P O L I T E C H N I K A R Z E S Z O W S K A
im. Ignacego Łukasiewicza

WYDZIAŁ MATEMATYKI i FIZYKI STOSOWANEJ

Małgorzata Radomska,
Adrianna Rapa

Nr Albumu 173203, 173204

Projekt w języku Python – model Pottsa

kierunek studiów: Inżynieria i Analiza Danych

Rzeszów 2022

Spis treści

1 Wstęp

Projekt realizowany w ramach przedmiotu „Wprowadzenie do programowania w języku Python” na kierunku Inżynieria i Analiza Danych, semestr pierwszy, grupa 5. Algorytm jest zapisany w środowisku PyCharm.

Przedstawić i opisać model Potts’a w języku Python.

2 Opisteoretyczny

Model Potts'a jest wykorzystywany do przedstawienia oddziaływania między cząsteczkami w układzie fizycznym. Każda cząsteczka jest opisywana przez stan jaki przyjmuje, wartościami są zazwyczaj liczby całkowite.

Stan całego układu jest zależny od stanów poszczególnych cząstek, co skutkuje zmianą stanu całego układu gdy zmieniają się stany cząstek. Energię całego systemu określa się za pomocą Hamiltionianu:

$$H = J \sum_{<i,j>} \delta(s_i, s_j) - H_0 \sum \delta(s_i, \alpha)$$

gdzie:

- J - to stała opisująca siłę oddziaływań między sąsiednimi punktami na sieci
- $\sum_{<i,j>}$ - to suma po sąsiednich parach punktów na sieci
- s_i - to stan punktu na i-tym miejscu sieci ($s_i \in \{1,2,3,...,q\}$)
- $\delta(s_i, s_j)$ - to delta Kroneckera, równa 1, jeśli $s_i = s_j$, lub 0 w przeciwnym przypadku
- H_0 - to stała opisująca wpływ zewnętrznego pola magnetycznego na układ
- α - to stan, który jest faworyzowany przez zewnętrzne pole magnetyczne

Hamiltonian opisuje całkowitą energię układu, a jego zmniejszenie oznacza stabilizację układu. W modelu Potts'a, zmiany stanów punktów na sieci są realizowane przez minimalizację wartości Hamiltonianu

Pierwszy składnik Hamiltonianu reprezentuje energię wynikającą z oddziaływań między sąsiednimi punktami o tym samym stanie. Im bardziej podobne stany, tym silniejsze jest oddziaływanie. Drugi składnik opisuje wpływ zewnętrznego pola magnetycznego, które faworyzuje określony stan α.

W modelu Potts'a wyliczane jest także prawdopodobieństwo przejścia układu do nowego stanu według wzoru:

$$P = e^{\frac{-d}{k \cdot T}}$$

gdzie:

- p to prawdopodobieństwo wystąpienia danego stanu energetycznego
- dE to zmiana energii między dwoma stanami
- k to stała Boltzmanna (k = 1.38 × 10^-23 J/K w jednostkach SI)
- T to temperatura systemu

Model Potts'a jest modelem spinowym, w którym spiny mogą przyjmować więcej niż dwa stany.

Algorytm Monte Carlo jest jedną z najpopularniejszych metod numerycznych stosowanych do symulacji modeli fizycznych. Polega on na losowym wyborze próbek z określonego rozkładu, a następnie na ich analizie, w celu uzyskania estymacji rozkładu prawdopodobieństwa.

W przypadku modelu Potts'a, metoda Monte Carlo jest szczególnie przydatna, ponieważ stanowiąc model spinowy, istnieje wiele możliwych stanów spinów, a ich liczba rośnie wraz z liczbą spinów. Metoda Monte Carlo umożliwia symulowanie dużych systemów spinowych i estymowanie ich zachowania, co byłoby trudne lub niemożliwe do osiągnięcia za pomocą innych metod analitycznych.

3 Działanie algorytmu

1) Ustawia się rozmiar siatki, liczbę wartości, jakie mogą przyjąć spiny, temperaturę, stałą oddziaływania i liczbę kroków MC.
2) Losowo ustala się siatkę początkowych stanów spinów w systemie.
3) Wybór węzłów, których stany będą zmieniane.
4) Propozycja nowych stanów węzłów na podstawie sąsiedztw z innymi węzłami.
5) Obliczenie zmiany energii systemu w wyniku zmiany stanów dla wcześniej wybranych węzłów z wykorzystaniem Hamiltonianu.
6) Rozpoczyna się pętla Monte Carlo, która składa się z następujących kroków:
    a. Losowo wybiera się spin na siatce.
    b. Oblicza się zmianę energii dla tego spinu.
    c. Wylicza się prawdopodobieństwo akceptacji zmiany stanu spinu.
    d. Akceptacja lub odrzucenie zmian (akceptacja, jeśli energia po zmianie stanu jest niższa od dotychczasowej wartości lub odrzucenie, o którym decyduje prawdopodobieństwo)
    e. Jeśli zmiana zostanie zaakceptowana, losuje się nowy stan spinu.
    f. Aktualizacja statystyk dotyczących liczby węzłów w poszczególnych stanach.
    g. Powtarzanie całego procesu do momentu uzyskania stanu równowagi termodynamicznej przez system.
7) Po zakończeniu pętli Monte Carlo, wyświetla się wynik, czyli ostateczne stany spinów na siatce.

Ponadto węzły przyjmują określone wartości dyskretne, w siatce dwuwymiarowej ma sąsiadów (góra, dół, prawo, lewo) lub istnieje też możliwość wprowadzenia dodatkowych relacji między węzłami.

Zmiana stanów poszczególnych węzłów jest realizowana iteracyjnie poprzez przeglądanie kolejnych węzłów i propozycję zmiany ich stanu na podstawie prawdopodobieństwa oraz akceptacje lub odrzucenie tej zmiany. Iteracje są powtarzane do momentu osiągnięcia określonej wartości przez system stany równowagi lub określonej wartości parametrów (np. temperatury, energii).

Możliwa jest także zmiana stanu węzłów w zawieszeniu systemu, gdzie sprawdzane są wszystkie stany oraz ich ewentualna zmiana, a następnie wznowienie działania algorytmu w miejscu gdzie nastąpiło zawieszenie. Jednak powyższa metoda może utrudniać zachowanie równowagi w układzie i zalecane jest realizowanie zmian w sposób iteracyjny.

Wynik symulacji można zinterpretować jako rozkład prawdopodobieństwa spinów w danym modelu. Dla każdej temperatury i stałej oddziaływania, wynik symulacji przedstawia z jakim prawdopodobieństwem system znajdzie się w różnych stanach spinów.

4 Opis działania programu krok po kroku

4.1 Opis funkcji siatka(q, N)

Funkcja siatka(q, N) służy do tworzenia siatki losowych stanów spinów o zadanych rozmiarach.

Opis funkcji siatka(q, N):

- q: liczba wartości, które mogą przyjąć spiny (w modelu Potts'a mogą to być dowolne wartości)
- N: rozmiar siatki (liczba wierszy i kolumn)

Kroki funkcji siatka(q, N):

1. Wykorzystuje bibliotekę NumPy i funkcję np.random.randint() do generowania macierzy losowych liczb całkowitych o rozmiarze (N, N).
2. Zakres losowanych liczb wynosi od 0 do q-1, co odpowiada możliwym wartościom spinów w siatce.
3. Wygenerowana macierz jest zwracana jako wynik funkcji.

```
def siatka(q, N):
    spins = np.random.randint(0, q, size=(N, N))
    return spins
```

4.2 Opis funkcji zmiana_energii(spins, i, j, J, N)

Funkcja zmiana_energii(spins, i, j, J, N) służy do obliczania zmiany energii przy zmianie pojedynczego spinu na pozycji (i, j) w siatce.

Opis funkcji zmiana_energii(spins, i, j, J, N):

- spins: siatka stanów spinów
- i: indeks wiersza spinu, którego zmiana energii jest obliczana
- j: indeks kolumny spinu, którego zmiana energii jest obliczana
- J: stała oddziaływania
- N: liczba wierszy i kolumn w siatce

Kroki funkcji zmiana_energii(spins, i, j, J, N):

1. Na podstawie siatki spins i indeksów (i, j) są obliczane indeksy sąsiadujących spinów. Wykorzystuje się modulo (%) w celu zapewnienia periodycznych warunków brzegowych, co oznacza, że spin na krańcu siatki ma sąsiadów na przeciwnym krańcu.
2. Sumowane są wartości spinów sąsiadów spinu (i, j) i wynik jest przypisywany do zmiennej s.
3. Obliczana jest zmiana energii (dE) zgodnie z równaniem: dE = -J * (s * spins[i, j]). W tym równaniu J oznacza stałą oddziaływania, a spins[i, j] to wartość spinu na pozycji (i, j) w siatce.
4. Zmiana energii dE jest zwracana jako wynik funkcji.

```
def zmiana_energii(spins, i, j, J, N):
    s = spins[(i + 1) % N, j] + spins[i - 1, j] + spins[i, (j + 1) % N] + spins[i, j - 1]
    dE = -J * (s * spins[i, j])
    return dE
```

4.3 Opis funkcji prawdopodobieństwo(dE, k, T)

Funkcja prawdopodobieństwo(dE, k, T) służy do obliczania prawdopodobieństwa akceptacji zmiany stanu spinu na podstawie zmiany energii (dE), temperatury (T) i stałej Boltzmanna (k).

Opis funkcji prawdopodobieństwo(dE, k, T):

- dE: zmiana energii spowodowana zmianą stanu spinu
- k: stała Boltzmanna
- T: temperatura

Kroki funkcji prawdopodobieństwo(dE, k, T):

1. Obliczane jest prawdopodobieństwo akceptacji zmiany stanu na podstawie wzoru: p = np.exp(-dE / (k * T)). Wykorzystuje się funkcję eksponencjalną (np.exp) z modułu NumPy, aby obliczyć wykładnik o wartości -dE / (k * T).
2. Obliczone prawdopodobieństwo akceptacji p jest zwracane jako wynik funkcji.

```
def prawdpodobienstwo(dE, k, T):
    p = np.exp(-dE / (k * T))
    return p
```

4.4 Opis funkcji hamiltonian(spins)

Funkcja "hamiltonian" służy do obliczenia wartości Hamiltonianu, czyli energii całej siatki spins.

Opis funkcji hamiltonian(spins):

- spins: siatka spinów przyjmujących q różnych wartości

Kroki funkcji hamiltonian(spins):

1. Wyzerowanie zmiennej energy.
2. Przejście przez wszystkie elementy siatki i obliczenie sumy wartości spinów będących sąsiadami spinu [i,j].
3. Obliczenie energii układu dla spinu [i,j] zgodnie ze wzorem na Hamiltonian energy += -J * s * nb_sum oraz dodanie wyniku do zmiennej energy.
4. Wynikiem funkcji jest wartość zmiennej energy po przejściu pętli przez całą macierz.

```
def hamiltonian(spins):
    energy = 0
    for i in range(N):
        for j in range(N):
            s = spins[i, j]
            nb_sum = spins[(i + 1) % N, j] + spins[i - 1, j] + spins[i, (j + 1) % N] + spins[i, j - 1]
            energy += -J * s * nb_sum
    return energy
```

4.5 Opis funkcji MonteCarlo(nsteps, k, T, spins)

Funkcja MonteCarlo(nsteps, k, T, spins) implementuje algorytm Monte Carlo, który wykonuje symulację procesu termodynamicznego na siatce spinów.

Opis funkcji MonteCarlo(nsteps, k, T, spins):

- nsteps: liczba kroków Monte Carlo, czyli liczba iteracji, które zostaną wykonane w symulacji
- k: stała Boltzmanna
- T: temperatura
- spins: początkowy stan siatki spinów
- q: liczba różnych stanów, jakie mogą przyjąć spiny

Kroki funkcji MonteCarlo(nsteps, k, T, spins):

1. Utworzenie pustych list state_counts (służy do przechowywania informacji o liczbie różnych stanów występujących w siatce dla kolejnych kroków symulacji), energies (służy do przechowywania informacji o wartościach Hamiltonianu dla kolejnych kroków symulacji), state_occurrences (służy do przechowywania informacji o liczbie wystąpień poszczególnych stanóww siatce dla kolejnych kroków symulacji).
2. Pętla for wykonuje się nsteps razy, czyli tyle, ile określono jako liczbę kroków Monte Carlo.
3. W każdej iteracji losowo wybierane są indeksy i i j, które reprezentują losowy spin w siatce.
4. Obliczana jest zmiana energii (dE) przy zmianie wybranego spinu, wywołując funkcję zmiana_energii(spins, i, j).
5. Wyliczane jest prawdopodobieństwo akceptacji zmiany stanu spinu, wywołując funkcję prawdopodobieństwo(dE, k, T).
6. Sprawdzane jest, czy zmiana zostanie zaakceptowana przez porównanie losowej liczby z prawdopodobieństwem akceptacji. Jeśli losowa liczba jest mniejsza od p, to zmiana jest akceptowana.
7. Jeśli zmiana zostanie zaakceptowana, losowany jest nowy stan spinu (new_spin) i aktualizowana jest siatka spins na pozycji (i, j) poprzez przypisanie new_spin.
8. Obliczana jest liczba unikatowych wartości przyjmowanych przez spiny i przypisana do zmiennej state_count, a następnie dodana do listy state_counts.
9. Aktualizowana jest lista state_occurrences dla każdej wartości q
10. Obliczana jest energia układu z wykorzystaniem funkcji hamiltonian(spins) i przypisana do zmiennej energy, a następnie dodana do listy energies.
11. Po wykonaniu wszystkich kroków Monte Carlo, zaktualizowana siatka spins jest zwracana jako wynik funkcji.Ponadto wynikiem funkcji są też listy: state_counts, state_occurrences i energies.

```python
def MonteCarlo(nsteps, k, T, spins, q):
    state_counts = []
    energies = []
    state_occurrences = {i: [] for i in range(q)}  # Liczba wystąpień poszczególnych stanów
    for step in range(nsteps):
        # Losowe wybieranie spinu
        i = random.randint(0, N - 1)
        j = random.randint(0, N - 1)

        # Obliczanie zmiany energii
        dE = zmiana_energii(spins, i, j, J, N)

        # Wyliczenie prawdopodobieństwa akceptacji zmiany stanu
        p = prawdpodobienstwo(dE, k, T)

        # Sprawdzenie, czy zmiana zostanie zaakceptowana
        if random.random() < p:
            # Losowe wybranie nowego stanu spinu
            new_spin = random.randint(0, q - 1)
            # Zaktualizowanie siatki
            spins[i, j] = new_spin

        state_count = len(np.unique(spins))
        state_counts.append(state_count)

        # Aktualizacja liczby wystąpień poszczególnych stanów
        for state in range(q):
            state_occurrences[state].append(np.count_nonzero(spins == state))
        energy = hamiltonian(spins)
        energies.append(energy)
    return spins, state_counts, state_occurrences, energies
```

4.6 Opis funkcji wizualizacja(spins)

Funkcja wizualizacja(spins) służy do wizualizacji siatki spinów przy użyciu biblioteki matplotlib.

Opis funkcji wizualizacja(spins):

- spins: siatka spinów, która ma zostać zwizualizowana.

Kroki funkcji wizualizacja(spins):

1. Wywołanie funkcji imshow(spins, cmap='hot', interpolation='nearest') z biblioteki matplotlib.pyplot. Ta funkcja służy do wyświetlania obrazów i przyjmuje jako argumenty siatkę spins oraz opcje konfiguracyjne: cmap='hot' określa paletę kolorów, która będzie użyta do wizualizacji, a interpolation='nearest' ustala metodę interpolacji pikseli, aby obraz był wyświetlany bez rozmycia.
2. Wywołanie funkcji colorbar() z biblioteki matplotlib.pyplot. Ta funkcja dodaje kolorową skalę (legendę) do wykresu, która informuje o przyporządkowaniu kolorów do wartości na siatce spinów.
3. Wywołanie funkcji title("Siatka stanów") z biblioteki matplotlib.pyplot. Ta funkcja ustawia tytuł wykresu na "Siatka stanów".

4. Wywołanie funkcji show() z biblioteki matplotlib.pyplot. Ta funkcja wyświetla wykres siatki spinów.

```
def wizualizacja(spins):
    plt.imshow(spins, cmap='hot', interpolation='nearest')
    plt.colorbar()
    plt.title("Siatka stanów")
    plt.show()
```

4.7 Opis funkcji wykres_liczebnosci(nsteps, state_counts)

Funkcja wykres_liczebnosci(nsteps, state_counts) służy do przedstawienia wykresu zależności liczby różnych stanów od kroku symulacji przy użyciu biblioteki matplotlib.

Opis funkcji wykres_liczebnosci(nsteps, state_counts):

- nsteps: liczba kroków symulacji
- state_counts: lista przechowująca informacje o liczbie wartości spinów wystepujących w siatce spins w każdym kroku symulacji

Kroki funkcji wizualizacja(spins):

1. Wywołanie funkcji plot(range(nsteps), state_counts)z biblioteki matplotlib.pyplot. Ta funkcja służy do utworzenia wykresu, gdzie w zakresie nsteps dla każdego kroku przypisana jest wartość odpowiadająca liczbie unikatowych stanów występujących w siatce w danym kroku symulacji.
2. Wywołanie funkcji xlabel("Krok symulacji") i ylabel("Liczba różnych stanów") z biblioteki matplotlib.pyplot. Ta funkcje dodaje odpowiadają za dodanie odpowiednich etykiet dla osi X i dla osi Y.
3. Wywołanie funkcji title("Zależność liczby różnych stanów od kroku symulacji") z biblioteki matplotlib.pyplot. Ta funkcja ustawia tytuł wykresu na "Zależność liczby różnych stanów od kroku symulacji".
4. Wywołanie funkcji show() z biblioteki matplotlib.pyplot. Ta funkcja wyświetla wykres zależności liczby różnych stanów od kroku symulacji.

```
def wykres_liczebnosci(nsteps, state_counts):
    plt.plot(range(nsteps), state_counts)
    plt.xlabel("Krok symulacji")
    plt.ylabel("Liczba różnych stanów")
    plt.title("Zależność liczby różnych stanów od kroku symulacji")
    plt.show()
```

4.8 Opis funkcji wykres_stanow(q, nsteps, state_occurrences)

Funkcja wykres_stanow(q, nsteps, state_occurrences) służy do przedstawienia wykresu zależności liczby wystąpień poszczególnychstanów od kroku symulacji przy użyciu biblioteki matplotlib.

Opis funkcji wykres_stanow(q, nsteps, state_occurrences):

- q: liczba stanów, które mogą przyjąć spiny
- nsteps: liczba kroków symulacji
- state_occurrences: lista przechowująca informacje o liczbie wystąpień poszczególnych stanóww siatce dla kolejnych kroków symulacji

Kroki funkcji wykres_stanow(q, nsteps, state_occurrences):

1. Wywołanie funkcji figure(figsize=(8, 6)) z biblioteki matplotlib.pyplot. Ta funkcja służy do utworzenia wykresu o wymiarach 8x6 jednostek.
2. W kolejnym kroku następuje iterowanie po kolejnych stanach spinów od 0 do q-1, gdzie w każdym kroku tworzona jest linia przedstawiająca zależność liczby wystąpień danego stanu (state_occurrences[state]) od kroku symulacji (nsteps). Dodatkowo dla każdego stanu przypisana jest etykieta informująca o numerze prezentowanego stanu.
3. Wywołanie funkcji xlabel("Krok symulacji") i ylabel("Liczba wystąpień") z biblioteki matplotlib.pyplot. Ta funkcje dodaje odpowiadają za dodanie odpowiednich etykiet dla osi X i dla osi Y.
4. Wywołanie funkcji title("Zależność liczby wystąpień poszczególnych stanów od kroku symulacji") z biblioteki matplotlib.pyplot. Ta funkcja ustawia tytuł wykresu na "Zależność liczby wystąpień poszczególnych stanów od kroku symulacji".
5. Wywołanie funkcji legend() z biblioteki matplotlib.pyplot. Ta funkcja wyświetla legendę przedstawiającą dopasowanie kolorów linii na wykresie do odpowiednich stanów.
6. Wywołanie funkcji show() z biblioteki matplotlib.pyplot. Ta funkcja wyświetla wykres zależności liczby wystąpień poszczególnych stanów od kroku symulacji.

```
def wykres_stanow(q, nsteps, state_occurrences):
    plt.figure(figsize=(8, 6))
    for state in range(q):
        plt.plot(range(nsteps), state_occurrences[state], label=f"Stan {state}")
    plt.xlabel("Krok symulacji")
    plt.ylabel("Liczba wystąpień")
    plt.title("Zależność liczby wystąpień poszczególnych stanów od kroku symulacji")
    plt.legend()
    plt.show()
```

4.9 Opis funkcji wykres_hamiltonian(nsteps, energies)

Funkcja wykres_hamiltonian(nsteps, energies) służy do przedstawienia wykresu zależności wartości Hamiltonianu od kroku symulacji przy użyciu biblioteki matplotlib.

Opis funkcji wykres_hamiltonian(nsteps, energies):

- nsteps: liczba kroków symulacji
- energies: lista przechowująca informacje o wartościach Hamiltonianu w każdym kroku symulacji

Kroki funkcji wykres_hamiltonian(nsteps, energies):

1. Wywołanie funkcji plot(range(nsteps), energies) z biblioteki matplotlib.pyplot. Ta funkcja służy do utworzenia wykresu, gdzie w zakresie nsteps dla każdego kroku przypisana jest wartość odpowiadająca wartości Hamiltonianu w danym kroku symulacji.
2. Wywołanie funkcji xlabel("Krok symulacji") i ylabel("Hamiltonian") z biblioteki matplotlib.pyplot. Ta funkcje dodaje odpowiadają za dodanie odpowiednich etykiet dla osi X i dla osi Y.
3. Wywołanie funkcji title("Wartość Hamiltonianu w kolejnych krokach symulacji") z biblioteki matplotlib.pyplot. Ta funkcja ustawia tytuł wykresu na "Wartość Hamiltonianu w kolejnych krokach symulacji".
4. Wywołanie funkcji show() z biblioteki matplotlib.pyplot. Ta funkcja wyświetla wykres zależności wartości Hamiltonianu od kroku symulacji.

```
def wykres_hamiltonian(nsteps, energies):
    plt.plot(range(nsteps), energies)
    plt.xlabel("Krok symulacji")
    plt.ylabel("Hamiltonian")
    plt.title("Wartość Hamiltonianu w kolejnych krokach symulacji")
    plt.show()
```

4.10 Kod główny programu

1. Poproszenie użytkownika o wprowadzenie liczby wierszy liczby i kolumn siatki spinów w postaci liczby całkowitej i przypisanie tej wartości do zmiennej N.
2. Poproszenie użytkownika o wprowadzenie liczby stanów, jakie mogą przyjmować spiny w postaci liczby całkowitej i przypisanie tej wartości do zmiennej q.
3. Poproszenie użytkownika o wprowadzenie wartości temperatury układu w postaci liczby zmiennoprzecinkowej i przypisanie tej wartości do zmiennej T.
4. Przypisanie wartości stałej Boltzmana (1.38 * 10 ^ -23) do zmiennej k.
5. Poproszenie użytkownika o wprowadzenie wartości stałej oddziaływania w układzie w postaci liczby zmiennoprzecinkowej i przypisanie tej wartości do zmiennej J.
6. Poproszenie użytkownika o wprowadzenie liczby kroków pętli Monte Carlo w postaci liczby całkowitej i przypisanie tej wartości do zmiennej nsteps.
7. Wywołanie funkcji siatka(q, N) i przypisanie jej wyniku do zmiennej spins. Następuje wygenerowanie siatki spinów.
8. Wywołanie funkcji hamiltonian(spins) i przypisanie jej wyniku do zmiennej H. Następuje obliczenie i wyświetlenie w konsoli wartości Hamiltonianu przed rozpoczęciem symulacji.
9. Wywołanie funkcji wizualizacja(spins). Następuje wyświetlenie siatki spinów przed rozpoczęciem symulacji.
10. Wywołanie funkcji MonteCarlo(nsteps, k, T, spins) i przypisanie jej wyniku do zmiennych: spins, state_counts, state_occurrences, energies. Następuje przeprowadzenie symulacji.
11. Wyświetlenie wyniku symulacji w konsoli postaci dwuwymiarowej macierzy przedstawiającej rozmieszczenie spinów o danych stanach.
12. Wywołanie funkcji wizualizacja(spins). Następuje wyświetlenie siatki spinów po przeprowadzeniu symulacji.
13. Wywołanie funkcji hamiltonian(spins) i przypisanie jej wyniku do zmiennej H. Następuje obliczenie i wyświetlenie w konsoli wartości Hamiltonianu po przeprowadzeniu symulacji.
14. Wywołanie funkcji wykres_liczebnosci(nsteps, state_counts). Następuje wyświetlenie wykresu zależności liczby wystąpień poszczególnych stanów od kroku symulacji.

15. Wywołanie funkcji wykres_stanow(q, nsteps, state_occurrences). Następuje wyświetlenie wykresu zależności liczby wystąpień poszczególnych stanów od kroku symulacji.
16. Wywołanie funkcji wykres_hamiltonian(nsteps, energies). Następuje wyświetlenie wykresu zależności wartości Hamiltonianu od kroku symulacji.

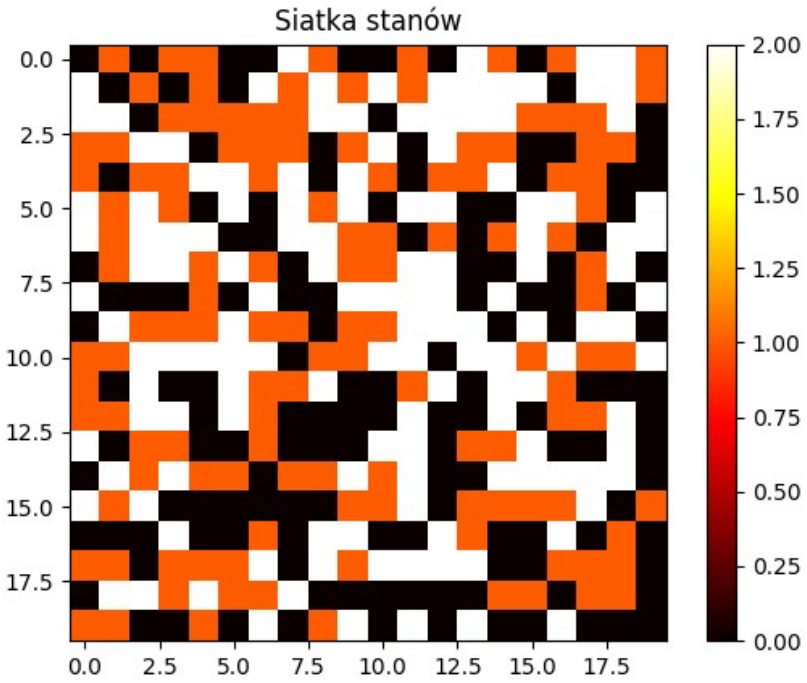
```
# Rozmiar siatki
print(
    "Rozmiar sieci określa utworzenie macierzy, na przykład, "
    "wprowadzenie N = 10 spowoduje utworzenie sieci o wymiarach 10x10")
N = int(input("Podaj rozmiar siatki: "))

# Liczba wartości, które mogą przyjąć spiny
print(
    "Liczba wartości, które mogą być przyjmowane przez spiny określa różnorodność stanów spinów. "
    "Na przykład, wprowadzenie q = 2 oznacza, że spiny mogą przyjąć tylko 2 różne wartości.")
q = int(input("Liczba wartości, które mogą przyjąć spiny: "))

# Temperatura
print("Temperatura symulacji określa jej intensywność."
      "Wprowadź wartość temperatury, na przykład 2.0.")
T = float(input("Temperatura: "))

# Stała Boltzmana
k = 1.38 * 10 ** -23

# Stała oddziaływania
print("Stała oddziaływania określa siłę oddziaływania między sąsiednimi spinami."
      "Im większa wartość stałej oddziaływania, tym większe znaczenie ma wzajemne wpływanie spinów na siebie.")
J = float(input("Stała oddziaływania: "))

# Liczba kroków MC
print("Liczba kroków MC określa liczbę iteracji w metodzie Monte Carlo."
      "Większa liczba kroków MC może prowadzić do dokładniejszych wyników, ale zwiększa czas obliczeń.")
nsteps = int(input("Liczba kroków MC: "))
```

```
print("Generowanie siatki losowych stanów spinów...")
spins = siatka(q, N)
print("Siatka stanów spinów została wygenerowana.")

H = hamiltonian(spins)
print("Wartość Hamiltonianu przed wykonaniem symulacji:", H)

wizualizacja(spins)

print("Przeprowadzanie symulacji Monte Carlo")
spins, state_counts, state_occurrences, energies = MonteCarlo(nsteps, k, T, spins, q)
print("Symulacja Monte Carlo została zakończona.")

print("Siatka spinów po przeprowadzeniu symulacji:")
print(spins)

wizualizacja(spins)

H = hamiltonian(spins)
print("Wartość Hamiltonianu po wykonaniu symulacji:", H)

wykres_liczebnosci(nsteps, state_counts)

wykres_stanow(q, nsteps, state_occurrences)

wykres_hamiltonian(nsteps, energies)
```

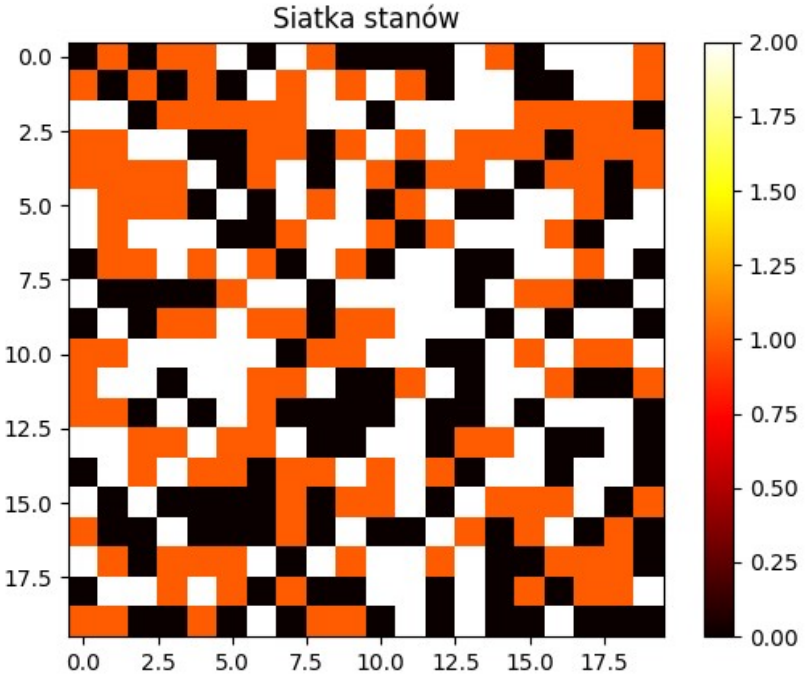
5 Przykłady działania programu

Wynik działania programu dla następujących danych:

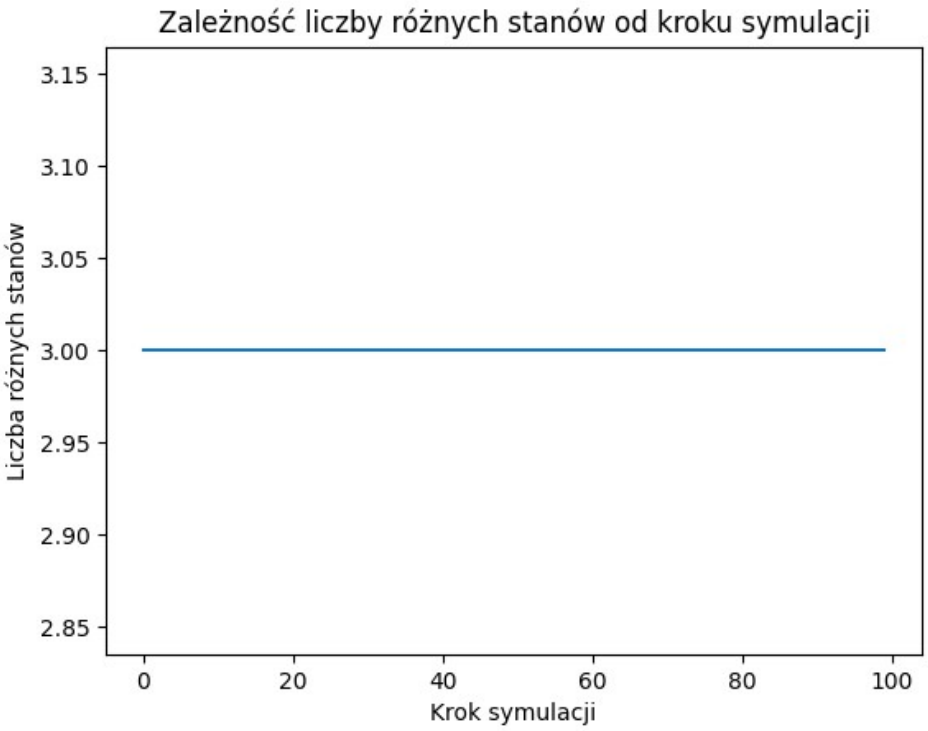
- N: 20
- q: 3
- T: 50.0
- J: 15.0
- nsteps: 100



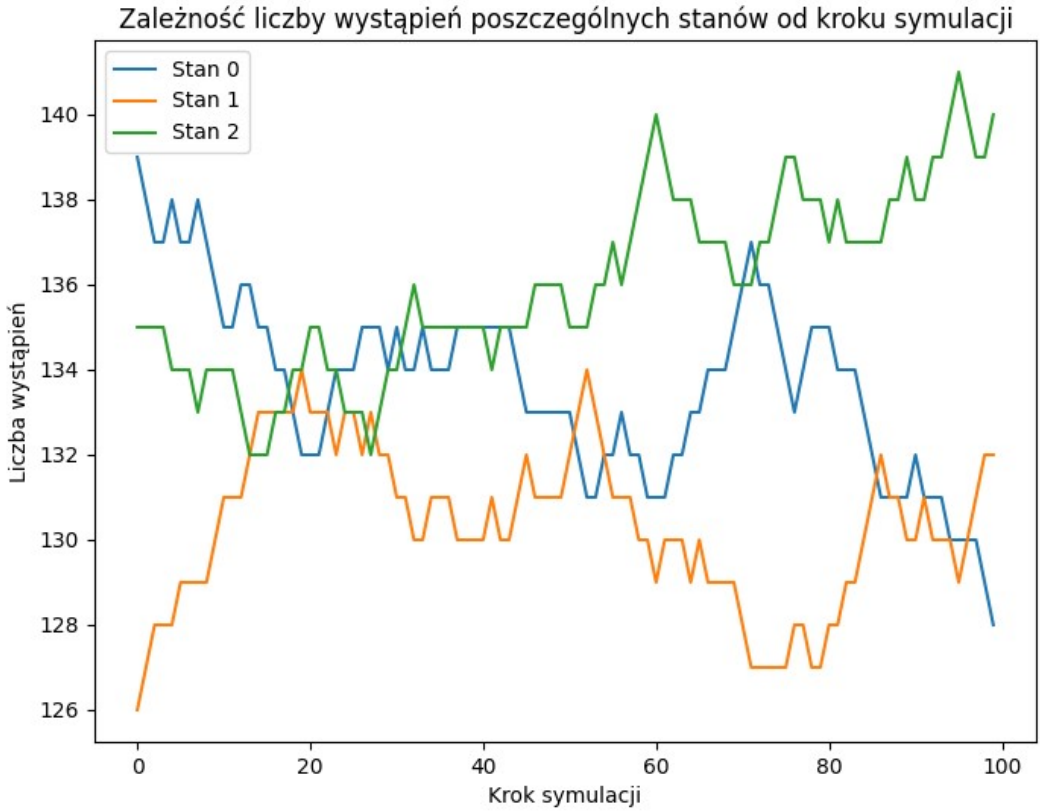
Wykres 1. Siatka spinów przed rozpoczęciem symulacji



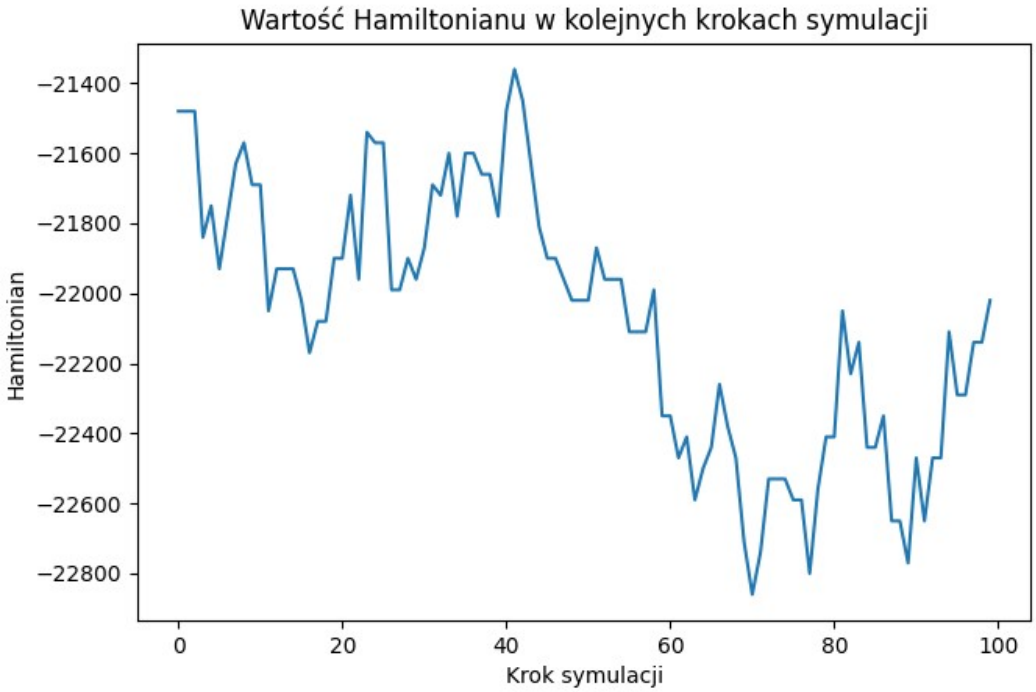
Wykres 2. Siatka spinów po zakończeniu symulacji



Wykres 3. Wykres zależności liczby unikatowych stanów od kroku symulacji



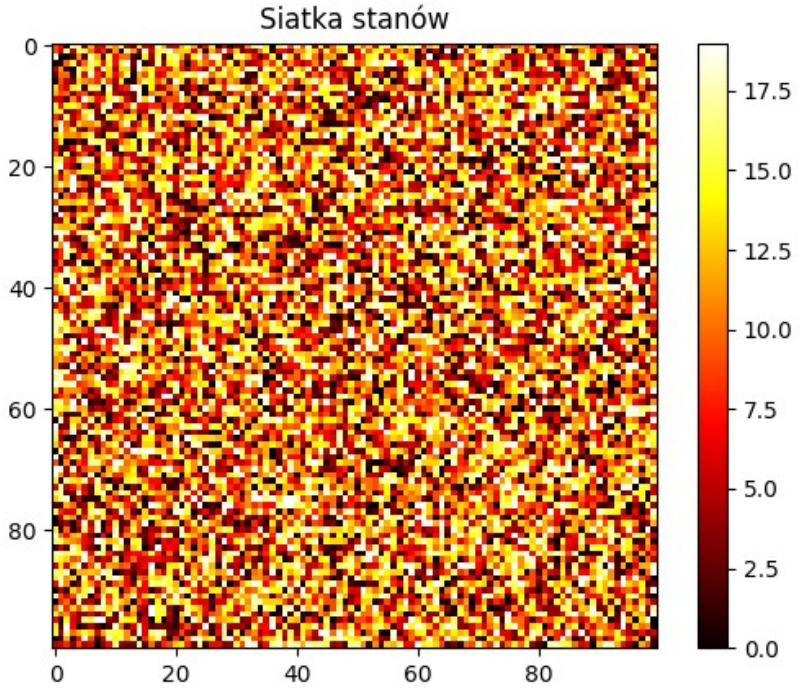
Wykres 4. Wykres zależności liczby wystąpień poszczególnych stanów od kroku symulacji



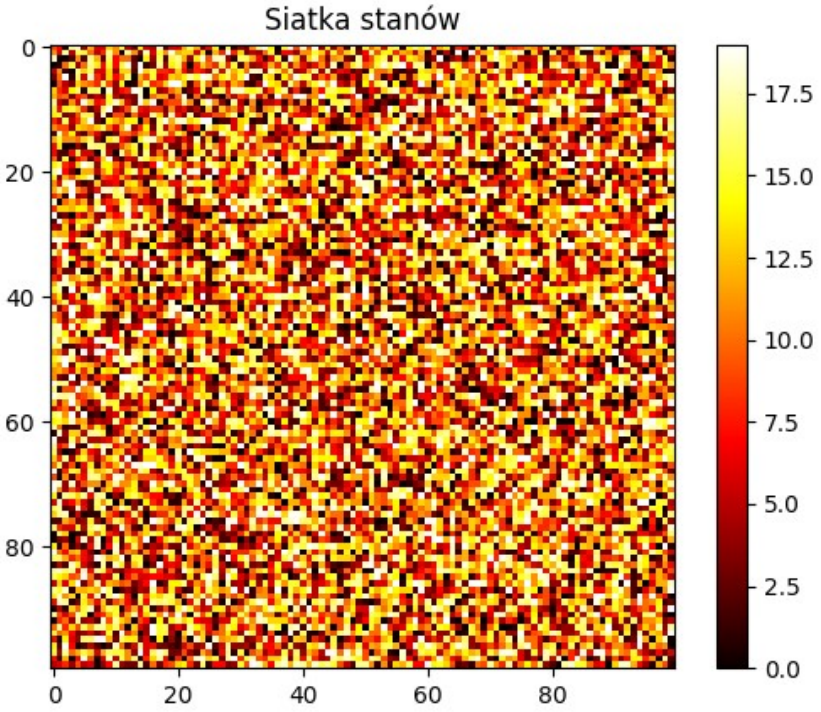
Wykres 5. Wykres zależności wartości Hamiltonianu od kolejnych kroków symulacji

Wynik działania programu dla następujących danych:

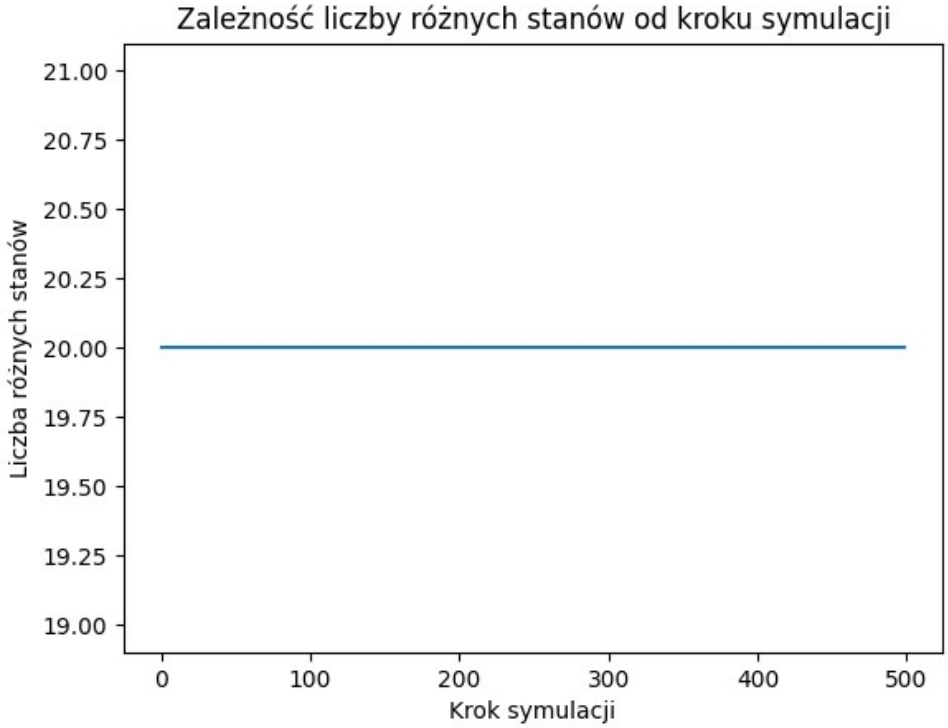
- N: 100
- q: 20
- T: 20.5
- J: 60.3
- nsteps: 500



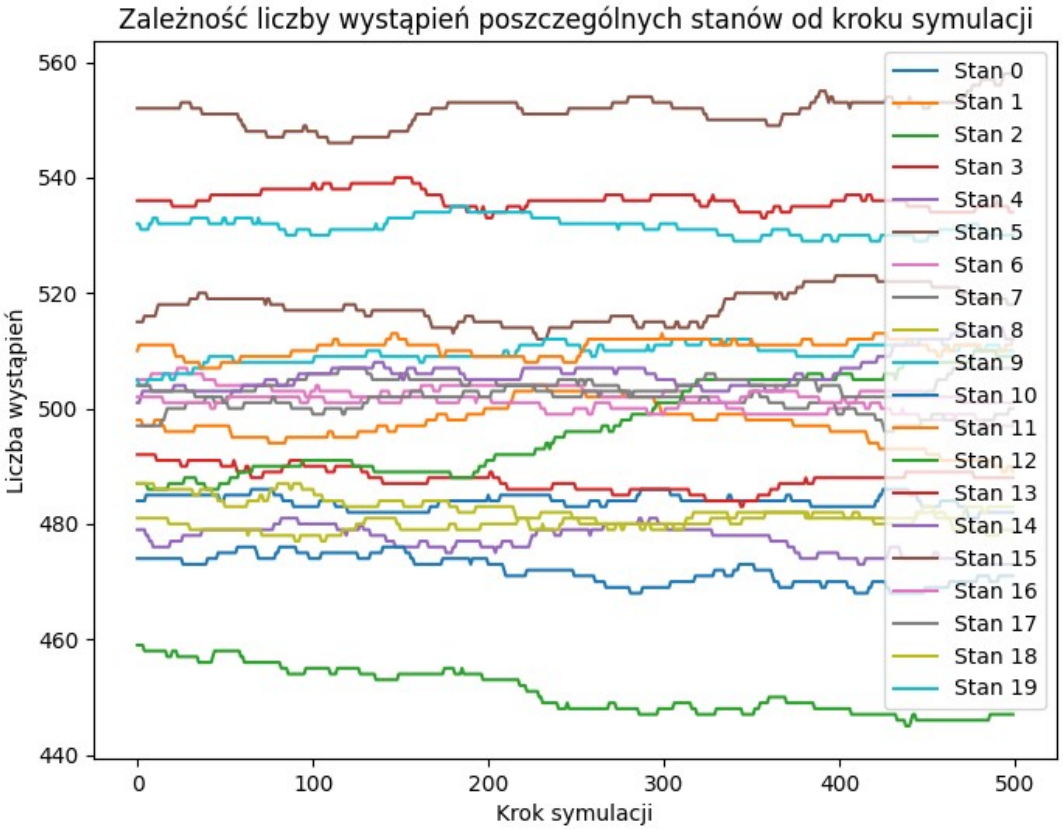
Wykres 6. Wykres siatki spinów przed rozpoczęciem symulacji



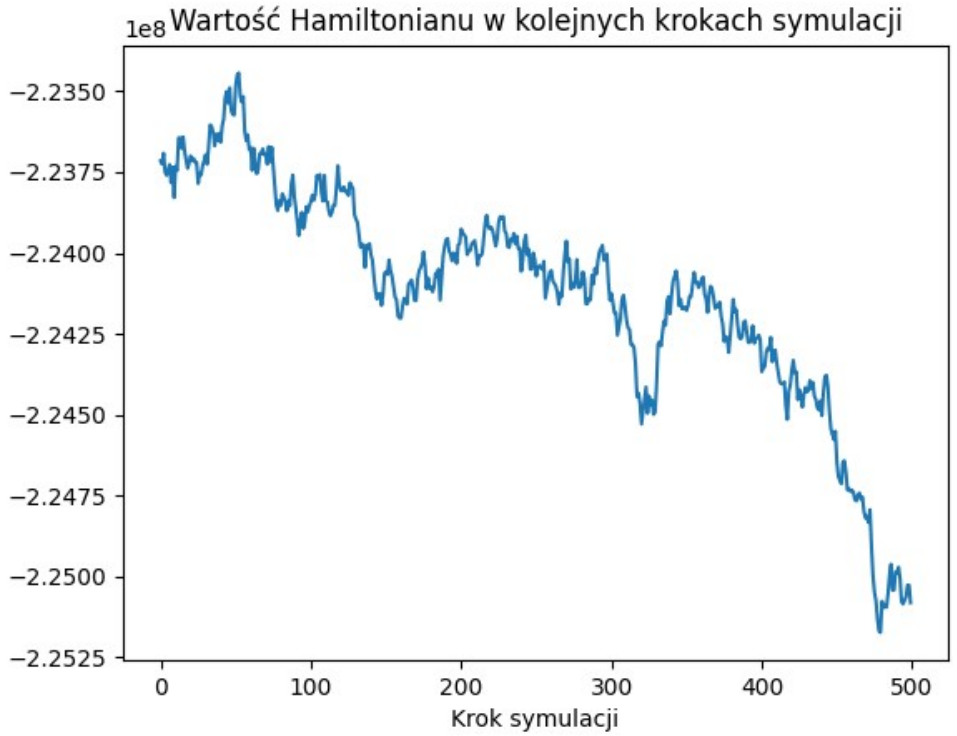
Wykres 7. Wykres siatki spinów po przeprowadzeniu symulacji



Wykres 8. Wykres zależności liczby unikatowych stanów od kroku symulacji



Wykres 9. Wykres zależności liczby wystąpień poszczególnych stanów od kroku symulacji



Wykres 10. Wykres zależności wartości Hamiltonianu od kolejnych kroków symulacji

6 Zastosowania modelu Potts'a

Model Potts'a znajdujezastosowanie w różnych dziedzinach naukowych i inżynieryjnych. Przykłady zastosowań tego modelu:

- Fizyka statystyczna: Model Potts'a jest często wykorzystywany w fizyce statystycznej do analizy zachowań układów magnetycznych. Może być stosowany do badania właściwości magnetycznych materiałów, takich jak ferromagnesy, antyferromagnesy i magnetyki spinowe. Przy użyciu modelu Potts'a można badać fazowe przejścia, analizować zjawisko uporządkowania magnetycznego i analizować zachowanie termodynamiczne systemów magnetycznych.

- Nauki o materiałach: Model Potts'a jest wykorzystywany w badaniach materiałowych do analizy struktury i właściwości różnych materiałów. Może pomóc w zrozumieniu struktury domen magnetycznych, efektów skaleń, wzorców uporządkowania oraz wpływu temperatury i oddziaływań międzyatomowych na materiały magnetyczne.

- Teoria informacji: Model Potts'a jest stosowany w teorii informacji do analizy kodowania i kompresji informacji. Może być wykorzystany do badania problemów związanych z przekazywaniem i przechowywaniem danych, takich jak rozpoznawanie wzorców, kompresja danych, kodowanie kanałów komunikacyjnych i optymalne planowanie sieci.

- Biologia i nauki o życiu: Model Potts'a jest używany w biologii i naukach o życiu do modelowania i analizy różnych aspektów biologicznych systemów. Może być stosowany do badania układów biologicznych, takich jak sieci neuronowe, białka i interakcje międzykomórkowe. Model Potts'a może pomóc w analizie wzorców występujących w układach biologicznych, takich jak struktury przestrzenne białek, dynamika reakcji biochemicznych i rozwoju komórek.

- Informatyka: Model Potts'a jest również wykorzystywany w informatyce do rozwiązywania problemów optymalizacyjnych. Może być stosowany do rozwiązywania problemów przypisania, układania grafów, planowania tras i analizy skomplikowanych systemów informatycznych.

Model Potts'a jest wszechstronnym narzędziem, które znajduje zastosowanie w wielu dziedzinach nauki i inżynierii. Dzięki swojej elastyczności i zdolności do analizy systemów złożonych, stanowi cenne narzędzie do badania i zrozumienia różnych aspektów naturalnych i sztucznych systemów.

7 Modyfikacje

W kodzie dokonano modyfikacji poprawiających skuteczność i działanie algorytmu. W tym celu zostały zmodyfikowane wzory na prawdopodobieństwo oraz hamiltonian, tak by był zgodny dla modelu Potts'a, a nie tylko dla szczególnego przypadku jakim jest model Ising'a. Kod odpowiedzialny za generowanie poszczególnych wykresów został umieszczony w osobnych funkcjach. Elementy tworzące listy zostały umieszczone w funkcji Monte Carlo. Dodano możliwość wpisania dowolnych wartości przez użytkownika, zamiast statycznego przypisywania wartości do zmiennych w kodzie.

8 Źródła

https://en.wikipedia.org/wiki/Potts_model

https://www.sciencedirect.com/topics/physics-and-astronomy/pott-model

https://scholar.rose-hulman.edu/cgi/viewcontent.cgi?article=1194&context=rhumj

https://www.ihes.fr/~duminil/publi/2015%20Currents%20developments%20in%20mathematics.pdf

https://chat.openai.com/