Programowanie i metody numeryczne

Projekt ***Ścisła diagonalizacja***

Krystian Jabłonowski
Krystian.Jablonowski@fuw.edu.pl

Ścisła diagonalizacja, często oznaczana skrótem ED od angielskiego *exact diagonalization*, jest jedną z podstawowych metod numerycznego badania skończonych układów kwantowych. Idea metody jest prosta: wybieramy skończoną bazę przestrzeni Hilberta, zapisujemy Hamiltonian jako macierz, a następnie obliczamy jego wartości i wektory własne. W ten sposób otrzymujemy bezpośredni dostęp do energii, stanów własnych oraz wartości oczekiwanych obserwabli.

Projekt ma dwa cele. Pierwszym jest napisanie programu, który buduje i diagonalizuje Hamiltoniany prostych modeli kwantowych na sieci. Drugim jest wykorzystanie otrzymanych stanów własnych do obliczenia wybranych wielkości fizycznych, takich jak korelacje spinowe, podwójne obsadzenie lub funkcja spektralna.

Najważniejsze ograniczenie tej metody wynika z faktu, że wymiar przestrzeni Hilberta rośnie wykładniczo z rozmiarem układu. Dlatego w projekcie należy zwrócić szczególną uwagę na wykorzystanie symetrii oraz na różnicę między pełną diagonalizacją macierzy gęstej a obliczaniem kilku wartości własnych macierzy rzadkiej metodą iteracyjną.

1. Zakres projektu i oczekiwane wyniki

Program powinien być napisany w języku Python. Zalecane biblioteki to `numpy`, `scipy`, `matplotlib` oraz, w przypadku macierzy rzadkich, `scipy.sparse`. Można użyć innego języka programowania, ale program musi realizować te same kroki numeryczne.

Minimalny zakres projektu obejmuje:

1. konstrukcję bazy dla łańcucha spinowego 1/2,
2. budowę Hamiltonianu modelu Heisenberga,
3. wykorzystanie sektora o zadanej wartości S^z_{tot},
4. konstrukcję bazy dla modelu Hubbarda przy zadanych $N_\uparrow$ i $N_\downarrow$,
5. budowę Hamiltonianu modelu Hubbarda z poprawnym znakiem fermionowym,
6. diagonalizację Hamiltonianu i obliczenie kilku prostych obserwabli,
7. przygotowanie wykresów pokazujących zależność czasu działania oraz wymiaru przestrzeni Hilberta od rozmiaru układu.

Część rozszerzona obejmuje obliczenie funkcji spektralnej w reprezentacji Lehmanna. Jest to najbardziej wymagający element projektu, dlatego należy wykonać go dla małych układów, dla których możliwa jest pełna diagonalizacja kilku sektorów liczby cząstek.

Program powinien wypisywać najważniejsze liczby na ekran, a wykresy oraz dane liczbowe powinny być zapisywane do plików, na przykład w formacie `.txt`, `.csv`, `.png` lub `.pdf`. Przykładowe pliki wyjściowe to `energies.txt`, `timing.csv`, `correlations.pdf` oraz `spectral_function.pdf`.

Przykładowy interfejs programu może wyglądać następująco:

```
model = HeisenbergChain(L=10, J=1.0, N_up=5, bc="open")
H = model.build_hamiltonian(sparse=True)
energies, states = model.diagonalize(k=6)
model.save_energies("energies_heisenberg_L10.txt")

hub = HubbardChain(L=6, t=1.0, U=4.0, N_up=3, N_down=3, bc="open")
H = hub.build_hamiltonian(sparse=True)
energies, states = hub.diagonalize(k=10)
hub.save_observables("observables_hubbard_L6_U4.txt")
```

Nie trzeba używać dokładnie takiej struktury klas, ale program powinien być podzielony na czytelne funkcje odpowiedzialne za konstrukcję bazy, budowę Hamiltonianu, diagonalizację i analizę wyników.

2. Model Heisenberga

Pierwszy model, który należy rozwiązać, to jednowymiarowy model Heisenberga dla spinów 1/2. Opisuje on lokalne momenty magnetyczne rozmieszczone na węzłach sieci. W tym projekcie przyjmujemy konwencję

$$\hat{H} = J \sum_{\langle i,j \rangle} \hat{\vec{S}}_i \cdot \hat{\vec{S}}_j, \tag{1}$$

gdzie $J > 0$ odpowiada oddziaływaniu antyferromagnetycznemu, a $J < 0$ oddziaływaniu ferromagnetycznemu. Sumowanie po $\langle i, j \rangle$ oznacza sumowanie po parach najbliższych sąsiadów. Dla otwartych warunków brzegowych są to pary $(0, 1), (1, 2), \ldots, (L-2, L-1)$.

Korzystając z operatorów drabinkowych, Hamiltonian można zapisać jako

$$\hat{H} = J \sum_{\langle i,j \rangle} \left[\frac{1}{2} \left(\hat{S}_i^+ \hat{S}_j^- + \hat{S}_i^- \hat{S}_j^+ \right) + \hat{S}_i^z \hat{S}_j^z \right]. \tag{2}$$

Na każdym węźle mamy dwa możliwe stany: $|\uparrow\rangle$ oraz $|\downarrow\rangle$. Wymiar pełnej przestrzeni Hilberta wynosi

$$\dim \mathcal{H} = 2^L. \tag{3}$$

W implementacji najwygodniej reprezentować stan spinowy jako liczbę całkowitą albo jako napis bitowy. Można przyjąć konwencję, że bit równy 1 oznacza spin skierowany w górę, a bit równy 0 spin skierowany w dół. Na przykład dla $L = 4$ stan bitowy `1010` oznacza konfigurację

$$|\uparrow\downarrow\uparrow\downarrow\rangle. \tag{4}$$

Działanie lokalnego członu Hamiltonianu na parę sąsiednich spinów jest następujące:

Konfiguracja pary	Wkład do Hamiltonianu
$\|\uparrow\uparrow\rangle$	diagonalny wkład $J/4$
$\|\downarrow\downarrow\rangle$	diagonalny wkład $J/4$
$\|\uparrow\downarrow\rangle$	diagonalny wkład $-J/4$ oraz przejście do $\|\downarrow\uparrow\rangle$ z amplitudą $J/2$
$\|\downarrow\uparrow\rangle$	diagonalny wkład $-J/4$ oraz przejście do $\|\uparrow\downarrow\rangle$ z amplitudą $J/2$

Ta tabela powinna być bezpośrednio wykorzystana do budowy macierzy Hamiltonianu. Dla każdego stanu bazowego należy przejść po wszystkich parach sąsiednich węzłów, dodać wkład diagonalny oraz, jeśli spiny są przeciwne, dodać element pozadiagonalny odpowiadający zamianie tych dwóch spinów.

Zadanie 1.1. Napisz funkcję `build_basis_spin(L)`, która tworzy pełną bazę spinową dla łańcucha o długości L. Funkcja powinna zwracać listę stanów oraz słownik `state_to_index`, który dla danego stanu zwraca jego indeks w bazie. Sprawdź działanie funkcji dla $L = 2, 3, 4$ i wypisz na ekran wszystkie stany bazowe.

Zadanie 1.2. Napisz funkcję `build_heisenberg_hamiltonian(L, J, bc)`, która buduje macierz Hamiltonianu (2). Najpierw zaimplementuj otwarte warunki brzegowe `bc="open"`. Okresowe warunki brzegowe `bc="periodic"` można potraktować jako rozszerzenie. Sprawdź, czy otrzymana macierz jest hermitowska. Dla $L = 2$ porównaj wartości własne z wynikiem analitycznym: stan singletowy ma energię $-3J/4$, a trzy stany trypletowe mają energię $J/4$.

Zadanie 1.3. Zdiagonalizuj Hamiltonian dla kilku rozmiarów układu, na przykład $L = 4, 6, 8, 10, 12$. Dla małych układów możesz użyć pełnej diagonalizacji `numpy.linalg.eigh`. Dla większych układów użyj macierzy rzadkich oraz funkcji `scipy.sparse.linalg.eigsh`. Narysuj zależność czasu diagonalizacji od wymiaru przestrzeni Hilberta.

2.1. Sektory symetrii

Hamiltonian Heisenberga zachowuje całkowitą magnetyzację w kierunku z,

$$\hat{S}^z_{\mathrm{tot}} = \sum_i \hat{S}^z_i. \tag{5}$$

Oznacza to, że Hamiltonian nie miesza stanów o różnych wartościach S^z_{tot}. Możemy więc diagonalizować Hamiltonian osobno w sektorach o ustalonej liczbie spinów skierowanych w górę $N_\uparrow$. Wymiar takiego sektora wynosi

$$\dim \mathcal{H}(L, N_\uparrow) = \binom{L}{N_\uparrow}. \tag{6}$$

Dla antyferromagnetycznego łańcucha przy parzystym L szczególnie ważny jest sektor $N_\uparrow = L/2$, czyli $S^z_{\mathrm{tot}} = 0$.

Zadanie 1.4. Rozbuduj kod tak, aby funkcja budująca bazę przyjmowała opcjonalny parametr `N_up`. Jeżeli parametr jest podany, program powinien wygenerować tylko te stany, w których liczba bitów równych 1 wynosi $N_\uparrow$. Następnie zbuduj Hamiltonian w tym sektorze. Porównaj wymiar pełnej przestrzeni Hilberta 2^L z wymiarem sektora $\binom{L}{N_\uparrow}$ dla kilku wartości L.

Zadanie 1.5. Dla stanu podstawowego w sektorze $S^z_{\text{tot}} = 0$ oblicz korelacje spinowe

$$C(r) = \frac{1}{L-r} \sum_{i=0}^{L-r-1} \left\langle \hat{S}^z_i \hat{S}^z_{i+r} \right\rangle . \tag{7}$$

Narysuj $C(r)$ jako funkcję odległości r dla kilku rozmiarów układu. Program powinien zapisać dane do pliku oraz utworzyć rysunek.

3. Model Hubbarda

Drugim modelem jest jednowymiarowy model Hubbarda. Jest to podstawowy model oddziałujących fermionów na sieci. Hamiltonian ma postać

$$\hat{H} = -t \sum_{\langle i,j\rangle,\sigma} \left(\hat{c}^\dagger_{i\sigma} \hat{c}_{j\sigma} + \hat{c}^\dagger_{j\sigma} \hat{c}_{i\sigma} \right) + U \sum_i \hat{n}_{i\uparrow} \hat{n}_{i\downarrow}. \tag{8}$$

Operator $\hat{c}^\dagger_{i\sigma}$ tworzy fermion o spinie σ na węźle i, a operator $\hat{c}_{i\sigma}$ usuwa taki fermion. Operator liczby cząstek ma postać

$$\hat{n}_{i\sigma} = \hat{c}^\dagger_{i\sigma} \hat{c}_{i\sigma}. \tag{9}$$

Parametr t określa amplitudę przeskoku między sąsiednimi węzłami, natomiast U opisuje oddziaływanie dwóch fermionów o przeciwnych spinach znajdujących się na tym samym węźle.

W projekcie należy pracować w sektorze o ustalonej liczbie cząstek $N_\uparrow$ i $N_\downarrow$. Wymiar przestrzeni Hilberta wynosi wtedy

$$\dim \mathcal{H}(L, N_\uparrow, N_\downarrow) = \binom{L}{N_\uparrow} \binom{L}{N_\downarrow}. \tag{10}$$

Bazę można zapisać jako pary bitstringów (u, d), gdzie u opisuje położenia fermionów ze spinem w górę, a d położenia fermionów ze spinem w dół. Na przykład dla $L = 4$ stan

$$(u, d) = (\texttt{1010}, \texttt{1001}) \tag{11}$$

oznacza, że fermiony $\uparrow$ znajdują się na węzłach 0 i 2, natomiast fermiony $\downarrow$ na węzłach 0 i 3. Węzeł 0 jest obsadzony podwójnie, więc człon oddziaływania dodaje do energii wkład U.

3.1. Znak fermionowy

Najważniejsza różnica między modelem spinowym a modelem Hubbarda polega na tym, że operatory fermionowe antykomutują. Dlatego podczas przeskoku cząstki trzeba poprawnie uwzględnić znak fermionowy. Najbezpieczniejsza metoda polega na przyjęciu jednego globalnego uporządkowania orbitali, na przykład

$$(0\uparrow, 0\downarrow, 1\uparrow, 1\downarrow, \ldots, (L-1)\uparrow, (L-1)\downarrow). \tag{12}$$

Następnie każdy stan można traktować jako jeden bitstring długości $2L$. Działanie operatora $\hat{c}^\dagger_p \hat{c}_q$ można zaimplementować w dwóch krokach:

1. zastosuj operator anihilacji $\hat{c}_q$; jeżeli orbital q nie jest obsadzony, wynik jest zerowy,

2. zastosuj operator kreacji $\hat{c}_p^\dagger$; jeżeli orbital p jest już obsadzony, wynik jest zerowy.

W każdym kroku znak wynosi $(-1)^m$, gdzie m jest liczbą obsadzonych orbitali znajdujących się przed danym orbitalem w przyjętym uporządkowaniu. Taka procedura automatycznie uwzględnia relacje antykomutacyjne.

Jeżeli zamiast jednego bitstringu długości $2L$ używasz dwóch osobnych bitstringów dla spinów $\uparrow$ i $\downarrow$, nadal musisz konsekwentnie odtworzyć znak odpowiadający wybranemu globalnemu uporządkowaniu orbitali.

Zadanie 2.1. Napisz funkcję `build_basis_hubbard(L, N_up, N_down)`, która generuje bazę modelu Hubbarda. Funkcja powinna sprawdzać poprawność parametrów, to znaczy warunki $0 \le N_\uparrow \le L$ oraz $0 \le N_\downarrow \le L$. W przypadku błędnych parametrów program powinien wypisać czytelny komunikat błędu.

Zadanie 2.2. Napisz funkcję `build_hubbard_hamiltonian(L, t, U, N_up, N_down, bc)`, która buduje Hamiltonian (8). Macierz powinna zawierać wkład diagonalny od oddziaływania oraz wkłady pozadiagonalne od przeskoków fermionów. Zadbaj o poprawny znak fermionowy. Sprawdź, czy macierz jest hermitowska.

Zadanie 2.3. Przetestuj kod w granicy $U = 0$. W tym przypadku energie wielu cząstek powinny być sumami energii jednocząstkowych. Dla otwartego łańcucha energie jednocząstkowe wynoszą

$$\varepsilon_m = -2t \cos\left(\frac{\pi m}{L+1}\right), \qquad m = 1, 2, \ldots, L. \tag{13}$$

Porównaj kilka najniższych energii otrzymanych numerycznie z wynikiem zbudowanym przez obsadzanie poziomów jednocząstkowych.

Zadanie 2.4. Dla połowy wypełnienia, czyli $N_\uparrow = N_\downarrow = L/2$ dla parzystego L, oblicz energię stanu podstawowego jako funkcję oddziaływania U. Następnie oblicz średnie podwójne obsadzenie

$$D = \frac{1}{L} \sum_i \langle \hat{n}_{i\uparrow} \hat{n}_{i\downarrow} \rangle \,. \tag{14}$$

Narysuj $D(U)$ dla kilku rozmiarów układu. Zinterpretuj wynik: dla dodatniego U podwójne obsadzenie powinno maleć, ponieważ dwa fermiony na tym samym węźle kosztują energię.

Zadanie 2.5. Porównaj czas budowy Hamiltonianu oraz czas diagonalizacji dla kilku rozmiarów L. Wykonaj osobny wykres pokazujący wymiar przestrzeni Hilberta jako funkcję L. W komentarzu wypisywanym przez program podaj, dla jakiego największego układu udało się wykonać pełną diagonalizację, a dla jakiego trzeba było użyć metody iteracyjnej.

4. Wartości oczekiwane obserwabli

Po diagonalizacji Hamiltonianu otrzymujemy wartości własne oraz wektory własne. Wektory własne są numerycznym zapisem stanów kwantowych w wybranej bazie. Jeżeli baza ma postać

$$\{|\alpha\rangle\}_{\alpha=0}^{\mathcal{D}-1}, \tag{15}$$

to dowolny stan można zapisać jako

$$|\psi\rangle = \sum_{\alpha=0}^{\mathcal{D}-1} c_\alpha |\alpha\rangle . \tag{16}$$

W praktyce liczby c_α są współrzędnymi wektora własnego zwróconego przez procedurę diagonalizacji. Na przykład w `numpy.linalg.eigh` kolumna macierzy wektorów własnych odpowiada jednemu stanowi własnemu. Przed obliczaniem obserwabli należy upewnić się, że stan jest znormalizowany, czyli

$$\sum_{\alpha=0}^{\mathcal{D}-1} |c_\alpha|^2 = 1. \tag{17}$$

Dla dowolnego operatora $\hat{O}$ wartość oczekiwana w stanie $|\psi\rangle$ wynosi

$$\langle \hat{O} \rangle_\psi = \langle \psi | \hat{O} | \psi \rangle = \sum_{\alpha,\beta} c_\alpha^* O_{\alpha\beta} c_\beta, \qquad O_{\alpha\beta} = \langle \alpha | \hat{O} | \beta \rangle . \tag{18}$$

Jeżeli macierz operatora jest jawnie zbudowana, to w Pythonie odpowiada temu operacja

```
expectation_value = np.vdot(psi, O @ psi)
```

Funkcja `np.vdot` wykonuje sprzężenie zespolone pierwszego argumentu, dlatego odpowiada dokładnie iloczynowi $\langle\psi| (\hat{O} |\psi\rangle)$.

W tym projekcie wiele najważniejszych obserwabli jest diagonalnych w bazie obsadzeń. Wtedy nie trzeba budować macierzy operatora. Jeżeli

$$\hat{O} |\alpha\rangle = O_\alpha |\alpha\rangle , \tag{19}$$

to wzór (18) upraszcza się do

$$\langle \hat{O} \rangle_\psi = \sum_\alpha |c_\alpha|^2 O_\alpha. \tag{20}$$

To jest najważniejszy wzór praktyczny w tym projekcie: dla każdego stanu bazowego obliczamy wartość obserwabli O_α, mnożymy ją przez prawdopodobieństwo $|c_\alpha|^2$ i sumujemy po całej bazie.

4.1. Przykład: korelacje spinowe w modelu Heisenberga

Dla modelu Heisenberga stan bazowy jest konfiguracją spinów. Dla danej konfiguracji $|\alpha\rangle$ definiujemy

$$s_i^z(\alpha) = \begin{cases} +\frac{1}{2}, & \text{jeżeli spin na węźle } i \text{ jest skierowany w górę,} \\ -\frac{1}{2}, & \text{jeżeli spin na węźle } i \text{ jest skierowany w dół.} \end{cases} \tag{21}$$

Wtedy operator $\hat{S}_i^z \hat{S}_j^z$ jest diagonalny w tej bazie i

$$\langle \alpha | \hat{S}_i^z \hat{S}_j^z | \alpha \rangle = s_i^z(\alpha) s_j^z(\alpha). \tag{22}$$

Dla stanu własnego $|\psi\rangle = \sum_\alpha c_\alpha |\alpha\rangle$ otrzymujemy więc

$$\left\langle \hat{S}_i^z \hat{S}_j^z \right\rangle_\psi = \sum_\alpha |c_\alpha|^2 s_i^z(\alpha) s_j^z(\alpha). \tag{23}$$

Na tej podstawie można policzyć uśrednioną korelację z odległością r:

$$C(r) = \frac{1}{L-r} \sum_{i=0}^{L-r-1} \left\langle \hat{S}_i^z \hat{S}_{i+r}^z \right\rangle . \tag{24}$$

W kodzie odpowiada temu schemat:

```
def spin_z_value(state, i):
    # Konwencja: bit 1 oznacza spin w gore, bit 0 spin w dol.
    return 0.5 if ((state >> i) & 1) else -0.5

def sz_sz_expectation(basis, psi, i, j):
    value = 0.0
    for alpha, state in enumerate(basis):
        prob = abs(psi[alpha])**2
        value += prob * spin_z_value(state, i) * spin_z_value(state, j)
    return value
```

4.2. Przykład: podwójne obsadzenie w modelu Hubbarda

W modelu Hubbarda naturalną obserwablą jest podwójne obsadzenie

$$\hat{D} = \frac{1}{L} \sum_i \hat{n}_{i\uparrow} \hat{n}_{i\downarrow}. \tag{25}$$

Dla stanu bazowego opisanego przez dwa bitstringi (u, d) liczba podwójnie obsadzonych węzłów jest równa liczbie jedynek w bitowym iloczynie `u & d`. Dlatego dla konfiguracji $\alpha = (u, d)$ mamy

$$D_\alpha = \frac{1}{L} \,\mathrm{popcount}(u \,\&\, d), \tag{26}$$

gdzie `popcount` oznacza liczbę bitów równych 1. Wartość oczekiwana w stanie $|\psi\rangle$ wynosi

$$\langle \hat{D} \rangle_\psi = \sum_\alpha |c_\alpha|^2 D_\alpha. \tag{27}$$

Przykładowa implementacja może wyglądać następująco:

```
def double_occupancy_expectation(basis, psi, L):
    value = 0.0
    for alpha, (up, down) in enumerate(basis):
        prob = abs(psi[alpha])**2
        double_occ = (up & down).bit_count() / L
        value += prob * double_occ
    return value
```

Analogicznie liczy się lokalną gęstość $\langle \hat{n}_{i\sigma} \rangle$, korelacje gęstości $\langle \hat{n}_i \hat{n}_j \rangle$ oraz korelacje spinowe w modelu Hubbarda. Wystarczy dla każdej konfiguracji bazowej obliczyć odpowiednią liczbę, a następnie użyć wzoru (20).

Dla modelu Hubbarda lokalna składowa spinowa jest zdefiniowana jako

$$\hat{S}_i^z = \frac{1}{2} \left(\hat{n}_{i\uparrow} - \hat{n}_{i\downarrow} \right). \tag{28}$$

Dla konfiguracji (u, d) otrzymujemy więc

$$s_i^z(u,d) = \frac{1}{2}\left[n_{i\uparrow}(u) - n_{i\downarrow}(d)\right], \tag{29}$$

gdzie $n_{i\sigma}$ jest równe 1, jeżeli odpowiedni bit jest obsadzony, oraz 0 w przeciwnym przypadku.

4.3. Operatory niediagonalne

Niektóre obserwable nie są diagonalne w bazie obsadzeń. Przykładem w modelu Heisenberga jest operator $\hat{S}_i^+ \hat{S}_j^-$, a w modelu Hubbarda operator przeskoku $\hat{c}_{i\sigma}^\dagger \hat{c}_{j\sigma}$. Wtedy należy użyć pełnego wzoru (18). Nie trzeba jednak zawsze budować pełnej macierzy operatora. Wystarczy działać operatorem na każdym stanie bazowym.

Algorytm jest następujący:

1. wybierz stan bazowy $|\beta\rangle$,
2. zastosuj operator $\hat{O}$ do tego stanu,
3. jeżeli wynik jest niezerowy i ma postać $A|\alpha\rangle$, dodaj wkład $c_\alpha^* A c_\beta$,
4. powtórz procedurę dla wszystkich stanów $|\beta\rangle$.

W zapisie symbolicznym oznacza to

$$\langle \hat{O} \rangle_\psi = \sum_\beta \sum_{\alpha:\, \hat{O}|\beta\rangle = A_{\alpha\beta}|\alpha\rangle} c_\alpha^* A_{\alpha\beta} c_\beta. \tag{30}$$

Ta sama idea będzie potrzebna w części rozszerzonej przy obliczaniu elementów macierzowych funkcji spektralnej.

4.4. Kontrola poprawności i zadania do wykonania

Program powinien zawierać zestaw prostych testów pozwalających wykryć najczęstsze błędy implementacyjne. W szczególności należy sprawdzić:

1. hermitowskość macierzy Hamiltonianu,
2. zgodność wymiaru bazy z odpowiednim wzorem kombinatorycznym,
3. normalizację wektorów własnych zwróconych przez procedurę diagonalizacji,
4. zgodność modelu Heisenberga dla $L = 2$ z wynikiem analitycznym,
5. zgodność modelu Hubbarda dla $U = 0$ z sumą energii jednocząstkowych,
6. niezależność podstawowych wyników od sposobu indeksowania stanów bazowych.

Oprócz energii własnych program powinien obliczać również obserwable. Dla modelu Heisenberga naturalną wielkością są korelacje spinowe $\langle \hat{S}_i^z \hat{S}_j^z \rangle$. Dla modelu Hubbarda naturalne są: liczba cząstek, podwójne obsadzenie, korelacje gęstości oraz korelacje spinowe.

Zadanie 4.1. Dodaj do programu funkcje obliczające wartości oczekiwane operatorów diagonalnych w bazie obsadzeń. Wykorzystaj wzór (20) do obliczenia podwójnego obsadzenia w modelu Hubbarda oraz korelacji $\langle \hat{S}_i^z \hat{S}_j^z \rangle$ w obu modelach. Program powinien wypisać na ekran, który stan własny jest analizowany, na przykład stan podstawowy albo pierwszy stan wzbudzony.

Zadanie 4.2. Sprawdź numerycznie, że dla stanu własnego prawdopodobieństwa $|c_\alpha|^2$ sumują się do jedności. Następnie dla jednego małego układu wypisz kilka największych składników $|c_\alpha|^2$ stanu podstawowego wraz z odpowiadającymi im konfiguracjami bazowymi.

Zadanie 4.3. Dla wybranego układu zapisz do pliku: parametry modelu, wymiar bazy, kilka najniższych energii, energię stanu podstawowego, podwójne obsadzenie oraz czas działania programu. Plik powinien być możliwy do późniejszego wczytania i narysowania wykresów bez ponownego uruchamiania diagonalizacji.

5. Funkcja spektralna: część rozszerzona

Najbardziej zaawansowaną częścią projektu jest obliczenie funkcji spektralnej modelu Hubbarda. Funkcja spektralna mówi, z jakim prawdopodobieństwem można dodać albo usunąć cząstkę o zadanej energii. Jest to wielkość bezpośrednio związana z jednocząstkową funkcją Greena.

Dla małych układów najwygodniej zacząć od lokalnej funkcji spektralnej. W temperaturze zerowej można ją zapisać jako

$$\begin{aligned} A_{i\sigma}(\omega) = &\sum_m \left| \left\langle m, N+1 \middle| \hat{c}_{i\sigma}^\dagger \middle| GS, N \right\rangle \right|^2 \frac{\eta/\pi}{\left[\omega - (E_m^{N+1} - E_{GS}^N)\right]^2 + \eta^2} \\ &+ \sum_m \left| \langle m, N-1 | \hat{c}_{i\sigma} | GS, N \rangle \right|^2 \frac{\eta/\pi}{\left[\omega - (E_{GS}^N - E_m^{N-1})\right]^2 + \eta^2}. \end{aligned} \tag{31}$$

Tutaj $|GS, N\rangle$ oznacza stan podstawowy w sektorze z $N = N_\uparrow + N_\downarrow$ cząstkami, natomiast $|m, N+1\rangle$ i $|m, N-1\rangle$ oznaczają stany własne w sektorach z jedną cząstką więcej albo mniej. Parametr $\eta > 0$ jest sztucznym poszerzeniem poziomów energetycznych. Bez tego poszerzenia funkcja spektralna skończonego układu byłaby sumą delt Diraca.

Wariant z rozdzielczością pędową można zdefiniować przez operator

$$\hat{c}_{k\sigma} = \frac{1}{\sqrt{L}} \sum_{j=0}^{L-1} e^{-ikj} \hat{c}_{j\sigma}, \tag{32}$$

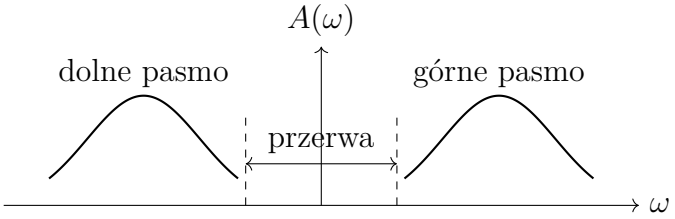
ale jest on najprostszy dla okresowych warunków brzegowych. Dlatego w projekcie wystarczy obliczyć lokalną funkcję spektralną $A_{i\sigma}(\omega)$ albo średnią po węzłach

$$A_\sigma(\omega) = \frac{1}{L} \sum_i A_{i\sigma}(\omega). \tag{33}$$

Zadanie 5.1. Dla małego układu, na przykład $L = 4$ albo $L = 6$, wykonaj pełną diagonalizację Hamiltonianu Hubbarda w sektorze $(N_\uparrow, N_\downarrow)$ oraz w sektorach różniących się o jedną cząstkę. Następnie oblicz lokalną funkcję spektralną zgodnie ze wzorem (31). Narysuj $A(\omega)$ dla kilku wartości U.

Zadanie 5.2. Sprawdź wpływ parametru poszerzenia η na otrzymany wykres. Dla bardzo małego η powinny być widoczne pojedyncze wąskie piki. Dla większego η piki zaczynają tworzyć gładsze struktury. Program powinien zapisać wykresy dla co najmniej trzech wartości η.

Zadanie 5.3. Dla połowy wypełnienia porównaj funkcję spektralną dla małego i dużego U. Jakościowo opisz, czy ciężar spektralny rozdziela się na dwie struktury energetyczne, które można interpretować jako dolne i górne pasmo Hubbardowskie. Nie oczekuj idealnie gładkiego widma: dla małego układu wynik zawsze będzie zawierał skończoną liczbę pików.



Rysunek 1: Schematyczny obraz funkcji spektralnej w układzie z silnym oddziaływaniem. W małym układzie numerycznym zamiast gładkich pasm otrzymujemy skończoną liczbę pików poszerzonych parametrem η.

6. Uwagi praktyczne

Najczęstszy błąd w tym projekcie polega na próbie budowy dużej macierzy gęstej. Dla małych układów jest to dopuszczalne, ponieważ ułatwia testowanie programu. Dla większych układów należy używać macierzy rzadkich. W praktyce wygodne jest zbieranie elementów macierzy w trzech listach `row`, `col`, `data`, a następnie utworzenie macierzy typu `scipy.sparse.coo_matrix` i konwersja do formatu `csr`.

Drugi częsty błąd dotyczy znaków fermionowych w modelu Hubbarda. Jeżeli wyniki dla $U = 0$ nie zgadzają się z sumą poziomów jednocząstkowych, należy najpierw sprawdzić właśnie działanie operatorów kreacji i anihilacji. Bez poprawnego znaku fermionowego widmo modelu Hubbarda będzie błędne.

Trzeci problem dotyczy interpretacji funkcji spektralnej. Dla skończonego układu widmo nie jest ciągłe. Otrzymujemy skończoną liczbę przejść, które dopiero po poszerzeniu Lorentzowskim tworzą funkcję podobną do gładkiego wykresu. Dlatego porównania dla różnych U powinny mieć charakter jakościowy, a nie polegać na dokładnym wyznaczaniu granic pasm.

7. Podsumowanie wymagań

Na koniec projektu program powinien umożliwiać wykonanie następujących czynności:

1. zbudowanie bazy modelu Heisenberga w pełnej przestrzeni i w sektorze $N_\uparrow$,
2. zbudowanie Hamiltonianu Heisenberga i obliczenie jego najniższych energii,
3. obliczenie korelacji spinowych w stanie podstawowym,

4. zbudowanie bazy modelu Hubbarda dla zadanych $N_\uparrow$ i $N_\downarrow$,

5. zbudowanie Hamiltonianu Hubbarda z poprawnym znakiem fermionowym,

6. obliczenie energii stanu podstawowego i podwójnego obsadzenia jako funkcji U,

7. porównanie czasu działania programu dla różnych rozmiarów układu,

8. w części rozszerzonej: obliczenie lokalnej funkcji spektralnej dla małych układów.

Projekt będzie dobrze wykonany, jeżeli kod będzie czytelny, podzielony na funkcje, wyposażony w podstawowe testy poprawności oraz będzie zapisywał dane potrzebne do odtworzenia wykresów. Najważniejszy nie jest sam rozmiar układu, który uda się policzyć, lecz poprawność konstrukcji bazy, Hamiltonianu i obserwabli.