# Programowanie równoległe i rozproszone

Projekt – Szybka transformata Fouriera (FFT)

*Michał Szlachetka 205684* | *Mateusz Kurpet 200905*

# Wstęp

Celem transformaty Fouriera jest zamiana sygnału zmierzonego w dziedzinie czasu na sygnał w dziedzinie częstotliwości. Dzięki temu można uzyskać nowe informacje na temat pewnych wielkości fizycznych, które nie były widoczne na przebiegu czasowym.

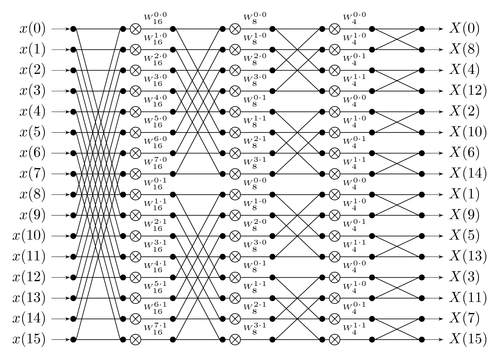
Matematycznie transformata Fouriera (DFT) polega na wykonaniu przekształcenia dla każdego zmierzonego punktu:

X(m) =

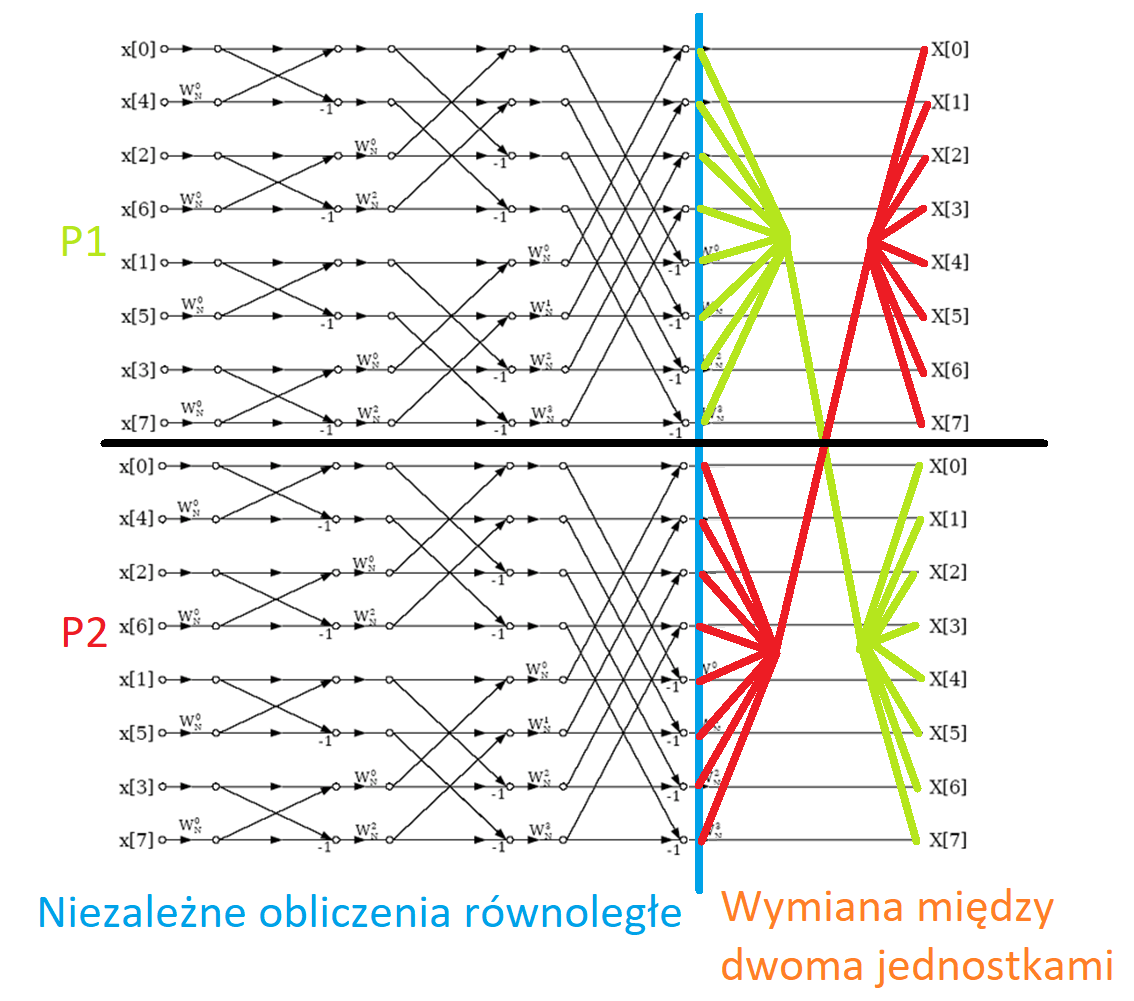
* X(m) – wartość transformaty
* x(n) – wartość danej próbki
* N – liczba zebranych próbek

Klasyczne podejście wymaga wykonania operacji między każdą parą próbek, więc złożoność obliczeniowa wynosi **O(**).

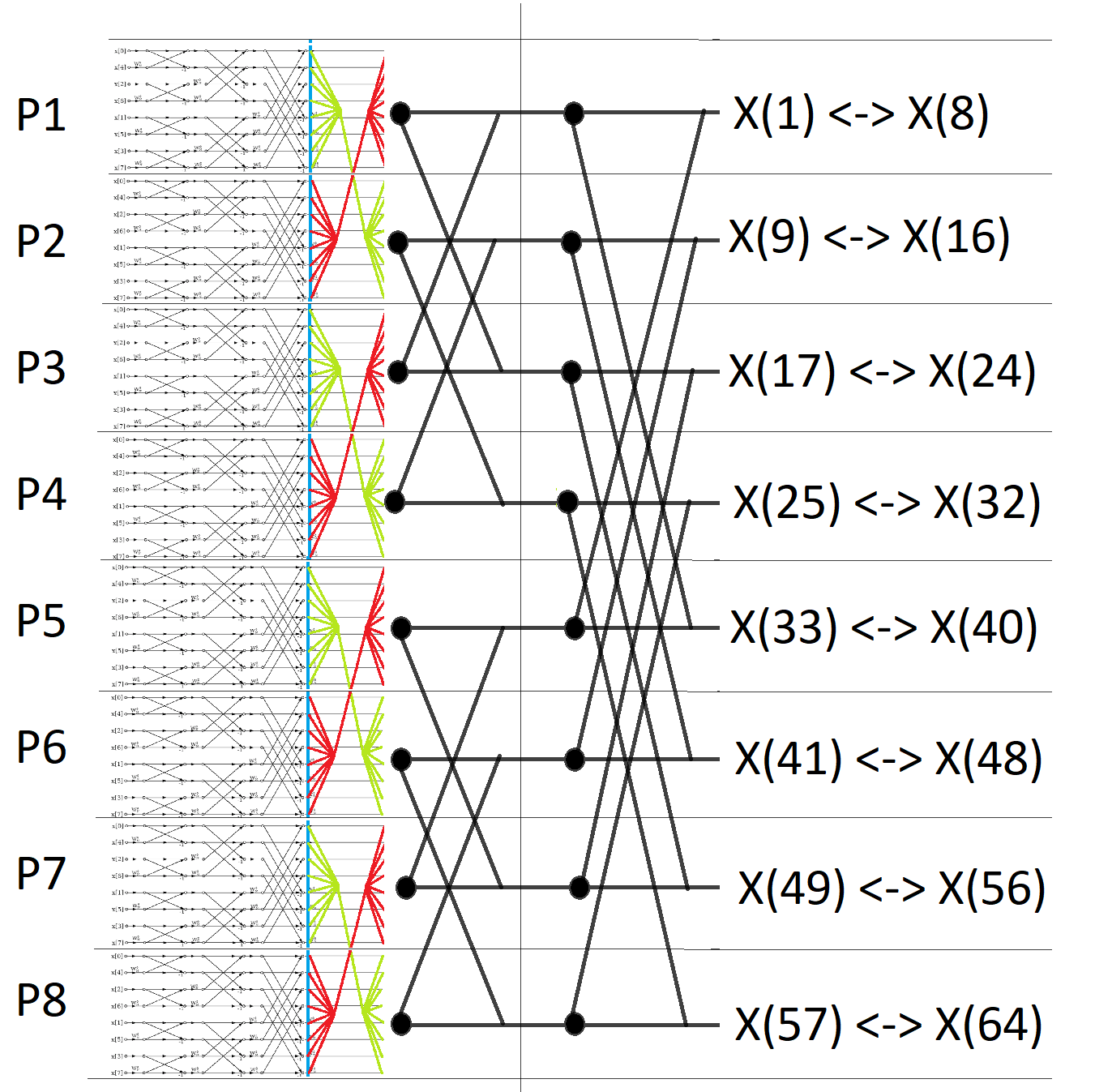
FFT – szybka transformata Fouriera wykorzystuje metodę dziel i rządź. Polega na rozłożeniu zadania na obliczanie mniejszych transformat: FFT(n) -> 2x FFT(n/2) -> 4x FFT(n/4) ---> n/2 x FFT(2). Następnie „małe” transformaty są składane w coraz większe w strukturze motylkowej. Złożoność takiego podejścia to tylko **O(n\*log(n)).**



FFT równolegle – polega również na wykorzystaniu metody dziel i rządź oraz możliwości składania transformat. Każda jednostka obliczeniowa dostaje N/p danych. Dopóki krok wymaga skrzyżowania mniejszej ilości punktów niż zostało przydzielone danej jednostce, operacje na wszystkich jednostkach wykonywane są w tym samym czasie zupełnie niezależnie. W momencie gdy ilość danych do skrzyżowania przekracza N/p rozpoczynają się wymiany pomiędzy jednostkami.



Można powiedzieć że od tego momentu zamiast scalania N/p punktowych transformat łączymy teraz p-jednostkowe transformaty. Wymiana między p1-p2 odpowiada dwu-punktowej transformacie Fouriera. Powielając ten proces dla 64-punktowej transformaty przy 8 jednostkach obliczeniowych otrzymamy poniższą strukturę motylkową:



Całkowity przebieg algorytmu (z uwzględnieniem specyficznych operacji wymaganych przez FFT) wygląda następująco:

* Zaokrąglij liczbę danych wejściowych do najbliższej wyższej potęgi 2
* Wykonaj odwrócenie bitowe na danych wejściowych, tak aby otrzymać wyniki w prawidłowej kolejności
* Wylicz współczynniki – jeden raz
* Rozdziel dane równo pomiędzy jednostki obliczeniowe
* Dopóki liczba elementów krzyżujących się jest mniejsza od ilości danych na jednostce, obliczaj FFT lokalnie
* W przeciwnym razie wymieniaj wszystkie aktualne wyliczenia z odpowiednią jednostką w strukturze motyla i wylicz FTT
* Po ostatniej wymianie zbierz wyniki

# Sposób uruchomienia programu

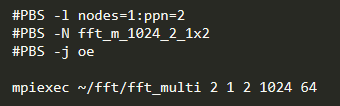
W ramach projektu zostały napisane 2 programy w języku C++:

* *fft\_single.cpp* – program w wersji sekwencyjnej
* *fft\_multi.cpp* – program w wersji zrównoleglonej

Poprawne uruchomienie programu wymaga podania w pliku *.pbs* 5-ciu parametrów wywołania skryptu:

1. liczba jednostek obliczeniowych (np. 2)
2. liczba wykorzystanych węzłów (np. 1)
3. liczba wykorzystanych jednostek obliczeniowych w ramach węzła (np. 2)
4. częstotliwość próbkowania (np. 1024)
5. czas pomiaru [s] (np. 64)

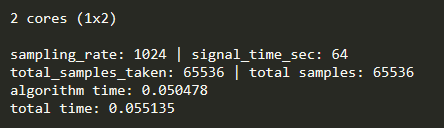
Nazwa pliku wynikowego (np. *fft\_m\_1024\_2\_1x2.o17477*) zawiera informacje o wersji programu (sekwencyjna/zrównoleglona), częstotliwości próbkowania oraz liczby jednostek obliczeniowych i sposobu ich wykorzystania na klastrze.



# Interpretacja wyników programu

Plik wynikowy zawiera wartości parametrów z jakimi został wywołany dany skrypt, oraz informacje na temat czasu wykonania algorytmu:

* N cores (nodes x ppn) – liczba wykorzystanych jednostek obliczeniowych
* sampling\_rate – częstotliwość próbkowania
* signal\_time\_sec – czas pomiaru
* total\_samples\_taken / total\_samples – liczba próbek sygnału (sampling\_rate x signal\_time\_sec)
* algorithm time – czas wykonania algorytmu właściwego szybkiej transformaty Fouriera
* total time – czas wykonania algorytmu właściwego + czas zebrania wyników ze wszystkich jednostek obliczeniowych + wyliczenie ostatecznego wyniku.



Dodatkowo, plik wynikowy z wykonania sekwencyjnej wersji programu, zawiera dane wejściowe (sygnał sinusoidalny) oraz dane wyjściowe (wynik szybkiej transformaty Fouriera) algorytmu.

# Przeprowadzone badania

W ramach projektu zostały przeprowadzone badania w wyniku których zostały wyliczone szybkie transformaty Fouriera dla sygnału będącego falą sinusoidalną:

)

Współczynniki:

* A = 1 - amplituda
* ω = 2π - pulsacja
* ϴ = 0 - przesunięcie fazowe

W ramach badań przyjęliśmy stały czas pomiaru, który wyniósł 64 sekund oraz 4 różne częstotliwości próbkowania: 1024, 8192, 32768, 131072. Dzięki temu mogliśmy sprawdzić czy wykorzystanie zrównoleglonego FFT ma sens w przypadku mierzenia sygnałów o dużej częstotliwości.

Na jednym procesorze został wykonany program w wersji sekwencyjnej, natomiast wersja zrównoleglona algorytmu, została wywołana na 2, 4 ,8, 16 i 32 jednostkach obliczeniowych.

Przykładowe wygenerowane sygnały oraz wyniki FFT:

Zgodnie z oczekiwaniami transformata sygnału sinusoidalnego o częstotliwości 1Hz wykazała wystąpienie dokładnie tej częstotliwości. Transformata sygnału złożonego z trzech sinusów o różnych amplitudach wykazała wystąpienie wszystkich trzech zadanych częstotliwości. Również wartości amplitud są widoczne na wykresie.

# Wyniki badań

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Czas pomiaru [s] | Częstotliwość próbkowania | Liczba procesorów | Średni czas [s] | Przyśpieszenie | Efektywność |
| 64 | **1024** | 1 | 0,082117 | 1,000 | 100% |
| 2 | 0,050478 | 1,627 | 81% |
| 4 | 0,037272 | 2,203 | 55% |
| 8 | 0,034475 | 2,382 | 30% |
| 16 | 0,031269 | 2,626 | 16% |
| 32 | 0,029964 | 2,741 | 9% |
| **8192** | 1 | 0,845486 | 1,000 | 100% |
| 2 | 0,518014 | 1,632 | 82% |
| 4 | 0,432885 | 1,953 | 49% |
| 8 | 0,313095 | 2,700 | 34% |
| 16 | 0,249949 | 3,383 | 21% |
| 32 | 0,235940 | 3,583 | 11% |
| **32768** | 1 | 4,473333 | 1,000 | 100% |
| 2 | 2,664443 | 1,679 | 84% |
| 4 | 2,280371 | 1,962 | 49% |
| 8 | 1,609433 | 2,779 | 35% |
| 16 | 1,126289 | 3,972 | 25% |
| 32 | 0,848957 | 5,269 | 16% |
| **131072** | 1 | 21,796237 | 1,000 | 100% |
| 2 | 12,131160 | 1,797 | 90% |
| 4 | 8,972502 | 2,429 | 61% |
| 8 | 6,590262 | 3,307 | 41% |
| 16 | 4,814520 | 4,527 | 28% |
| 32 | 3,722278 | 5,856 | 18% |

# Wnioski

* Zrównoleglenie obliczeń przyniosło pożądany efekt. Dla wszystkich badanych częstotliwości można było zaobserwować przyśpieszenie przy zrównolegleniu obliczeń.
* Najlepsze wyniki zostały osiągnięte dla największych badanych wartości częstotliwości próbkowania: 131072 oraz 32768.
* Wykorzystanie 32 jednostek obliczeniowych dla częstotliwości próbkowania 8192 nie daje znaczącego wzrostu przyśpieszenia.
* Wykorzystanie więcej niż 4 jednostek obliczeniowych dla częstotliwości próbkowania 1024, nie daje zbyt dużego wzrostu przyśpieszenia.
* Im większa częstotliwości próbkowania tym dłużej można obserwować wzrost przyśpieszenia przy równoczesnym wzroście liczby jednostek obliczeniowych.
* Zależność efektywności zrównoleglania względem liczby wykorzystanych jednostek obliczeniowych jest podobna dla wszystkich badanych częstotliwości próbkowania – efektywność w podobny sposób maleje wraz ze wzrostem liczby wykorzystanych jednostek obliczeniowych.  
  Najlepszą efektywność osiąga największa badana częstotliwość próbkowania (131072), natomiast najgorszą efektywność osiąga najmniejsza częstotliwość próbkowania (1024).