Rozwiązywanie układu równań Ax=b dla macierzy podanej w postaci macierzy klatkowej

Maciej Chylak, grupa 1

27 grudnia 2020

Spis treści

1 Temat

1.1 Treść zadania

Rozwiązywanie układu równań liniowych Ax=b, gdzie

$$\begin{bmatrix} A_1 & 0 & 0 \\ A_2 & I & 0 \\ A_3 & A_4 & A_5 \end{bmatrix}$$

gdzie $A_i(pxp)$ A_1 i A_5 są macierzami trójdiagonalnymi, symetrycznymi i dodatnio określonymi.

Warto zauważyć, że równanie przy podanych warunkach będzie miało albo jedno rozwiązanie albo będzie ono sprzeczne. Do rozwiązania podanego układu posłużymy się metodą eliminacji Gaussa (GEPP), gdyż tak skonstruowana macierz jest macierzą nieosobliwą, a GEPP jest w stanie rozwiazać każdy z takich układów.

2 Program obliczeniowy

2.1 Opis funkcji

Program obliczeniowy składa się z kilku funkcji pomocniczych oraz jednej funkcji głównej, korzystającej z pozostałych.
Użyte w programie funkcje:

- matrix_builder - funkcja ta służy do zbudowania macierzy o wymiarze 3p x 3p odpowiadającej macierzy blokowej podanej w treści zadania; Dodatkowo sprawdza ona także, czy macierze spełniają warunki podane w poleceniu zadania;

- gauss_elimination - funkcja ta służy do rozwiązania układu równań typu Ax=b przy pomocy metody eliminacji Gaussa (GEPP), gdzie A jest macierzą kwadratową. Zwraca ona wektor rozwiązań lub komunikat, że układ jest sprzeczny.

- gauss_plus_builder - funkcja ta łączy funkcje matrix_builder oraz gauss_elimination. Pozwala na wywowałnie obu funkcji jednocześnie

- determinant - funkcja służąca do obliczania wyznacznika macierzy kwadratowej. Korzysta ona z metody Laplace'a.

- inverse_matrix - funkcja ta pozwala nam znaleźć macierz odwrotną do macierzy wejściowej. W przypadku, gdy macierz jest osobliwa, zostaje wyświetlony odpowiedni komunikat

- condition_number - funkcja ta oblicza wskaźnik uwarunkowania macierzy nieosobliwych.

- inverse_matrix - funkcja ta wyznacza macierz odwrotną do macierzy A
- test - skrypt, w którym zostały wczytane wszystkie dane wejściowe, które następnie posłużyły mi jako przykłady użycia funkcji. Dodatkowo plik ten posłużył mi do tworzenia wykresów.

2.2 Sposób obsługi programu

Programu używa się poprzez wywołanie funkcji gauss_elimination na macierzy zbudowanej poprzez użycie funkcji matrix_builder lub poprzez użycie funkcji gauss_plus_builder, która łączy obie funkcje w sobie. Prostrzymi słowami najpierw wymagane jest zbudowanie macierzy przedstawionej w treści zadania, a następnie wykonanie na niej eliminacji Gaussa (GEPP):

```
>> gauss_elimination(matrix_builder(A1,A2,A3,A4,A5), B)
```

```
>> gauss_plus_builder(A1, A2, A3, A4, A5, B)
```

gdzie $A1, A2, A3, A4, A5$ oznaczają odpowiednie macierze podane w treści zadania a B wektor wyrazów wolnych

3 Przykłady obliczeniowe

3.1 Przykład obliczeniowy nr 1

W tym przykładzie posłużymy się następującymi danymi wejściowymi:

$$A_1 = \begin{bmatrix} 4 & 2 \\ 2 & 2 \end{bmatrix}, A_2 = \begin{bmatrix} -3 & 3 \\ 5 & -2 \end{bmatrix}, A_3 = \begin{bmatrix} 4 & 1 \\ 5 & 2 \end{bmatrix},$$

$$A_4 = \begin{bmatrix} 3 & 3 \\ -5 & 2 \end{bmatrix}, A_5 = \begin{bmatrix} 5 & 2 \\ 2 & 4 \end{bmatrix}, B = \begin{bmatrix} 4 \\ 1 \\ 5 \\ 1 \\ 2 \\ 1 \end{bmatrix}$$

Wywołanie funkcji gauss_elimination oraz linsolve zwraca nam następujące wyniki:

```
>> gauss_plus_builder(A1_1, A2_1, A3_1, A4_1, A5_1, B_1)

ans =

    1.5000
   -1.0000
   12.5000
   -8.5000
  -13.1250
   25.3125

>> linsolve(matrix_builder(A1_1, A2_1, A3_1, A4_1, A5_1), B_t_1)

ans =

    1.5000
   -1.0000
   12.5000
   -8.5000
  -13.1250
   25.3125
```

Jak możemy zauważyć, nasze wyniki są niemalże dokładne. Każdy z elementów wektora x obliczony przy pomocy funkcji gauss_elimination różni się dokładnie o:

```
>> A_1_diff_prev

A_1_diff_prev =

   1.0e-12 *

   -0.0111
    0.0160
   -0.0782
    0.0870
    0.0639
   -0.1670
```

od x uzyskanego w przypadku wywołania funkcji linsolve.

3.2 Przykład obliczeniowy nr 2

W tym przykładzie posłużymy się następującymi danymi wejściowymi:

$$A_1 = \begin{bmatrix} 7 & 3 & 0 \\ 3 & 4 & 3 \\ 0 & 3 & 5 \end{bmatrix}, A_2 = \begin{bmatrix} 1 & 3 & 2 \\ -4 & 2 & 1 \\ 5 & -2 & 1 \end{bmatrix}, A_3 = \begin{bmatrix} 4 & 1 & -4 \\ 5 & 2 & 2 \\ 1 & 5 & 1 \end{bmatrix},$$

$$A_4 = \begin{bmatrix} 9 & 3 & -4 \\ 2 & -5 & 2 \\ -4 & 2 & 1 \end{bmatrix}, A_5 = \begin{bmatrix} 10 & 2 & 0 \\ 2 & 5 & 4 \\ 0 & 4 & 5 \end{bmatrix}, B = \begin{bmatrix} 4 \\ 1 \\ 2 \\ 5 \\ 2 \\ -1 \\ 5 \\ 2 \\ 1 \end{bmatrix}$$

Wywołanie funkcji gauss_elimination oraz linsolve zwraca nam następujące wyniki:

```
>> gauss_plus_builder(A1_2, A2_2, A3_2, A4_2, A5_2, B_2)

ans =

    1.4687
   -2.0937
    1.6562
    6.5000
   10.4062
  -14.1875
  -23.6464
   48.9196
  -33.5920

>> linsolve(matrix_builder(A1_2, A2_2, A3_2, A4_2, A5_2), B_t_2)

ans =

    1.4687
   -2.0937
    1.6562
    6.5000
   10.4062
  -14.1875
  -23.6464
   48.9196
  -33.5920
```

Jak możemy zauważyć, nasze wyniki są niemalże dokładne. Każdy z elementów wektora x obliczony przy pomocy funkcji gauss_elimination różni się dokładnie o:

```
A_2_diff_prev =

   1.0e-13 *

    0.0466
   -0.1066
    0.0488
    0.0622
    0.0355
    0.5329
    0.2842
   -0.6395
    0.4974
```

od x uzyskanego w przypadku wywołania funkcji linsolve.

3.3 Przykład obliczeniowy nr 3

W tym przykładzie posłużymy się następującymi danymi wejściowymi:

$$A_1 = \begin{bmatrix} 10 & 2 & 0 & 0 \\ 2 & 5 & 4 & 0 \\ 0 & 4 & 7 & 1 \\ 0 & 0 & 1 & 8 \end{bmatrix}, A_2 = \begin{bmatrix} 1 & 3 & 1 & 2 \\ -4 & 2 & 1 & -1 \\ 5 & -22 & 1 & 2 \\ 2 & 3 & 4 & -4 \end{bmatrix}, A_3 = \begin{bmatrix} 4 & 15 & 4 & -4 \\ 5 & 2 & 2 & 2 \\ 1 & 5 & 1 & -2 \\ 2 & 3 & 4 & -4 \end{bmatrix},$$

$$A_4 = \begin{bmatrix} 9 & 3 & -4 & 2 \\ 2 & -6 & -5 & 2 \\ -4 & 2 & -4 & 1 \\ 2 & 3 & 4 & -4 \end{bmatrix}, A_5 = \begin{bmatrix} 7 & 1 & 0 & 0 \\ 1 & 8 & 1 & 0 \\ 0 & 1 & 10 & 4 \\ 0 & 0 & 4 & 5 \end{bmatrix}, B = \begin{bmatrix} 4 \\ 1 \\ 2 \\ 5 \\ 2 \\ 1 \\ 5 \\ 2 \\ 1 \\ 1 \\ 2 \\ 3 \end{bmatrix}$$

Wywołanie funkcji gauss_elimination oraz linsolve zwraca nam następujące wyniki:

```
>> gauss_plus_builder(A1_3, A2_3, A3_3, A4_3, A5_3, B_3)

ans =

    0.4528
   -0.2640
    0.3536
    0.5808
    0.8240
    3.5664
   -4.5872
    2.7952
   -5.4216
   -0.0456
   -5.5944
    8.6711

>> linsolve(matrix_builder(A1_3, A2_3, A3_3, A4_3, A5_3), B_t_3)

ans =

    0.4528
   -0.2640
    0.3536
    0.5808
    0.8240
    3.5664
   -4.5872
    2.7952
   -5.4216
   -0.0456
   -5.5944
    8.6711
```

Jak możemy zauważyć, nasze wyniki są niemalże dokładne. Każdy z elementów wektora x obliczony przy pomocy funkcji gauss_elimination różni się dokładnie o:

```
A_3_diff_prev =

   1.0e-14 *

         0
    0.0167
   -0.0111
         0
   -0.0333
   -0.2220
    0.5329
   -0.0888
    0.4441
    0.0611
    0.5329
   -0.7105
```
od x uzyskanego w przypadku wywołania funkcji linsolve.

3.4 Przykład obliczeniowy nr 4

W tym przykładzie posłużymy się następującymi danymi wejściowymi:

$$A_1 = \begin{bmatrix} 10 & 72 & 0 & 0 & 0 \\ 7 & 9 & 2 & 0 & 0 \\ 0 & 2 & 7 & 1 & 0 \\ 0 & 0 & 1 & 8 & 3 \\ 0 & 0 & 0 & 3 & 10 \end{bmatrix}, A_2 = \begin{bmatrix} 4 & 15 & 4 & 2 & -4 \\ 5 & 2 & 4 & 2 & 2 \\ 1 & 2 & 5 & 1 & -2 \\ 2 & 3 & 1 & 4 & -4 \\ 4 & 2 & -4 & 1 & 2 \end{bmatrix}, A_3 = \begin{bmatrix} 1 & 3 & 1 & -5 & 2 \\ -4 & 2 & 1 & 1 & -1 \\ 5 & -22 & 1 & 2 & 2 \\ 2 & 3 & 4 & 4 & -4 \\ 4 & 2 & -4 & 1 & 2 \end{bmatrix},$$

$$A_4 = \begin{bmatrix} 9 & 3 & -4 & 4 & 2 \\ 2 & -6 & -5 & 2 & 1 \\ -4 & 2 & -4 & 1 & -2 \\ 2 & 3 & 4 & -4 & -1 \\ 4 & 2 & 1 & 5 & -4 \end{bmatrix}, A_5 = \begin{bmatrix} 9 & 1 & 0 & 0 \\ 0 \\ 1 & 8 & 1 & 0 \\ 0 \\ 0 & 1 & 8 & 4 \\ 0 \\ 0 & 0 & 4 & 9 \\ 5 \\ 0 & 0 & 0 & 5 \\ 10 \end{bmatrix}, B = \begin{bmatrix} 4 \\ 1 \\ 2 \\ -5 \\ 2 \\ 1 \\ 5 \\ 2 \\ 1 \\ 4 \\ 2 \\ 1 \\ 5 \\ 2 \\ 1 \end{bmatrix}$$

Wywołanie funkcji gauss_elimination oraz linsolve zwraca nam następujące wyniki:

```
>> gauss_plus_builder(A1_4, A2_4, A3_4, A4_4, A5_4, B_4)

ans =

    0.9201
   -0.7430
    0.6232
   -0.8765
    0.4630
    9.5770
    0.2197
    1.2522
    6.1235
    4.2491
  -12.9134
   -1.2371
    1.6224
    4.4417
   -7.4580

>> linsolve(matrix_builder(A1_4, A2_4, A3_4, A4_4, A5_4), B_t_4)

ans =

    0.9201
   -0.7430
    0.6232
   -0.8765
    0.4630
    9.5770
    0.2197
    1.2522
    6.1235
    4.2491
  -12.9134
   -1.2371
    1.6224
    4.4417
   -7.4580
```

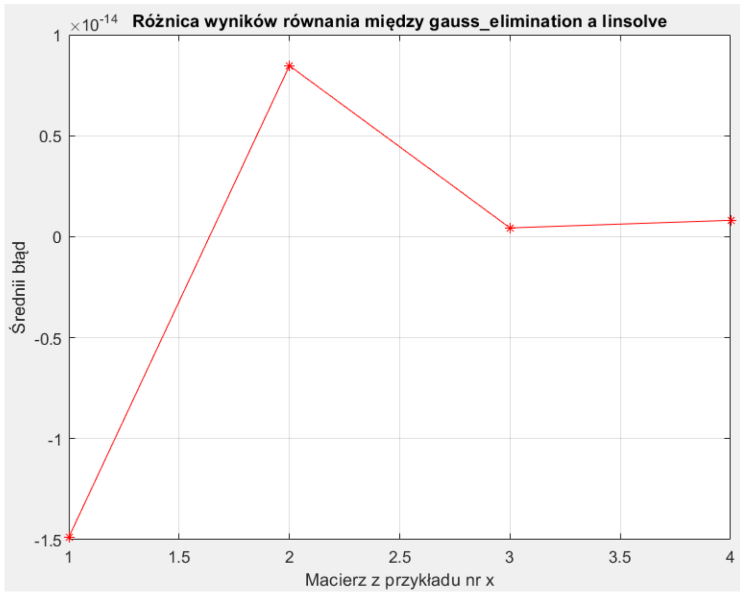
Jak możemy zauważyć, nasze wyniki są niemalże dokładne. Każdy z elementów wektora x obliczony przy pomocy funkcji gauss_elimination różni się dokładnie o:

```
A_4_diff_prev =

   1.0e-13 *

  -0.0056
   0.0067
  -0.0022
        0
  -0.0011
  -0.1066
   0.0172
   0.0400
   0.0178
  -0.0178
   0.1066
   0.0533
  -0.0044
  -0.0444
   0.0622
```

od x uzyskanego w przypadku wywołania funkcji linsolve.

3.5 Dokładność wyników

Jak mogliśmy zauważyć, wyniki działania naszej funkcji są niemalże zgodne z wynikami uzyskanymi przy pomocy funkcji linsolve. Przedstawmy nasze różnice wyników na wykresie:



Największe odchylenie względem zera jest widoczne w przypadku macierzy o wymiarze 6 x 6. Co ciekawe, im większy był wymiar macierzy blokowej, tym obliczenia wychodziły bardziej dokładne, lecz należy pamiętać, że równie ważną rolę odgrywa tutaj wskaźnik uwarunkowania macierzy.

3.6 Przykład obliczeniowy nr 5

W tym przykładzie posłużymy się następującymi danymi wejściowymi:

$$A_1 = \begin{bmatrix} -45 & 3 & 0 \\ 3 & 6 & 3 \\ 0 & 3 & 2 \end{bmatrix}, A_2 = \begin{bmatrix} 10 & 3 & 2 \\ -4 & 2 & 1 \\ 5 & -22 & 1 \end{bmatrix}, A_3 = \begin{bmatrix} 40 & 135 & -4 \\ 5 & 2 & 2 \\ 1 & 5 & 1 \end{bmatrix},$$

$$A_4 = \begin{bmatrix} 93 & 3 & -4 \\ 2 & -5 & 2 \\ -4 & 2 & 1 \end{bmatrix}, A_5 = \begin{bmatrix} 100 & 2 & 0 \\ 2 & 500 & 4 \\ 0 & 43 & 21 \end{bmatrix}, B = \begin{bmatrix} 4 \\ 1 \\ 2 \\ 5 \\ 2 \\ -1 \\ 5 \\ 2 \\ 1 \end{bmatrix}$$

Wywołanie funkcji gauss_elimination oraz linsolve zwraca nam następujący wynik:

```
>> gauss_plus_builder(A1_5, A2_5, A3_5, A4_5, A5_5, B_5)
   -45

Macierz A1 nie jest dodatnio określona

ans =

   NaN
```

Macierz A1 jest symetryczna. Wyznacznik minoru głównego M1 wynosi -45, więc z kryterium Sylwestera nie jest ona dodatnio określona. Komunikat był słuszny

3.7 Przykład obliczeniowy nr 6

W tym przykładzie posłużymy się następującymi danymi wejściowymi:

$$A_1 = \begin{bmatrix} 7 & 3 & 1 \\ 3 & 4 & 3 \\ 1 & 3 & 5 \end{bmatrix}, A_2 = \begin{bmatrix} 10 & 3 & 2 \\ -4 & 2 & 1 \\ 5 & -22 & 1 \end{bmatrix}, A_3 = \begin{bmatrix} 40 & 135 & -4 \\ 5 & 2 & 2 \\ 1 & 5 & 1 \end{bmatrix},$$

$$A_4 = \begin{bmatrix} 93 & 3 & -4 \\ 2 & -5 & 2 \\ -4 & 2 & 1 \end{bmatrix}, A_5 = \begin{bmatrix} 10 & 2 & 0 \\ 2 & 5 & 4 \\ 0 & 4 & 5 \end{bmatrix}, B = \begin{bmatrix} 4 \\ 1 \\ 2 \\ 5 \\ 2 \\ -1 \\ 5 \\ 2 \\ 1 \end{bmatrix}$$

Wywołanie funkcji gauss_elimination oraz linsolve zwraca nam następujący wynik:

```
>> gauss_plus_builder(A1_6, A2_6, A3_6, A4_6, A5_6, B_6)
Macierz A1 nie jest trójdiagonalna

ans =

   NaN
```

Jak możemy łatwo zauważyć, komunikat jest słuszny. Macierz A1 faktycznie nie jest trójdiagonalna.

3.8 Przykład obliczeniowy nr 7

W tym przykładzie posłużymy się następującymi danymi wejściowymi:

$$A_1 = \begin{bmatrix} 7 & 3 & 0 \\ 3 & 6 & 3 \\ 0 & 3 & 8 \end{bmatrix}, A_2 = \begin{bmatrix} 10 & 3 & 2 \\ -4 & 2 & 1 \\ 5 & -22 & 1 \end{bmatrix}, A_3 = \begin{bmatrix} 40 & 135 & -4 \\ 5 & 2 & 2 \\ 1 & 5 & 1 \end{bmatrix},$$

$$A_4 = \begin{bmatrix} 93 & 3 & -4 \\ 2 & -5 & 2 \\ -4 & 2 & 1 \end{bmatrix}, A_5 = \begin{bmatrix} 100 & 2 & 0 \\ 2 & 50 & 4 \\ 0 & 43 & 21 \end{bmatrix}, B = \begin{bmatrix} 4 \\ 1 \\ 2 \\ 5 \\ 2 \\ -1 \\ 5 \\ 2 \\ 1 \end{bmatrix}$$

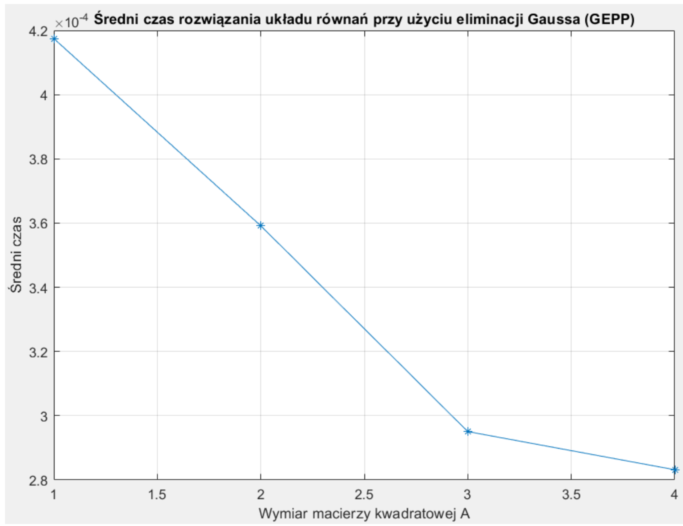
Wywołanie funkcji gauss_elimination oraz linsolve zwraca nam następujący wynik:

```
>> gauss_plus_builder(A1_7, A2_7, A3_7, A4_7, A5_7, B_7)
Macierz A5 nie jest symetryczna

ans =

   NaN
```

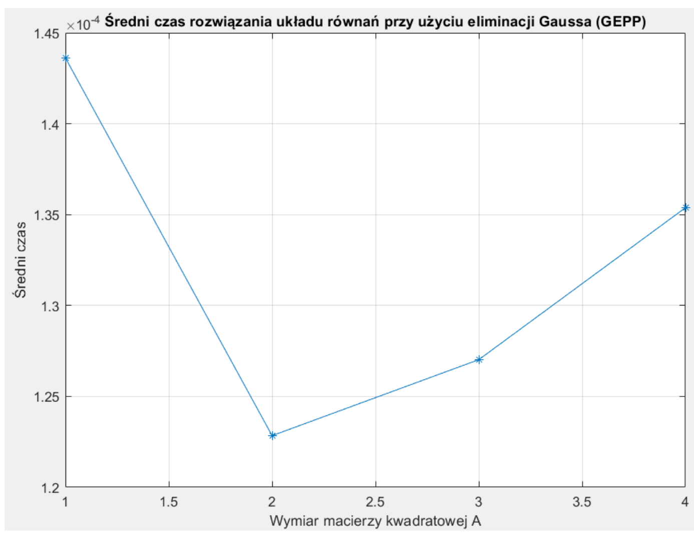
Jak możemy łatwo zauważyć, komunikat jest słuszny. Macierz A5 nie jest symetryczna

4 Analiza prędkości działania funkcji w zależności od wymiaru macierzy

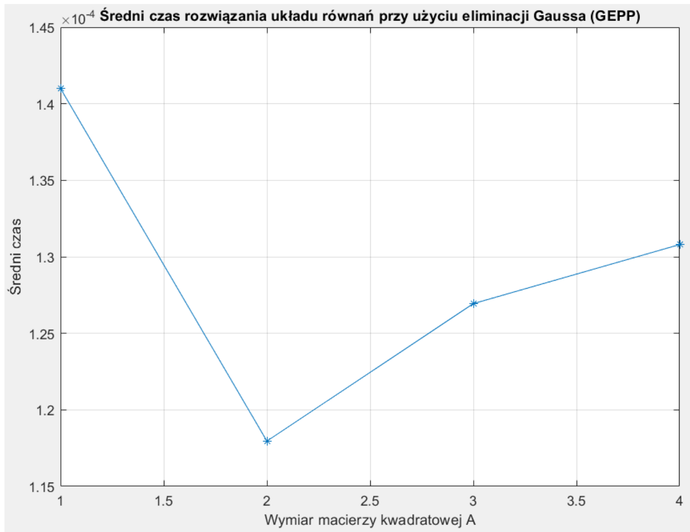
W podanych poniżej zależnościach zostały użyte macierze z przykładów 1, 2, 3 oraz 4. Celem tego doświadczenia było sprawdzenie, czy wraz ze wzrostem wymiarów macierzy, wzrośnie także czas działania funkcji.
Pomiar nr 1:



Pomiar nr 2:



Pomiar nr 3:



Z podanych wykresu wynika, że czasy wykonywania się podanych funkcji nie różnią się zbytnio od siebie. Należy jednak pamiętać, iż dużą rolę odgrywa także dobór elementów naszej macierzy, czas nie tylko zależy od wymiaru.

5 Wskaźnik uwarunkowania układu równań

Jak powszechnie wiadomo, obliczenia wykonywane przez komputer nie zawsze są dokładne. Uwarunkowanie układu równań można oszacować przy pomocy wskaźnika uwarunkowania macierzy, który jest niczym innym jak iloczynem normy zgodnej z normą wektorową macierzy A oraz normy odwrotności macierzy A.

Aby sprawdzić dokładność naszych obliczeń, postanowiłem stworzyć funkcję condition_number. Nie została ona użyta w funkcji gauss_elimination, gdyż dla większych wymiarów macierzy wejściowej A, funkcja nie wykonuję się w przyzwoitym czasie, przez co uzyskanie rozwiązania równania Ax=b zajęłoby zbyt wiele czasu.

Sprawdźmy więc, jakie uwarunkowanie macierzy posiadają macierze z przykładu 1 oraz 2.

Przykład 1:

```
>> condition_number(A_1)

ans =

    36
```

Przykład 2:

```
>> condition_number(A_2)

ans =

  119.4375
```

Jak widać, macierze z obu przykładów są dobrze uwarunkowane, zatem rozwiązania x nie powinny odbiegać od rozwiązań rzeczywistych.

6 Bibliografia

- https://www.mathworks.com/help/matlab/
- http://www.mini.pw.edu.pl/(tylda)iwrobel/MADMN_zima_2020-21/mn_pliki/Zapiski_numeryczne_MAD.pdf
- https://pl.wikipedia.org/wiki/Kryterium_Sylvestera
- https://pl.wikipedia.org/wiki/Macierz_klatkowa