Rachunek macierzowy – program 1

Ernest Roszak, Maksymilian Wojnar

1 Wstęp

W programie zostały zaimplementowane dwie metody mnożenia macierzy – tradycyjna, oraz metoda Bineta. Językiem implementacji jest Matlab.

2 Pseudokod rozwiązania

Definiujemy funkcję `Matmul`, która wykonuje mnożenie macierzy A i B, używając jednej z dwóch metod w zależności od rozmiaru macierzy. Jako argumenty przyjmuje ona dwie macierze – A i B, oraz parametr l. Jeśli liczba wierszy A jest mniejsza lub równa parametrowi l, funkcja korzysta z klasycznej metody mnożenia macierzy. W przeciwnym razie używamy rekurencyjnej metody Bineta.

Funkcja `ClassicMatmul` wykonuje klasyczne mnożenie macierzy. Jej argumentami są dwie macierze A i B. Dla każdego wiersza macierzy A oraz kolumny B funkcja oblicza iloczyn skalarny, a wynik jest zapisywany w odpowiednim miejscu macierzy C.

Funkcja `BinetMatmul` wykorzystuje rekurencyjną metodę mnożenia macierzy, która dzieli macierze A i B na cztery mniejsze podmacierze, następnie wykonuje operacje mnożenia na tych podmacierzach i sumuje je, aby utworzyć wynikową macierz C. Argumentami funkcji są macierze A i B, oraz parametr l. Proces jest powtarzany aż do osiągnięcia wartości parametru l, w którym funkcja przełącza się na klasyczną metodę mnożenia macierzy.

function MATMUL(A, B, l)
  if $size(A, 1) \leqslant l$ **then**
    $C \leftarrow ClassicMatmul(A, B)$
  else
    $C \leftarrow BinetMatmul(A, B, l)$
  end if
  return C
end function

function CLASSICMATMUL(A, B)
  $n, m \leftarrow size(A)$
  $m, k \leftarrow size(B)$
  $C \leftarrow$ matrix of size $n \times k$
  for $a \leftarrow 1$ to n **do**
    for $b \leftarrow 1$ to k **do**

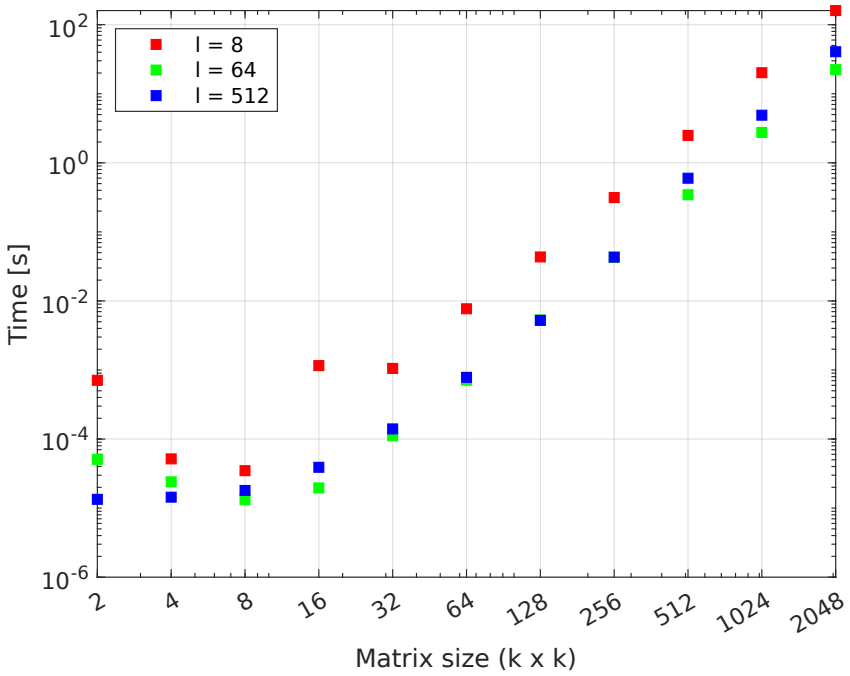
```
            sum ← 0
            for c ← 1 to m do
                sum ← sum + A(a, c) · B(c, b)
            end for
            C(a, b) ← sum
        end for
    end for
    return C
end function

function BinetMatmul(A, B, l)
    n, m ← size(A)
    A11 ← A(1 : n/2, 1 : m/2)
    A12 ← A(1 : n/2, m/2 + 1 : end)
    A21 ← A(n/2 + 1 : end, 1 : m/2)
    A22 ← A(n/2 + 1 : end, m/2 + 1 : end)

    m, k ← size(B)
    B11 ← B(1 : m/2, 1 : k/2)
    B12 ← B(1 : m/2, k/2 + 1 : end)
    B21 ← B(m/2 + 1 : end, 1 : k/2)
    B22 ← B(m/2 + 1 : end, k/2 + 1 : end)

    C11 ← Matmul(A11, B11, l) + Matmul(A12, B21, l)
    C12 ← Matmul(A11, B12, l) + Matmul(A12, B22, l)
    C21 ← Matmul(A21, B11, l) + Matmul(A22, B21, l)
    C22 ← Matmul(A21, B12, l) + Matmul(A22, B22, l)

    C ← [C11, C12;
         C21, C22]
    return C
end function
```

3 Wyniki

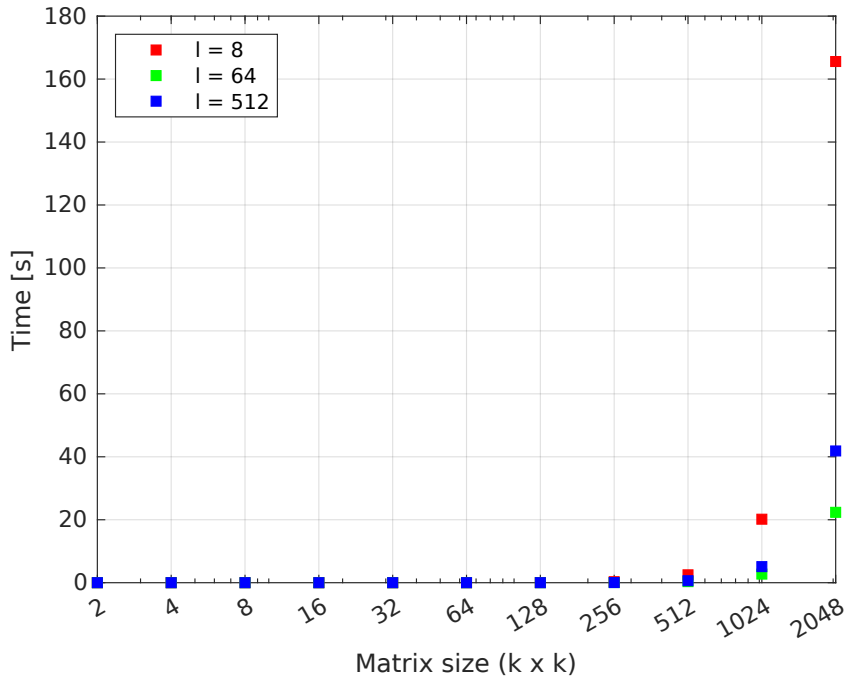
Przygotowaliśmy skrypt, który przeprowadza eksperymenty dotyczące czasu i liczby operacji zmiennoprzecinkowych podczas mnożenia macierzy. Skrypt generuje losowe macierze A i B o zadanych rozmiarach oraz wykonuje mnożenie macierzy $C = A \times B$ za pomocą funkcji `Matmul`. Następnie mierzy czas mnożenia macierzy oraz liczbę operacji i zwraca średnią wartość dla każdego rozmiaru macierzy oraz wybranego parametru l. Każdy eksperyment został powtórzony 5 razy na wylosowanych macierzach zadanego rozmiaru w celu zwiększenia wiarygodności wyników. Do operacji zaliczamy działania dodawania, odejmowania, mnożenia i dzielenia, **nie** jest natomiast zliczana liczba przypisań do zmiennej.

Wykresy zostały wygenerowane dla trzech różnych wartości $l = \{8, 64, 512\}$. Wyniki zostały zaprezentowane na dwóch rodzajach wykresów (log-log oraz log-lin), aby odpowiednio pokazać różnice w czasie i liczbie operacji, a jednocześnie uchwycić skalę różnic pomiędzy poszczególnymi eksperymentami.

3.1 Czas mnożenia macierzy

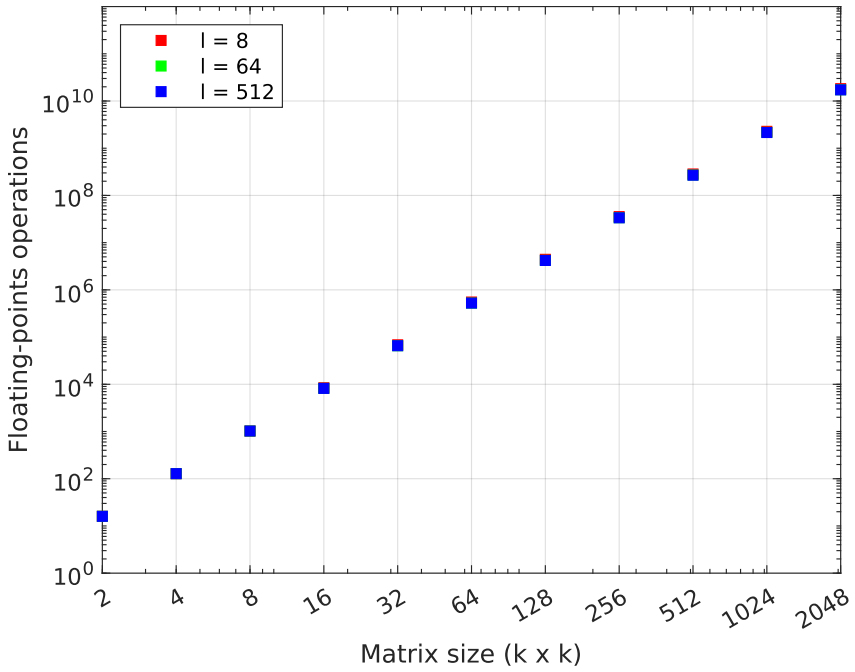


Rysunek 1: Wykres log-log

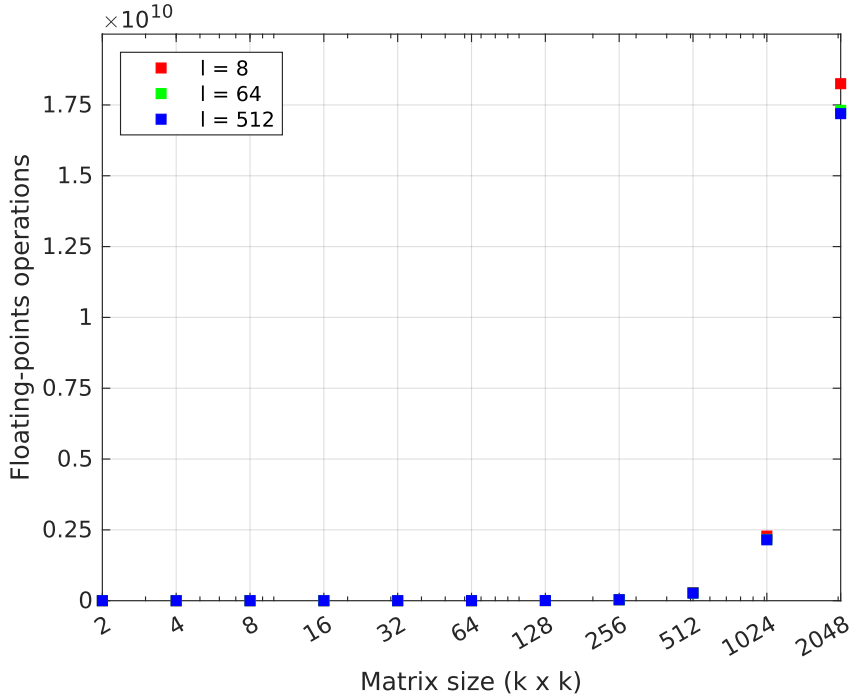


Rysunek 2: Wykres log(X)-lin(Y)

3.2 Liczba operacji zmiennoprzecinkowych



Rysunek 3: Wykres log-log



Rysunek 4: Wykres log(X)-lin(Y)

4 Wnioski

- Dla różnych wartości parametru l liczba wykonywanych operacji jest podobna, jednak czas mnożenia jest znacząco różny. Może to wynikać z ilości danych przesyłanych pomiędzy pamięcią podręczną, a pamięcią cache. Parametr l zmienia nie tylko liczbę operacji zmiennoprzecinkowych, ale również wielkość macierzy przesyłanych pomiędzy różnymi poziomami pamięci komputera.

- Z trzech testowanych wartości parametru l najkrótszy czas mnożenia został osiągnięty dla $l = 64$. Najgorsze wyniki dla wszystkich testowanych wartości zostały osiągnięte dla $l = 8$. Największa wartość $l = 512$ radzi sobie odrobinę gorzej niż $l = 64$ dla dużych macierzy, co sugeruje, że zwiększenie wartości l nie zawsze wiąże się ze zmniejszeniem czasu mnożenia macierzy.