Dokumentacja projektu zespołowego nr 3

Anna Ćwiklińska, Krystian Gronkowski, Ihor Malyi

Maj 2023

Spis treści

1 Treść zadania

Zaimplementować znajdowanie macierzy odwrotnej metodą Gaussa-Jordana (metoda powinna działać przynajmniej dla macierzy do stopnia 6).

2 Teoretyczny opis metody

Metoda Gaussa-Jordana jest jedną z technik rozwiązujących układy równań liniowych i znajdujących macierze odwrotne. Jej celem jest przekształcenie macierzy wejściowej do postaci macierzy jednostkowej poprzez elementarne operacje na wierszach. Główna idea metody polega na wykorzystaniu operacji elementarnych takich jak dodawanie wielokrotności jednego wiersza do innego w celu wyzerowania elementów pozostających pod i nad główną przekątną. Po wykonaniu tych operacji na macierzy wejściowej, otrzymuje się macierz jednostkową po lewej stronie, a macierz odwrotna znajduje się po prawej stronie.

Niech A będzie macierzą kwadratową stopnia n.

1. Konstruujemy macierz rozszerzoną $[A|I]$, gdzie I jest macierzą jednostkową stopnia n.

2. Dla k od 1 do n wykonujemy następujące kroki:

   (a) Jeśli element a_{kk} jest równy zero, zamieniamy wiersze k i j z poniższego kroku, gdzie j jest najmniejszym indeksem większym lub równym k taki, że $a_{jk} \neq 0$. Jeżeli taki indeks j nie istnieje, macierz A jest osobliwa, a metoda Gaussa-Jordana nie może być zastosowana.

   (b) Mnożymy k-ty wiersz przez $\frac{1}{a_{kk}}$, aby uzyskać jedynkę na przekątnej.

   (c) Dla każdego wiersza i różnego od k, od 1 do n, wykonujemy operację elementarną odejmowania wielokrotności a_{ik}-krotności k-tego wiersza od i-tego wiersza, aby wyzerować element a_{ik}.

   Otrzymamy macierz rozszerzoną w postaci $[I|B]$, gdzie B jest macierzą odwrotną do macierzy A.

Warto zauważyć, że jeśli macierz A jest osobliwa, czyli nie ma odwrotności, to nie będzie możliwe wykonanie wszystkich kroków powyższej metody, ponieważ krok (2a) wymaga istnienia elementu niezerowego w każdym kroku.

Przykład 1 *Szukamy macierzy odwrotnej do macierzy 3. stopnia:*

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 1 & 1 & 0 \\ 0 & 2 & 3 \end{bmatrix}$$

$$\left[\begin{array}{ccc|ccc} 1 & 2 & 3 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 0 \\ 0 & 2 & 3 & 0 & 0 & 1 \end{array}\right] \underset{w_2 - w_1}{\sim} \left[\begin{array}{ccc|ccc} 1 & 2 & 3 & 1 & 0 & 0 \\ 0 & -1 & -3 & -1 & 1 & 0 \\ 0 & 2 & 3 & 0 & 0 & 1 \end{array}\right]$$

$$\underset{-1w_2}{\sim} \left[\begin{array}{ccc|ccc} 1 & 2 & 3 & 1 & 0 & 0 \\ 0 & 1 & 3 & 1 & -1 & 0 \\ 0 & 2 & 3 & 0 & 0 & 1 \end{array}\right] \underset{w_3 - 2w_2}{\overset{w_1 - 2w_2}{\sim}} \left[\begin{array}{ccc|ccc} 1 & 0 & -3 & -1 & 2 & 0 \\ 0 & 1 & 3 & 1 & -1 & 0 \\ 0 & 0 & -3 & -2 & 2 & 1 \end{array}\right]$$

$$\underset{-\frac{1}{3}w_3}{\sim} \left[\begin{array}{ccc|ccc} 1 & 0 & -3 & -1 & 2 & 0 \\ 0 & 1 & 3 & 1 & -1 & 0 \\ 0 & 0 & 1 & \frac{2}{3} & -\frac{2}{3} & -\frac{1}{3} \end{array}\right] \underset{w_2 - 3w_3}{\overset{w_1 + 3w_3}{\sim}} \left[\begin{array}{ccc|ccc} 1 & 0 & 0 & 1 & 0 & -1 \\ 0 & 1 & 0 & -1 & 1 & 1 \\ 0 & 0 & 1 & \frac{2}{3} & -\frac{2}{3} & -\frac{1}{3} \end{array}\right]$$

Zatem rozwiązaniem jest macierz:

$$A^{-1} = \begin{bmatrix} 1 & 0 & -1 \\ -1 & 1 & 1 \\ \frac{2}{3} & -\frac{2}{3} & -\frac{1}{3} \end{bmatrix}$$

3 Opis programu

Celem programu jest wyznaczenie macierzy odwrotnej macierzy podanej przez użytkownika za pomocą metody Gaussa-Jordana.Program jest napisany w języku Python3. Nie korzysta z zewnętrznych bibliotek, importuje tylko bibliotekę standardową Pythona `math`, która jest używana w pliku `get_data.py` do wykonywania operacji matematycznych.

4 Opis algorytmu

1. Początek programu znajduje się w funkcji `main()`. W pierwszej kolejności użytkownik informowany jest o dokładności obliczeń.

2. Program importuje dane do macierzy przy użyciu funkcji `import_data` z pliku `getData.py`. Zaimportowane dane są przekazywane do konstruktora obiektu klasy `Matrix`.

3. Konstruktor klasy `Matrix` inicjalizuje macierz wejściową i macierz odwrotną. Macierz odwrotna jest inicjalizowana jako macierz jednostkowa.

4. Metoda `findInverseMatrix()` znajduje macierz odwrotną.

   (a) Najpierw wywoływana jest metoda `checkSwaps()`, która sprawdza, czy występują zera na diagonalnej macierzy wejściowej i w razie potrzeby zamienia wiersze za pomocą metody `swapRows()`. Jeśli nie jest możliwe ustalenie macierzy odwrotnej, program wypisuje komunikat i kończy działanie.

   (b) Następnie, przy użyciu pomocniczej metody `divideRow()`, dla każdego wiersza macierzy wejściowej dokonywany jest podział wiersza przez element diagonalny, aby przekształcić go w jedynkę.

   (c) Dla każdego wiersza różnego od aktualnego, za pomocą metody `substractRow()` aktualny wiersz jest odejmowany od pozostałych wierszy odpowiednią ilość razy, aby przekształcić ich elementy w zera.

5. W funkcji `main()` wywoływana jest metoda `printInverseMatrix()`, która wypisuje macierz odwrotną w sformatowanej formie.

5 Instrukcja użytkowania

1. Należy upewnić się, że program jest kompletny i folder główny zawiera wszystkie pliki potrzebne do uruchomienia programu, w tym plik z danymi wejściowymi.

2. W pliku `"data.txt"` należy umieścić macierz, której odwrotność chcemy obliczyć (w poprawnym formacie, opisanym w sekcji Dane wejściowe.

3. W następnej kolejności uruchamiamy główny program odpowiednim poleceniem w terminalu, np. dla środowiska Windows `py main.py`.

4. Po uruchomieniu programu zostanie wyświetlony komunikat informujący o dokładności obliczeń. Następnie, jeśli macierz odwrotna istnieje, zostanie wyświetlona w konsoli, natomiast jeśli nie istnieje, zostanie wyświetlony odpowiedni komunikat.

5. Po wyświetleniu macierzy odwrotnej lub komunikatu o niemożności obliczenia macierzy odwrotnej, program zakończy swoje działanie.

5.1 Dane wejściowe

Dane wejściowe są wczytywane z pliku tekstowego o nazwie `"data.txt`, który powinien znajdować się w tym samym katalogu, co inne pliki programu. Plik ten powinien zawierać kwadratową macierz liczb, dla której chcemy obliczyć macierz odwrotną.

- Każdy wiersz macierzy powinien być umieszczony w osobnej linii.
- Poszczególne elementy wiersza powinny być oddzielone spacją.
- Macierz musi być kwadratowa, czyli liczba wierszy powinna być równa liczbie kolumn. W przeciwnym przypadku program wyświetli odpowiedni komunikat o błędzie i zakończy działanie.
- Upewnij się, że wartości liczbowe w macierzy są poprawnie zapisane i oddzielone spacjami.
- Jeśli wartość liczby przekracza zakres, tj. jest większa niż 10^{20}, program wyświetli komunikat o błędzie i zakończy działanie.
- Jeśli w pliku `"data.txt"` wystąpią inne dane niż liczby lub jeśli nie będzie możliwe ich przekonwertowanie na liczby, program wyświetli odpowiedni komunikat o błędzie i zakończy działanie.

Przykład poprawnych danych wejściowych w pliku `"data.txt"` dla macierzy 3×3:

```
4 7 4
8 1 1
9 5 9
```

5.2 Dane wyjściowe

Po wykonaniu obliczeń, program wyświetla wynik na standardowym wyjściu (w konsoli) w postaci macierzy odwrotnej lub komunikatu informującego o niemożności obliczenia macierzy odwrotnej dla danej macierzy.

Jeśli macierz odwrotna istnieje:

- Wynik jest wyświetlany w postaci macierzy, gdzie poszczególne elementy są wypisywane w postaci liczby zmiennoprzecinkowej z dokładnością do 7 miejsc po przecinku.
- Każdy wiersz macierzy odwrotnej jest wypisywany w osobnej linii.
- Elementy w wierszach są oddzielone odstępami w celu zwiększenia czytelności macierzy i uniknięcia pomyłki przy jej odczytywaniu.

Przykład wyniku dla macierzy odwrotnej:

```
 0.4166667  -0.6666667   0.0833333
 0.0555556   0.1111111  -0.0555556
-0.3055556   0.4444444   0.1388889
```

Jeśli macierz odwrotna nie istnieje:

- Wyświetlany jest odpowiedni komunikat informujący o niemożności obliczenia macierzy odwrotnej dla danej macierzy.

Program na wyjściu może również, zamiast wyniku, informować o niepoprawności wprowadzonych danych, np. o niewłaściwym znaku w macierzy wejściowej lub o jej nieprawidłowym kształcie.

6 Struktura plików

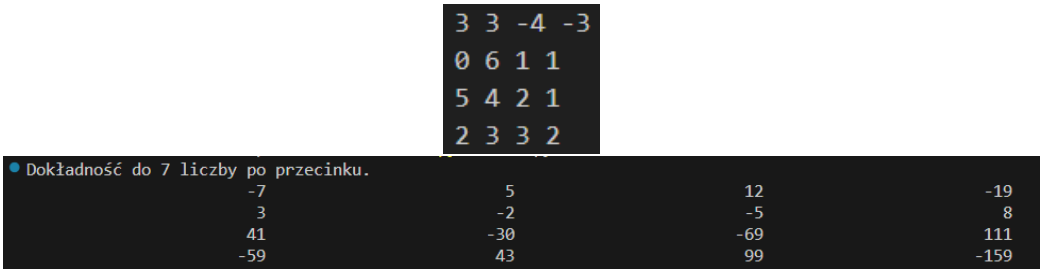
```
MacierzOdwrotna
├── data.txt
├── getData.py
├── main.py
└── matrix.py
```

6.1 Opis struktury plików

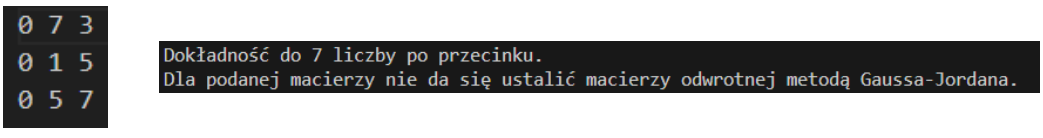
- Folder `MacierzOdwrotna` zawiera wszystkie pliki niezbędne do funkcjonowania programu.
- Plik `data.txt` jest plikiem tekstowym, który przeznaczony jest do przekazania danych wejściowych do programu. Plik ten zawiera macierz kwadratową, której odwrotność ma za zadanie wyznaczyć program.
- Plik `getData.py` odpowiada za pobranie danych z pliku `data.txt`. Zawiera funkcję `import_data()`, która otwiera plik z danymi i pobiera z niego macierz wejściową, sprawdzając przy tym poprawność wartości.
- Plik `main.py`, który odpowiada za główne wykonanie programu.
- `matrix.py` zawiera klasę `Matrix`, która implementuje funkcje niezbędne do obliczania macierzy odwrotnej:
  - `findInverseMatrix()`: Ta funkcja wykonuje główną logikę obliczania macierzy odwrotnej za pomocą metody Gaussa-Jordana. Przechodzi przez macierz i wykonuje odpowiednie operacje na wierszach, takie jak dzielenie wiersza przez element na głównej przekątnej i odejmowanie wielokrotności innych wierszy w celu wyzerowania ich elementów.
  - `checkSwaps()`: Funkcja ta sprawdza, czy konieczne są zamiany wierszy w przypadku, gdy element na głównej przekątnej jest równy zero. Przeszukuje macierz w poszukiwaniu innego wiersza, który może być zamieniony z bieżącym, aby uniknąć dzielenia przez zero. Jeśli nie można znaleźć takiego wiersza, program wyświetla komunikat informujący o niemożności obliczenia macierzy odwrotnej.

- `swapRows(swap1, swap2)`: Ta funkcja zamienia miejscami dwa wiersze macierzy. Jest wykorzystywana w przypadku, gdy konieczne jest wykonanie zamiany wierszy, aby uniknąć dzielenia przez zero podczas obliczania macierzy odwrotnej.
- `printInverseMatrix()`: Funkcja ta odpowiedzialna jest za wyświetlanie macierzy odwrotnej na standardowym wyjściu. Przechodzi przez każdy wiersz macierzy odwrotnej i formatuje jego elementy w odpowiedniej postaci. Wynik jest wyświetlany w formie czytelnej macierzy, gdzie każdy wiersz jest wypisywany w osobnej linii.
- `substractRow(subFrom, tosub, howManyTimes)`: Ta funkcja odejmuje od jednego wiersza macierzy wielokrotność innego wiersza. Jest używana w metodzie Gaussa-Jordana do wyzerowania elementów macierzy poza główną przekątną.
- `divideRow(divideFrom, divider)`: Funkcja ta dzieli elementy jednego wiersza macierzy przez określony dzielnik.

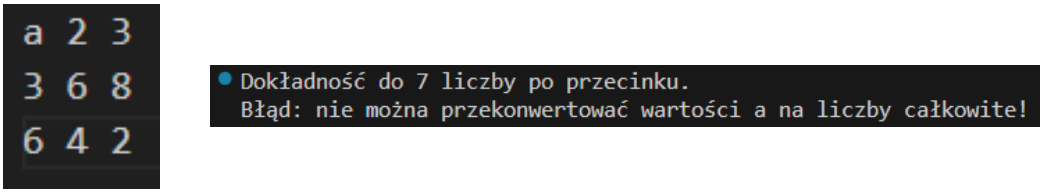
7 Raport z demonstracji



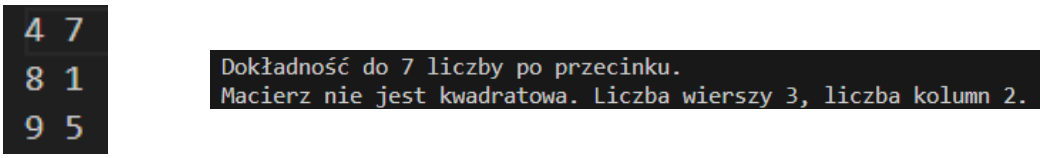
Rysunek 1: Rozwiązanie dla poprawnych danych i macierzy nieosobliwej



Rysunek 2: Komunikat w przypadku wprowadzenia macierzy osobliwej



Rysunek 3: Komunikat w przypadku niewłaściwego znaku w macierzy wejściowej



Rysunek 4: Komunikat w przypadku niewłaściwego kształtu macierzy

8 Wnioski

Metoda Gaussa-Jordana może być użyta do znalezienia macierzy odwrotnej dla macierzy kwadratowej. Jednak istnieją pewne warunki, które muszą być spełnione. Jednym z tych warunków jest to, że macierz musi mieć pełny układ liniowo niezależnych wektorów. Jeśli macierz nie posiada takiego układu, metoda nie może być użyta.

Metoda Gaussa-Jordana jest podatna na błędy zaokrągleń, zwłaszcza gdy liczby w macierzy są bliskie zeru lub bardzo duże. Może to wpływać na precyzję wyników, szczególnie gdy obliczenia są wykonywane na komputerze, który pracuje na skończonej liczbie bitów.

9 Kod programu

W kodzie programu nie jest uwzględniony plik `data.txt` ze względu na jego unikalność. Mimo, że jest on niezbędny do prawidłowego działania programu, dane w nim zawarte mogą się różnić z każdym wywołaniem, ponieważ mogą być zmieniane przez użytkownika. Struktura danych wejściowych została przedstawiona w sekcji Dane wejściowe.

9.1 get_data.py

```
import math


def import_data():
    data = []
    try:
        with open("./data.txt") as f:
            for line in f:
                line = line.strip()
                if not line:
                    continue
                values = line.split()
                row = []
                for x in range(len(values)):
                    try:
                        if abs(float(values[x])) > math.pow(10,
                        ↪  20):
                            print(
                                f"Liczba musi być <
                                ↪  {math.pow(10, 20)}. Błąd
                                ↪  przy odczycie liczby
                                ↪  {values[x]}"
                            )
                            exit()
                        row.append(float(values[x]))
                    except ValueError:
                        print(
                            f"Błąd: nie można przekonwertować
                            ↪  wartości {values[x]} na
                            ↪  liczbę!"
                        )
                        exit()
                data.append(row)
    except FileNotFoundError:
        print("Błąd: plik nie istnieje!")
        exit()
```

```
    rows_count = len(data)
    for row in data:
        if len(row) != rows_count:
            print(
                f"Macierz nie jest kwadratowa. Liczba wierszy
                ↪ {rows_count}, liczba kolumn {len(row)}."
            )
            exit()

    return data
```

9.2 main.py

```
from matrix import Matrix
from getData import import_data


def main():
    print("Dokładność do 7 liczby po przecinku.")
    matrix = Matrix(import_data())
    matrix.findInverseMatrix()
    matrix.printInverseMatrix()


if __name__ == "__main__":
    main()
```

9.3 matrix.py

```
class Matrix:
    def __init__(self, matrix) -> None:
        self.matrix = matrix
        self.inverse = [[0 for x in range(len(matrix[0]))] for
        ↪ x in range(len(matrix))]
        for x in range(len(self.inverse)):
            for y in range(len(self.inverse[x])):
                if x != y:
                    self.inverse[x][y] = 0
```

```
            else:
                self.inverse[x][y] = 1

    def findInverseMatrix(self):
        self.checkSwaps()
        for x in range(len(self.matrix)):
            if self.matrix[x][x] == 0:
                print(
                    "Dla podanej macierzy nie da się ustalić
                    ↪ macierzy odwrotnej metodą
                    ↪ Gaussa-Jordana."
                )
                exit()
            self.divideRow(x, 1 / self.matrix[x][x])
            for y in range(len(self.matrix)):
                if y != x:
                    self.substractRow(y, x, self.matrix[y][x] /
                    ↪ self.matrix[x][x])

    def checkSwaps(self):
        for x in range(len(self.matrix)):
            if self.matrix[x][x] == 0:
                for y in range(len(self.matrix)):
                    if self.matrix[y][x] != 0 and
                    ↪ self.matrix[x][y] != 0:
                        self.swapRows(x, y)
                        break
                    elif y == len(self.matrix) - 1:
                        print(
                            "Dla podanej macierzy nie da się
                            ↪ ustalić macierzy odwrotnej
                            ↪ metodą Gaussa-Jordana."
                        )
                        exit()

    def swapRows(self, swap1, swap2):
        for x in range(len(self.matrix[swap1])):
            tmp = self.matrix[swap1][x]
```

```
            tmp2 = self.inverse[swap1][x]
            self.inverse[swap1][x] = self.inverse[swap2][x]
            self.inverse[swap2][x] = tmp2
            self.matrix[swap1][x] = self.matrix[swap2][x]
            self.matrix[swap2][x] = tmp

    def printInverseMatrix(self):
        pattern = "{:>25.7g} " * len(self.inverse[0])
        for row in self.inverse:
            print(pattern.format(*row))

    def substractRow(self, subFrom, tosub, howManyTimes):
        for x in range(len(self.matrix[subFrom])):
            self.matrix[subFrom][x] -= self.matrix[tosub][x] *
            ↪  howManyTimes
            self.inverse[subFrom][x] -= self.inverse[tosub][x]
            ↪  * howManyTimes

    def divideRow(self, divideFrom, divider):
        for x in range(len(self.matrix[divideFrom])):
            self.matrix[divideFrom][x] *= divider
            self.inverse[divideFrom][x] *= divider
```

Literatura

[1] Jaruszewska-Walczak, D. Wykład z Algorytmów numerycznych.

[2] Marlewski, A. *Algebra macierzy liczbowych.*

[3] Pańczyk, B., Łukasik, E., Sikora, J. *Metody numeryczne w przykładach*