Problematyka:

- Program ma rozwiązać równania macierzowe $A_i y = b$ dla i = 1, 2 dla dwóch macierzy i wektora podanych w zadaniu:

$$\mathbf{A}_1 = \begin{pmatrix} 2.554219275 & 0.871733993 & 0.052575899 & 0.240740262 & 0.316022841 \\ 0.871733993 & 0.553460938 & -0.070921727 & 0.255463951 & 0.707334556 \\ 0.052575899 & -0.070921727 & 3.409888776 & 0.293510439 & 0.847758171 \\ 0.240740262 & 0.255463951 & 0.293510439 & 1.108336850 & -0.206925123 \\ 0.316022841 & 0.707334556 & 0.847758171 & -0.206925123 & 2.374094162 \end{pmatrix}$$

$$\mathbf{A}_2 = \begin{pmatrix} 2.645152285 & 0.544589368 & 0.009976745 & 0.327869824 & 0.424193304 \\ 0.544589368 & 1.730410927 & 0.082334875 & -0.057997220 & 0.318175706 \\ 0.009976745 & 0.082334875 & 3.429845092 & 0.252693077 & 0.797083832 \\ 0.327869824 & -0.057997220 & 0.252693077 & 1.191822050 & -0.103279098 \\ 0.424193304 & 0.318175706 & 0.797083832 & -0.103279098 & 2.502769647 \end{pmatrix}$$

$$\mathbf{b} \equiv (-0.642912346, -1.408195475, 4.595622394, -5.073473196, 2.178020609)^T$$

- Następnie ma to zrobić ponownie lecz z zaburzeniem wektora b:

$$\mathbf{A}_i \mathbf{y} = \mathbf{b} + \Delta \mathbf{b}$$

- Zaburzenie ma być wygenerowane jako losowy wektor o małej normie euklidesowej.

Wyjaśnienie programu:

- W programie wykorzystane są funkcję z dodatkowej biblioteki (numpy) aby “nie wynajdywać koła od nowa”
- Funkcję stworzone przeze mnie dla potrzeby rozwiązania problemu:
    - printOperation: przyjmuje trzy argumenty: pierwszy jest wypisywane w konsoli, kolejne argumenty są macierzą i wektorem które funkcja używa do rozwiązania równania macierzowego i wypisania wyniku

- Główna logika programu:
    - Program oblicza równanie macierzowe dla:
        - Przez zaburzeniem:
            - Macierzy A_1 i wektora b
            - Macierzy A_2 i wektora b
    - Następnie program zaburza wektor b
    - Później program ponownie oblicza równanie macierzowe dla:
        - Po zaburzeniu:
            - Macierzy A_1 i wektora b
            - Macierzy A_2 i wektora b

Wyjaśnienie problematyki:

Zacznijmy od wyjaśnienia, czym jest wrażliwość układów równań na zaburzenia danych. Równania mogą być mało podatne na zmiany w danych, co oznacza, że drobne zmiany nie wpłyną znacząco na wyniki. W przeciwnym razie mówimy, że układ równań jest źle uwarunkowany. Zrozumienie, które układy są dobrze uwarunkowane, ma kluczowe znaczenie, ponieważ wpływa to na dokładność obliczeń w systemach o ograniczonej precyzji oraz na wydajność używanych algorytmów.

Do oceny uwarunkowania układów równań wykorzystuje się współczynnik uwarunkowania K (kappa). Dla macierzy symetrycznych można go obliczyć za pomocą wzoru:

$$K = ||A|| \cdot ||A^{-1}||$$

Gdzie |A| oznacza normę macierzy symetrycznej, czyli jej największą wartość własną pod względem modułu, a $||A^{-1}||$ to norma macierzy odwrotnej do A, czyli odwrotność jej najmniejszej wartości własnej.

Im mniejszy współczynnik uwarunkowania K, tym układ równań jest lepiej uwarunkowany i bardziej odporny na zaburzenia danych. W praktyce do rozwiązywania układów równań można wykorzystać funkcje biblioteki numerycznej, np. NumPy, takie jak linalg.solve(). Ta funkcja przeprowadza faktoryzację LU z częściowym pivotingiem i oblicza rozwiązania układów równań.

Wnioski:

- Patrząc się na wyniki równań macierzowych przed i po zaburzeniu wektora b, widać że macierz A_2 jest dobrze uwarunkowana lecz macierz A_1 jest źle uwarunkowana.

Uruchamianie programu:

- make all / make NUM2 : uruchamia program

  W razie problemów można zmienić w pliku make wywołanie pythona z “python” na “python3”

Mateusz Kałwa

Wyniki programu:

```
Przed zaburzeniem:
A1y = b:
[ 0.22508493 -0.00602226  1.84183182 -5.15344244 -0.2176225 ]

A2y = b:
[ 0.57747172 -1.27378458  1.67675008 -4.8157949   0.20156347]



Po zaburzeniu:
A1y = b:
[-525.67509771 1891.99576055  248.20894002 -509.05683604 -625.80855111]

A2y = b:
[ 0.57747159 -1.27378402  1.67675012 -4.81579404  0.20156366]
```