

Sprawozdanie

Metody Numeryczne 2, laboratorium 1

Grzegorz Rozdzialik (D4, grupa lab. 2)

18 października 2016

1 Zadanie

Temat 1, zadanie 7:

Rozwiązywanie układu równań z macierzą trójdagonalną w dziedzinie zespolonej metodą Gaussa-Seidla w tył (Backwards Gauss-Seidel).

Szukamy rozwiązania układu równań $Ax = b$, gdzie $A \in \mathbb{C}^{n \times n}$ (A - macierz trójdagonalna, $\det A \neq 0$), $b \in \mathbb{C}^n$, $n \in \mathbb{N}$.

Układ $Ax = b$ można zapisać w postaci:

$$\begin{bmatrix}
 d_1 & u_1 & 0 & 0 & 0 & \dots & 0 \\
 l_2 & d_2 & u_2 & 0 & 0 & \dots & 0 \\
 0 & l_3 & d_3 & u_3 & 0 & \dots & 0 \\
 0 & 0 & l_4 & d_4 & u_4 & \dots & 0 \\
 & & & \ddots & \ddots & \ddots & \vdots \\
 \vdots & \vdots & & 0 & l_{n-3} & d_{n-3} & u_{n-3} & 0 & 0 \\
 & & & & 0 & l_{n-2} & d_{n-2} & u_{n-2} & 0 \\
 & & & & & 0 & l_{n-1} & d_{n-1} & u_{n-1} \\
 0 & 0 & \dots & & & & 0 & l_n & d_n
 \end{bmatrix}
 \begin{bmatrix}
 x_1 \\
 x_2 \\
 x_3 \\
 \vdots \\
 x_{n-1} \\
 x_n
 \end{bmatrix}
 =
 \begin{bmatrix}
 b_1 \\
 b_2 \\
 b_3 \\
 \vdots \\
 b_{n-1} \\
 b_n
 \end{bmatrix} \quad (1)$$

gdzie $l_i, d_i, u_i, b_i, x_i \in \mathbb{C}^n$ dla każdego $i = 1, \dots, n$.

2 Opis metody

Metoda Gaussa-Seidla w tył jest iteracyjną metodą rozwiązywania układów równań liniowych. Jest ona podobna do zwykłej metody Gaussa-Seidla, przy czym w tym przypadku obliczenia rozpoczynamy od ostatnich elementów wektora rozwiązań.

Mając przybliżenie początkowe $x^{(0)} \in \mathbb{C}^n$ tworzymy ciąg kolejnych przybliżeń rozwiązań $\{x^{(k)}\}$ ($k = 1, 2, \dots$) taki, że

$$\lim_{k \rightarrow +\infty} x^{(k)} = x$$

Po wymnożeniu lewej strony układu (1) otrzymujemy następujący układ równań:

$$\left\{ \begin{array}{l} d_1 x_1 + u_1 x_2 = b_1 \\ l_2 x_1 + d_2 x_2 + u_2 x_3 = b_2 \\ l_3 x_2 + d_3 x_3 + u_3 x_4 = b_3 \\ \vdots \\ l_i x_{i-1} + d_i x_i + u_i x_{i+1} = b_i \\ \vdots \\ l_{n-1} x_{n-2} + d_{n-1} x_{n-1} + u_{n-1} x_n = b_{n-1} \\ l_n x_{n-1} + d_n x_n = b_n \end{array} \right.$$

Następnie z i -tego równania wyliczamy:

$$x_i = \frac{b_i - l_i x_{i-1} - u_i x_{i+1}}{d_i} \quad \text{dla } i = 2, \dots, n-1$$

Wyjątkiem są równania pierwsze oraz ostatnie, ponieważ nie istnieją współczynniki odpowiednio l_1 oraz u_n , stąd:

$$x_1 = \frac{b_1 - u_1 x_2}{d_1}$$

$$x_n = \frac{b_n - l_n x_{n-1}}{d_n}$$

W k -tym kroku obliczamy przybliżenie $x^{(k+1)}$ w taki sposób, że najpierw obliczamy $x_n^{(k+1)}$, następnie zmniejszamy indeks dolny, obliczając kolejno $x_{n-1}^{(k+1)}, \dots, x_1^{(k+1)}$.

Przy wyliczaniu $x_i^{(k+1)}$ używamy najbardziej aktualnych przybliżeń (za-
miast $x_{i+1}^{(k)}$ używamy $x_{i+1}^{(k+1)}$, wyjątkiem jest $x_n^{(k)}$).

Należy oddzielnie przybliżać część rzeczywistą oraz część urojoną.

3 Implementacja metody

Wprowadźmy następujące oznaczenia:

$$\begin{array}{ll} u_{r,j} = \operatorname{Re}(u_j) & u_{i,j} = \operatorname{Im}(u_j) \\ d_{r,j} = \operatorname{Re}(d_j) & d_{i,j} = \operatorname{Im}(d_j) \\ l_{r,j} = \operatorname{Re}(l_j) & l_{i,j} = \operatorname{Im}(l_j) \\ x_{r,j}^{(k)} = \operatorname{Re}(x_j^{(k)}) & x_{i,j}^{(k)} = \operatorname{Im}(x_j^{(k)}) \\ b_{r,j} = \operatorname{Re}(b_j) & b_{i,j} = \operatorname{Im}(b_j) \end{array}$$

dla $j = 1, \dots, n$.

Poniższe wzory opisują sposób obliczenia kolejnego przybliżenia $x^{(k+1)}$:

$$x_{r,n}^{(k+1)} = \frac{b_{r,n} + d_{i,n}x_{i,n}^{(k)} - l_{r,n}x_{r,n-1}^{(k)} + l_{i,n}x_{i,n-1}^{(k)}}{d_{r,n}}$$

$$x_{i,n}^{(k+1)} = \frac{b_{i,n} - d_{i,n}x_{r,n}^{(k)} - l_{r,n}x_{i,n-1}^{(k)} - l_{i,n}x_{r,n-1}^{(k)}}{d_{r,n}}$$

dla $j = (n-1), \dots, 2$

$$x_{r,j}^{(k+1)} = \frac{b_{r,j} + d_{i,j}x_{i,j}^{(k)} - l_{r,j}x_{r,j-1}^{(k)} + l_{i,j}x_{i,j-1}^{(k)} - u_{r,j}x_{r,j+1}^{(k+1)} + u_{i,j}x_{i,j+1}^{(k+1)}}{d_{r,j}}$$

$$x_{i,j}^{(k+1)} = \frac{b_{i,j} - d_{r,j}x_{i,j}^{(k)} - l_{r,j}x_{i,j-1}^{(k)} - l_{i,j}x_{r,j-1}^{(k)} - u_{i,j}x_{r,j+1}^{(k+1)} - u_{r,j}x_{i,j+1}^{(k+1)}}{d_{r,j}}$$

$$x_{r,1}^{(k+1)} = \frac{b_{r,1} + d_{i,1}x_{i,1}^{(k)} - u_{r,1}x_{r,2}^{(k+1)} + u_{i,1}x_{i,2}^{(k+1)}}{d_{r,1}}$$

$$x_{i,1}^{(k+1)} = \frac{b_{i,1} - d_{r,1}x_{i,1}^{(k)} - u_{i,1}x_{r,2}^{(k+1)} - u_{r,1}x_{i,2}^{(k+1)}}{d_{r,1}}$$

4 Warunek stopu

Z możliwych warunków stopu wybrałem warunek Gill'a. Obliczenia trwają dopóki spełniona jest nierówność

$$\|x^{(k+1)} - x^{(k)}\| \geq \varepsilon \|x^{(k)}\| + \delta$$

gdzie $\varepsilon, \delta > 0$.

Dodatkowo w implementacji nałożyłem restrykcję na maksymalną ilość iteracji. Po 10000 iteracjach obliczenia nie będą kontynuowane.

5 Poprawność algorytmu

W ogólności metoda Gaussa-Seidla, a więc i metoda Gaussa-Seidla w tył, jest zbieżna globalnie dla macierzy diagonalnie silnie dominujących (kolumnowo lub wierszowo) oraz macierzy dodatnio określonych.

Po przeprowadzeniu testów odkryłem, że zaimplementowany algorytm działa dobrze, jeżeli część rzeczywista elementów z diagonalu dominuje w rzędzie lub kolumnie, a część urojona tego elementu jest znacznie mniejsza od części rzeczywistej.

Jeżeli część urojona elementów z diagonalu jest porównywalna do części rzeczywistej, to algorytm daje złe wyniki.

6 Przykłady

We wszystkich przykładach $\varepsilon = \text{eps} \approx 2.22 * 10^{-16}$, $\delta = 0$. Rzędy błędu zostały obliczone na podstawie porównania rozwiązania z rozwiązaniem obliczonym przez funkcję *linsolve* dostępną w Matlabie.

Układ 1 $n = 5$

Elementy na diagonalu są znacznie większe od elementów na dwóch innych pasmach. Dodatkowo nie zawierają części urojonej.

$$\begin{bmatrix} 61354 & 44 + 59i & 0 & 0 & 0 \\ 60 + 88i & 83844 & 54 + 27i & 0 & 0 \\ 0 & 84 + 40i & 89650 & 4 + 42i & 0 \\ 0 & 0 & 81 + 84i & 48721 & 55 + 50i \\ 0 & 0 & 0 & 80 + 63i & 25148 \end{bmatrix} x = \begin{bmatrix} 20 + 62i \\ 38 + 10i \\ 71 + 37i \\ 2 + 50i \\ 28 + 32i \end{bmatrix}$$

Znalezione rozwiązanie:

$$x = \begin{bmatrix} 0.000325764927490767 + 0.00101000840330470i \\ 0.000453671976952820 + 0.000117676750075580i \\ 0.000792072530469199 + 0.000411492905043098i \\ 4.04854673760366e - 05 + 0.00102061129748172i \\ 0.00111583663410018 + 0.00126911883695549i \end{bmatrix}$$

Liczba iteracji: 9
 Rząd błędu: -6
 Czas działania: 0.290354ms

Układ 2 $n = 4$

Elementy na diagonalu są znacznie większe od elementów na dwóch innych pasmach. Ich część urojona jest porównywalna z rzeczywistą.

$$\begin{bmatrix} 12339 + 18978i & 22 + 91i & 0 & 0 \\ 39 + 74i & 17852 + 19484i & 60 + 64i & 0 \\ 0 & 95 + 35i & 18710 + 12993i & 24 + 14i \\ 0 & 0 & 94 + 34i & 15539 + 14995i \end{bmatrix} x = \begin{bmatrix} 79 + 40i \\ 80 + 56i \\ 35 + 70i \\ 33 + 83i \end{bmatrix}$$

Znalezione rozwiązanie: **brak** - przekroczono maksymalną liczbę iteracji.

Czas działania: 97.203925ms

Układ 3 $n = 4$

Elementy na diagonalu są porównywalne z elementami na dwóch innych pasmach. Ich część urojona jest porównywalna z rzeczywistą.

$$\begin{bmatrix} 19 + 53i & 43 + 19i & 0 & 0 \\ 71 + 11i & 76i & 44 + 50i & 0 \\ 0 & 48 + 54i & 21 + 65i & 48 + 88i \\ 0 & 0 & 28 + 55i & 54 + 68i \end{bmatrix} x = \begin{bmatrix} 83 + 32i \\ 69 + 16i \\ 71 + 74i \\ 18 + 57i \end{bmatrix}$$

Znalezione rozwiązanie: **brak** - przekroczono maksymalną liczbę iteracji.

Czas działania: 94.710700ms

Układ 4 $n = 5$

Elementy na diagonalu są porównywalne z elementami na dwóch innych pasmach. Nie mają części urojonej.

$$\begin{bmatrix} 95 & 16 + 7i & 0 & 0 & 0 \\ 17 + 72i & 35 & 27 + 88i & 0 & 0 \\ 0 & 83 + 93i & 5 & 34 + 15i & 0 \\ 0 & 0 & 3 + 56i & 93 & 20 + 80i \\ 0 & 0 & 0 & 1 + 83i & 78 \end{bmatrix} x = \begin{bmatrix} 36 + 89i \\ 30 + 41i \\ 68 + 50i \\ 5 + 36i \\ 33 + 99i \end{bmatrix}$$

Znalezione rozwiązanie: **brak** - przekroczono maksymalną liczbę iteracji.

Czas działania: 101.725476ms

Układ 5 $n = 5$

Elementy na diagonalu są większe od elementów na dwóch innych pasmach. Nie mają części urojonej.

$$\begin{bmatrix} 317 & 15 + 66i & 0 & 0 & 0 \\ 19 + 4i & 363 & 67 + 41i & 0 & 0 \\ 0 & 4 + 80i & 337 & 90 + 66i & 0 \\ 0 & 0 & 61 + 39i & 394 & 68 + 40i \\ 0 & 0 & 0 & 73 + 6i & 258 \end{bmatrix} x = \begin{bmatrix} 62 + 86i \\ 16 + 43i \\ 33 + 58i \\ 27 + 13i \\ 54 + 58i \end{bmatrix}$$

Znalezione rozwiązanie:

$$x = \begin{bmatrix} 0.210003815360937 + 0.259969953085646i \\ 0.0367475440953636 + 0.0776124641037519i \\ 0.0937110364191769 + 0.160199771371148i \\ 0.0603979474198726 - 0.0309935213767811i \\ 0.191492204302669 + 0.232171082852658i \end{bmatrix}$$

Liczba iteracji: 19

Rząd błędu: -1

Czas działania: 0.722562ms

Układ 6 $n = 100$

Elementy na diagonalu porównywalne z elementami na dwóch innych pasmach (zarówno część rzeczywista, jak i urojona).

	BGS	linsolve
Rząd błędu	<i>brak</i>	4
Czas działania	901.49ms	0.43ms
Ilość iteracji	<i>nie dotyczy</i>	<i>nie dotyczy</i>

Układ 7 $n = 100$

Elementy na diagonalu są większe (około ośmiokrotnie) od elementów na dwóch innych pasmach. Nie mają części urojonej.

	BGS	linsolve
Rząd błędu	1	4
Czas działania	0.26ms	0.45ms
Ilość iteracji	30	<i>nie dotyczy</i>

Układ 8 $n = 100$

Elementy na diagonalu są znacznie większe (około pięćdziesięciokrotnie) od elementów na dwóch innych pasmach. Ich część urojona jest porównywalna z częściami urojonymi elementów z innych pasm.

	BGS	linsolve
Rząd błędu	1	4
Czas działania	0.11ms	0.37ms
Ilość iteracji	12	<i>nie dotyczy</i>

Układ 9 $n = 100$

Elementy na diagonalu są znacznie większe (około pięćdziesięciokrotnie) od elementów na dwóch innych pasmach, zarówno część urojona, jak i rzeczywista.

	BGS	linsolve
Rząd błędu	<i>brak</i>	4
Czas działania	748.37ms	0.41ms
Ilość iteracji	10000	<i>nie dotyczy</i>

7 Wnioski

1. Aby algorytm zadziałał, część rzeczywista elementów na diagonalu musi być większa od części urojonej tychże elementów. Dodatkowo macierz powinna być diagonalnie dominująca (wierszowo lub kolumnowo).
2. Im elementy z głównej diagonalu są większe od elementów z pozostałych dwóch pasm, tym dokładność algorytmu jest lepsza. Można to zauważyć porównując układ 1, w którym elementy na diagonalu były znacznie większe od pozostałych, oraz układ 5, gdzie elementy z diagonalu były maksymalnie pięciokrotnie większe.
3. Wraz ze wzrostem ilości równań (a więc i niewiadomych) metoda Gaussa-Seidla w tył staje się szybsza w porównaniu ze zwykłą funkcją *linsolve* (przykład: układ 7 oraz 8), o ile tylko metoda jest zbieżna.

4. Metoda Gaussa-Seidla w tył działa znacznie dłużej od *linsolve* dla dużych macierzy, jeżeli metoda nie jest zbieżna (przykład: układ 9).

8 Skrypt testujący

Do metody został dołączony skrypt testujący, pozwalający na wygenerowanie losowych wartości elementów i sprawdzenie poprawności działania metody. Możliwe parametry to ustawienie oddzielnie zakresu części rzeczywistej i części urojonej dla:

- rozmiaru układu równań n
- elementów na diagonalu macierzy A
- elementów poza diagonalą macierzy A
- elementów wektora b
- elementów wektora przybliżenia początkowego $x^{(0)}$
- parametrów ε, δ
- maksymalnej liczby iteracji

Skrypt pokazuje rząd błędu (porównując najpierw z rozwiązaniem obliczonym przez *linsolve*, następnie jako $\|Ax - b\|$), ilość iteracji oraz czas potrzebny na obliczenia.

9 Interfejs graficzny

Do metody został dołączony interfejs graficzny. Znajduje się on w pliku *bgs-GUI.fig*. Pozwala na zmianę zakresu losowanych elementów z podziałem na elementy na diagonalu macierzy A , poza diagonalą (ale nadal w macierzy A), wektora b oraz wektora przybliżenia początkowego $x^{(0)}$. Dodatkowo możliwe

jest zmienienie rozmiaru n układu równań oraz dostosowanie parametrów stopu ε, δ oraz maksymalnej liczby iteracji.

Rozwiązanie układu $Ax = b$ z macierzą trójdziagonalną o elementach zespolonych metodą Gaussa-Seidla w tył (BGS)

Zakres elementów z diagonalą
 Część rzeczywista: 4000 do 9000
 Część urojona: 0 do 100

Zakres elementów pod i nad diagonalą
 Część rzeczywista: 0 do 100
 Część urojona: 0 do 100

Zakres elementów wektora b
 Część rzeczywista: 0 do 100
 Część urojona: 0 do 100

Zakres elementów do przybliżenia początkowego
 Część rzeczywista: 0 do 100
 Część urojona: 0 do 100

Parametry
 Epsilon: 1e-17
 Delta: 0
 Max. iter: 10000
 n: 20

Oblicz

Statystyki
 Rząd błędu BGS: 0
 Ilość iteracji: 12
 Czas działania BGS: 0.08ms
 Rząd błędu linsolve: 3
 Czas działania linsolve: 0.06ms

Autor: Grzegorz Rozdziałik (D4)
 Grupa laboratoryjna nr 2

Interfejs graficzny jest graficzną wersją skryptu testującego *testScript.m* i pokazuje następujące wyniki:

- rząd błędu rozwiązania obliczonego za pomocą *linsolve* dostępnego w Matlabie oraz czas działania tych obliczeń (w milisekundach)
- rząd błędu liczony względem wektora b ($\|Ax - b\|$)
- ilość iteracji
- czas działania metody Gaussa-Seidla w tył (w milisekundach)

10 Bibliografia

- Metoda Gaussa-Seidla - Wikipedia, wolna encyklopedia
- P. Tatjewski *Metody numeryczne*, Warszawa: Oficyna Wydawnicza PW, 2013