# Zadanie E
# SOR - metoda nadrelaksacji

Język programowania: C++

Metoda nadrelaksacyjna SOR jest jedną z metod iteracyjnego rozwiązywania układów równań liniowych

$$Ax = y \text{ gdzie } A \in \mathbb{R}^{N\times N}, x, y \in \mathbb{R}^N.$$

Dla danego parametru $\omega$ i wektora startowego $x^0$ wyznacza ona ciąg przybliżeń rozwiązania $x^k$.

Dla macierzy symetrycznych, dodatnio określonych i $\omega \in (0, 2)$ metoda SOR jest zbieżna do rozwiązania powyższego układu dla dowolnego wektora startowego.

Celem zadania jest zaimplementowanie metody SOR dla układów, w których macierz A jest symetryczną macierzą rzadką (większość elementów jest równych 0), która ma postać wielodiagonalną. Oznacza to, że elementy niezerowe mogą występować na diagonali oraz na pewnej liczbie "wstęg" znajdujących powyżej i poniżej diagonali. Jeżeli liczba wstęg powyzej diagonalii to M to w wierszu k-tym niezerowe mogą być elementy $a_{k,j}$ gdzie $k - M \leqslant j \leqslant k + M$.

Przykładowa macierz $B$ (rozmiaru N=8 z M = 2) może wyglądać następująco

$$B = \begin{bmatrix} c_1 & b_1 & a_1 & 0 & 0 & 0 & 0 & 0 \\ b_1 & c_2 & b_2 & a_2 & 0 & 0 & 0 & 0 \\ a_1 & b_2 & c_3 & b_3 & a_3 & 0 & 0 & 0 \\ 0 & a_2 & b_3 & c_4 & b_4 & a_4 & 0 & 0 \\ 0 & 0 & a_3 & b_4 & c_5 & b_5 & a_5 & 0 \\ 0 & 0 & 0 & a_4 & b_5 & c_6 & b_6 & a_6 \\ 0 & 0 & 0 & 0 & a_5 & b_6 & c_7 & b_7 \\ 0 & 0 & 0 & 0 & 0 & a_6 & b_7 & c_8 \end{bmatrix}$$

## Wejście

Dane wejściowe są w formacie

*N*
*M*
*wstęga pierwsza*
*wstęga druga*
...
*wstęga M-ta*
*diagonala*
*prawa strona równania*
*wektor początkowy*
$\omega$
*L*

gdzie $N$ oznacza rozmiar macierzy, $M$ liczbę niezerowych "wstęg", $\omega$ jest parametrem metody SOR, a $L$ oznacza liczbę iteracji, którą mamy wykonać. Przykładowy plik dla macierzy $B$ może wyglądać następująco:

8
2
$a_1$ $a_2$ $a_3$ $a_4$ $a_5$ $a_6$
$b_1$ $b_2$ $b_3$ $b_4$ $b_5$ $b_6$ $b_7$
$c_1$ $c_2$ $c_3$ $c_4$ $c_5$ $c_6$ $c_7$ $c_8$
$y_1$ $y_2$ $y_3$ $y_4$ $y_5$ $y_6$ $y_7$ $y_8$
$x_1^0$ $x_2^0$ $x_3^0$ $x_4^0$ $x_5^0$ $x_6^0$ $x_7^0$ $x_8^0$
1.5
4

definiuje on zadanie wykonania 4 iteracji metody SOR dla układu $Bx = y$ gdzie $y = (y_1, y_2, \ldots, y_8)$, parametru $\omega = 1.5$ i wektora startowego $x^0 = (x_1^0, x_2^0, x_3^0, x_4^0, x_5^0, x_6^0, x_7^0, x_8^0)$

## Wyjście

Wyjście powinno zwierać w kolejnych liniach współrzędne wektora $x^L$ otrzymanego w wyniku wykonania $L$ iteracji metody SOR dla podanych danych wejściowych. Wyniki wypisujemy w notacji naukowej z precyzją 10 miejsc po przecinku.

Dla powyższych przykładowych danych wyjście powinno wyglądać następująco

$x_1^4$
$x_2^4$
$x_3^4$
$x_4^4$
$x_5^4$
$x_6^4$
$x_7^4$
$x_8^4$

## Uwagi

Z uwagi na błędy zaokrągleń wyniki będą porównywane z pewnym dopuszczalnym błędem zarówno względnym $\varepsilon_r$ (zazwyczaj około $10^-8$) jak i bezwzględnym $\varepsilon_a$ (zazwyczaj około $10^{-12}$). Dokładnie: jeżeli oczekiwany wynik to $x$ a program zwróci $\bar(x)$, to wynik ten będzie odrzucony jeżeli $|x - \bar{x}| > \varepsilon_a$ oraz $|\frac{x-\bar{x}}{x}| > \varepsilon_r$.

## Przykład

Dane wejściowe (plik1.in)

```
7
2
1 2 1 2 1
2 -1 3 1 3 -1
5 6 7 8 9 10 11
8 9 11 16 15 14 11
2 3 2 3 2 3 2
1.5
1
```

Oczekiwane wyjście (plik1.out)

```
-1.0000000000000000e+000
2.5000000000000000e-001
-7.3214285714285721e-001
3.1808035714285765e-001
-2.6432291666666696e-001
9.2352120535714288e-001
6.6197874391233791e-001
```

## Przykład

Dane wejściowe (plik2.in)

```
10
1
1 -1 2 1 3 -1 1 2 2
8 9 10 9 8 5 10 9 7 5
7 -8 5 -7 3 -5 7 -5 4 -3
2 1 2 1 2 1 2 1 2 1
1.3
5
```

Oczekiwane wyjście (plik2.out)

```
9.9940568551471820e-01
-9.0904662229329669e-01
6.0991796223201045e-01
-1.0130102944878934e+00
1.0831324689900357e+00
-1.5137961825745920e+00
6.1540772283026224e-01
-8.7787343355916414e-01
1.1233322463475970e+00
-1.0523066239504728e+00
```