

Współbieżny algorytm eliminacji Gaussa-Jordana

Szymon Żuk

December 15, 2025

1 Część teoretyczna

W części teoretycznej przyjąłem numerację elementów macierzy od zera, ponieważ ułatwia to indeksowanie w późniejszej implementacji algorytmu. Wprowadziłem też pivoting, co poprawia stabilność numeryczną algorytmu i rozwiązuje problem dzielenia przez zero.

1.1 Definicja niepodzielnych zadań

Niech $M \in \mathbb{R}^{n \times (n+1)}$ będzie macierzą rozszerzoną (z wektorem wyrazów wolnych).

- A_i — wybór pivota w kolumnie i (element o największej wartości bezwzględnej spośród $M_{k,i}$ dla $k \geq i$)

$$p_i = \operatorname{argmax}_{k \in \{i, i+1, \dots, n-1\}} |M_{k,i}|$$

- $B_{i,j,k}$ — zamiana elementów $M_{i,j}$ i $M_{k,j}$:

$$\operatorname{swap}(M_{i,j}, M_{k,j})$$

- $C_{i,k}$ — obliczenie mnożnika eliminacji dla wiersza k :

$$m_{k,i} = \frac{M_{k,i}}{M_{i,i}}$$

- $D_{i,j,k}$ — obliczenie składnika do odejmowania dla elementu $M_{k,j}$:

$$d_{i,j,k} = M_{i,j} \cdot m_{k,i}$$

- $E_{i,j}$ — normalizacja elementu j wiersza i :

$$M_{i,j} = \frac{M_{i,j}}{M_{i,i}}$$

- $F_{i,j,k}$ — odejmowanie składnika od elementu $M_{k,j}$:

$$M_{k,j} = M_{k,j} - d_{i,j,k}$$

1.2 Algorytm sekwencyjny

Algorytm składa się z dwóch etapów: eliminacji dolnej części macierzy oraz eliminacji górnej części macierzy.

1.2.1 Faza pierwsza

Faza pierwsza polega na zerowaniu elementów poniżej przekątnej i normalizacji wierszy. Algorytm przetwarza kolejne wiersze $i = 0, 1, \dots, n-2$. Dla ustalonego etapu i wykonywane są następujące zadania:

- A_i
- $B_{i,j,p_i}; j = i, i+1, \dots, n$
- $C_{i,k}; k = i, i+1, \dots, n-1$
- $D_{i,j,k}; j = i, i+1, \dots, n; k = i+1, i+2, \dots, n-1$
- $E_{i,j}; j = i, i+1, \dots, n$
- $F_{i,j,k}; j = i, i+1, \dots, n; k = i+1, i+2, \dots, n-1$

Następnie wykonywana jest normalizacja ostatniego wiersza:

$$E_{n-1,n-1}, E_{n-1,n}$$

1.2.2 Faza druga

Faza druga polega na zerowaniu elementów powyżej przekątnej. Wiersze przetwarzane są w kolejności $i = n-1, n-2, \dots, 1$. Dla ustalonego etapu i wykonywane są następujące zadania:

- $C_{i,k}; k = i-1, i-2, \dots, 0$
- $D_{i,i,k}; k = i-1, i-2, \dots, 0$
- $D_{i,n,k}; k = i-1, i-2, \dots, 0$
- $F_{i,i,k}; k = i-1, i-2, \dots, 0$
- $F_{i,n,k}; k = i-1, i-2, \dots, 0$

1.3 Alfabet zadań

Definiuję pomocnicze zbiory indeksów macierzy:

$$I = \{0, \dots, n-1\}, \quad I^\downarrow = \{0, \dots, n-2\}, \quad I^\uparrow = \{1, \dots, n-1\}, \quad J = \{0, \dots, n\}.$$

$$J_{\geq i} = \{j \in J \mid j \geq i\}, \quad K_{> i} = \{k \in I \mid k > i\}, \quad K_{< i} = \{k \in I \mid k < i\}.$$

Zbiór zadań Σ rozbiłem na podzbiory odpowiadające zakresom indeksów wynikającym z przebiegu algorytmu:

$$\begin{aligned}\Sigma &= \Sigma_A \cup \Sigma_B \cup \Sigma_C \cup \Sigma_D \cup \Sigma_E \cup \Sigma_F \cup \Sigma_G \\ \Sigma_A &= \{A_i \mid i \in I\}, \\ \Sigma_B &= \{B_{i,j,p} \mid i \in I, j \in J_{\geq i}\}, \\ \Sigma_C &= \{C_{i,k} \mid i \in I, k \in K_{>i} \cup K_{<i}\}, \\ \Sigma_D &= \{D_{i,j,k} \mid i \in I^\downarrow, j \in J_{\geq i}, k \in K_{>i}\} \cup \{D_{i,j,k} \mid i \in I^\uparrow, j \in \{i, n\}, k \in K_{<i}\}, \\ \Sigma_E &= \{E_{i,j} \mid i \in I, j \in J, j \geq i\}, \\ \Sigma_F &= \{F_{i,j,k} \mid i \in I^\downarrow, j \in J_{\geq i}, k \in K_{>i}\} \cup \{F_{i,j,k} \mid i \in I^\uparrow, j \in \{i, n\}, k \in K_{<i}\}\end{aligned}$$

1.4 Relacja zależności

Relację rozbiłem na cztery grupy podzbiorów.

1.4.1 Faza pierwsza — zależności w obrębie etapu

Dla ustalonego $i = 0, 1, \dots, n-2$ definiuję relację

$$\begin{aligned}D_i^\downarrow &= \{(A_i, B_{i,j,p}) \mid j \in J_{\geq i}\} \\ &\cup \{(B_{i,i,p}, C_{i,k}) \mid k \in K_{>i}\} \\ &\cup \{(C_{i,k}, D_{i,j,k}) \mid j \in J_{\geq i}, k \in K_{>i}\} \\ &\cup \{(D_{i,j,k}, E_{i,j,k}) \mid j \in J_{\geq i}, k \in K_{>i}\} \\ &\cup \{(D_{i,j,k}, F_{i,j,k}) \mid j \in J_{\geq i}, k \in K_{>i}\}\end{aligned}$$

1.4.2 Faza pierwsza — zależności pomiędzy etapami

Dla ustalonego $i = 0, 1, \dots, n-2$ definiuję relację

$$\begin{aligned}D_{i,i+1}^\downarrow &= \{(F_{i,i+1,k}, A_{i+1}) \mid k \in K_{>i}\} \\ &\cup \{(F_{i,j,i+1}, D_{i+1,j,k}) \mid j \in J_{>i}, k \in K_{>i}\} \\ &\cup \{(F_{i,j,k}, F_{i+1,j,k}) \mid j \in J_{>i}, k \in K_{>i+1}\}\end{aligned}$$

Definiuję również relację zależności między przedostatnim i ostatnim etapem fazy pierwszej:

$$D_{n-2,n-1}^\downarrow = \{(F_{n-2,n-1,n-1}, E_{n-1,n-1}), (F_{n-2,n,n-1}, E_{n-1,n})\}$$

Oraz relację zależności między ostatnim etapem fazy pierwszej a pierwszym etapem fazy drugiej:

$$D_{n-1,n-1}^{\downarrow\uparrow} = \{(E_{n-1,n-1}, C_{n-1,k}) \mid k \in K_{<n-1}\}$$

1.4.3 Faza druga — zależności w obrębie etapu

Dla ustalonego $i = n - 1, n - 2, \dots, 1$ definiuję relację

$$D_i^\uparrow = \{(C_{i,k}, D_{i,j,k}) \mid j \in \{i, n\}, k \in K_{<i}\} \\ \cup \{(D_{i,j,k}, F_{i,j,k}) \mid j \in \{i, n\}, k \in K_{<i}\}.$$

1.4.4 Faza druga — zależności pomiędzy etapami

Dla ustalonego $i = n - 1, n - 2, \dots, 2$ definiuję relację

$$D_{i,i-1}^\uparrow = \{(F_{i,j,k}, C_{i-1,k}) \mid j \in \{i, n\}, k \in K_{<i}\}$$

1.4.5 Pełna relacja zależności

Pełna relacja zależności ma postać:

$$D = \text{sym} \left(\left\{ \left(\bigcup_{i=0}^{n-2} D_i^\downarrow \right) \cup \left(\bigcup_{i=0}^{n-2} D_{i,i+1}^\downarrow \right) \cup D_{n-1,n-1}^{\downarrow\uparrow} \cup \left(\bigcup_{i=1}^{n-1} D_i^\uparrow \right) \cup \left(\bigcup_{i=2}^{n-1} D_{i,i-1}^\uparrow \right) \right\}^+ \right) \cup I_\Sigma,$$

1.5 Graf zależności diekerta

Struktura grafu zależności Diekerta $G = (V, E)$ wynika wprost z wcześniej zdefiniowanych podzbiorów relacji zależności.

$$V = \Sigma, \quad E = \left(\bigcup_{i=0}^{n-2} D_i^\downarrow \right) \cup \left(\bigcup_{i=0}^{n-2} D_{i,i+1}^\downarrow \right) \cup D_{n-1,n-1}^{\downarrow\uparrow} \cup \left(\bigcup_{i=1}^{n-1} D_i^\uparrow \right) \cup \left(\bigcup_{i=2}^{n-1} D_{i,i-1}^\uparrow \right)$$

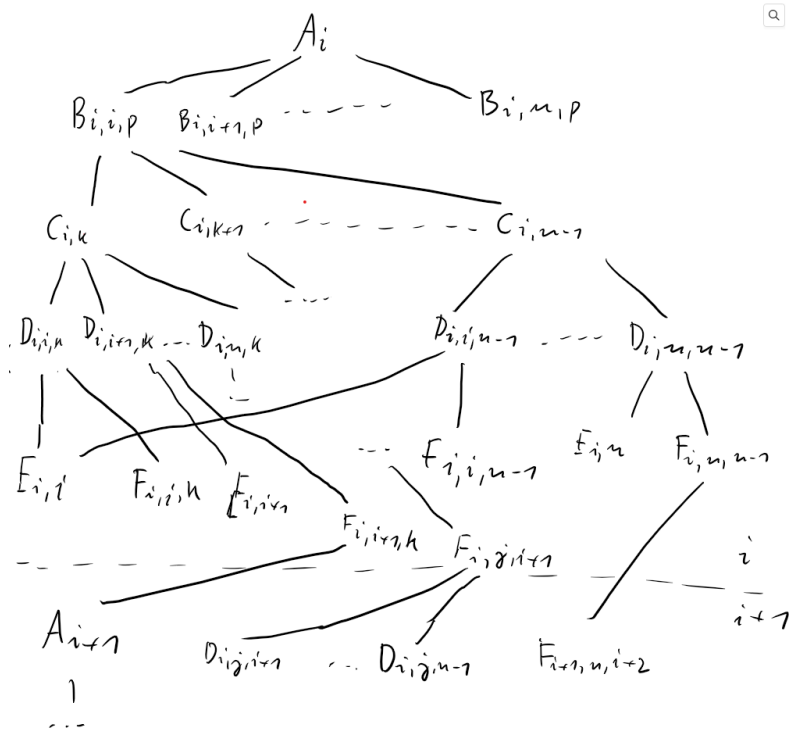


Figure 1: Fragment grafu Diekerta dla fazy pierwszej

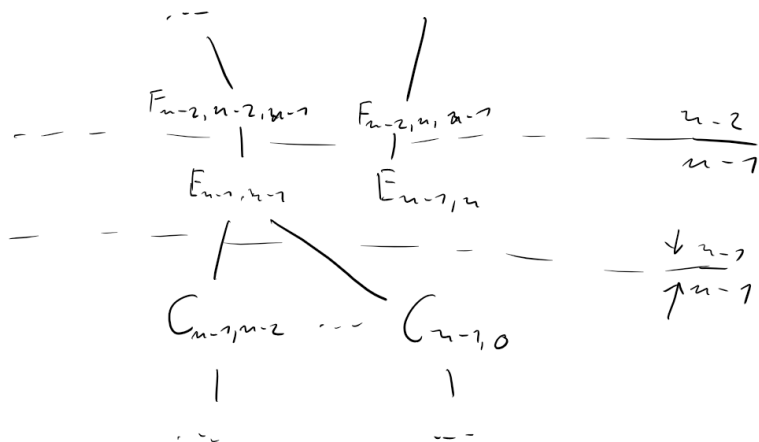


Figure 2: Fragment grafu Diekerta dla przejścia między fazami

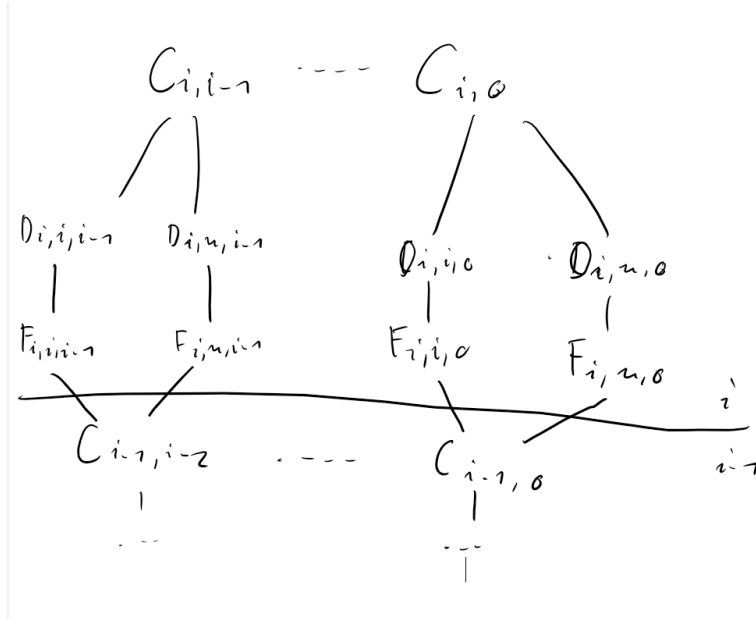


Figure 3: Fragment grafu Diekerta dla fazy drugiej

1.6 Postać normalna Foaty

Niech zbiory zadań wykonywanych równoległe w kolejnych warstwach Foaty będą oznaczone przez

$$\begin{aligned}
 \mathcal{A}_i &= \{A_i\}, & 0 \leq i \leq n-2, \\
 \mathcal{B}_i &= \{B_{i,j,p_i} \mid j \in J_{\geq i}\}, & 0 \leq i \leq n-2, \\
 \mathcal{C}_i^\downarrow &= \{C_{i,k} \mid k \in K_{>i}\}, & 0 \leq i \leq n-2, \\
 \mathcal{DE}_i^\downarrow &= \{D_{i,j,k} \mid j \in J_{\geq i}, k \in K_{>i}\} \cup \{E_{i,j} \mid j \in J_{\geq i}\}, & 0 \leq i \leq n-2, \\
 \mathcal{F}_i^\downarrow &= \{F_{i,j,k} \mid j \in J_{\geq i}, k \in K_{>i}\}, & 0 \leq i \leq n-2, \\
 \mathcal{E}_{n-1} &= \{E_{n-1,j} \mid j \in \{n-1, n\}\}, \\
 \mathcal{C}_i^\uparrow &= \{C_{i,k} \mid k \in K_{<i}\}, & 1 \leq i \leq n-1, \\
 \mathcal{D}_i^\uparrow &= \{D_{i,j,k} \mid j \in \{i, n\}, k \in K_{<i}\}, & 1 \leq i \leq n-1, \\
 \mathcal{F}_i^\uparrow &= \{F_{i,j,k} \mid j \in \{i, n\}, k \in K_{<i}\}, & 1 \leq i \leq n-1.
 \end{aligned}$$

Postać normalna Foaty przyjmuje zatem formę

$$\begin{aligned}
FNF = & [\mathcal{A}_0]_{\equiv_I^\dagger} \frown [\mathcal{B}_0]_{\equiv_I^\dagger} \frown [\mathcal{C}_0^\downarrow]_{\equiv_I^\dagger} \frown [\mathcal{DE}_0^\downarrow]_{\equiv_I^\dagger} \frown [\mathcal{F}_0^\downarrow]_{\equiv_I^\dagger} \\
& \frown [\mathcal{A}_1]_{\equiv_I^\dagger} \frown [\mathcal{B}_1]_{\equiv_I^\dagger} \frown [\mathcal{C}_1^\downarrow]_{\equiv_I^\dagger} \frown [\mathcal{DE}_1^\downarrow]_{\equiv_I^\dagger} \frown [\mathcal{F}_1^\downarrow]_{\equiv_I^\dagger} \\
& \frown \dots \\
& \frown [\mathcal{A}_{n-2}]_{\equiv_I^\dagger} \frown [\mathcal{B}_{n-2}]_{\equiv_I^\dagger} \frown [\mathcal{C}_{n-2}^\downarrow]_{\equiv_I^\dagger} \frown [\mathcal{DE}_{n-2}^\downarrow]_{\equiv_I^\dagger} \frown [\mathcal{F}_{n-2}^\downarrow]_{\equiv_I^\dagger} \\
& \frown [\mathcal{E}_{n-1}]_{\equiv_I^\dagger} \\
& \frown [\mathcal{C}_{n-1}^\uparrow]_{\equiv_I^\dagger} \frown [\mathcal{D}_{n-1}^\uparrow]_{\equiv_I^\dagger} \frown [\mathcal{F}_{n-1}^\uparrow]_{\equiv_I^\dagger} \\
& \frown \dots \\
& \frown [\mathcal{C}_1^\uparrow]_{\equiv_I^\dagger} \frown [\mathcal{D}_1^\uparrow]_{\equiv_I^\dagger} \frown [\mathcal{F}_1^\uparrow]_{\equiv_I^\dagger}.
\end{aligned}$$

2 Część implementacyjna

Implementację zorganizowałem wokół warstw Foaty opisanych w części teoretycznej. Każde zadanie (oprócz A) ma swój kernel CUDA, a kod hosta sekwencyjnie przechodzi po kolejnych warstwach Foaty.

2.1 Sterowanie na hoście

Funkcja `gauss_jordan_cuda` w pliku `gauss_jordan.cu` utrzymuje macierz w pamięci GPU i za pomocą kerneli z pliku `kernels.cu` realizuje kolejne warstwy Foaty. Wykorzystuje przy tym dodatkowy wektor na mnożniki eliminacji oraz macierz na składniki do odejmowania. Znajdywanie pivotów odbywa się na CPU. Macierz jest reprezentowana przez strukturę ‘`CudaMatrix`’ i jest zaimplementowana za pomocą 1-wymiarowej tablicy, aby przyspieszyć działanie GPU.

2.2 Kernele

Kernele są uruchamiane z liczbą wątków równą 256. Każdy kernel realizuje zadanie odpowiadające jednej literze alfabetu zadań. Kernel oblicza indeks elementu, który ma przetworzyć na podstawie indeksu wątku i bloku. Ze względu na to, że wymiary macierzy mogą nie być wielokrotnością 256 każdy kernel sprawdza czy jego indeks znajduje się w zakresie macierzy przed wykonaniem obliczeń.