# Metody Numeryczne - Projekt 2

Zuzanna Piróg

styczeń 2023

Celem projektu jest rozwiązywanie układu równań liniowych $Ax = b$ gdzie $A \in \mathbb{R}^{nxn}$ jest macierzą trójdiagonalną, $b \in \mathbb{R}^n$, metodą Backward SOR. Macierz A będziemy przechowywać w postaci macierzy $3xn$.

Wydział Matematyki
i Nauk Informacyjnych
POLITECHNIKA WARSZAWSKA

# Spis treści

# 1 Metoda iteracyjna rozwiązywania układów równań liniowych

Nazwa BSOR (ang.Backward Successive OverRelaxation), nazywana też metodą nadrelaksacji jest to metoda iteracyjna służąca do rozwiązywania układów równań liniowych.
Niech $Ax = b$, będzie układem równań liniowych. który chcemy rozwiązać, przy czym $A \in \mathbb{R}^{nxn}$ jest nieosobliwa a $x, b \in \mathbb{R}^n$. Idea metod iteracyjnych jest następująca: startując z danego przybliżenia początkowego

$$x^0 = \begin{bmatrix} x_1^0 \\ x_2^0 \\ \vdots \\ x_n^0 \end{bmatrix} \in \mathbb{R}^n$$

tworzymy ciąg kolejnych przybliżeń $x^k$, taki, że $x^k \to x$ przy $k \to$ inf.Każdy element tego ciągu jest wektorem (tj. $x^k \in \mathbb{R}^n$ n dla $k = 0, 1, ...$).Zbieżność jest rozumiana w sensie normy, tj. $||x^k - x|| \to 0$.
Metody iteracyjne mają duże znaczenie zwłaszcza przy rozwiązywaniu wielkich układów równań. Mogą być szybsze i mogą mieć mniejsze wymagania pamięciowe niż metody bezpośrednie. W praktyce nierzadko spotyka się układy, które mają kilkaset tysięcy czy kilka milionów niewiadomych. Często też zdarza się, że układ ma określoną strukturę, np. elementy niezerowe znajdują się tylko na określonych miejscach w macierzy, najczęściej w pobliżu diagonali. Można (a często trzeba, jeśli chcemy uzyskać rozwiązanie w rozsądnym czasie) to odpowiednio wykorzystać w implementacji.

Metoda BSOR omawiana w niniejszym dokumencie, należy do grupy tzw. metod iteracji prostej. Zasada ich konstrukcji jest taka, że układ wyjściowy $Ax = b$ zapisujemy w postaci równoważnej $x = Bx + c$, a następnie kolejne przybliżenia obliczamy według wzoru

$$x^{k+1} = Bx^k + c, \text{dla } k = 0, 1, ...$$

Macierz B nazywamy macierzą iteracji. Jeśli nie znamy przybliżonego rozwiązania, to jako przybliżenie początkowe $x^0$ można wziąć wektor o elementach losowych albo wektor zerowy.

Ciężko jest sprawdzić czy dany wektor $x^0$ jest dobtym przybliżeniem początkowym gwarantującym zbieżność metody. Dlatego szukamy warunków zapewniających zbieżność dla każdego $x^0 \in \mathbb{R}^n$. Wykorzystujemy do tego celu twierdzenie mówiące o tym, że dana metoda iteracyjna postaci $x^{k+1} = Bx^k + c$, dla $k = 0, 1, ...$ jest zbieżna globalnie (czyli dla każdego przybliżenia początkowego $x^0$) wtedy i tylko wtedy gdy $\rho(B) < 1$, gdzie $\rho(B)$ jest promieniem spektralnym macierzy $B$.

# 2 Metoda Backward SOR

Nazwa BSOR (ang. Backward Successive OverRelaxation), nazywana też metodą nadrelaksacji, jest uogólnieniem innej metody iteracyjnej - Gaussa - Seidla. W metodzie BSOR występuje dodatkowo parametr $\omega \in \mathbb{R}$, zwany parametrem relaksacji, który umożliwia szybszą zbieżność metody w porównaniu do metody Gausa - Seidla. BSOR wykorzystuje podobnie jak Gaus - Seidel wcześniej wyliczone rozwiązania wektora $x_n^{k+1}, ...x_{i+1}^{k+1}$ w celu obliczenia najnowszego przybliżenia $x_i^{k+1}$. Parametr $\omega$ może przyjmować wartości jedynie z przedziału $(0, 2)$ w pozostałych przypadkach metoda nie będzie zbieżna dla pewnych przybliżeń początkowych. Wartym zauważenia jest fakt, że dla wartości paramteru $\omega = 1$ otrzymamy metodę Gausa - Seidla. Gdy współczynnik $\omega > 1$, mówimy o metodzie kolejnych (sukcesywnych) nadrelaksacji (ang. Successive Over-Relaxation, SOR). Takie wartości współczynnika relaksacji zazwyczaj przyśpieszającbieżność metody Gaussa-Seidla. Odpowiednio dobrana wartość parametru $\omega < 1$ pozwala uzyskać zbieżność w przypadku układów, które normalnie nie spełniają warunku zbieżności metody GaussaSeidla.

## 2.1 Algorytm Metody BSOR

Poniżej znajduje się algorytm służący do wyliczania kolejnych przybliżeń

$x^0 = (x_1^0, ..., x_n^0)^T$ - przybliżenie początkowe
for $k = 0, 1, ...$ (dopóki nie będzie spełniony wybrany warunek stopu)
for $i = n, n-1, ...1$
$x_i^{k+1} = (1-\omega)x_i^k + (\omega/a_{ij})(b_i - \sum_{j=1,j<i}^{n} a_{ij}x_j^{k+1} - \sum_{j=1,j>i}^{n} a_{ij}x_j^k)$
*end*
*end*

Możemy zauważyć, że gdy $\omega = 1$, metoda SOR staje się metodą Gaussa-Seidla. Metodę SOR można również wyprowadzić zapisując macierz A jako

$$A = L + D + U,$$

gdzie $D$ to macierz diagonalna zawierająca główną przekątną macierzy A, $L$ to macierz dolnotrójkątna, a $U$ to macierz górnotrójkątna. Następnie mnożąc powyższe równanie przez $\omega \neq 1$ a potem dodając stronami macierz $D$ i odpowiednio grupując wyrazy, otrzymujemy:

$$\omega A = (D + \omega L) - ((1-\omega)D - \omega U)$$

Podstawiając powyższą zależność do równania $\omega Ax = \omega b$, równoważnemu równaniu $Ax = b$, otrzymamy:

$$(D + \omega L)x - ((1-\omega)D - \omega U)x = \omega b$$

a dalej

$$x = (D = \omega L)^{-1}((1-\omega)D - \omega U)x + \omega(D + \omega L)^{-1}b.$$

Wzór iteracyjny jest więc postaci:

$$x^{k+1} = B_{BSOR}x^k + c_{BSOR}$$

gdzie

$$B_{BSOR} = (D + \omega L)^{-1}((1-\omega)D - \omega U)$$

oraz

$$c_{BSOR} = \omega(D + \omega L)^{-1}b$$

# 3 Zastosowanie algorytmu metody BSOR w zadaniu

Celem projektu jest rozwiązywanie układu równań liniowych $Ax = b$, gdzie $A \in \mathbb{R}^{nxn}$ jest macierzą trójdiagonalną, $b \in \mathbb{R}^n$ wektorem wyrazów wolnych, omawianą metodą iteracyjną Backward SOR (BSOR). W pamięci komputera będziemy przechowywać niezerowe przekątne macierzy $A$ w postaci macierzy $3xn$.

## 3.1 Implementacja i pierwszy przykład

Zaprezentujemy działanie i implementację algorytmu na podstawie pierwszego przykładu. Przyjżyjmy się zatem macierzy trójdiagonalnej $A$ oraz wektorowi wyrazów wolnych $b$:

$$A = \begin{pmatrix} 5 & -1 & 0 & 0 \\ -1 & 5 & 2 & 0 \\ 0 & 2 & 5 & -2 \\ 0 & 0 & -2 & 6 \end{pmatrix}, b = \begin{pmatrix} 1 \\ 2 \\ 3 \\ 4 \end{pmatrix}$$

### 3.1.1 Badanie zbieżności macierzy iteracji $B_{BSOR}$ oraz wskaźnika uwarunkowania macierzy A

Musimy najpierw sprawdzić czy dla danego parametru $\omega$ macierz iteracji $B_{BSOR}$ będzie zbieżna, czyli czy $\rho(B_{BSOR}) < 1$. Wykorzystamy do tego funkcję zaimplementowaną w Matlabie. Obliczymy przy okazji wskaźnik uwarunkowania macierzy $A$ zgodnie ze wzorem:

$$cond(A) = ||A^{-1}|| * ||A||.$$

Implementacja funkcji sprawdzającej zbieżność macierzy i wyliczającą wskaźnik uwarunkowania macierzy $A$ w Matlabie:

```
1  function [rho_B, cond] = zbieznosci(A,omega)
2  %funckja oblicza promien spektralny macierzy iteracji B dla ...
       metody BSOR, na podstawie przekazanej macierzy A
3
4  %Input:
5  %A - macierz ukladu rownan
6  %omega - parametr relaksacji
7
8  %Output:
9  %rho_B - promien spektralny macierzy iteracji B
10 %cond - wskaznik uwarunkowania macierzy A
11
```

```
%rozbicie macierzy A na A = D + U + L (D - macierz diagonalna, U ...
    - macierz
%dolno trojkatna, L - macierz gorno trojkatna)
D = diag(diag(A));
L = tril(A) - D;
U = triu(A) - D;

%macierz Iteracji
B = inv(D+omega*L)*((1-omega)*D - omega*U);
%promien spektralny
rho_B = max(abs(eig(B)));
if rho_B ≥ 1
    disp('Promien spektralny wiekszy niz 1, brak zbieznosci')
    return
end
cond = norm(inv(A)) * norm(A);

end
```

Sprawdźmy jakie wyniki otrzymamy dla macierzy podanej w zadaniu i parametru relaksacji $\omega = 0.8$

```
>> [rho_B, cond] = zbieznosci(A,omega)

rho_B =

    0.5266


cond =

    3.6395
```

Zatem macierz $A$ jest zbieżna, jej promień spektralny wynosi $\rho(B_{BSOR}) = 0.5266$ dla parametru $\omega = 0.8$, natomiast wskaźnik uwarunkowania macierzy wynosi $cond(A) = 3.6395$.

### 3.1.2 Przechowywanie trójdiagonalnej macierzy A jako macierzy $3xn$

Przystąpmy do dalszej części polecenia. Mamy przechowywać macierz trójdiagonalną $A$ w postaci macierzy o wymiarach $3xn$ zawierającej przekątne macierzy $A$. Aby to zrobić zastosujemy funkcję przekształcania macierzy zaimplementowaną w Matlabie.

```
function [B] = store_as_3n_matrix(A)
%funkcja zmienia przekazana macierz trojdiagonalna na macierz o ...
    wymiarach
%3xn i przechowuje wartosci diagonali
%A - macierz trojdiagonalna
%B - macierz ktorej pierwszy wiersz to gorny diagonal, drugi ...
    wiersz to
%glowny diagonal macierzy A, trzeci wiersz to dolny diagonal ...
    macierzy A
```

```
7
8
9  n = size(A,1);
10 B = zeros(3,n);
11
12 %wartosci diagonali
13 d3 = diag(A,-1);
14 d2 = diag(A,0);
15 d1 = diag(A,1);
16
17 %przedluzanie krotszych przekatnych zerami
18 d1 = [d1; zeros(n-length(d1),1)];
19 d3 = [zeros(n-length(d3),1); d3];
20
21 %przypisanie do wierszy
22 B(1,:) = d1;
23 B(2,:) = d2;
24 B(3,:) = d3;
25
26 end
```

Po zastosowaniu funkcji store_as_3n_matrix(A), macierz A będzie wyglądać następująco:

$$A = \begin{pmatrix} 5 & -1 & 0 & 0 \\ -1 & 5 & 2 & 0 \\ 0 & 2 & 5 & -2 \\ 0 & 0 & -2 & 6 \end{pmatrix} \longrightarrow A = \begin{pmatrix} -1 & 2 & -2 & 0 \\ 5 & 5 & 5 & 6 \\ 0 & -1 & 2 & -2 \end{pmatrix}$$

Pierwszy wiersz nowej macierzy to górna przekątna macierzy A znajdująca się nad główną przekątną, wiersz został dopełniony zerem aby zgadzały się wymiary nowej macierzy. Drugi wiersz to główna przekątna macierzy A, a trzeci wiersz to dolna przekątna, dopełniona zerem na początku wiersza.
Tak przedstawioną macierz będziemy stosować w implementacji algorytmu BSOR w Matlabie.

### 3.1.3 Implementacja metody BSOR w Matlabie

```
1  function x = funkcja_BSOR(A, b, omega, x0, tol, maxiter)
2  %funkcja rozwiazuje rownanie postaci Ax = b przy uzyciu metody ...
       iteracyjnej
3  %BSOR
4  %Input:
5  %A - macierz trojdiagonalna w postaci n x n
6  %b - wektor wyrazow wolnych w postaci n x 1
7  %omega - parametr relaksacji
8  %x0 - pierwotny wektor przyblizenia
9  %tol - akceptowalny blad
10 %maxiter - maksymalna liczba iteracji
11 %Output:
12 %x - szukane rozwiazanie
```

```
13
14 %zgodnie z poleceniem mamy przechowywac macierz A w pamieci ...
       komputera jako
15 %macierz nx3 lub 3xn
16
17 if size(A, 1)≠ size(A,2)
18     disp('Zle wymiary macierzy')
19     return
20 end
21 if size(b,1) ≠ size(A,1) ||size(b,2) ≠ 1
22     disp('Zle rozmiary danych')
23     return
24 end
25 if size(x0,1) ≠ size(A,1) ||size(x0,2) ≠ 1
26     disp('Zle rozmiary danych')
27     return
28 end
29
30 A = store_as_3n_matrix(A);
31 n = size(A,2);
32
33 iteracja = 1;
34 while iteracja ≤ maxiter
35     err = 0;
36     s = 0;
37     for i = n:-1: 1
38         if i == 1
39             s = s - A(2,i)*x0(i) - A(i,i)*x0(i+1);
40         elseif i == n
41             s = s - A(2,i)*x0(i) - A(3,i)*x0(i-1);
42         else
43             s = s - A(2,i)*x0(i) - A(1,i)*x0(i+1) - A(3,i)*x0(i-1);
44         end
45         s = omega*(s + b(i))/A(2,i);
46
47         if abs(s) > err
48             err = abs(s);
49         end
50         x0(i) = x0(i) + s;
51
52     end
53     if err ≤ tol
54         break;
55     else
56         iteracja = iteracja + 1;
57     end
58 end
59
60 fprintf('Rozwiazanie znalezione po %d iteracjach', iteracja);
61 x = x0;
62
63 end
```

Jako że w zadaniu rozważamy macierz A, która jest trójdiagonalna, wydajność naszego algorytmu poprawiła się i możemy uniknąć obliczania dodatkowej pętli przechodzącej po kolumnach macierzy A, nie musimy również niepotrzebnie iterować przez 0'we wartości pierwotnej macierzy, oszczędzając tym samym

pamięć i czas wykonania polecenia.

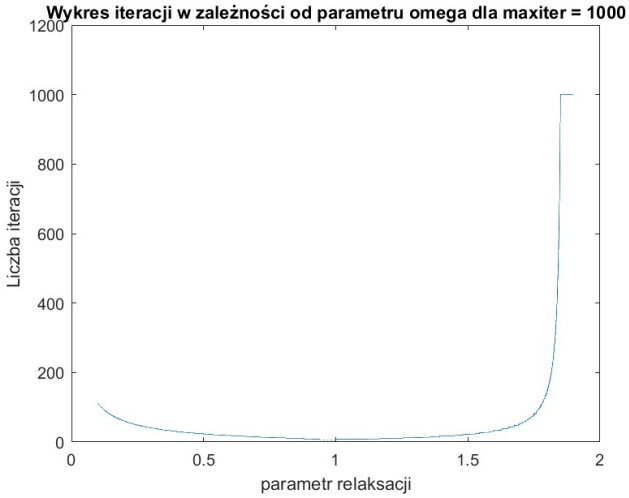
Przyjmijmy, że początkowe przybliżenie to wektor $x^0 = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}$, akceptowalny błąd to $tol = 0.0002$, a maksymalna liczba iteracji $maxiter = 1000$. Porównajmy wyniki uzyskane przy pomocy algorytmu dla różnych wartości parametru relaksacji $\omega$.

Tabela przedstawiająca zależność uzyskanych wyników i ilości potrzebnych iteracji do ich osiągnięcia dla różnych wartości paramteru relaksacji:

| | $\omega = 0.2$ | $\omega = 0.7$ | $\omega = 1.2$ | $\omega = 1.8$ | $\omega = 1.85$ | *dokl.rozw* |
|---|---|---|---|---|---|---|
| *iteracje* | 60 | 15 | 10 | 144 | 974 | – |
| $x_1$ | 0.2108 | 0.2103 | 0.2103 | 0.2103 | 0.2104 | 0.2103 |
| $x_2$ | 0.0531 | 0.0518 | 0.0516 | 0.0517 | 0.0517 | 0.0516 |
| $x_3$ | 0.9743 | 0.9760 | 0.9762 | 0.9763 | 0.9762 | 0.9762 |
| $x_4$ | 0.9909 | 0.9919 | 0.9921 | 0.9920 | 0.9920 | 0.9921 |

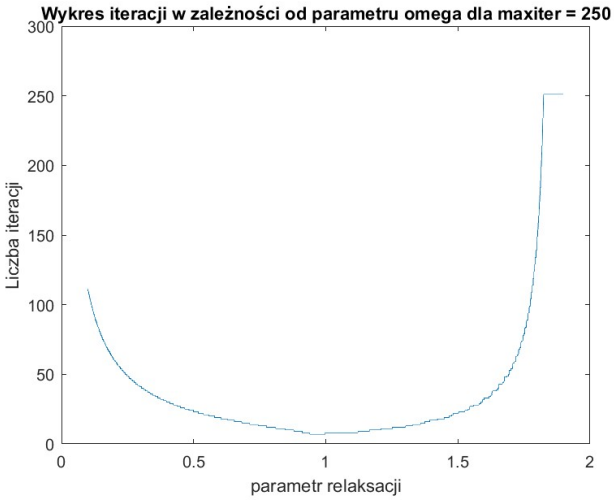
Jak możemy zauważyc metoda niezwykle dobrze radzi sobie z wyznaczaniem poprawnych rozwiązań. To co jest warte naszej uwagi to fakt, że wartości te są prawie takie same jak pożądany wynik. Uwagę przykuwa fakt, że dla $\omega = 1.8$ oraz $\omega = 1.85$ różnica wymaganej liczby iteracji jest tak ogromna, wzrosła ona około 4 razy. Co ciekawe nie miało to miejsca dla parametrów dalej od siebie oddalonych $\omega = 0.7$ oraz $\omega = 1.2$, które znalazły rozwiązanie w niewielkiej liczbie iteracji.

Spójrzmy zatem na wykres liczby iteracji w zależności od parametru $\omega$:




Rysunek 1: Wizualizacja zbieżności dla **Przykładu 1**

Jak możemy zauważyć metoda BSOR radzi sobie z niniejszym układem liniowym nadzwyczaj sprawnie. Dla większości wartości parametru $\omega$ metoda bardzo szybko znajduje rozwiązania i są one wyjątkowo dokładne jak możemy odczytać z tabeli. Problemy zdają się pojawiać w momencie gdy parametr $\omega$ zbliża się do wartości 2. Wtedy liczba iteracji gwałtownie rośnie co reprezentuje na wykresie prawie pionowa, ostra linia, chociaż mimo tego zjawiska przybliżenia pozostają dokładne.




Rysunek 2: Wizualizacja zbieżności dla **Przykładu 1**

Co ciekawe, gdy zmniejszymy maksymalną liczbę iteracji o 3/4 poprzedniej liczby to funkcja iteracji od parametru jeszcze bardziej przyjmuje postać kołyski. Zarówno dla skrajnie małych wartości parametru jaki skrajnie dużych funkcja zwiększa konieczną liczbę iteracji wymaganą do odnalezienia przybliżeń.

## 3.2 Pozostałe przykłady

Zaprezentujemy przykłady, dla których metoda jest zbieżna bądź nie jest zbieżna.

### 3.2.1 Przykład drugi

Rozpatrzmy macierz $A \in \mathbb{R}^{nxn}$ oraz wektor $b \in \mathbb{R}^n$ takie, że

$$A = \begin{pmatrix} 4 & 3 & 0 \\ 3 & 4 & -1 \\ 0 & -1 & 4 \end{pmatrix}, b = \begin{pmatrix} 9 \\ 9 \\ 6 \end{pmatrix}.$$

Sprawdźmy zbieżność macierzy iteracyjnej $B_{BSOR}$ dla parametru $\omega = 0.8$ oraz policzmy wskaźnik uwarunkowania macierzy A.

```
>> [rho_B, cond] = zbieznosci(A,omega)

rho_B =

    0.7464

cond =

    8.5497
```

Zatem macierz A zbieżna zgodnie z metodą BSOR. Promeień spekralny o wartości $\rho(B_{BSOR}) = 0.7464$ jest mniejszy od 1. Wskaźnik uwarunkwoania macierzy A zaś wynosi $cond(A) = 8.5497$. Jest on większy w porównaniu do poprzedniego zadania.

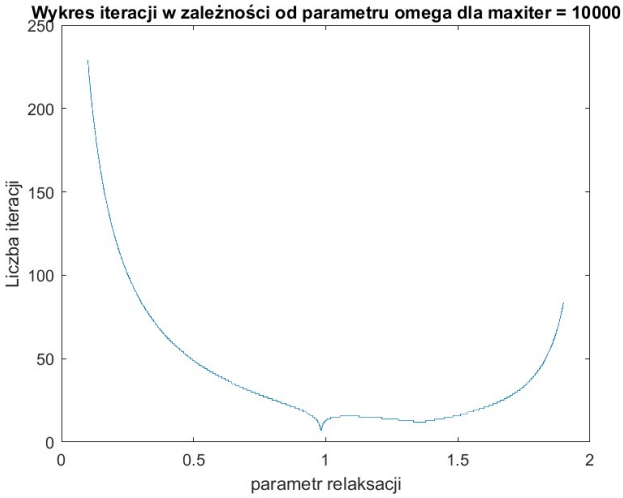
Macierz trójdiagonalną A przekształćmy do postaci macierzy 3xn.

$$A = \begin{pmatrix} 4 & 3 & 0 \\ 3 & 4 & -1 \\ 0 & -1 & 4 \end{pmatrix} \longrightarrow A = \begin{pmatrix} 3 & -1 & 0 \\ 4 & 4 & 4 \\ 0 & 3 & -1 \end{pmatrix}$$

Ponownie przyjmijmy, że początkowe przybliżenie to wektor $x^0 = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$, akceptowalny błąd to $tol = 0.0002$, a maksymalna liczba iteracji $maxiter = 1000$. Porównajmy wyniki uzyskane przy pomocy algorytmu dla różnych wartości parametru relaksacji $\omega$.

| | $\omega = 0.2$ | $\omega = 0.6$ | $\omega = 1.1$ | $\omega = 1.5$ | $\omega = 1.9$ | $dokl.rozw$ |
|---|---|---|---|---|---|---|
| $iteracje$ | 124 | 39 | 16 | 16 | 84 | – |
| $x_1$ | 0.3790 | 0.3759 | 0.3748 | 0.3750 | 0.3749 | 0.3748 |
| $x_2$ | 2.4958 | 2.4990 | 2.5002 | 2.5000 | 2.5000 | 2.5002 |
| $x_3$ | 2.1236 | 2.1247 | 2.1251 | 2.12513 | 2.1251 | 2.12512 |

Jak możemy odczytac dane z tabelki metoda BSOR radzi sobie nadzwyczaj sprawnie w znajdowaniu przybliżeń dla równania $Ax = b$. W przeciwieństwie do wcześniejszego przykładu, nawet dla dużego parametru relaksacji $\omega$ metoda w niezbyt dużej liczbie iteracji znajduje poprawne przybliżenia. Przyjrzyjmy się jak wygląda wykres iteracji od wartości parametru:




Rysunek 3: Wizualizacja zbieżności dla **Przykładu 2**

Tym razem metoda potrzebuje wielu iteracji dla małych wartości parametru $\omega$ w okolicy wartości paramtru równej 1 metoda potrzebuje najmniej iteracji.

### 3.2.2 Przykład trzeci

Rozpatrzmy macierz A i wektor b o następujących wartościach:

$$A = \begin{pmatrix} 3 & 4 & 0 & 0 & 0 & 0 \\ 6 & 7 & 8 & 0 & 0 & 0 \\ 0 & 2 & 1 & 1 & 0 & 0 \\ 0 & 0 & 5 & 1 & 6 & 0 \\ 0 & 0 & 0 & 8 & 9 & 7 \\ 0 & 0 & 0 & 0 & 1 & 5 \end{pmatrix}, b = \begin{pmatrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \end{pmatrix}$$

Sprawdźmy czy dla danej macierzy A jej macierz iteracji $B_{BSOR}$ spełnia warunki zbieżności, a więc czy metoda BSOR będzie zbieżna dla przedstawionej macierzy A.

```
1 >> [rho_B, cond] = zbieznosci(A,0.8)
2 Promie spektralny wi kszy ni 1, wynosi 7.930415e+00, brak ...
      zbie no ci
```

Zatem dla macirezy A metoda Backward SOR nie będzie zbieżna, ponieważ jej promień spektralny jest większy niż lub równy 1.

### 3.2.3 Przykład czwarty

Rozpatrzmy teraz macierz A, oraz wektor wyrazów wolnych b, o następujących wartościach:

$$A = \begin{pmatrix} 2 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 2 \end{pmatrix}, b = \begin{pmatrix} 7 \\ 1 \\ 1 \end{pmatrix}.$$

Sprawdźmy zbieżność macierzy gdy parametr relaksacji wynosi $\omega = 0.8$.

```
>> [rho_B, cond] = zbieznosci(A,omega)
rho_B =

    0.6593

cond =

    5.8284
```

Zatem metoda będzie zbieżna dla macierzy A zaprezentowanej w przykładzie. Promień spektralny o wartości $\rho(B_{BSOR}) = 0.6593$ jest mniejszy od 1. Wskaźnik uwarunkowania macierzy A zaś wynosi $cond(A) = 5.8284$.
Macierz A przekształcona na macierz zawierającą jedynie przekątne prezentuje się następująco:

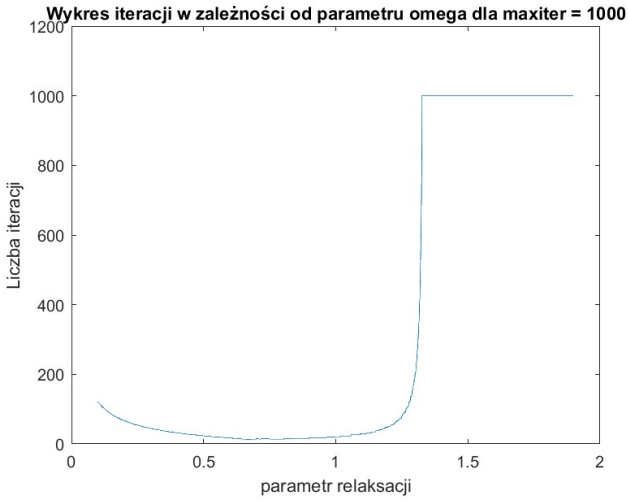
$$A = \begin{pmatrix} 2 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 2 \end{pmatrix} \longrightarrow A = \begin{pmatrix} -1 & -1 & 0 \\ 2 & 2 & 2 \\ 0 & -1 & -1 \end{pmatrix}$$

Ponownie przyjmijmy, że początkowe przybliżenie to wektor $x^0 = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$, jednak tym razem akceptowalny błąd to $tol = 0.004$, a maksymalna liczba iteracji $maxiter = 1000$. Porównajmy wyniki uzyskane przy pomocy algorytmu dla różnych wartości parametru relaksacji $\omega$.

| | $\omega = 0.1$ | $\omega = 0.5$ | $\omega = 1$ | $\omega = 1.3$ | $\omega = 1.4$ | $dokl.rozw$ |
|---|---|---|---|---|---|---|
| $iteracje$ | 121 | 23 | 20 | 191 | 1000 | – |
| $x_1$ | 5.9179 | 5.9943 | 6.0039 | 6.0048 | $1.0e+29*0.3721$ | 6 |
| $x_2$ | 4.8800 | 4.9886 | 5.0049 | 5.0083 | $1.0e+29*1.2129$ | 5 |
| $x_3$ | 2.9124 | 2.9886 | 3.0010 | 3.0037 | $1.0e+29*0.7366$ | 3 |

Tym razem dokładne rozwiązania to liczby całkowite, dlatego łatwo odczytać jak dokładna i szybka będzie metoda BSOR. Wraz ze wzrostem parametru ilość potrzebnych iteracji najpierw maleje, to zjawisko utrzymuje się do wartości parametru $\omega = 1.4$.

Spójrzmy na wykres:




Rysunek 4: Wizualizacja zbieżności dla **Przykładu 4**

Faktycznie dla wartości parametru większego od $\omega = 1.4$ ilość iteracji drastycznie wzrasta. Funkcja przymuje prawie pionową linię a od $\omega = 1.42$ algortym nie znajduje przybliżenia w ograniczeniu do 1000 iteracji.

### 3.2.4 Przykład piąty

Rozpatrzmy teraz macierz A, oraz wektor wyrazów wolnych b, o następujących wartościach:

$$A = \begin{pmatrix} 1 & 4 & 0 & 0 \\ 3 & 4 & 1 & 0 \\ 0 & 2 & 3 & 4 \\ 0 & 0 & 1 & 3 \end{pmatrix}, b = \begin{pmatrix} 1 \\ 4 \\ 1 \\ 0 \end{pmatrix}.$$

Sprawdźmy czy metoda jest zbieżna dla macierzy A.

```
1 >> [rho_B, cond] = zbieznosci(A,0.8)
2 Promie spektralny wi kszy niz 1, wynosi 2.427433e+00, brak ...
      zbieznosci
```

Zatem metoda BSOR nie będzie zbieżna dla następującej macierzy A, promień spektralny macierzy iteracjui $B_{BSOR}$ jest większy niż 1.

### 3.2.5 Przykład szósty

Rozpatrzmy teraz macierz A, oraz wektor wyrazów wolnych b, o następujących wartościach:

$$A = \begin{pmatrix} 7 & -1 & 0 \\ 2 & 7 & 3 \\ 0 & 3 & 7 \end{pmatrix}, b = \begin{pmatrix} 3 \\ 3 \\ 5 \end{pmatrix}.$$

Sprawdźmy czy metoda jest zbieżna dla macierzy A, przy parametrze relaksacji $\omega = 0.8$.

```
>> [rho_B, cond] = zbieznosci(A,0.8)
rho_B =
    0.3885
cond =
    2.4902
```

Zatem metoda będzie zbieżna dla macierzy A zaprezentowanej w przykładzie. Promień spektralny o wartości $\rho(B_{BSOR}) = 0.3885$ jest mniejszy od 1. Wskaźnik uwarunkwoania macierzy A zaś wynosi $cond(A) = 2.4902$.

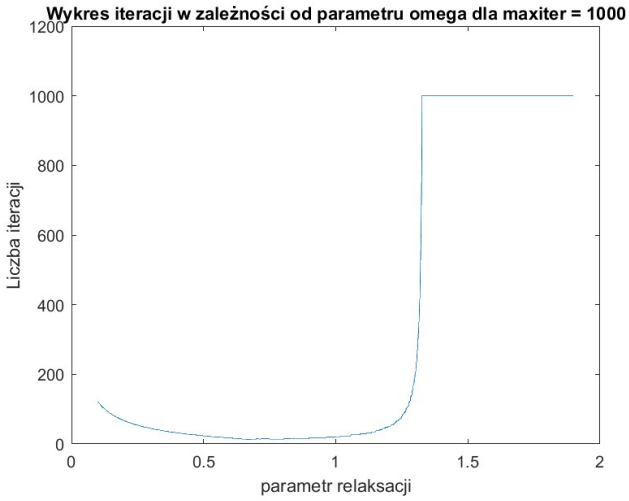
Zobaczmy jak wygląda macierz trójdiagonalna A przekształcona na macierz zawierającą jedynie przekątne.

$$A = \begin{pmatrix} 7 & -1 & 0 \\ 2 & 7 & 3 \\ 0 & 3 & 7 \end{pmatrix} \longrightarrow A = \begin{pmatrix} -1 & 3 & 0 \\ 7 & 7 & 7 \\ 0 & 2 & 3 \end{pmatrix}$$

Ponownie przyjmijmy, że początkowe przybliżenie to wektor $x^0 = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$, akceptowalny błąd to $tol = 0.0004$, a maksymalna liczba iteracji $maxiter = 1000$. Porównajmy wyniki uzyskane przy pomocy algorytmu dla różnych wartości parametru relaksacji $\omega$.

| | $\omega = 0.2$ | $\omega = 0.6$ | $\omega = 1.1$ | $\omega = 1.5$ | $\omega = 1.55$ | *dokl.rozw* |
|---|---|---|---|---|---|---|
| *iteracje* | 40 | 14 | 10 | 62 | 123 | – |
| $x_1$ | 0.4292 | 0.4286 | 0.4285 | 0.4284 | 0.4288 | 0.4286 |
| $x_2$ | 0.0024 | 0.0005 | −0.0001 | −0.0001 | 0.0001 | 0 |
| $x_3$ | 0.7115 | 0.7137 | 0.7142 | 0.7141 | 0.7145 | 0.7141 |

Ponownie jak w niektórych wcześniejszych przykładach dla niewielkich wartości paramtetru $\omega$ ilość iteracji jest niewielka dopiero w okolicy parametru $\omega = 1.5$ metoda nie potrafi znaleźć przybliżenia w mniej niż 1000 iteracjach.



Rysunek 5: Wizualizacja zbieżności dla **Przykładu 6**

Podobnie jak w przykładzie 4 funkcja iteracji gwałtownie rośnie od pewnej wartości parametru. Dla wartości parametru $\omega \in (0.5, 1)$ ilość iteracji jest niezwykle niska, dla takich wartości $\omega$ znajdujemy optymalne rozwiązanie.

## 3.3 Podsumowanie

Dobór parametru $\omega$ jest wysoce zależny od problemu. Przy ustalonym współczynniku relaksacji metoda BSOR może być szybciej zbieżna, ale wolniejsza dla innego. W większości omawianych przykładów od pewnej wartości parametru liczba iteracji gwałtownie wzrastała Metoda BSOR nie była zatem wydajna dla większych wartości $\omega$ od danej granicy. Warto zauważyć że w przykładzie czwartym gdy zwiększyła się tolerancja na błąd to zmniejszyła sie liczba potrzebnych iteracji do osiągnięcia przybliżenia równania $Ax = b$. Podsumowując funckja Backward SOR wyznacza przybliżenia bardzo dokładnie w większości przykładów. Dobór parametru $\omega$, dla którego algorytm będzie najwydajniejszy jest uzależniony od konkretnego zadania.

# 4 Dodatek

Ponizej znajduje się implementacja funkcji służącej do wyświetlania wykresu ilości iteracji w zależnnosci od wartości parametru $\omega$.

```
function plotIteracje(A, b,x0, tol, maxiter)
%Funckja tworzy wykres liczby iteracji potrzebnych do ...
    znalezienia rozwi zania Ax = b w zale no ci od parametru ...
    relaksacji
%Input:
%A - macierz tr jdiagonalna w postaci n x n
%b - wektor wyraz w wolnych w postaci n x 1
%x0 - pierwotny wektor przybli enia
%tol - akceptowalny b   d
%maxiter - maksymalna liczba iteracji
%Output:
%Wykres Liczby iteracji w zale no ci od omega

    %warto ci parametru relaksacji
    wVals = [0.1:0.00001:1.9];

    %inicjalizacja wektora liczby iteracji
    y = zeros(size(wVals));

    %znajdowanie liczby iteracji
    for i = 1:length(wVals)
        y(1,i) = funkcja_BSOR(A, b, wVals(i), x0, tol, maxiter);
    end

    %tworzenie wykresu
    plot(wVals, y);
    xline(0);
    yline(0)
    xlabel('parametr relaksacji');
    ylabel('Liczba iteracji');
    title(sprintf('Wykres iteracji w zale no ci od parametru ...
        omega dla maxiter = %d', maxiter))
end
```

# 5 Literatura


http://jacek.zlydach.pl/old-blog/download/numerki/sprawozdanka/zad1$_s or_t eoria.pdf$
*https* : *//tu.kielce.pl/ rokach/instr/mncc*$_w$*yklad*$_0$7*.pdf*
*https* : *//www.math.hkust.edu.hk/ mamu/courses*/231/*Slides*/$CH07_4A.pdf$