Teoria współbieżności - zadanie domowe nr 3: Współbieżna eliminacja Gaussa

Wojciech Michaluk

30.12.2024

1 Opis zadania

Zadanie polega na wykonaniu dla macierzy o dowolnym rozmiarze N następujących etapów:

1. Zlokalizowanie niepodzielnych czynności wykonywanych przez algorytm, nazwanie ich oraz zbudowanie alfabetu w sensie teorii śladów,
2. Skonstruowanie relacji zależności dla alfabetu opisującego algorytm eliminacji Gaussa,
3. Przedstawienie algorytmu eliminacji Gaussa w postaci ciągu symboli alfabetu,
4. Wygenerowanie grafu zależności Diekerta,
5. Przekształcenie ciągu symboli opisujących algorytm do postaci normalnej Foaty.

Należy zaprojektować i zaimplementować współbieżny algorytm eliminacji Gaussa, w szczególności zwrócić uwagę na implementację jak najlepiej odwzorowującą graf zależności lub postać normalną Foaty. Program ma działać dla zadanych rozmiarów macierzy N.

2 Zaprojektowanie algorytmu

Aby zaprojektować algorytm dla ogólnego przypadku, najpierw zacznę rozważania od nazwania niepodzielnych czynności wykonywanych w ramach algorytmu, następnie skupię się na przypadkach szczególnych - dla $N = 2$ oraz $N = 3$ i postaram się, na podstawie analogii, uogólnić pomysł dla dowolnego N.

2.1 Niepodzielne czynności wykonywane przez algorytm

Zgodnie z omówieniem problemu na ćwiczeniach oraz na wykładzie, przyjmuję następujące oznaczenia niepodzielnych czynności wykonywanych przez algorytm - dla ogólnego przypadku:

(a) $A_{i,k}$ oznacza znalezienie mnożnika dla i-tego wiersza do odejmowania go od k-tego wiersza. Zachodzi

$$m_{k,i} = \frac{M_{k,i}}{M_{i,i}}, \tag{1}$$

gdzie M to macierz współczynników wraz z "doklejoną" kolumną wyrazów wolnych - tzw. macierz *uzupełniona* układu - przykładowo, $M_{k,i}$ oznacza wyraz z k-tego wiersza i i-tej kolumny tej macierzy, natomiast $m_{k,i}$ to szukany mnożnik.

(b) $B_{i,j,k}$ oznacza pomnożenie j-tego elementu i-tego wiersza przez mnożnik - do odejmowania od k-tego wiersza. Dla szukanego wyniku pomnożenia $n_{k,i,j}$ mamy:

$$n_{k,i,j} = M_{i,j} \cdot m_{k,i}, \tag{2}$$

gdzie oznaczenia są jak powyżej.

(c) $C_{i,j,k}$ oznacza odjęcie przemnożonego j-tego elementu i-tego wiersza od k-tego wiersza, czyli zachodzi:

$$M_{k,j} = M_{k,j} - n_{k,i,j}, \tag{3}$$

gdzie oznaczenia są jak powyżej.

2.2 Omówienie przypadku $N = 2$

Układ równań jest postaci:

$$\begin{bmatrix} M_{11} & M_{12} \\ M_{21} & M_{22} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \end{bmatrix}.$$

Macierz uzupełniona układu:

$$\left[\begin{array}{cc|c} M_{11} & M_{12} & M_{13}(= y_1) \\ M_{21} & M_{22} & M_{23}(= y_2) \end{array}\right].$$

Czynności, jakie należy wykonać:

- Odjąć od drugiego wiersza odpowiednio przemnożony pierwszy wiersz - są to kolejno zadania: $A_{1,2}, B_{1,1,2}, C_{1,1,2}, B_{1,2,2}, C_{1,2,2}, B_{1,3,2}, C_{1,3,2}$

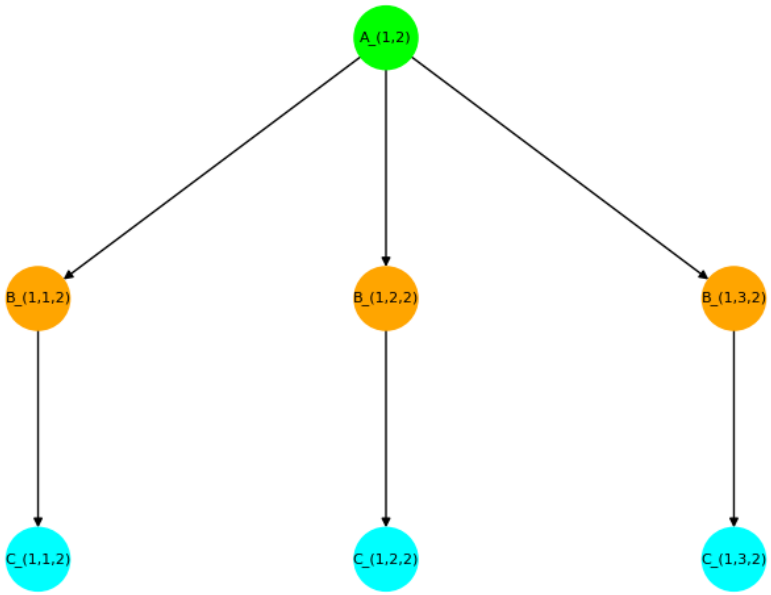
Zatem alfabet w sensie teorii śladów to $\Sigma = \{A_{1,2}, B_{1,1,2}, C_{1,1,2}, B_{1,2,2}, C_{1,2,2}, B_{1,3,2}, C_{1,3,2}\}$.
Relacja zależności:

$$D = \mathrm{sym}\{\{(A_{1,2}, B_{1,1,2}), (A_{1,2}, B_{1,2,2}), (A_{1,2}, B_{1,3,2}), (B_{1,1,2}, C_{1,1,2}), (B_{1,2,2}, C_{1,2,2}), (B_{1,3,2}, C_{1,3,2})\}^{+}\} \cup I_{\Sigma},$$

gdzie sym oznacza uwzględnienie par symetrycznych (wynika to z symetryczności relacji zależności), $^+$ oznacza domknięcie przechodnie zbioru, natomiast I_Σ to "przekątna" relacji, czyli uwzględnienie par z dwoma takimi samymi elementami (wynika to ze zwrotności relacji). Zatem w zbiorze D powyżej wyszczególniłem tylko bezpośrednie zależności. Jest to także przydatne przy wyznaczaniu grafu zależności Diekerta (a konkretnie diagramu Hassego, czyli kiedy usuwamy krótsze ścieżki w grafie skierowanym).
Przykładowy ślad wykonania algorytmu - przedstawienie algorytmu w postaci ciągu symboli alfabetu:

$$w = A_{1,2}B_{1,1,2}C_{1,1,2}B_{1,2,2}C_{1,2,2}B_{1,3,2}C_{1,3,2}.$$

Graf zależności Diekerta można wyznaczyć na podstawie relacji zależności (biorąc bezpośrednie zależności), ten poniżej został wygenerowany przez program (więc już po etapie implementacji), natomiast dla zachowania spójności rozważań umieszczam go w tym miejscu.



Rys. 1: Graf zależności Diekerta dla $N = 2$, z kolorowaniem

Jak można zauważyć na rysunku, wierzchołki grafu (operacje) są nieco inaczej podpisane niż w dokumencie, ale intuicyjnie, np. **C_(1,1,2)** odpowiada $C_{1,1,2}$.
Postać normalna Foaty (niejako klasy Foaty zostały zaznaczone poprzez kolorowanie grafu powyżej):

$$FNF([w]) = [A_{1,2}][B_{1,1,2}B_{1,2,2}B_{1,3,2}][C_{1,1,2}C_{1,2,2}C_{1,3,2}].$$

2.3 Omówienie przypadku $N = 3$

Ten przypadek został szczegółowo omówiony na wykładzie i przedstawiony na ćwiczeniach, natomiast przytoczę tu to rozumowanie, żeby łatwiej dokonać analizy tego przypadku i dokonać próby uogólnienia pomysłu.
Układ równań jest opisany analogicznie jak dla $N = 2$, macierz uzupełniona tego układu to:

$$\left[\begin{array}{ccc|c} M_{11} & M_{12} & M_{13} & M_{14}(=y_1) \\ M_{21} & M_{22} & M_{23} & M_{24}(=y_2) \\ M_{31} & M_{32} & M_{33} & M_{34}(=y_3) \end{array}\right].$$

Czynności, jakie należy wykonać:

- Odjąć od drugiego wiersza odpowiednio przemnożony pierwszy wiersz - są to kolejno zadania: $A_{1,2}, B_{1,1,2}, C_{1,1,2}, B_{1,2,2}, C_{1,2,2}, B_{1,3,2}, C_{1,3,2}, B_{1,4,2}, C_{1,4,2}$
- Odjąć od trzeciego wiersza odpowiednio przemnożony pierwszy wiersz - są to kolejno zadania: $A_{1,3}, B_{1,1,3}, C_{1,1,3}, B_{1,2,3}, C_{1,2,3}, B_{1,3,3}, C_{1,3,3}, B_{1,4,3}, C_{1,4,3}$
- Odjąć od trzeciego wiersza odpowiednio przemnożony drugi wiersz - są to kolejno zadania: $A_{2,3}, B_{2,2,3}, C_{2,2,3}, B_{2,3,3}, C_{2,3,3}, B_{2,4,3}, C_{2,4,3}$

Zatem alfabet w sensie teorii śladów to:

$$\begin{aligned}\Sigma = \{&A_{1,2}, B_{1,1,2}, C_{1,1,2}, B_{1,2,2}, C_{1,2,2}, B_{1,3,2}, C_{1,3,2}, B_{1,4,2}, C_{1,4,2},\\ &A_{1,3}, B_{1,1,3}, C_{1,1,3}, B_{1,2,3}, C_{1,2,3}, B_{1,3,3}, C_{1,3,3}, B_{1,4,3}, C_{1,4,3},\\ &A_{2,3}, B_{2,2,3}, C_{2,2,3}, B_{2,3,3}, C_{2,3,3}, B_{2,4,3}, C_{2,4,3}\}.\end{aligned}$$

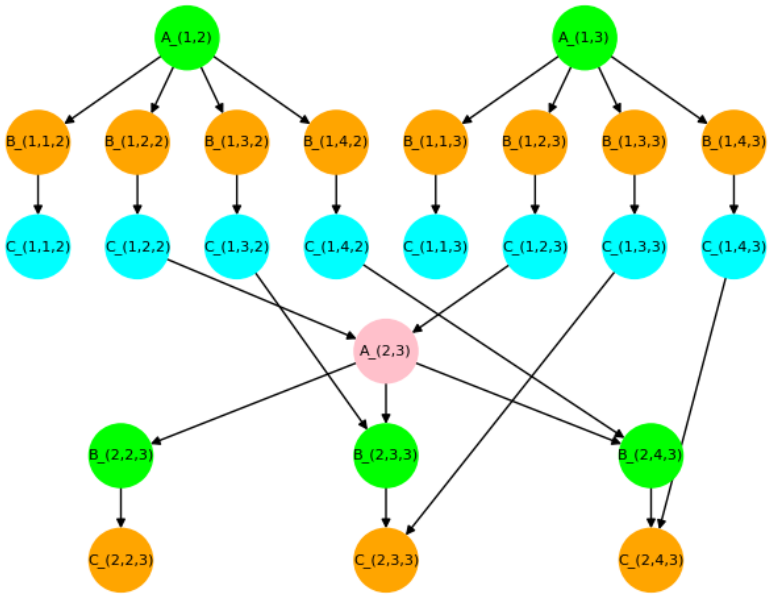
Relacja zależności:

$$\begin{aligned}D = \mathrm{sym}\{\{&(A_{1,2}, B_{1,1,2}), (A_{1,2}, B_{1,2,2}), (A_{1,2}, B_{1,3,2}), (A_{1,2}, B_{1,4,2}),\\ &(B_{1,1,2}, C_{1,1,2}), (B_{1,2,2}, C_{1,2,2}), (B_{1,3,2}, C_{1,3,2}), (B_{1,4,2}, C_{1,4,2}),\\ &(A_{1,3}, B_{1,1,3}), (A_{1,3}, B_{1,2,3}), (A_{1,3}, B_{1,3,3}), (A_{1,3}, B_{1,4,3}),\\ &(B_{1,1,3}, C_{1,1,3}), (B_{1,2,3}, C_{1,2,3}), (B_{1,3,3}, C_{1,3,3}), (B_{1,4,3}, C_{1,4,3}),\\ &(A_{2,3}, B_{2,2,3}), (A_{2,3}, B_{2,3,3}), (A_{2,3}, B_{2,4,3}),\\ &(B_{2,2,3}, C_{2,2,3}), (B_{2,3,3}, C_{2,3,3}), (B_{2,4,3}, C_{2,4,3}),\\ &(C_{1,2,2}, A_{2,3}), (C_{1,2,3}, A_{2,3}), (C_{1,3,2}, B_{2,3,3}), (C_{1,3,3}, C_{2,3,3}), (C_{1,4,2}, B_{2,4,3}), (C_{1,4,3}, C_{2,4,3})\}^+\} \cup I_\Sigma,\end{aligned}$$

gdzie oznaczenia są analogiczne jak dla przypadku $N = 2$.
Ślad wykonania algorytmu:

$$\begin{aligned}w = &A_{1,2}B_{1,1,2}C_{1,1,2}B_{1,2,2}C_{1,2,2}B_{1,3,2}C_{1,3,2}B_{1,4,2}C_{1,4,2}\\ &A_{1,3}B_{1,1,3}C_{1,1,3}B_{1,2,3}C_{1,2,3}B_{1,3,3}C_{1,3,3}B_{1,4,3}C_{1,4,3}\\ &A_{2,3}B_{2,2,3}C_{2,2,3}B_{2,3,3}C_{2,3,3}B_{2,4,3}C_{2,4,3}.\end{aligned}$$

Graf zależności Diekerta:



Rys. 2: Graf zależności Diekerta dla $N = 3$, z kolorowaniem

Postać normalna Foaty:

$$\begin{aligned} FNF([w]) =& [A_{1,2}A_{1,3}][B_{1,1,2}B_{1,2,2}B_{1,3,2}B_{1,4,2}B_{1,1,3}B_{1,2,3}B_{1,3,3}B_{1,4,3}] \\ & [C_{1,1,2}C_{1,2,2}C_{1,3,2}C_{1,4,2}C_{1,1,3}C_{1,2,3}C_{1,3,3}C_{1,4,3}][A_{2,3}] \\ & [B_{2,2,3}B_{2,3,3}B_{2,4,3}][C_{2,2,3}C_{2,3,3}C_{2,4,3}]. \end{aligned}$$

2.4 Wnioski ze szczególnych przypadków i próba uogólnienia

Przypadek dla $N = 2$ jest bardzo prosty, występują w nim zależności jedynie od A do B "w danym wierszu" oraz analogicznie od B do C. Z kolei przypadek dla $N = 3$ pozwala wyciągnąć więcej wniosków na temat algorytmu - poza wspomnianymi powyżej zależnościami, które również zachodzą tutaj (**jak i dla każdej wartości** N**!**), mamy także zależności "pomiędzy" wierszami - co, jeśli przypomnimy sobie znaczenie zdefiniowanych operacji A, B i C - jak najbardziej ma sens, bo np. przy ustaleniu mnożnika, kiedy chcemy odjąć drugi wiersz od trzeciego (zadanie $A_{2,3}$), poprzednie operacje, w których odejmowaliśmy przemnożony pierwszy wiersz od drugiego i trzeciego (zadania odpowiednio $C_{1,2,2}$ i $C_{1,2,3}$) mają wpływ na używane do tej operacji elementy - zmieniają je, przez co muszą zostać wykonane wcześniej.
Patrząc dokładnie na grafy zależności dla przeanalizowanych szczególnych przypadków, można próbować dokonać pewnego rodzaju "indukcji" (oczywiście nie jest to sformalizowane), mianowicie częścią grafu dla $N = 3$ jest graf dla $N = 2$, gdybyśmy patrzyli na odjęcie drugiego wiersza od trzeciego (oznaczenia są inne, ale analogiczne). Dodane są operacje odejmowania pierwszego wiersza od pozostałych i wspomniane odpowiednie zależności pomiędzy operacjami działającymi na różnych parach wierszy.
Spróbujmy podać ogólną postać odpowiednio alfabetu, relacji zależności, śladu wykonania algorytmu, budowy grafu zależności Diekerta (jakie połączenia - skierowane krawędzie - występują pomiędzy jakimi wierzchołkami, czyli zadaniami) oraz postaci normalnej Foaty - jak wyglądają klasy Foaty **dla dowolnego** N.
W poniższych rozważaniach głównie bazuję na przypadku $N = 3$, ale obserwacje przypadku $N = 2$ również są przydatne.

2.4.1 Alfabet w sensie teorii śladów

Jako że działamy na macierzy uzupełnionej układu, to liczba wierszy wynosi N, natomiast liczba kolumn to $N + 1$. Zauważmy, że używając i-tego wiersza, działamy jedynie na wierszach poniżej ("zerujemy" w nich

odpowiednią kolumnę), czyli są to wiersze $i+1, ..., N$. Z kolei w tych wierszach działamy na kolumnach $i, ..., N+1$ (nie "ruszamy" poprzednich kolumn, bo tam już są oczekiwane 0).
Stąd alfabet jest postaci:

$$\begin{aligned}\Sigma = \{&A_{1,2}, B_{1,1,2}, C_{1,1,2}, \ldots, B_{1,N+1,2}, C_{1,N+1,2},\\ &A_{1,3}, B_{1,1,3}, C_{1,1,3}, \ldots, B_{1,N+1,3}, C_{1,N+1,3},\\ &\ldots\\ &A_{1,N}, B_{1,1,N}, C_{1,1,N}, \ldots, B_{1,N+1,N}, C_{1,N+1,N},\\ &A_{2,3}, B_{2,2,3}, C_{2,2,3}, \ldots, B_{2,N+1,3}, C_{2,N+1,3},\\ &\ldots\\ &A_{N-1,N}, B_{N-1,N-1,N}, C_{N-1,N-1,N}, B_{N-1,N,N}, C_{N-1,N,N}, B_{N-1,N+1,N}, C_{N-1,N+1,N}\}.\end{aligned}$$

2.4.2 Relacja zależności dla alfabetu

Tak jak w szczególnych przypadkach, skupimy się na bezpośrednich zależnościach, bowiem wszystkie zależności uzyskamy, uwzględniając domknięcie przechodnie zbudowanego zbioru i pary symetryczne.
Oczywistymi zależnościami są "w danym wierszu" pary postaci $(A_{i,k}, B_{i,j,k})$ - bo zadanie B korzysta z mnożnika obliczonego w zadaniu A oraz, z podobnych powodów, pary postaci $(B_{i,j,k}, C_{i,j,k})$.
Jeżeli chodzi o zależności "między wierszami", są to konkretnie zależności operacji od operacji C, które muszą być wykonane wcześniej. Rozpatrzmy to "parami", dla danego i oraz k i k', $k' > k$:

- $C_{i,i,k}$ oraz $C_{i,i,k'}$ - od tych operacji nie zależą żadne dalsze operacje, bo "wyzerowanej komórki" już nigdzie dalej nie używamy,
- $C_{i,i+1,k}$ oraz $C_{i,i+1,k'}$ - od tych operacji zależy operacja $A_{k,k'}$, bowiem zmieniają one oba elementy, które są w tej operacji A używane,
- $C_{i,j,k}$ dla $j = i+2, \ldots, N+1$ - od każdej z tych operacji zależy odpowiadająca operacja $B_{k,j,k'}$, bowiem zmienia ona element używany w tej operacji B,
- $C_{i,j,k'}$ dla $j = i+2, \ldots, N+1$ - od każdej z tych operacji zależy odpowiadająca operacja $C_{k,j,k'}$, z powodów jak powyżej.

Zatem relacja zależności wygląda następująco:

$$\begin{aligned}D = \mathrm{sym}\{\{&(A_{1,2}, B_{1,1,2}), \ldots, (A_{1,2}, B_{1,N+1,2}),\\ &(B_{1,1,2}, C_{1,1,2}), \ldots, (B_{1,N+1,2}, C_{1,N+1,2}),\\ &(A_{1,3}, B_{1,1,3}), \ldots, (A_{1,3}, B_{1,N+1,3}),\\ &(B_{1,1,3}, C_{1,1,3}), \ldots, (B_{1,N+1,3}, C_{1,N+1,3}),\\ &\ldots\\ &(A_{1,N}, B_{1,1,N}), \ldots, (A_{1,N}, B_{1,N+1,N}),\\ &(B_{1,1,N}, C_{1,1,N}), \ldots, (B_{1,N+1,N}, C_{1,N+1,N}),\\ &(A_{2,3}, B_{2,2,3}), \ldots, (A_{2,3}, B_{2,N+1,3}),\\ &(B_{2,2,3}, C_{2,2,3}), \ldots, (B_{2,N+1,3}, C_{2,N+1,3}),\\ &\ldots\\ &(A_{N-1,N}, B_{N-1,N-1,N}), (A_{N-1,N}, B_{N-1,N,N}), (A_{N-1,N}, B_{N-1,N+1,N}),\\ &(B_{N-1,N-1,N}, C_{N-1,N-1,N}), (B_{N-1,N,N}, C_{N-1,N,N}), (B_{N-1,N+1,N}, C_{N-1,N+1,N}),\\ &(C_{1,2,2}, A_{2,3}), (C_{1,2,3}, A_{2,3}), (C_{1,3,2}, B_{2,3,3}), (C_{1,3,3}, C_{2,3,3}), \ldots, (C_{1,N+1,2}, B_{2,N+1,3}), (C_{1,N+1,3}, C_{2,N+1,3}),\\ &\ldots\\ &(C_{N-2,N-1,N-1}, A_{N-1,N}), (C_{N-2,N-1,N}, A_{N-1,N}), (C_{N-2,N,N-1}, B_{N-1,N,N}),\\ &(C_{N-2,N,N}, C_{N-1,N,N}), (C_{N-2,N+1,N-1}, B_{N-1,N+1,N}), (C_{N-2,N+1,N}, C_{N-1,N+1,N})\}^+\} \cup I_\Sigma.\end{aligned}$$

2.4.3 Przedstawienie algorytmu w postaci ciągu symboli alfabetu

Patrząc na alfabet, możemy w prosty sposób przedstawić przykładowy ślad wykonania algorytmu:

$$\begin{aligned} w =& A_{1,2}B_{1,1,2}C_{1,1,2}\dots B_{1,N+1,2}C_{1,N+1,2} \\ & A_{1,3}B_{1,1,3}C_{1,1,3}\dots B_{1,N+1,3}C_{1,N+1,3} \\ & \dots \\ & A_{1,N}B_{1,1,N}C_{1,1,N}\dots B_{1,N+1,N}C_{1,N+1,N} \\ & A_{2,3}B_{2,2,3}C_{2,2,3}\dots B_{2,N+1,3}C_{2,N+1,3} \\ & \dots \\ & A_{N-1,N}B_{N-1,N-1,N}C_{N-1,N-1,N}B_{N-1,N,N}C_{N-1,N,N}B_{N-1,N+1,N}C_{N-1,N+1,N}. \end{aligned}$$

2.4.4 Graf zależności Diekerta

Tak jak było wspomniane wcześniej, grafy zależności są wyznaczone przez relację zależności, biorąc pod uwagę bezpośrednie zależności - czyli te, które w jednym z poprzednich punktów zostały wyszczególnione.
Dla $N = 2$ oraz $N = 3$ grafy zostały pokazane, niestety już dla $N = 4$ graf jest bardzo "skomplikowany", w efekcie dla większych N takie przedstawienie graficzne jest mało czytelne, dlatego tu ograniczam się do opisu słownego.

2.4.5 Postać normalna Foaty

Potocznie mówiąc, kolejne klasy Foaty układają się "warstwami", tzn. najpierw mamy klasę z zadaniami $A_{1,k}$ dla $k = 2, \dots, N$, później klasę z zadaniami $B_{1,j,k}$, gdzie $j = 1, \dots, N + 1$, następnie analogicznie klasę z zadaniami $C_{1,j,k}$ itd. Ogólna postać postaci normalnej Foaty:

$$\begin{aligned} FNF([w]) =& [A_{1,2}\dots A_{1,N}][B_{1,1,2}\dots B_{1,N+1,2}\dots B_{1,1,N}\dots B_{1,N+1,N}] \\ & [C_{1,1,2}\dots C_{1,N+1,2}\dots C_{1,1,N}\dots C_{1,N+1,N}] \\ & \dots \\ & [A_{N-1,N}][B_{N-1,N-1,N}B_{N-1,N,N}B_{N-1,N+1,N}][C_{N-1,N-1,N}C_{N-1,N,N}C_{N-1,N+1,N}]. \end{aligned}$$

Uzbrojony w tę wiedzę przechodzę do implementacji algorytmu.

3 Implementacja algorytmu

3.1 Struktura programu realizującego algorytm

Program jest napisany w całości w języku **Python**. Do wygenerowania grafu zależności Diekerta korzystam z bibliotek Pythona: *networkx* oraz *matplotlib*. Z kolei do zaimplementowania części wykonanywanej współbieżnie korzystam z wątków - biblioteka *threading*.
W przesłanej paczce zip znajdują się następujące pliki z kodem:

- *theo_ utils.py* - ten plik zawiera definicje funkcji pomocniczych związanych z wyznaczaniem zagadnień z części "teoretycznej", używanych w głównej części programu,
- *gauss_ utils.py* - ten plik zawiera definicje funkcji pomocniczych związanych z głównym zadaniem implementacyjnym - współbieżne wykonanie eliminacji Gaussa,
- *program.py* - główna część programu, przedstawiająca kolejne etapy obliczeń, rysowanie grafu zależności Diekerta, odpowiedzialna za wykonanie współbieżnego algorytmu eliminacji Gaussa. Wykorzystuje funkcje zdefiniowane w *theo_ utils.py* oraz *gauss_ utils.py*.

Poza tym, paczka obejmuje katalog *examples*, który zawiera przykładowe pliki testowe (z wejściem do programu, w formacie tak jak w poleceniu):

- plik *in2.txt* testuje działanie programu dla przypadku $N = 2$, został napisany ręcznie,

- plik *in3.txt* testuje działanie programu dla przypadku $N = 3$, został dostarczony do zadania (pod nazwą *in.txt*),
- pozostałe pliki testują przypadki (zgodnie z liczbą w nazwie) dla $N = 5, 10, 25, 50, 100$. Zostały wygenerowane przez podlinkowaną w poleceniu *sprawdzarkę*.

Pliki z kodem są okraszone komentarzami (przy funkcji / fragmencie kodu), które wskazują, za co jest odpowiedzialny ten kod.

3.2 Uwagi do programu

Żeby móc uruchomić program, trzeba mieć zainstalowaną jedną z nowszych wersji Pythona (np. 3.11) oraz wspomniane na początku rozdziału biblioteki.
Zgodnie ze strukturą plików w przesłanej paczce zip, przykładowe uruchomienie programu jest następujące:

```
py .\program.py examples/in2.txt
```

Jako że program oblicza i wypisuje po kolei alfabet, relację zależności, ślad wykonania algorytmu, wyświetla w oddzielnym oknie graf zależności Diekerta, podaje postać normalną Foaty i dopiero potem przechodzi do realizacji algorytmu współbieżnego eliminacji Gaussa, to dla większych N jest to nieczytelne (komunikaty są bardzo długie), więc można zakomentować odpowiednie części dotyczące wyświetlania. Również graf zależności (już dla $N = 4$) z powodu dużej ilości wierzchołków i krawędzi między nimi jest raczej nieczytelny (choć poprawny) i jego wygenerowanie może zająć więcej czasu.
Dla testowanych przypadków, idąc po kolei aż do $N = 50$ wyniki są poprawne, sprawdzarka je zatwierdza (różnice między wynikami są w zakresie przyjętej tolerancji). Niestety dla $N = 100$ wyniki są już błędne, zapewne dochodzi do kumulacji błędu numerycznego i albo od pewnego momentu (którego nie szukałem, ale dla N pomiędzy 50 a 100), albo akurat dla tego "niewygodnego" wygenerowanego przypadku następuje "eksplozja" błędu, bowiem uzyskane wyniki znacząco odbiegają od poprawnych. Ponadto, Python jest dość wolnym językiem i dla $N = 100$ program wykonywał się kilka minut (dla $N = 50$ trzeba było odczekać chwilę, ale nie było to zbytnio uciążliwe).
Z wymienionych wyżej powodów zaniechałem już dalszych testów dla wartości $N = 100$ i większych, ale wydaje mi się, że mój algorytm jest ogólnie poprawny, a te błędy wynikają z reprezentacji liczb zmiennoprzecinkowych na komputerze. W każdym razie nie ukrywam faktu zaistnienia tego problemu, jest to także część wniosków, jakie można wyciągnąć z tego zadania i stanowi dobry punkt wyjścia do rozważań i szukania powodów, czemu tak się dzieje.