One-Dimensional Cellular Automata

Modelowanie Komputerowe

Gabriela Godek, Gabriela Białoskórska, Wojciech Kura, Ignacy Tekieli
Fizyka Techniczna III.

Czerwiec 2021

Spis treści

1 Wprowadzenie

1.1 Cel pracy

Celem niniejszego projektu była implementacja kodu w wybranym języku programowania (Python/Cpp), na podstawie przykładowego programu zamieszczonego w książce *"The nonlinear workbook"* (Willi-Hans Steeb, 3rd edition). Wybrany przez nas rozdział książki (9.2) dotyczył *One-Dimensional Cellular Automata*. Na potrzeby projektu napisano program `spin.py` w języku Python, na podstawie kodu zamieszczonego w książce w języku Cpp.

1.2 Wstęp teoretyczny

Rozważmy jednowymiarowy pierścień o N punktach (ang.:*sites*). Każdy punkt jest oznaczony odpowiednim indeksem, kolejno od zera, tj. $i = 0, 1, ..., N$. Narzucamy okresowe warunki brzegowe $N = 0$. Każdy punkt i może przyjmować wartości $a_i = 0$ oraz $a_i = 1$. Niech a_i zmienia się w funkcji czasu $a_i(t)$ dla dyskretnych momentów czasowych $t = 0, 1, 2...$ zgodnie z równaniem:

$$a_i(t+1) = (a_{i-1}(t) + a_{i+1}(t))$$

Odwzorowanie to zawiera sumę $(a_{i-1}(t) + a_{i+1}(t))$, jest zatem nieliniowe. Może zostać wyrażone za pomocą zmiennych spinowych $S_j \varepsilon(+1, -1)$, gdzie $S_i = -1$ odpowiada $a_i = 0$ oraz $S_i = +1$ odpowiada $a_i = 1$. Zatem równanie to można zapisać w postaci:

$$S_i(t+1) = -S_{i-1}(t)S_{i+1}(t)$$

Odwzorowanie to ma następujące właściwości:

- Dla $t = 1, 2...$
$$S_1(t)S_2(t)...S_N(t) = (-1)^N$$
- Dla $t = 0, 1...$ oraz $n = 0, 1...$
$$S_i(t + 2^n) = -S_{i-2^n}(t)S_{i+2^n}(t)$$

Dla $N = 2^k$, $k = 1, 2...$ otrzymujemy wszystkie $S_i(t) = -1$ dla wszystkich momentów czasowych $t \geqslant 2^{k-1}$, niezależnie od początkowej konfiguracji spinu pierścienia.

2 Materiały i Metody

2.1 Wiadomości wstępne

Program `spin.py` powstał na podstawie projektu zamieszczonego w książce *"The nonlinear workbook"* (Willi-Hans Steeb, 3rd edition) (`spin.cpp`). Różni się on jednak pod wieloma aspektami, choć idea pozostaje wspólna. Nasz program charakteryzuje się znacznie większą interaktywnością, oczekując danych od użytkownika (ilość komórek oraz kroki czasowe) - w oryginalnym kodzie były one zdefiniowane na wstępie. Ponadto dane początkowe są generowane losowo, dzięki zastosowaniu biblioteki `random`.

2.2 Implementacja

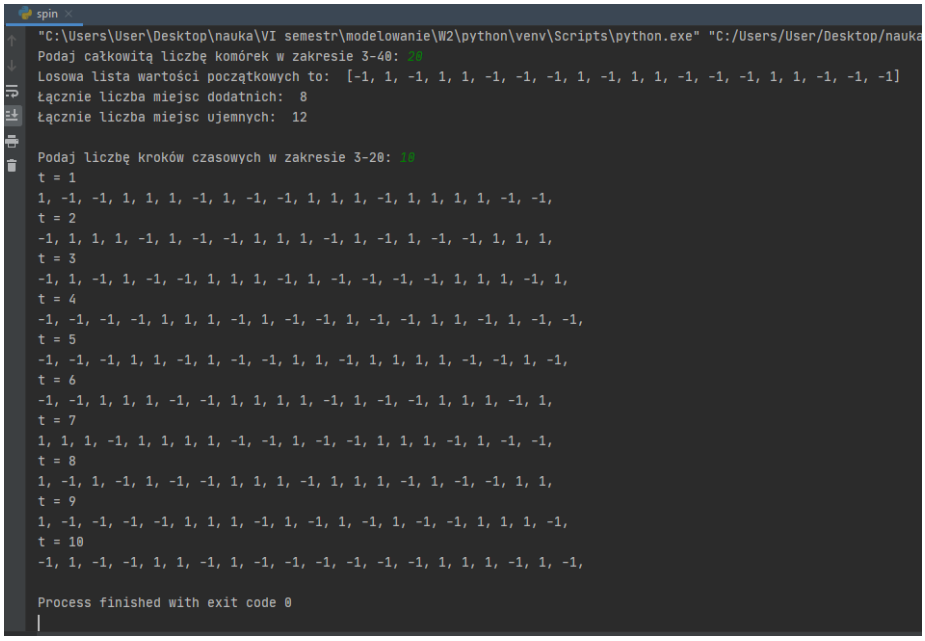
```
import random


# stworzenie klas do wyswietlenia bledow w przypadku podania
wartosci spoza okreslonego zakresu

class Error(Exception):
    """Klasa do tworzenia wlasnych bledow

    """
    pass


class ValueTooSmallError(Error):
    """Blad: zbyt mala wartosc
    Klasa do wywolania bledu dla wartosci ponizej
    okreslonej wartosci granicznej.
    """
    pass


class ValueTooBigError(Error):
    """Blad: zbyt duza wartosc
    Klasa do wywolania bledu dla wartosci powyzej
    okreslonej wartosci granicznej.

    """
    pass


# ustalenie ilosci komorek, zakres wynika z osobistego
osadu sensownosci ilosci komorek, mozna to zmienic
```

```
N = None
while True:
    try:
        N = int(input("Podaj calkowita liczbe komorek w zakresie 3-40: "))
        if N < 3:
            raise ValueTooSmallError
        elif N > 40:
            raise ValueTooBigError
        break
    except ValueError:
        print("To nie jest liczba calkowita. Podaj liczbe calkowita.\n")
        print()
    except ValueTooSmallError:
        print("Liczba poza zakresem, za mala.")
        print()
    except ValueTooBigError:
        print("Liczba poza zakresem, za duza.")
        print()

# wygenerowanie losowej zawartosci dla tablicy S
i wyswietlenie wyniku operacji

minus_count = random.randint(1, N - 1)

# zakres (1, N - 1) gwarantuje zawsze przynajmniej jedno pole ujemne i dodatnie

plus_count = N - minus_count
S = [-1] * minus_count + [1] * plus_count
random.shuffle(S)
print("Losowa lista wartosci poczatkowych to: ", S,
"\nLacznie liczba miejsc dodatnich: ", plus_count,
      "\nLacznie liczba miejsc ujemnych: ", minus_count)
print()
# ustalenie ilosci krokow czasowych dla dzialania programu,
zakres tak samo jak w przypadku ilosci komorek

T = None
while True:
    try:
        T = int(input("Podaj liczbe krokow czasowych w zakresie 3-20: ")) + 1
        if T < 3:
            raise ValueTooSmallError
        elif T > 20:
            raise ValueTooBigError
        break
```

```
        except ValueError:
            print("To nie jest liczba calkowita. Podaj liczbe calkowita.\n")
            print()
        except ValueTooSmallError:
            print("Liczba poza zakresem, za mala.")
            print()
        except ValueTooBigError:
            print("Liczba poza zakresem, za duza.")
            print()
    W = [N]
    for t in range(1, T):
        W = S.copy()
        for j in range(0, N):
            S[j] = W[((j - 1) + N) % N] * W[(j + 1) % N]
        # wypisanie rezulatow dla kazdego kroku czasowego
        print("t =", t)
        for j in range(0, N):
            print(S[j], end=", ")
        print()

    print("\n:)")
```

3 Wyniki

Wyniki dla przykładowych danych wpisanych przez użytkownika:



4 Podsumowanie

W projekcie wykonano program `spin.py`, bazując na kodzie w języku cpp. Pomimo znacznie większej ilości funkcji i nieco większym poziomie skomplikowania, program działa zgodnie z oczekiwaniami. W rozważanym przypadku S[N] oraz S[0] są od siebie zależne, co widać w wynikach.

5 Bibliografia

Literatura wykorzystana do napisania programów oraz wstępu:

- *"The nonlinear workbook"* - Willi-Hans Steeb, 3rd edition
  Chapter 9.2 - *One-Dimensional Cellular Automata*