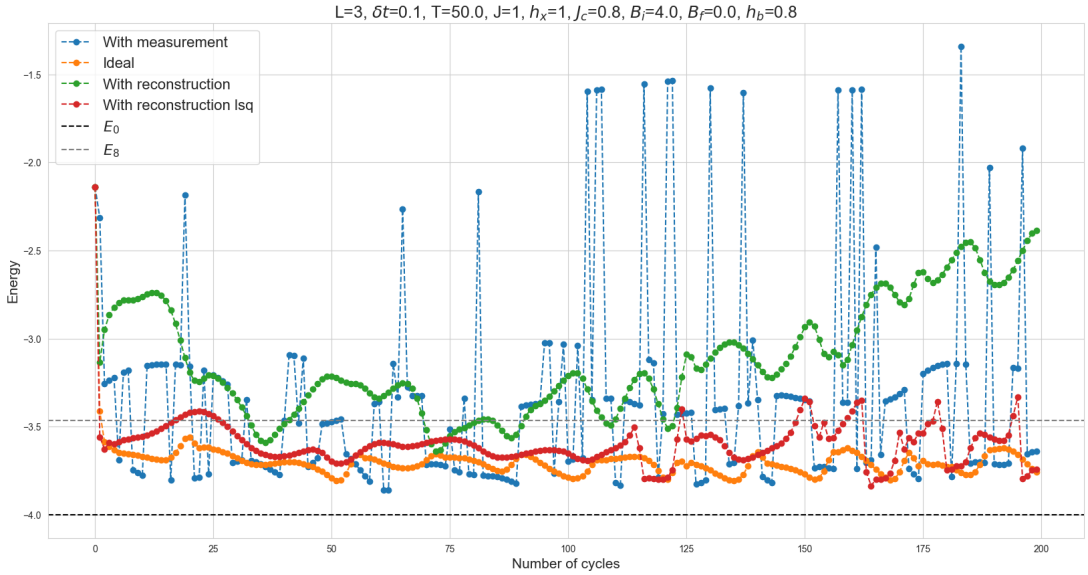
5 Enostaven hamiltonian

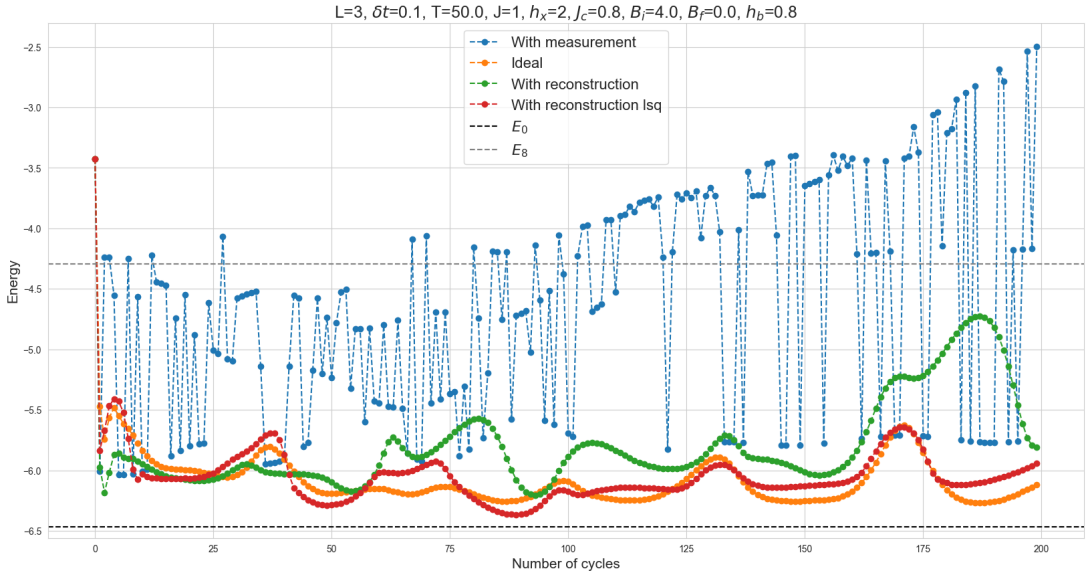
Kot prvo bomo iskali osnovno stanje enodimenzionalnega transferse field ising modela z dožino 3 spinov:

$$H = -J \sum_{i=0}^{L-1} s_i^z s_{i+1}^z + h_x \sum_{i=0}^{L-1} s_i^x$$

Če primerjamo različne metode, ki smo jih do zdaj sestavili, dobimo naslednje slike







Kot bi pričakovali, metod, ki naprej nesejo več informacije delujejo bolje za širši spekter vrednosti J in h_x. Včasih, pa vidimo, da delujejo morda celo predobro, na primer metoda z meritvjo v z smeri, ki v trenutku najde osnovno stanje, ko je h_x majhen, saj je to usmerjeno skoraj samo v z smer. Da iz ocene uporabnosti teh metod izločimo takšne primere, je dobra ideja, da pogledamo kompleksnejši hamiltonian.

5.1 Animacije

Poleg grafov energije, se izplača ogledati tudi projekcije na lastna stanja. Te sem predstavil v različnih animacijah, ki se nahajajo na Githubu.

V IJS2024/Animation Generation se v podmapah, ki označata izbiro parametrov, nahajajo animirani grafi, ki kažejo spremembo energije in projekcij na lastna stanja celotnega sistema in TFIM verige z različnimi metodami za podajanje stanj iz cikla v cikel.

Druge animacije, ki kažejo iste projekcije, ampak narisane na energijskem spektru, pa se nahajajo v IJS2024/Version 3.0/V3 Animation

6 Frustriran hamiltonian

Tu sem poskusil izvesti proceduro na frustriranem hamiltonianu z $L = 5$, kar se je izkazalo za nepraktično, saj čas računanja skoči na več kot 11 ur, saj matrike, s katerimi računamo, verjetno niso več redke in n-diagonalne in moramo uporabiti za delo z njimi splošne algoritme.

7 Kratka predstavitev programske kode

V github repositoriju https://github.com/konj5/IJS2024/tree/main se nahaja več verzij. Verzija 1 in 2 sta še v čisto začetni obliki in vsebujeta le ločene Jupyter datoteke za generiranje grafov in animacij. Verzija 3 je že v končni obliki, torej je optimizirana in ne shranjuje vseh stanj. Deluje tako, da opazljivke izmerimo tako, da programu podamo funkcijo. Verzija 4 je hitrejša kot verzija 3, ampak je zato omejene na določene opazljivke, ki jih lahko podamo z pričakovani vrednostjo operatorja. Verzija 5 pa združuje obe funkcionalnosti.

V nadaljevanju se vse nanaša na verzijo 5. Glavna dva objekta, s katerima ima zunanji uporabnik kontakt sta measurables.py, ki je kolekcija uporabnih opazljivk in objekt Procedure v procedure.py. Primer uporabe lahko najdemo kar na dnu procedure.py. V splošnem najprej ustvarimo objekt Procedure

```
proc=Procedure()
```

in nato nastavimo relevantne parametre. Vsi imajo že privzeto vrednost, tako ta nastavimo samo tiste, za katere želimo, da so drugačni.

```
proc.setParameters(L=3)
```

Na koncu moramo le še izvesti proceduro z klicem metode proc.runProcedure. Tej moramo nujno podati število ciklov, ki jih želimo izvesti (N_Cycles), hamiltnian, katerega osnovno stanje iščemo (chain_hamiltonian) in opazljivke, ki jih želimo izmeriti (measure). Opazljivke so lahko podane kot funkcija, ki vzame kot argument stanje in objekt Procedure, ter vrne list izmerjenih opazljivk, ali pa kot seznam operatorev, katerih pričakovane vrednosti izmerimo. Izven tega lahko nastavimo tudi to, kako se parametri sklopitve spreminjajo z časom, začetno srtanje in kako se stanja podajajo iz cikla v cikel.

Pomembno je tudi to, da določimo, ali želimo delati z vektorji stanj ali gostotnimi matrikami, ter izbrati vse ostale metode primerno v skladu z to odločitvjo.

Na koncu dobimo meritve v obliki 3D numpy arraya. Njegov prvi indeks določi opazljivko, drugi zaporedno številko cikla, tretji pa trenutek v času znotraj cikla.

8 Zaključek

Vseskozi sem potrdil, da dodajanje informacij o projekciji spina na druge osi res pomaga do neke mere, vendar pa ostaja veliko vprašanj glede uporabnosti na zahtevnejših sistemih še odprtih.