Równanie Ciepła 2D

1. Wstęp Teoretyczny

Równanie ciepła w 2D może być zapisane jako:

Gdzie:

* u(t, x, y) jest funkcją ciepła zależną od czasu i położenia. Nazywany potem obrazem.
* t to czas
* x,y to współrzędne

Do rozwiązania równania zostanie użyta metoda skończonych metod. Może być zapisana jako:

Oznacza to, że przyblizamy pochodną skończonymi różnicami.

Jeśli oznaczymy:

To możemy zapisać:

Co można także zapisać jako:

Ten wzór będzie bezpośrednio implementowany w programach.

Przy czym elementy skrajne obrazu będą miały stałe wartości i będą stanowić tzw. warunki przegowe. Ich niezmienna temperatura będzie wymuszać zmiany temperatury wewnątrz obrazu.

Wszystkie poniższe równania można także zastąpić opisem intujcyjnym. Zmiana temperatury w czasie w danym punkcie zależy od różnicy temperatur wokół tego punktu. Jeśli różnica jest duża temperatura szybko się zmienia. Jest to zwykłe zjawisko dyfuzji.

1. Implemencja OpenMP

W implementacji OpenMP użyto pętli for z dyrektywą #pragma omp parallel for.

Program oblicza kolejno pewną ilość iteracji i w każdej akutalizuje wartość funkcji 2D. Iteracja po kolejnych rzędach odbywa się sekwencyjnie. Natomiast elementy rzędu są dzielone pomiędzy ilość wątków w każdej iteracji.

Jest to zobrazowane poniżej:

Obraz zawierający kwadrat

Opis wygenerowany automatycznie

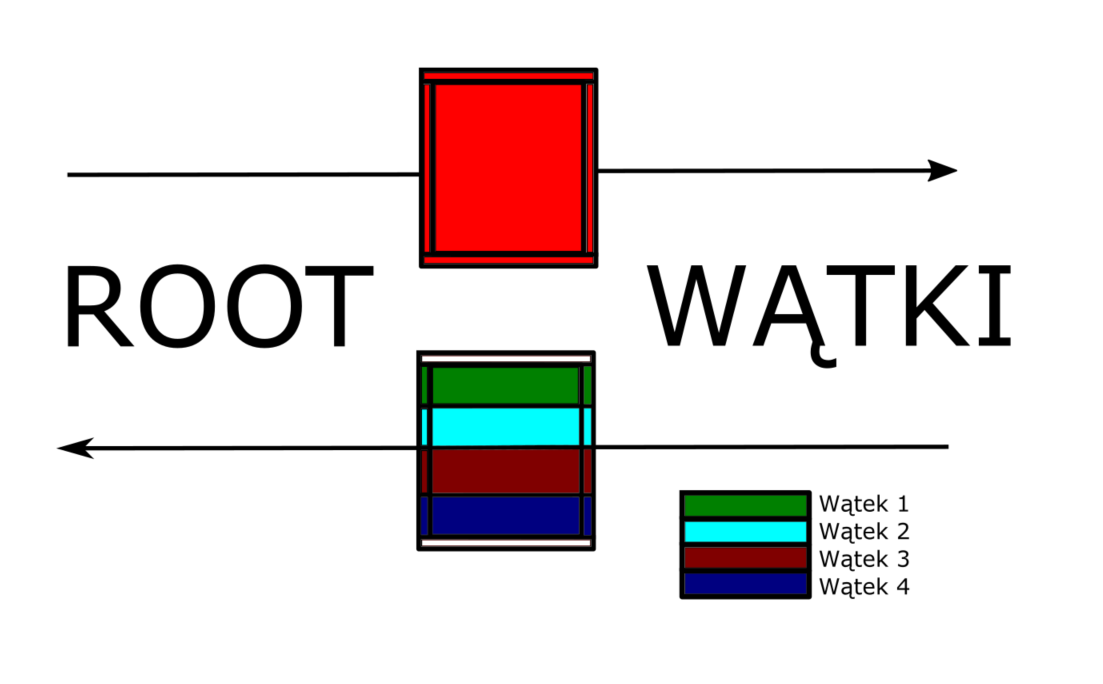
Kod jest dostępny w folderze heat\_openmp w ząłączonym pliku.

1. Implementacja MPI

W implementacji MPI jest użyty następujący algorytm:

* Root rozgłasza pełny obraz za pomocą *Broadcast*.
* Każdy z wątków oblicza swoją część obrazu – tym razem obraz podzielony rzędami, nie kolumnami.
* Za pomocą *Gather* root zbiera części obrazu poza górną i dolną krawędzią, które nie są obliczane.
* Krawędź górna i dolna są kopiowane z obrazu początkowego.
* Całość jest powtarzana aż do wyczerpania się ustalonej liczby iteracji.

Ograniczeniem algorytmu jest to, że (liczba rzędów - 2) jest musi być podzielna przez ilość wątków. Poniżej ilustracja algorytmu.



Kod jest dostępny w folderze heat\_mpi.c.

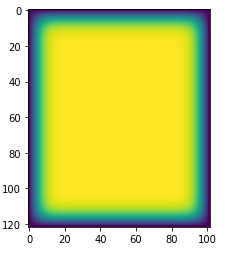
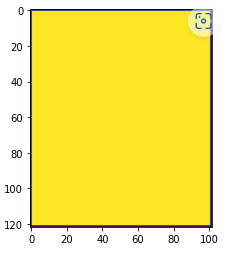
1. Implementacja CUDA
2. Format wejścia wyjścia

Warunki początkowe i rozmiar obrazu ustawiane są makrami w programie. Możliwe jest wybranie temperatury poszczególnych krawędzi i pozostałej części obrazu (np. górna krawędź 100, pozostałe krawędzie 0, reszta elementów 0).

Wyjście programu jest podawane w postaci tablicy kompatybilnej z np.array w Pytonie. Możliwe jest obejrzenie wyników za pomocą matplotlib używając skryptu show.py.

Przykładowy uzyskany obrazy.

**Warunki początkowe: Brzegi 0, reszta obrazu nagrzana do 100.**

****

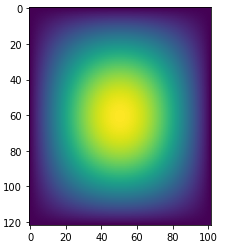
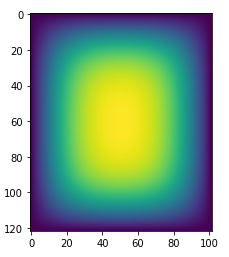
****

Fig. Obraz po kolejno: 0, 100, 1000 i 1000 iteracji.

Można zaobserwować wychładzanie środkowej części obrazu.

Przykładowy obraz ma wymiary 100 x 122. W przypadku większych obrazów może być problem z kopiowaniem tekstu zawierającego macierz.

1. Analiza wydajności algorytmów

W analizie sprawdzano jak szybko algorytmy policzą 1000 iteracji algorytmu dla obrazu o wymiarach 8002 x 2002 (8000 x 2000 plus brzegi).

Uzyskane czasy:

|  |  |  |  |
| --- | --- | --- | --- |
| Biblioteka | 1 wątek | 2 wątki | 4 wątki |
| OpenMP | 27742ms | 13439ms | 9981ms |
| OpenMPI | 190299ms | 138083ms | 147892ms |
|  |  |  |  |