

**SPRAWOZDANIE**

**LABORATORIUM NR 2**

**TEMAT:**

**Rozwiązywanie układów równań liniowych metodami iteracyjnymi**

**PROWADZĄCY:**

Dr inż. Barbara Głut

**AUTOR:**

Małgorzata Olszewska

1. **Zagadnienia dotyczące wykonywanych ćwiczeń.**

Oprócz klasycznych metod rozwiązywania układów równań liniowych, istnieją również metody iteracyjne, których głównym założeniem jest „przybliżanie” wektora rozwiązań do pożądanego wektora wynikowego w kolejnych iteracjach algorytmu. W zależności od rodzaju metody iteracyjnej, stosowane są przy tym macierze iteracyjne różnych postaci. W ćwiczeniu będę badać dwie metody iteracyjne rozwiązywania układów równań liniowych: metodę Jacobiego oraz metodę kolejnych nadrelaksacji (SOR).

Metody iteracyjne są w dużym stopniu sparametryzowane. Umiejętne dobieranie ich wartości pozwala na uzyskiwanie różnych wyników, w zależności od potrzeb. W ćwiczeniu badano wpływ: rozmiaru układu, zastosowanego kryterium stopu i jego dokładności oraz wektora początkowego na: dokładność wektora wynikowego, liczbę iteracji prowadzących do rozwiązania oraz czas wykonywania algorytmu. W metodzie SOR wprowadzony jest również dodatkowy parametr *ω*, którego wartość silnie determinuje zbieżność układu

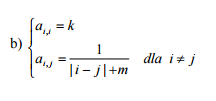
Metody iteracyjne nie zawsze są zbieżne lub są zbieżne dla nie wszystkich wektorów początkowych. Podstawowym kryterium badania zbieżności jest sprawdzenie, czy promień spektralny macierzy iteracji nie przekracza jedynki. Kryterium to jest często niepraktyczne, przez co często stosuje się inne, opisujące zbieżność metody iteracyjnej dla macierzy konkretnych postaci. Takie twierdzenia dotyczą między innymi macierzy przekątniowio dominującej dla metody Jacobiego. W metodzie SOR bada się przede wszystkim wartość parametru *ω*, co opisuje lemat Kahana.

1. **Cel ćwiczenia.**

Celem ćwiczenia było zapoznanie się z metodami iteracyjnymi rozwiązywania układów równań liniowych na przykładzie metody Jacobiego, a także zbadanie wpływu warunków początkowych oraz rodzajów kryteriów stopu na wyniki i czas wykonywania algorytmu. W ramach zadania domowego należało zapoznać się również z pojęciem promienia spektralnego i zbieżności algorytmów iteracyjnych oraz z metodą nadrelaksacyjną SOR.

1. **Przebieg ćwiczenia.**

Pierwsze z ćwiczeń, składało się z wielu etapów. Na początku należało przyjąć zadaną postać macierzy współczynników **A** oraz postać wektora rozwiązań wzorcowych **X** (jako permutację liczb: 1 i -1). Na tej podstawie obliczono wartości współrzędnych wektora wyrazów wolnych **B** wykorzystując równanie: ***AX = B***. Następnie, przy pomocy metody Jacobiego wyznaczono wartość wektora rozwiązań, przyjmując za znane: wektor wyrazów wolnych oraz macierz współczynników. Badano wpływ: kryteriów stopu, zastosowanych norm i ich dokładności, wektorów początkowych oraz wielkości układu na: ilość kroków algorytmu, czas obliczeń oraz dokładność rozwiązań (różnicę pomiędzy wektorem wzorcowym, a otrzymanym w odpowiednich normach). Nasza macierz A miała postać:



**Dla k=5, m=10.**

Pozostałe dwa zadania wykonano w domu. Pierwsze z nich polegało na znalezieniu promienia spektralnego macierzy iteracji wykorzystywanej w zadaniu pierwszym, dla różnych rozmiarów zadania. Badano przy tym zbieżność zadanego układu. Ostatnie zadanie polegało na wykonaniu programu rozwiązującego problem z zadania pierwszego przy pomocy metody nadrelaksacyjnej SOR oraz porównaniu jej działania z metodą Jacobiego.

1. **Opracowanie wyników.**

**4.1. Informacje wstępne.**

W każdym z zadań przyjęto, iż wektor wzorcowy **X** jest naprzemienną permutacją liczb -1 i 1, to jest:



dla *i = 1, 2, …, N*. Wartość wektora współczynników **B** obliczana była ze wzoru na postać macierzową układu równań:

***AX = B****.*

W każdym z zadań przyjęto, iż maksymalna liczba iteracji wynosić będzie 100.

Pomiary wykonywane były na komputerze przenośnym *ASUS*. Z kompilatorem G++ version 4.8.2 działającym pod systemem Windows 10 Education.

Wszystkie programy zostały napisane w języku C++. Użyto programu Code Blocks, która ułatwiała użycie biblioteki chrono i standardu C++ wersji 2011. Do pomiaru czasu zastosowana została biblioteka *chrono* dostępna w standardzie C++11 (flaga kompilacji: *-std=c++11*).

**4.2. Zadanie pierwsze**

W ramach zadania pierwszego badano zależność parametrów układu dla metody Jacobiego. Wykonano łącznie 11 serii pomiarowych.

* margines błędu: 10-5;
* wektor początkowy: 
* wyznaczanie warunku stopu: norma maksimum
* kryterium stopu: błąd wektora różnicy **X**

Poniższa tabela przedstawia otrzymane wyniki

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **N** | **Iteracje** |  |  | **Czas [s]** |
| 10 | 5 | 2.468e-008 | 1.874e-008 | - |
| 30 | 6 | 1.247e-007 | 4.666e-008 | - |
| 50 | 6 | 8.3e-007 | 3.733e-007 | - |
| 100 | 7 | 7.947e-007 | 2.102e-007 | - |
| 150 | 7 | 2.494e-006 | 7.547e-007 | - |
| 250 | 8 | 1.553e-006 | 3.506e-007 | - |
| 500 | 8 | 7.236e-006 | 1.517e-006 | - |
| 750 | 8 | 1.444e-005 | 1.494e-06 | - |
| 1 000 | 9 | 6.445e-006 | 1.095e-006 | 0,04903 |
| 5 000 | 9 | 3.687e-005 | 2.942e-006 | 1,062 |
| 10 000 | 9 | 5.366e-005 | 3.042e-006 | 4,036 |

**Zmiana kryterium stopu.**

* margines błędu: 10-5;
* wektor początkowy: 
* wyznaczanie warunku stopu: norma maksimum
* kryterium stopu: błąd rozwiązania układu

Poniższa tabela przedstawia otrzymane wyniki

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **N** | **Iteracje** |  |  | **Czas [s]** |
| 10 | 6 | 2.287e-07 | 1.347e-07 | - |
| 30 | 7 | 8.502e-07 | 3.132e-07 | - |
| 50 | 7 | 5.836e-07 | 1.545e-07 | - |
| 100 | 8 | 3.490e-06 | 1.432e-06 | - |
| 150 | 8 | 1.763e-06 | 6.306e-07 | - |
| 250 | 9 | 1.243e-06 | 3.852e-07 | - |
| 500 | 9 | 4.331e-06 | 1.050e-06 | - |
| 750 | 10 | 2.536e-06 | 5.593e-07 | - |
| 1 000 | 10 | 3.770e-06 | 7.316e-07 | 0,1146 |
| 5 000 | 11 | 7.325e-06 | 7.421e-07 | 2,963 |
| 10 000 | 11 | 1.114e-05 | 8.105e-07 | 11,72 |

**Zmiana marginesu błędu.**

* margines błędu: 10-3;
* wektor początkowy: 
* wyznaczanie warunku stopu: norma maksimum
* kryterium stopu: błąd wektora różnicy **X**

Poniższa tabela przedstawia otrzymane wyniki

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **N** | **Iteracje** |  |  | **Czas [s]** |
| 1 000 | 3 | 0.002542 | 0.0001128 | 0,02455 |
| 5 000 | 3 | 0.00267 | 0.0001081 | 0,5349 |
| 10 000 | 3 | 0.002682 | 0.0001075 | 2,042 |

* margines błędu: 10-3;
* wektor początkowy: 
* wyznaczanie warunku stopu: norma maksimum
* kryterium stopu: błąd rozwiązania układu

Poniższa tabela przedstawia otrzymane wyniki

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| N | Iteracje |  |  | Czas [s] |
| 1 000 | 4 | 0,0007512 | 5.124e-05 | 0,07964 |
| 5 000 | 4 | 0,0008657 | 2.292e-05 | 1,399 |
| 10 000 | 4 | 0,0008858 | 2.262e-05 | 5,516 |

**Zmiana sposobu wyznaczania warunku stopu i wektora początkowego.**

* margines błędu: 10-5;
* wektor początkowy: 
* wyznaczanie warunku stopu: norma euklidesowa
* kryterium stopu: błąd wektora różnicy **X**

Poniższa tabela przedstawia otrzymane wyniki

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **N** | **Iteracje** |  |  | **Czas [s]** |
| 1 000 | 27 | 2.516e-06 | 2.177e-07 | 0,117 |
| 2 500 | 32 | 1.370e-06 | 1.284e-07 | 0,7959 |
| 5 000 | 35 | 1.874e-06 | 1.290e-07 | 3,402 |
| 10 000 | 38 | 2.580e-06 | 1.280e-07 | 15,91 |

* margines błędu: 10-5;
* wektor początkowy: 
* wyznaczanie warunku stopu: norma euklidesowa
* kryterium stopu: błąd rozwiązania układu

Poniższa tabela przedstawia otrzymane wyniki

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **N** | **Iteracje** |  |  | **Czas [s]** |
| 1 000 | 29 | 3.459e-07 | 5.656e-08 | 0,3001 |
| 5 000 | 33 | 5.692e-07 | 6.612e-08 | 1,953 |
| 10 000 | 37 | 3.783e-07 | 3.681e-08 | 8,62 |
| 15 000 | 41 | 2.879e-07 | 2.235e-08 | 41,4 |

**Kolejna zmiana wektora początkowego.**

* margines błędu: 10-5;
* wektor początkowy: 
* wyznaczanie warunku stopu: norma maksimum
* kryterium stopu: błąd wektora różnicy **X**

Poniższa tabela przedstawia otrzymane wyniki

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **N** | **Iteracje** |  |  | **Czas [s]** |
| 1 000 | 17 | 8.896e-006 | 5.397e-007 | 0,0816 |
| 5 000 | 20 | 3.388e-005 | 3.004e-006 | 2,091 |
| 10 000 | 20 | 0.0001317 | 5.393e-006 | 9,039 |

**Kolejna zmiana wektora początkowego.**

* margines błędu: 10-5;
* wektor początkowy: 
* wyznaczanie warunku stopu: norma maksimum
* kryterium stopu: błąd wektora różnicy **X**

Poniższa tabela przedstawia otrzymane wyniki

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **N** | **Iteracje** |  |  | **Czas [s]** |
| 1 000 | 9 | 9.667e-006 | 1.642e-006 | 0,04512 |
| 5 000 | 10 | 2.45e-005 | 2.234e-006 | 1,06 |
| 10 000 | 10 | 4.019e-005 | 2.633e-006 | 4,747 |

**Kolejna zmiana wektora początkowego.**

* margines błędu: 10-5;
* wektor początkowy: 
* wyznaczanie warunku stopu: norma maksimum
* kryterium stopu: błąd wektora różnicy **X**

Poniższa tabela przedstawia otrzymane wyniki

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **N** | **Iteracje** |  |  | **Czas [s]** |
| 1 000 | 8 | 1.077e-005 | 1.643e-006 | 0,04428 |
| 5 000 | 8 | 3.968e-005 | 2.713e-006 | 0,8682 |
| 10 000 | 7 | 9.842e-005 | 3.849e-006 | 3,122 |

**WNIOSKI DO ĆWICZENIA:**

* **Zmiana rozmiaru układu**

Zgodnie z przewidywaniami, zmiana skali problemu ma wpływ na liczbę iteracji potrzebnych do otrzymania wyniku – im większa wartość liczby *N*, tym więcej kroków algorytm musi wykonać, aby wyznaczyć wynik z zadaną dokładnością. Zakończenie algorytmu zdeterminowane jest przez wartości normy wektora błędu między kolejnymi iteracjami lub normy wektora ***AX-B***, dlatego możliwe jest niewielkie zachwianie monotoniczności liczby iteracji.

Analogicznie, im większa liczba kroków, tym dłuższy czas realizacji algorytmu prowadzącego do otrzymania rozwiązania. Ponadto, czas wykonania każdego kroku w oczywisty sposób zwiększa się wraz ze wzrostem rozmiaru układu. Istotny wpływ na czas działania algorytmu ma również zastosowane kryterium stopu, co zostało omówione później.

Można było również przewidzieć, iż wzrost rozmiarów układu może generować większe błędy wektora rozwiązań. Istotnie, tę sytuację najlepiej obrazują: *Tabela 2.*, *Tabela 3*. W każdej z nich, warunek stopu sprawdzano dla normy maksimum wektora błędu pomiędzy kolejnymi iteracjami. Dlatego też norma maksimum otrzymanego wektora rozwiązań nie przekracza wartości 1e-5, jednak wartość normy euklidesowej tego wektora wzrasta nawet o dwa rzędy wielkości. Oznacza to, iż otrzymany wektor rozwiązań posiada dużo niewielkich błędów: dla wektora wzorcowego **x** oraz wektora otrzymanego **X** zachodzi związek:



błąd ten występuje na wielu współrzędnych otrzymanego wektora rozwiązania, co szczególnie ujawnia się dla dużych wartości rozmiaru układu *N*

* **Rodzaj kryterium stopu**

Analizując początkowe tabele umieszczone koło siebie można stwierdzić, iż wybór kryterium stopu ma niewielki wpływ na liczbę wykonywanych operacji. W każdej z dwóch tabel występujących obok siebie zostały zebrane wyniki dwóch eksperymentów przeprowadzonych z tymi samymi parametrami, różniącymi się jedynie warunkami stopu. Zauważalne jest, iż dla większych rozmiarów układu (z reguły dla *N = 1000*), wynik z zadaną dokładnością osiągany jest kilka iteracji wcześniej dla kryterium wektora różnicy **X** między kolejnymi iteracjami, niż dla kryterium normy wektora ***AX-B***.

Istotnym problemem doboru kryteriów stopu jest czas działania algorytmu. Różnica pomiędzy iteracjami jest niewielka (algorytm z kryterium normy wektora ***AX-B*** wykonuje średnio 110% ilości iteracji algorytmu z kryterium normy wektora różnicy **X** między kolejnymi iteracjami), zaś zmierzone czasy działania znacznie różnią się od siebie (pierwszy z wymienionych algorytmów osiąga czas działania przekraczający nawet 200% czasu działania drugiego algorytmu). Ten znaczny narzut czasowy zdeterminowany jest właśnie przez warunki stopu. Obliczenie różnicy dwóch wektorów jest wykonalne w czasie liniowym, natomiast pomnożenie wektora przez macierz można wykonać co najmniej w czasie kwadratowym (*N* standardowych iloczynów skalarnych dwóch wektorów o rozmiarze *N*).

Dobór kryterium stopu ma znikomy wpływ na dokładność wyników. Na ogół norma euklidesowa wektora błędu rozwiązania jest mniejsza dla kryterium normy wektora ***AX-B***.

* **Zmiana normy kryterium stopu**

Dowolnie dobrane kryterium stopu wymaga obliczenia normy *N*-elementowego wektora – zarówno w przypadku wektora różnicy **X** między kolejnymi iteracjami, jak i wektora rozwiązania ***AX-B***. Tę normę można wyznaczać na różne sposoby. W większości eksperymentów wykorzystywano normę maksimum, gdyż jest ona mniej restrykcyjna (generuje mniejsze błędy przy tej samej ilości operacji, niż norma euklidesowa).

Norma euklidesowa generuje większe błędy niż norma maksimum, dlatego spełniły się przewidywania, iż ilość kroków potrzebnych do zakończenia algorytmu przy tym samym rozmiarze macierzy jest większa*.* Szczególnie widoczne jest to, gdy obliczana jest norma wektora rozwiązania ***AX-B***.

Większa liczba iteracji przekłada się w oczywisty sposób na dłuższy czas wykonywania algorytmu. Ponownie widoczne jest to dla normy wektora rozwiązania, ponieważ czas potrzebny na wyznaczenie tego wektora jest kwadratowy, w przeciwieństwie do normy wektora różnicy **X** między kolejnymi iteracjami (którą obliczamy w czasie liniowym).

Dobór normy kryterium stopu ma również wpływ na dokładność obliczeń. Przy zastosowaniu normy maksimum w kryterium stopu, norma euklidesowa była większa nawet o dwa rzędy wielkości. Zastosowanie bardziej restrykcyjnej normy euklidesowej (oraz wymuszenie większej ilości iteracji do osiągnięcia wyniku dla tych samych parametrów), w naturalny sposób zwiększa dokładność obliczeń.

* **Wektor początkowy**

Dobór wektora początkowego ma znaczący wpływ na liczbę iteracji algorytmu potrzebną do uzyskania poprawnego wyniku. Dlatego bardzo ważna jest wiedza na temat oczekiwanej postaci rozwiązania – np. znak wartości *k*-tej współrzędnej wektora lub przedział, w jakim ta wartość się mieści. Analizując dane łatwo można stwierdzić, iż źle dobrany wektor początkowy potrafi wydłużyć liczbę iteracji do osiągnięcia tych samych danych nawet kilkukrotnie.

Większa liczba iteracji istotnie wydłuża również czas wykonywania się algorytmu dla tych samych parametrów oraz rozmiarów. Dlatego algorytmy z źle dobranymi wektorami początkowymi wykonują się znacznie dłużej, niż algorytmy z wektorami początkowymi bliskimi rozwiązaniu właściwemu. Dobór wektora początkowego ma znikomy wpływ na dokładność rozwiązań – w wyniku działania algorytmu z źle dobranym wektorem początkowym otrzymywany jest równie dokładny wynik jak dla wektora początkowego zbliżonego do oczekiwanego rozwiązania. Znacznemu pogorszeniu ulega wyłącznie liczba kroków wykonanych przez ten algorytm oraz czas jego wykonywania.

* + - * + Wektor *xi = 10000*. Ilość kroków potrzebnych do uzyskania wyniku dla tak dobranego wektora początkowego jest kilkukrotnie większa, niż dla wektora zerowego. Istotnie, wartości tego wektora są odległe od wartości wektora początkowego oraz nie mają cechy zmienności znaku
        + Wektor zerowy. Jest to uniwersalna postać wektora, szczególnie przydatna przy wektorach zadanych naprzemiennie, jak wektor wzorcowy postaci: *-1, 1, -1, 1, …* Warto zauważyć, iż wartość 0 leży dokładnie „pośrodku” wartości pojawiających się w wektorze wzorcowym oraz różnica wartości na dowolnej współrzędnej wektora wzorcowego i wektora zerowego jest niewielka (i wynosi dokładnie 1). W związku z tym, wektor zerowy faktycznie szybko przybliża do rozwiązania układu równań.
        + Wektor *-10000, 10000, -10000, 10000, …* gwarantuje on zakończenie algorytmu w mniejszej liczbie kroków niż wektor *xi = 10000*, jednakże nadal znacznie szybsza zbieżność jest uzyskiwana dla wektora zerowego
        + Wektor *0.5, -0.5, 0.5, -0.5, …* Powyższy wektor gwarantuje szybszą zbieżność niż wektor zerowy, liczba iteracji dla obu wektorów początkowych jest zbliżona. Wadą przedstawionego wektora jest niezgodność znaków w stosunku do wektora wzorcowego
        + Wektor *-0.5, 0.5, -0.5, 0.5, …* Przedstawiony wektor początkowy gwarantuje najszybszą zbieżność spośród wszystkich przedstawionych wariantów. Jego wartości są bardzo zbliżone do wartości wektora wzorcowego, na dowolnej, *k*-tej pozycji, wartości wektora wzorcowego oraz wektora początkowego mają te same znaki
* **Zmiana dokładności kryterium stopu**

Ponieważ dokładne wyznaczenie wartości wektora rozwiązań jest praktycznie niemożliwe, konieczne jest wprowadzenie pewnej tolerancji epsilonowej na normę wektora w kryterium zakończenia algorytmu. W prawie wszystkich eksperymentach przyjęto, że jest ona równa 1e-5. W niektórychdla normy o wartości 1e-3 *.*

W tym wypadku zastosowano mniejsze wymagania wobec oczekiwanego wektora rozwiązań. Spowodowało to znaczne zmniejszenie ilości kroków algorytmu wymaganych do osiągnięcia wyniku. Dla tolerancji 1e-3, algorytm wykonuje o ponad połowę mniej kroków, niż dla tolerancji 1e-5.

Wraz ze zmniejszeniem liczby iteracji, znacznemu skróceniu uległ również czas działania algorytmu.

Zwiększona tolerancja wektora wyniku doprowadza do pogorszenia jakości wektora rozwiązań -są znaczne błędy. Norma euklidesowa osiąga rząd 10-3. Z drugiej strony, dopuszczanie większych błędów i jednoczesne skrócenie czasu działania algorytmu pozwala na znalezienie w akceptowalnym czasie rozwiązań dla dużo większych układów.

**4.3. Zadanie drugie**

Zadanie to polegało na obliczeniu promienia spektralnego macierzy iteracji. Pozwala ona na stwierdzenie, czy używana metoda iteracyjna jest zbieżna dla zadanego układu. Macierz iteracji **G** w postaci ogólnej wyraża się wzorem:

***G = I – Q-1A***

gdzie: **I** – macierz jednostkowa, **Q** – ustalona macierz charakterystyczna dla zadanej metody iteracyjnej (z metodzie Jacobiego: **Q** jest macierzą diagonalną o elementach: ***Qi,i = Ai,i***), **A** – macierz współczynników równania.

Jeżeli promień spektralny macierzy **G** jest mniejszy od 1, wówczas metoda iteracyjna Jacobiego generuje ciąg zbieżny przybliżeń rozwiązania układu równań ***AX=B*** dla dowolnego wektora początkowego **X(0)**.

Badanie zbieżności przy pomocy promienia spektralnego jest z reguły niepraktyczne, chociaż powstało wiele metod przybliżania jego wartości (metoda potęgowa, metoda Kryłowa, metoda rozkładu QR i inne wraz z usprawnieniami). Dla metody Jacobiego istnieją prostsze kryteria rozstrzygalności o zbieżności, między innymi sprawdzanie przekątniowej dominacji macierzy **A**.

* Na potrzeby tego zadania, do obliczenia promienia spektralnego macierzy iteracji **G** wykorzystano metodę potęgową (iteracji wektorów). Pozwala ona na znalezienie w pierwszej kolejności największej co do modułu wartości własnej (czyli promienia spektralnego) oraz jej wektora własnego. Na początku wybrano wektor początkowy przybliżający wstępnie poszukiwany wektor własny. Składał się on wyłącznie z jedynek. Następnie, w pętli obliczano kolejne przybliżenia wektora własnego wykorzystując wzór:



* Unormowanie obliczonego wektora zapobiega zbieżności ciągu otrzymywanych wektorów do 0 oraz nieograniczonemu wzrostowi ich składowych. Obliczenia wykonywano do momentu, gdy:



gdzie: ε – dobrana stała dokładności.

Iloraz dwóch wektorów zrealizowano jako iloraz ich wartości na poszczególnych pozycjach *i = 1, 2, …, N.*

Otrzymany w wyniku działania algorytmu wektor **X(k)** jest szukanym wektorem własnym macierzy odpowiadającym promieniowi spektralnemu, którego wartość obliczono za pomocą wzoru:



Poniższa tabela przedstawia wyniki promieni spektralnych

|  |  |  |
| --- | --- | --- |
| **N** |  | **Zbieżność** |
| 10 | 0.135328 | TAK |
| 30 | 0.321965 | TAK |
| 50 | 0.444618 | TAK |
| 100 | 0.643517 | TAK |
| 150 | 0.774098 | TAK |
| 250 | 0.94998 | TAK |
| 500 | 1.20318 | NIE |
| 750 | 1.35658 | NIE |
| 1 000 | 1.46706 | NIE |
| 5 000 | 2.09898 | NIE |
| 10 000 | 2.37461 | NIE |

**WNIOSKI DO ĆWICZENIA:**

Dla rozmiarów układu *N* *> 250*, metoda iteracyjna Jacobiego traci cechę zbieżności dla dowolnego wektora początkowego **X(0)**. Pomimo tego, otrzymano rozwiązanie z satysfakcjonującą dokładnością i dla wybranych wektorów startowych. Wartość promienia spektralnego dla zadanych rozmiarów układu nie jest duża - nie przekracza 2.5. Można z tego wywnioskować że cecha zbieżności będzie zachowana dla większości przypadków.

**4.4. Zadanie trzecie**

Zadanie to polegało na zaimplementowaniu oraz przetestowaniu metody kolejnych nadrelaksacji SOR. Wykorzystano wzór:

***X(k) = (I – Q-1A)X(k) + Q-1B***

dla macierzy **Q** określonej wzorem:



**D** jest macierzą diagonalną o elementach: ***Di,i = Ai,i***, macierz **C** jest macierzą trójkątną złożoną z tych elementów macierzy **A**, które leżą pod główną diagonalą (to jest: ***Ci,j*** *= 0**dla* *j ≥ i*). Wartość rzeczywista *ω* jest parametrem rzeczywistym, którego dobór ma znaczny wpływ na zbieżność metody nadrelaksacji.

Wykorzystałam lemat Kahana oraz twierdzenie o dodatnio określonej macierzy hermitowskiej. Zgodnie z lematem Kahana, jeśli macierz **A** ma niezerową diagonalę i metoda SOR jest zbieżna dla tej macierzy, to zachodzi: *0 < ω < 2*. Inne twierdzenie głosi, iż dla dodatnio określonej macierzy hermitowskiej **A**, nieosobliwej macierzy **Q** oraz *ω < 2*, metoda SOR jest zbieżna dla dowolnie wybranego wektora początkowego.

Metoda SOR została zaimplementowana w sposób nieefektywny. Wyprowadzenie ogólnego wzoru na postać macierzy iteracji nie jest tak proste, jak w wypadku metody Jacobiego (nie występuje w tym przypadku macierz diagonalna, ale trójkątna), dlatego też wykonywanych jest wiele kosztownych operacji (między innymi odwracanie macierzy metodą Gaussa oraz mnożenie dwóch macierzy kwadratowych bez pomijania niepotrzebnych zer). Skutkiem tego jest pojawienie się działań o złożoności obliczeniowej rzędu *N3*, co oczywiście nie pozwala na wykonanie pomiarów dla dużych rozmiarów układu. Celem tego ćwiczenia jest jednak ogólne porównanie metody Jacobiego oraz metody SOR, a także zbadanie wpływu parametru rzeczywistego na jakość działania metody nadrelaksacji.

* margines błędu: 10-5;
* wektor początkowy: 
* maksymalna liczba dopuszczalnych iteracji algorytmu: 100
* kryterium stopu: błąd wektora różnicy **X**;
* wyznaczanie warunku stopu: zastosowano normę maksimum

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **N** | **Metoda kolejnych nadrelaksacji (SOR)** | | | | **Metoda Jacobiego** | | |
| **Parametr *ω*** |  |  | **Liczba kroków** |  |  | **Liczba kroków** |
| 10 | 1.5 | 2.795e-008 | 1.564e-008 | 4 | 1.559e-008 | 9.189e-009 | 8 |
| 30 | 1.5 | 6.081e-014 | 2.265e-014 | 2 | 7.937e-007 | 3.69e-007 | 10 |
| 50 | 1.5 | 8.909e-011 | 3.039e-011 | 1 | 2.277e-007 | 8.158e-008 | 12 |
| 100 | 1.5 | 2.074e-015 | 4.441e-016 | 1 | 1.027e-006 | 2.062e-007 | 14 |
| 150 | 1.5 | 3.911e-015 | 8.882e-016 | 1 | 3.105e-006 | 5.343e-007 | 15 |
| 250 | 1.5 | 5.108e-015 | 1.11e-015 | 1 | 1.549e-005 | 2.064e-006 | 16 |
| 500 | 0.0001 | 43.12 | 13.8 | 100 | 1.403e-005 | 2.036e-006 | 19 |
| 0.001 | 0.01907 | 0.008569 | 24 |
| 0.01 | 0.000767 | 0.0003677 | 3 |
| 0.1 | 1.04e-013 | 1.865e-014 | 1 |
| 0.5 | 2.298e-014 | 3.997e-015 | 1 |
| 1 | 7.588e-015 | 8.882e-016 | 1 |
| 1.5 | 8.661e-015 | 1.221e-015 | 1 |
| 750 | 1.5 | 2.539e-014 | 2.665e-015 | 1 | 1.036e-005 | 1.555e-006 | 21 |
| 1 000 | 1.5 | 7.554e-014 | 7.994e-015 | 1 | 1.496e-005 | 2.023e-006 | 22 |

**WNIOSKI DO ĆWICZENIA:**

Duży zysk w metodzie nadrelaksacji uzyskiwany jest dzięki znacznie mniejszej liczbie kroków potrzebnych do osiągnięcia poszukiwanego rozwiązania. Zysk ten osiąga się tylko wtedy kiedy parametr *ω* będzie odpowiednio dobrany. Możemy wywnioskować, że efektywniejsze implementacje algorytmu SOR są znacznie wydajniejsze od metody Jacobiego .

Wybór wektora początkowego dla algorytmu SOR ma dużo mniejszy wpływ na szybkość zbieżności (liczbę kroków), niż jest to w wypadku metody Jacobiego. Bardzo szybka zbieżność dla zaprezentowanych powyżej danych oraz optymalnego parametru *ω* następowała zarówno dla wektora zerowego, jak i wektora: *xi = 1e12, i = 1, 2, …, N.*

Wielkość układu nie decyduje w znaczny sposób o szybkości zbieżności metody Jacobiego. Przy optymalnej wartości parametru *ω* oraz wielkości układu *N = 1000*, oczekiwany wynik osiągany był z bardzo dobrą dokładnością już po pierwszej iteracji algorytmu. Podobnie, zastosowana norma w kryterium stopu ma znikomy wpływ na liczbę iteracji. Dużo bardziej opłacalne w tym przypadku jest stosowanie normy różnicy wektora **X** w kolejnych iteracjach, ponieważ wprowadzony sposób obliczania macierzy iteracji generuje duże obciążenie dla komputera. Należy jednak pamiętać, iż stosowanie tej normy nie sprawdza dokładności wyniku obliczeń, a „skok” wektora wyniku pomiędzy iteracjami. W wyniku tego możliwe są do otrzymania niedokładne wyniki (co doskonale ilustruje przykład: *N = 500, ω = 0.001*; z reguły przyjmuje się jednak: *ω > 1*).

**Parametrem silnie decydującym o zbieżności jest natomiast liczba *ω*.** Im mniejsza wartość parametru tym większe błędy i większa ilość kroków do wykonania. Co więcej, optymalna wartość parametru dla pewnej wielkości danej macierzy może już nie być optymalna dla macierzy zadanej tym samym wzorem, ale o innym rozmiarze.