Sprawozdanie 5 - Obliczenia Naukowe

Michał Kallas

5 stycznia 2025

1 Wstęp

W zadaniu musimy rozwiązać układ równań liniowych $Ax = b$, gdzie:

- $A \in \mathbb{R}^{n \times n}$ jest macierzą rzadką i blokową, opisaną równaniem (1),
- $b \in \mathbb{R}^n$ jest wektorem prawych stron,
- n jest dużą liczbą, podzielną przez ℓ, gdzie $\ell \geqslant 2$ to rozmiar kwadratowych bloków macierzy.

Struktura macierzy A jest zdefiniowana jako:

$$\begin{pmatrix} A_1 & C_1 & 0 & 0 & 0 & \cdots & 0 \\ B_2 & A_2 & C_2 & 0 & 0 & \cdots & 0 \\ 0 & B_3 & A_3 & C_3 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & B_{v-2} & A_{v-2} & C_{v-2} & 0 \\ 0 & \cdots & 0 & 0 & B_{v-1} & A_{v-1} & C_{v-1} \\ 0 & \cdots & 0 & 0 & 0 & B_v & A_v \end{pmatrix}, \tag{1}$$

gdzie $v = \frac{n}{\ell}$ oraz:

- $A_k \in \mathbb{R}^{\ell \times \ell}$ $(k = 1, \ldots, v)$ to macierze gęste (wypełnione niezerowymi elementami),
- $B_k \in \mathbb{R}^{\ell \times \ell}$ $(k = 2, \ldots, v)$ to macierze, w których tylko dwie ostatnie kolumny są niezerowe,
- $C_k \in \mathbb{R}^{\ell \times \ell}$ $(k = 1, \ldots, v-1)$ to macierze diagonalne,
- $0 \in \mathbb{R}^{\ell \times \ell}$ to macierze zerowe.

1.1 Macierze B_k i C_k

Macierz B_k ma tylko dwie ostatnie kolumny niezerowe:

$$B_k = \begin{pmatrix} 0 & \cdots & 0 & b^k_{1,\ell-1} & b^k_{1,\ell} \\ 0 & \cdots & 0 & b^k_{2,\ell-1} & b^k_{2,\ell} \\ \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & \cdots & 0 & b^k_{\ell,\ell-1} & b^k_{\ell,\ell} \end{pmatrix}$$

Macierz C_k jest diagonalna:

$$C_k = \begin{pmatrix} c_1^k & 0 & 0 & \cdots & 0 \\ 0 & c_2^k & 0 & \cdots & 0 \\ 0 & 0 & c_3^k & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & c_{\ell-1}^k \\ 0 & 0 & 0 & \cdots & c_\ell^k \end{pmatrix}$$

2 Zadanie 1

2.1 Opis problemu

Napisać funkcję rozwiązującą układ $Ax = b$ metodą eliminacji Gaussa uwzględniającą specyficzną postać (1) macierzy A dla dwóch wariantów:

(a) bez wyboru elementu głównego,

(b) z częściowym wyborem elementu głównego.

2.2 Metoda eliminacji Gaussa (bez wyboru elementu głównego)

2.2.1 Opis metody

Eliminacja Gaussa jest podstawową metodą rozwiązywania układów równań liniowych $Ax = b$. Składa się z dwóch kroków - sprowadzenia macierzy A do macierzy górnotrójkątnej oraz rozwiązania otrzymanego układu równań.

Sprowadzenie do macierzy górnotrójkątnej zaczynamy od zerowania elementów w pierwszej kolumnie, począwszy od dołu. Dokonujemy tego odejmując wielokrotności innych wierszy od aktualnego wiersza. W pierwszej kolumnie zerujemy $n-1$ elementów(wszystkie poza pierwszym elementem macierzy), a w każdej kolejnej o jeden mniej na górze. Po wyzerowaniu elementów w $n-1$ kolumn otrzymujemy macierz górnotrójkątną.

Następnym krokiem jest rozwiązanie otrzymanego układu równań. Ostatnią niewiadomą możemy wyliczyć w prosty sposób:

$$x_n = \frac{b_n}{a_{n,n}}$$

Kolejne wyliczamy podobnie, odejmując poprzednie wartości w celu otrzymania jednej niewiadomej w danym wierszu:

$$x_i = \frac{b_i - \sum_{j=i+1}^{n} x_j}{a_{i,i}}$$

W ten sposób otrzymujemy rozwiązanie układu równań $Ax = b$.

2.2.2 Złożoność

Złożoność pierwszego etapu wynosi $O(n^3)$, dlatego że potrzeba wykonać $\frac{n(n-1)}{2}$ kroków eliminacji wierszy, w każdym kroku wykonując operacje na $n-k$ wierszach, gdzie k to numer aktualnego kroku. Złożoność drugiego etapu wynosi $O(n^2)$, dlatego że musimy wyznaczyć wartości n niewiadomych i dla każdej z nich wykonać operacje proporcjonalne do liczby kolumn. W związku z tym złożoność całego algorytmu to $O(n^3) + O(n^2) = O(n^3)$.

Dzięki specyficznej strukturze macierzy A możemy zmodyfikować algorytm w celu zredukowania złożoności. Łatwo zauważyć, że A posiada dużo elementów zerowych, co pozwoli nam przyspieszyć pierwszy etap algorytmu. W każdej kolumnie pod przekątną może występować co najwyżej ℓ elementów niezerowych. To pozwala ograniczyć liczbę elementów do wyzerowania do $\ell \cdot (n-\ell)$. Co więcej, złożoność odejmowania wierszy jest tutaj zależna od ℓ, a nie od n. W takim wypadku, jako że ℓ jest stałą, złożoność etapu eliminacji wynosi:

$$O(\ell \cdot \ell \cdot (n-\ell)) = O(n)$$

Złożoność w drugim etapie także ulegnie zmniejszeniu, jako że suma potrzebna do wyliczenia x_n nie będzie teraz zależna od n, tylko od ℓ. Dzięki temu złożoność w tym etapie także będzie liniowa, co sprawia że cały algorytm będzie działał w czasie liniowym:

$$\underbrace{O(n)}_{\text{1. etap}} + \underbrace{O(n)}_{\text{2. etap}} = O(n)$$

2.2.3 Pseudokod

Korzystając z powyższych obserwacji możemy stworzyć liniową wersję eliminacji Gaussa. W celu zmniejszenia liczby operacji musimy wyliczać indeksy ostatniego wiersza mającego niezerowy element w danej kolumnie oraz ostatniej kolumny mającej niezerowy element w danym wierszu. Będziemy wyliczali indeks takiego wiersza jako $\min\{k+\ell+1, n\}$, a indeks takiej kolumny jako $\min\{w+\ell+1, n\}$, gdzie k to indeks kolumny, a w to indeks wiersza.

W implementacji skorzystałem z mojej struktury `SparseBlockMatrix` do przechowywania macierzy. To po prostu słownik, którego kluczem są współrzędne elementu, a wartością wartość elementu. W słowniku nie przechowuję zer, co optymalizuje zużycie pamięci.

Algorytm 1 Eliminacja Gaussa dla macierzy A

Dane: Macierz A, wektor prawych stron b, rozmiar macierzy n, rozmiar bloku ℓ
Wynik: Wektor rozwiązań x

▷ Sprowadzenie do macierzy górnotrójkątnej

for $k = 1$ to $n - 1$ **do**
  $indeks\_ostatniego\_wiersza \leftarrow \min(k + \ell + 1, n)$
  $indeks\_ostatniej\_kolumny \leftarrow \min(k + \ell, n)$
  for $i = k + 1$ to $indeks\_ostatniego\_wiersza$ **do**
    $wspolczynnik \leftarrow A[i, k]/A[k, k]$
    for $j = k$ to $indeks\_ostatniej\_kolumny$ **do**
      $A[i, j] \leftarrow A[i, j] - wspolczynnik \cdot A[k, j]$
    end for
    $b[i] \leftarrow b[i] - wspolczynnik \cdot b[k]$
  end for
end for

▷ Rozwiązanie układu równań

$x \leftarrow$ wektor zer wielkości n
for $k = n$ to 1 step -1 **do**
  $indeks\_ostatniej\_kolumny \leftarrow \min(k + \ell + 1, n)$
  $x[k] \leftarrow b[k]$
  for $i = k + 1$ to $indeks\_ostatniej\_kolumny$ **do**
    $x[k] \leftarrow x[k] - A[k, i] \cdot x[i]$
  end for
  $x[k] \leftarrow x[k]/A[k, k]$
end for
return x

2.3 Metoda eliminacji Gaussa z częściowym wyborem elementu głównego

2.3.1 Opis metody

Podstawowa metoda eliminacji Gaussa ma jeden poważny problem w kontekście obliczeń numerycznych. Mianowicie, jeśli elementy na przekątnej są zbyt bliskie zeru, to prowadzi to do bardzo dużych błędów, jako że w algorytmie występuje dzielenie przez te elementy. Tutaj z pomocą przychodzi nam zmodyfikowana eliminacja Gaussa z wyborem elementu głównego.

Element główny to termin odnoszący się do wartości, którą wybieramy jako punkt odniesienia w procesie zerowania macierzy. Służy do zerowania pozostałych wartości w danej kolumnie. W poprzedniej wersji algorytmu elementem głównym był zawsze $a_{k,k}$.

W wersji z częściowym wyborem elementu głównego będziemy chcieli poprzestawiać wiersze w taki sposób, aby na przekątną trafiły największe elementy. Będzie to jedyna większa różnica w stosunku do poprzedniej wersji algorytmu. Element główny będzie znajdował się w wierszu w, który zostanie

zamieniony z aktualnym wierszem. Wiersz w spełnia następujący warunek:

$$|a_{w,k}| = \max_{k \leqslant i \leqslant n} |a_{i,k}|$$

2.3.2 Złożoność

Jedyną różnicą mogącą wpłynąć na złożoność w stosunku do poprzedniej wersji algorytmu jest potrzeba znalezienia elementu głównego. Koszt tej operacji wynosi $O(n)$, więc nowy algorytm także jest liniowy.

2.3.3 Pseudokod

Algorytm 2 Eliminacja Gaussa z częściowym wyborem elementu głównego dla macierzy A

Dane: Macierz A, wektor prawych stron b, rozmiar macierzy n, rozmiar bloku ℓ
Wynik: Wektor rozwiązań x

▷ Sprowadzenie do macierzy górnotrójkątnej

$p \leftarrow$ wektor [1, 2, ..., n]
for $k = 1$ to $n - 1$ **do**
    $indeks\_ostatniego\_wiersza \leftarrow \min(k + \ell + 1, n)$
    $indeks\_ostatniej\_kolumny \leftarrow \min(k + 2 \cdot \ell, n)$
    ▷ Znalezienie wiersza w z elementem głównym
    $w \leftarrow k$
    $max\_wartosc \leftarrow |A[p[k], k]|$
    for $i = k + 1$ to $indeks\_ostatniego\_wiersza$ **do**
        $aktualna\_wartosc \leftarrow |A[p[i], k]|$
        if $aktualna\_wartosc > max\_wartosc$ **then**
            $max\_wartosc \leftarrow aktualna\_value$
            $w \leftarrow i$
        end if
    end for
    ▷ Zamiana znalezionego wiersza z aktualnym wierszem k
    $p[k], p[w] \leftarrow p[w], p[k]$
    ▷ Zerowanie analogicznie jak w podstawowej wersji
    for $i = k + 1$ to $indeks\_ostatniego\_wiersza$ **do**
        $wspolczynnik \leftarrow A[p[i], k]/A[p[k], k]$
        for $j = k$ to $indeks\_ostatniej\_kolumny$ **do**
            $A[p[i], j] \leftarrow A[p[i], j] - wspolczynnik \cdot A[p[k], j]$
        end for
        $b[p[i]] \leftarrow b[p[i]] - wspolczynnik \cdot b[p[k]]$
    end for
end for
▷ Rozwiązanie układu równań analogicznie jak w podstawowej wersji
$x \leftarrow$ wektor zer wielkości n
for $k = n$ to 1 step -1 **do**
    $indeks\_ostatniej\_kolumny \leftarrow \min(k + 2 \cdot \ell, n)$
    $x[k] \leftarrow b[p[k]]$
    for $i = k + 1$ to $indeks\_ostatniej\_kolumny$ **do**
        $x[k] \leftarrow x[k] - A[p[k], i] \cdot x[i]$
    end for
    $x[k] \leftarrow x[k]/A[p[k], k]$
end for
return x

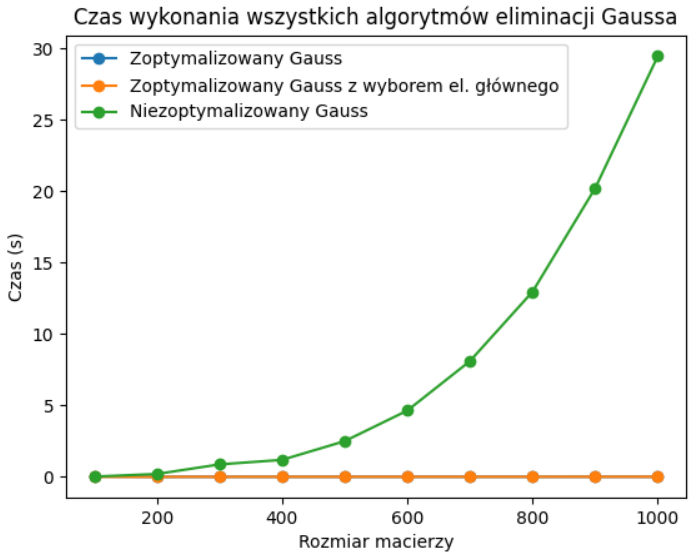
2.4 Eksperyment

Przeprowadziłem eksperyment, który miał na celu sprawdzenie szybkości działania oraz obciążenia pamięciowego algorytmów. Czas w sekundach oraz ilość zajętych bajtów obliczyłem za pomocą makra *@timed* z Julii. Wykresy generowałem za pomocą języka Python.

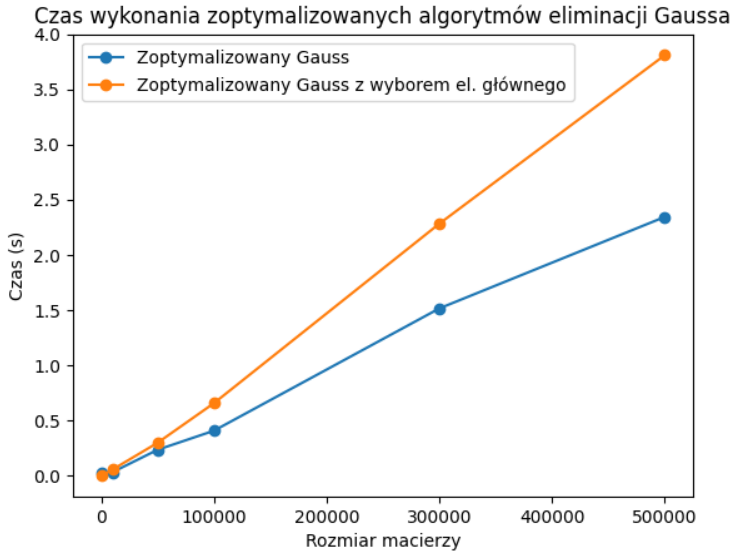
Najpierw porównałem ze sobą zoptymalizowane wersje eliminacji Gaussa wraz z wersją niezoptymalizowaną. Ten eksperyment przeprowadziłem dla macierzy wielkości $100, 200, ..., 1000$ generowanych przez funkcję `matcond`.

Potem porównałem ze sobą wersje zoptymalizowane na macierzach testowych.

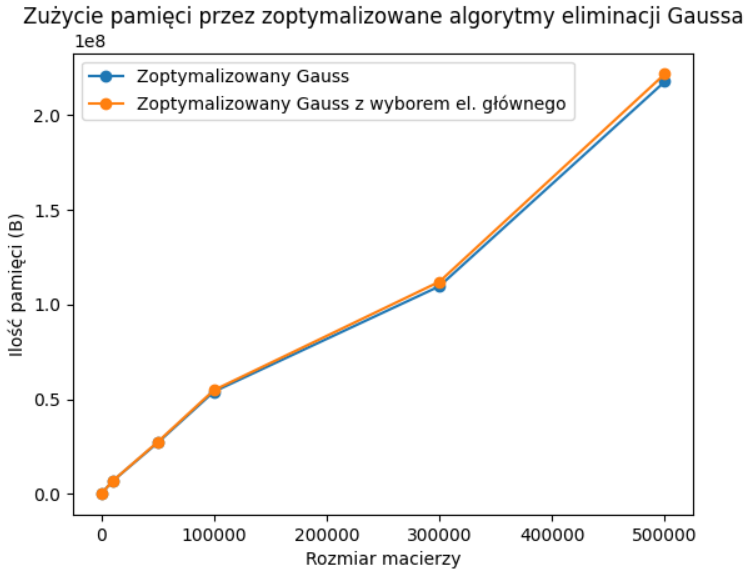
2.5 Wyniki eksperymentu



Rysunek 1: Czas wykonania wszystkich wersji eliminacji Gaussa w sekundach na losowych macierzach wygenerowanych przez funkcję `matcond`.



Rysunek 2: Czas wykonania zoptymalizowanych wersji eliminacji Gaussa w sekundach na macierzach testowych.



Rysunek 3: Zużycie pamięci przez zoptymalizowane wersje eliminacji Gaussa w bajtach na macierzach testowych.

2.6 Obserwacje i wnioski

Tak jak się spodziewaliśmy, zoptymalizowane wersje elimiacji Gaussa działają znacznie szybciej od wersji niedostosowanej pod zadane macierze. Faktycznie działają one w czasie liniowym. Wersja bez wyboru elementu głównego jest szybsza, ale trzeba pamiętać o jej ograniczeniach w kontekście obliczeń numerycznych. Na wykresach widać, że wersja z częściowym wyborem elementu głównego zużywa odrobinę więcej pamięci, jako że musi jeszcze przechować wektor permutacji.

Poczyniona w tym zadaniu optymalizacja była ogromna - złożoność została ograniczona z $O(n^3)$ do $O(n)$. Co ciekawe, zoptymalizowana wersja eliminacji Gaussa była bardzo zbliżona do oryginalnej. Wystraczyło ograniczyć liczbę elementów do wyzerowania, co świetnie wpłynęło na osiągi.

Wniosek z tego taki, że należy poświęcać czas na dokładną analizę problemu i tego z jakimi danymi pracujemy. To zadanie dobrze pokazało, że czasami duże optymalizacje da się poczynić małym kosztem.