Zadanie numeryczne nr 5

Kamila Dmowska

December 2021

1 Wstęp

Zadanie polegało na zaimplementowaniu czterech iteracyjnych metod rozwiązywania układu równań.

- relaksacyjną (Richardsona)

$$x^{(n+1)} = x^{(n)} + \gamma \left(b - Ax^{(n)} \right) ,$$

- Jacobiego:

$$x^{(n+1)} = D^{-1} \left(b - Rx^{(n)} \right) ,$$

- Gausa-Seidla

$$x^{(n+1)} = L^{-1} \left(b - Ux^{(n)} \right) ,$$

- Successive OverRelaxation

$$x_i^{(n+1)} = (1-\omega)x_i^{(n)} + \frac{\omega}{a_{ii}} \left(b_i - \sum_{j<i} a_{ij} x_j^{(n+1)} - \sum_{j>i} a_{ij} x_j^{(n)} \right), \quad i = 1, 2, \ldots, N .$$

Do rozwiązania, skorzystałam z podanej macierzy

$$A = \begin{pmatrix} 4 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 4 \end{pmatrix}, \qquad b = \begin{pmatrix} 2 \\ 6 \\ 2 \end{pmatrix}$$

2 Implementacja

2.1 Metoda relaksacyjna (Richardsona)

Jako pierwszą, zaimplementowałam metodę relaksacyjną. Jest ona najprostsza w zaimplementowaniu pośród przedstawionych czterech metod.

```
def relaxation(a, b, gamma, n):

    x = np.zeros(len(a[0]))

    for i in range(0, n):
        x = x + gamma * (b - (a @ x))

    return x
```

Na początku tworzę tablicę x wypełnioną zerami od długości jednego wiersza podanej w argumentach macierzy a. Następnie, robię pętlę w zależności od n, które przekazałam w argumentach funkcji. Im większe będzie n, tym dokładniejszy otrzymany wynik. W środku pętli dokonuję wyliczania x z podanego wcześniej wzoru. Gamma jest współczynnikiem, od właściwego dobrania zależy jak dokładny bedzie otrzymany wynik. Znak @ oznacza mnożenie macierzowe. Na końcu zwracam x.

2.2 Metoda Jacobiego

Aby skorzystać z tej metody, należy wyliczyć macierz R. Jest to macierz A bez diagonali.

```
def jacobi(A, b, n):
    x = np.zeros(len(A[0]))

    D = np.diagflat(np.diag(A))
    Dinv = np.linalg.inv(D)
    R = A - D

    for i in range(n):
        x = Dinv @ (b - R @ x)

    return x
```

Aby otrzymać diagonalę, używam wbudowanej funkcji z biblioteki numpy np.diagflat(np.diag(A)), (np.diag() storzy nam po prostu tablice z wartościami na przekątnej, zaś nam potrzebna jest cała macierz, więc korzystany z funkcji np.diagflat()). Następnie, musimy obliczyć D^{-1}. Aby to zrobić, korzystam z wbudowanej funkcji np.linalg.inv(). Następnym krokiem, jest wykoannie n-krotej iteracji. Podstawiam więc odpowiednie zmienne i w zależności od tego, jak dużo wykonam iteracji, dostanę dokładniejszy wynik.

2.3 Metoda Gausa-Seidla

```
def Gauss_Seidel(A, b, n ):

    x = np.zeros(len(A))
    L = np.tril(A)
    L_1 = np.linalg.inv(L)
    U = np.triu(A,1)

    for i in range(n):
        x = L_1 @ (b - (U @ x))

    return x
```

Aby skorzystać z tej metody, koniecznie jest obliczenie L^{-1} oraz U. Aby wyliczyć L oraz U, korzystamy z wbudowanych funkcji z biblioteki numpy, kolejno np.tril(), oraz np.triu(). Do obliczenia macierzy odwrotnej L^{-1}, korzystamy z funkcji np.linalg.inv(). Następnym krokiem jest wykonanie n ilości iteracji, którą podaliśmy w argumentach funkcji. Im większa liczba n, tym dokładniejszy wynik.

2.4 Succesive OverRelaxation

```
def Succesive_OverRelaxation(a,b,x,w,it):

    n = len(a)

    for k in range(it):

        for i in range(0, n):

            c = b[i]

            for j in range(0, n):
                if (i != j):
                    c -= a[i][j] * x[j]

            x[i] = (1 - w) * x[i] + (w/a[i][i])*(c)
    return x
```

W podanym wzorze widzimy dwie sumy. Aby uniknąć tworzenia dwóch osobnych pętli, możemy zwrócić uwagę na fakt, iż pierwsza pętla wykonuje się dla elementów $i < j$, natomiast druga pętla wykonuje się dla elementów dla $i > j$. Widząc to można napisać jedną pętlę, w której wykluczymy sytuację $i == j$. We wzorze, widnieje również współczynnik w. Wynik jest zależny od tego jak dobrze go dobierzemy.

3 Wyniki

Dla wszystkich metod, otrzymałam taki sam wynik:

$$x = \begin{pmatrix} 1 \\ 2 \\ 1 \end{pmatrix}$$

4 Wnioski

Dla wszystkich podanych metod, wyliczone wartości są poprawne. Najszybszą metodą jest metoda Succesive OverRelaxation, ponieważ nie trzeba w niej w każdej iteracji dokonywać mnożenia macierzy, co jest złożonym procesem .