24 października 2019

Michał Gozdera
grupa: G1
nr indeksu: 298869

# 18. Rozwiązanie układu równań z macierzą trójdiagonalną w dziedzinie zespolonej metodą Jacobiego

Projekt nr 1

## 1 Wstęp

Celem projektu jest rozwiązanie układu równań z macierzą trójdiagonalną w dziedzinie zespolonej. Wyjściowy układ (czyli macierz układu i wektor wyrazów wolnych) sprowadzamy do układu o elementach rzeczywistych, a następnie rozwiązujemy go metodą Jacobiego (macierz układu jest dana jako trzy wektory - niezerowe diagonale).

Ideą metody Jacobiego jest stworzenie macierzy iteracji i wektora wyrazów wolnych oraz nadpisywanie wektora rozwiązań w kolejnych iteracjach aż do momentu osiągnięcia zadowalającej dokładności rozwiązania. Jeżeli metoda Jacobiego okaże się rozbieżna dla podanego układu wejściowego, zaznaczamy to odpowiednim komentarzem.

Fakt, że macierz układu jest trójdiagonalna pozwala na swego rodzaju "skompresowanie" macierzy iteracji tak, by zawierała tylko niezerowe diagonale. Jest to kluczowe w szybkości działania algorytmu i umożliwia obliczenia na dużych macierzach.

# 2 Opis metody i programu obliczeniowego

Pierwszą czynnością, jaką należy wykonać jest sprowadzenie układu o elementach zespolonych do układu o elementach rzeczywistych oraz skonstruowanie macierzy iteracji w metodzie Jacobiego. Dla układu $Cz = d$, gdzie $C = A + iB$ ($C$ jest rozmiaru $N \times N$), $z = q + ip$, tworzymy macierz $M$ o rozmiarze $2N \times 2N$ w następujący sposób:

$$M = \begin{pmatrix} A & -B \\ B & A \end{pmatrix}.$$

Wektor $z$ o elementach zespolonych sprowadzamy do wektora $s$ o elementach rzeczywistych jak poniżej:

$$s = \begin{pmatrix} q \\ p \end{pmatrix}.$$

Analogicznie wektor $d$. Macierzą układu, dla którego wykonywać będziemy iteracje jest $M$, a wektorem wyrazów wolnych - wektor powstały z $d$. W ten sposób sprowadziliśmy układ zespolony do rzeczywistego. Teraz należy stworzyć macierz iteracji.

Za powyższą część algorytmu odpowiedzialna jest w programie funkcja `[G] = get_iter_matrix(a, c, b)`. Argumentami wejściowymi `get_iter_matrix` są trzy kolejne diagonale pierwotnej macierzy układu $C$ ($c$ - główna diagonala, $a$ - przekątna poniżej $c$, $b$ - przekątna powyżej $c$). Argumentem wyjściowym jest macierz G o rozmiarze $N \times 5$. Ponieważ $A$ i $B$ są trójdiagonalne, M ma 9 niezerowych diagonali i wygląda następująco:

$$M = \begin{pmatrix} c_{r1} & b_{r1} & 0 & \dots & 0 & -c_{i1} & -b_{i1} & 0 & \dots & 0 \\ a_{r1} & c_{r2} & b_{r2} & \ddots & \vdots & -a_{i1} & -c_{i2} & -b_{i2} & \ddots & \vdots \\ 0 & a_{r2} & c_{r3} & \ddots & 0 & 0 & -a_{i2} & -c_{i3} & \ddots & 0 \\ \vdots & \ddots & \ddots & \ddots & b_{rN-1} & \vdots & \ddots & \ddots & \ddots & -b_{iN-1} \\ 0 & \dots & 0 & a_{rN-1} & c_{rN} & 0 & \dots & 0 & -a_{iN-1} & -c_{iN} \\ c_{i1} & b_{i1} & 0 & \dots & 0 & c_{r1} & b_{r1} & 0 & \dots & 0 \\ a_{i1} & c_{i2} & b_{i2} & \ddots & \vdots & a_{r1} & c_{r2} & b_{r2} & \ddots & \vdots \\ 0 & a_{i2} & c_{i3} & \ddots & 0 & 0 & a_{r2} & c_{r3} & \ddots & 0 \\ \vdots & \ddots & \ddots & \ddots & b_{iN-1} & \vdots & \ddots & \ddots & \ddots & b_{rN-1} \\ 0 & \dots & 0 & a_{iN-1} & c_{iN} & 0 & \dots & 0 & a_{rN-1} & c_{rN} \end{pmatrix},$$

gdzie $a_{rk}$, $c_{rl}$, $b_{rk}$ - części rzeczywiste, $a_{ik}$, $c_{il}$, $b_{ik}$ - części urojone elementów diagonali odpowiednio: a, c, b ($1 \leqslant k \leqslant N-1$, $1 \leqslant l \leqslant N$). "Tradycyjna" macierz iteracji w metodzie Jacobiego wyglądałaby zatem tak:

$$\begin{pmatrix}
0 & -\frac{b_{r1}}{c_{r1}} & 0 & \dots & 0 & \frac{c_{i1}}{c_{r1}} & \frac{b_{i1}}{c_{r1}} & 0 & \dots & 0 \\
-\frac{a_{r1}}{c_{r2}} & 0 & -\frac{b_{r2}}{c_{r2}} & \ddots & \vdots & \frac{a_{i1}}{c_{r2}} & \frac{c_{i2}}{c_{r2}} & \frac{b_{i2}}{c_{r2}} & \ddots & \vdots \\
0 & -\frac{a_{r2}}{c_{r3}} & 0 & \ddots & 0 & 0 & \frac{a_{i2}}{c_{r3}} & \frac{c_{i3}}{c_{r3}} & \ddots & 0 \\
\vdots & \ddots & \ddots & \ddots & -\frac{b_{rN-1}}{c_{rN-1}} & \vdots & \ddots & \ddots & \ddots & \frac{b_{iN-1}}{c_{rN-1}} \\
0 & \dots & 0 & -\frac{a_{rN-1}}{c_{rN}} & 0 & 0 & \dots & 0 & \frac{a_{iN-1}}{c_{rN}} & \frac{c_{iN}}{c_{rN}} \\
-\frac{c_{i1}}{c_{r1}} & -\frac{b_{i1}}{c_{r1}} & 0 & \dots & 0 & 0 & -\frac{b_{r1}}{c_{r1}} & 0 & \dots & 0 \\
-\frac{a_{i1}}{c_{r2}} & -\frac{c_{i2}}{c_{r2}} & -\frac{b_{i2}}{c_{r2}} & \ddots & \vdots & -\frac{a_{r1}}{c_{r2}} & 0 & -\frac{b_{r2}}{c_{r2}} & \ddots & \vdots \\
0 & -\frac{a_{i2}}{c_{r3}} & -\frac{c_{i3}}{c_{r3}} & \ddots & 0 & 0 & -\frac{a_{r2}}{c_{r3}} & 0 & \ddots & 0 \\
\vdots & \ddots & \ddots & \ddots & -\frac{b_{iN-1}}{c_{rN-1}} & \vdots & \ddots & \ddots & \ddots & -\frac{b_{rN-1}}{c_{rN-1}} \\
0 & \dots & 0 & -\frac{a_{iN-1}}{c_{rN}} & -\frac{c_{iN}}{c_{rN}} & 0 & \dots & 0 & -\frac{a_{rN-1}}{c_{rN}} & 0
\end{pmatrix}. \quad (1)$$

Łatwo można jednak zauważyć, że przez to, iż $C$ jest macierzą trójdiagonalną, tylko niektóre (a ponadto ściśle określone) elementy macierzy iteracji (1) są niezerowe, mianowicie jest to osiem przekątnych. Informacja o tym, jakie liczby znajdują się na tych diagonalach wystarczy, aby zrealizować metodę Jacobiego.

Macierz G, zastępującą macierz (1) tworzymy jak poniżej:

$$G = \begin{pmatrix}
0 & -\frac{b_{r1}}{c_{r1}} & 0 & \frac{c_{i1}}{c_{r1}} & \frac{b_{i1}}{c_{r1}} \\
-\frac{a_{r1}}{c_{r2}} & -\frac{b_{r2}}{c_{r2}} & \frac{a_{i1}}{c_{r2}} & \frac{c_{i2}}{c_{r2}} & \frac{b_{i2}}{c_{r2}} \\
\vdots & \vdots & \vdots & \vdots & \vdots \\
-\frac{a_{rN-2}}{c_rN-1} & -\frac{b_{rN-1}}{c_{rN-1}} & \frac{a_{iN-2}}{c_{rN-1}} & \frac{c_{iN-1}}{c_{rN-1}} & \frac{b_{iN-1}}{c_{rN-1}} \\
-\frac{a_{rN-1}}{c_{rN}} & 0 & \frac{a_{iN-1}}{c_{rN}} & \frac{c_{iN}}{c_{rN}} & 0
\end{pmatrix}.$$

Pierwsza kolumna $G$ to połowa czwartej (licząc od dołu) niezerowej przekątnej macierzy (1), druga kolumna $G$ - połowa piątej. Drugą połowę każdej z tych przekątnych pomijamy, ponieważ jest ona taka sama jak pierwsza. Trzecia, czwarta i piąta kolumna $G$ to odpowiednio szósta, siódma i ósma przekątna macierzy (1) licząc od dołu. Przekątne: pierwszą, drugą i trzecią pomijamy, ponieważ różnią się one względem szóstej, siódmej i ósmej jedynie znakiem. Ponieważ przekątne mają różne długości, w $G$ dodajemy pojedyncze zera w pierwszym lub ostatnim wierszu. Ułatwi to implementację algorytmu.

Otrzymujemy w powyżej opisany sposób macierz rozmiaru $N \times 5$.

Przystępujemy teraz do rozwiązania właściwego problemu, tzn. układu równań liniowych metodą Jacobiego. Funkcja, którą się posłużymy to `[`$x$`, `$iter$`] = jacobi(`$a$`, `$c$`, `$b$`, `$d$`, `$tol$`)`. Argumentami wejściowymi tej funkcji są trzy diagonale wejściowej macierzy: $a$, $c$, $b$, wektor wyrazów wolnych układu równań $d$ oraz dokładność obliczeń $tol$, określająca, kiedy należy przerwać iteracje. Na wyjściu otrzymujemy wektor rozwiązań $x$ (sprowadzony z powrotem do postaci zespolonej) oraz liczbę iteracji jako drugi argument (w przypadku rozbieżności wyświetlany jest odpowiedni komunikat).

Funkcja `jacobi`, zgodnie z metodą Jacobiego sprowadza wektor zespolony $d$ do postaci:

$$d = \begin{pmatrix} \frac{d_{r1}}{c_{r1}} \\ \vdots \\ \frac{d_{rN}}{c_{rN}} \\ \frac{d_{i1}}{c_{r1}} \\ \vdots \\ \frac{d_{iN}}{c_{rN}} \end{pmatrix},$$

gdzie $d_{rl}$ - części rzeczywiste, $d_{il}$ - części urojone pierwotnego (wejściowego) wektora $d$ ($1 \leqslant l \leqslant N$). Jako początkową wartość wektora $x$ przyjmujemy wektor zerowy o $2N$ wierszach.Niech $G1$ - macierz zwracana przez `get_iter_matrix`, $G2$ - macierz powstała z $G1$ poprzez zmianę znaku wszystkich elementów kolumn nr 3, 4 i 5 oraz zamianę miejscami kolumn nr 1 i 2 z 3, 4 i 5. Następnie rozpoczynamy iteracje (niech $k$ oznacza $k$-tą iterację, $x_k$ - wektor rozwiązań w $k$-tej iteracji). Pierwsza współrzędna wektora $x_{k+1}$ to suma pierwszej współrzędnej wektora $d$ i iloczynu pierwszego wiersza macierzy $G$ oraz wektora $x_m$ złożonego z odpowiednich współrzędnych wektora $x$ (takich, że w macierzy (1) element w pierwszym wierszu i kolumnie o numerze danej współrzędnej wektora $x$ jest niezerowy), tzn.:

$$x_m = [0, x_k(2), 0, x_k(N+1), x_k(N+2)]';$$
$$x_{k+1}(1) = G1(1,:) * x_m + d(1);$$

Współrzędne od drugiej do N-1 obliczamy z analogicznym uzasadnieniem:

$$x_m = [x_k(i-1), x_k(i+1), x_k(N+i-1), x_k(N+i), x_k(N+i+1)]';$$

$$x_{k+1}(i) = G1(i,:) * x_m + d(i);$$

Dla współrzędnej nr $N$:

$$x_m = [x_k(N-1), 0, x_k(2*N-1), x_k(2*N), 0]'$$

$$x_{k+1}(N) = G1(N,:) * x_m + d(N);$$

Dla współrzędnych od $N+1$ do $2N$ postępujemy analogicznie, używając tym razem macierzy $G2$.

Iteracje wykonujemy dopóki norma różnicy $x_k - x_{k+1}$ będzie większa lub równa wartości *tol*. Jeżeli liczba iteracji przekroczy 100000 lub norma wektora osiągnie wartość większą niż $\frac{1}{\sqrt{eps}}$, oznacza to, że dla podanego przypadku metoda Jacobiego jest rozbieżna. Informujemy o tym komunikatem: "Function stopped - in this case Jacobi method is divergent.".

# 3 Opis eksperymentów

Podczas badania poprawności powyższego algorytmu należy powołać się na poniższe twierdzenie:

**Twierdzenie.** (O ZBIEŻNOŚCI METOD ITERACYJNYCH) *Metoda iteracyjna* $x_{k+1} = B * x_k + c$ *jest zbieżna globalnie* $\iff \rho(B) < 1$.

Funkcja `[`$M, d$`] = get_matrix(`$N, spectral\_radius\_greater\_than\_1$`)` zwraca losowy wektor wyrazów wolnych $d$ oraz macierz trójdiagonalną $M$ rozmiaru $N \times N$, przy czym jeżeli $spectral\_radius\_greater\_than\_1$ jest równe zero, $M$ jest macierzą, której macierz iteracji w metodzie Jacobiego ma promień zbieżności mniejszy od 1, a jeżeli $spectral\_radius\_greater\_than\_1$ jest równe jeden - większy od 1. Kilkukrotnie losujemy różnej wielkości macierze, modyfikując również drugi argument wywołania `get_matrix`. Uzyskujemy w ten sposób przykłady, dla których łatwo przetestować zbieżność i poprawność obliczeń. Modyfikować możemy także argument *tol* określający dokładność oraz porównywać liczbę iteracji (Funkcja ta nie służy do generowania macierzy bardzo dużych rozmiarów, ponieważ musimy w niej utworzyć macierz iteracji; jest ona przeznaczona do przetestowania poprawności algorytmu).

# 4 Przykłady obliczeniowe

1. Przykładowa macierz (której promień spektralny macierzy iteracji jest mniejszy od 1) i wektor wyrazów wolnych:

$$M1 = \begin{pmatrix} 0.7572 + 0.3804i & 0.0759 + 0.0540i \\ 0.9172 + 0.2858i & 0.7537 + 0.5678i \end{pmatrix} \qquad d1 = \begin{pmatrix} 0.8147 + 0.1270i \\ 0.9058 + 0.9134i \end{pmatrix}$$

   Wyniki wywołania: `jacobi(diag(M1, -1), diag(M1), diag(M1, 1), d1,` $10^{-9}$`)`:

$$ans = \begin{pmatrix} 0.8701 - 0.4044i \\ 0.6536 + 0.8815i \end{pmatrix}$$

   Wyniki wywołania: `linsolve(M1, d1)`:

$$ans = \begin{pmatrix} 0.8701 - 0.4044i \\ 0.6536 + 0.8815i \end{pmatrix}$$

2. Przykładowa macierz (której promień spektralny macierzy iteracji jest mniejszy od 1) i wektor wyrazów wolnych:

$$M2 = \begin{pmatrix} 0.8872 + 0.2144i & 0.3157 + 0.2309i & 0.0000 + 0.0000i & 0.0000 + 0.0000i \\ 0.3112 + 0.7653i & 0.7943 + 0.0807i & 0.1700 + 0.6474i & 0.0000 + 0.0000i \\ 0.0000 + 0.0000i & 0.0724 + 0.0939i & 0.8910 + 0.3638i & 0.8258 + 0.2274i \\ 0.0000 + 0.0000i & 0.0000 + 0.0000i & 0.6143 + 0.1963i & 0.8751 + 0.2278i \end{pmatrix}$$

$$d2 = \begin{pmatrix} 0.9373 + 0.9422i \\ 0.5997 + 0.7822i \\ 0.5354 + 0.9840i \\ 0.2413 + 0.8097i \end{pmatrix}$$

Wyniki wywołania: `jacobi(diag(M2, -1), diag(M2), diag(M2, 1), d2, 10^-3)` (mała dokładność):

$$ans = \begin{pmatrix} 0.6644 + 0.8166i \\ 1.1931 - 0.6336i \\ 0.6243 + 0.1344i \\ 0.0449 + 0.6792i \end{pmatrix}.$$

Liczba iteracji = 384.

Wyniki wywołania: `jacobi(diag(M2, -1), diag(M2), diag(M2, 1), d2, 10^-9)` (duża dokładność):

$$ans = \begin{pmatrix} 0.6643 + 0.8164i \\ 1.1929 - 0.6339i \\ 0.6241 + 0.1344i \\ 0.0447 + 0.6793i \end{pmatrix}.$$

Liczba iteracji = 1016.

Wyniki wywołania: `linsolve(M2, d2)`:

$$ans = \begin{pmatrix} 0.6643 + 0.8164i \\ 1.1929 - 0.6339i \\ 0.6241 + 0.1344i \\ 0.0447 + 0.6793i \end{pmatrix}.$$

3. Przykładowa macierz (której promień spektralny macierzy iteracji jest większy od 1) i wektor wyrazów wolnych:

$$M3 = \begin{pmatrix} 0.8797 + 0.1785i & 0.6638 + 0.6436i & 0.0000 + 0.0000i & 0.0000 + 0.0000i \\ 0.9267 + 0.2680i & 0.0039 + 0.5213i & 0.2319 + 0.9308i & 0.0000 + 0.0000i \\ 0.0000 + 0.0000i & 0.0213 + 0.1504i & 0.9813 + 0.5549i & 0.1726 + 0.2581i \\ 0.0000 + 0.0000i & 0.0000 + 0.0000i & 0.2209 + 0.0686i & 0.3077 + 0.1866i \end{pmatrix}$$

$$d3 = \begin{pmatrix} 0.0610 + 0.3795i \\ 0.7000 + 0.3729i \\ 0.0534 + 0.7439i \\ 0.1214 + 0.1324i \end{pmatrix}$$

Wyniki wywołania: `jacobi(diag(M3, -1), diag(M3), diag(M3, 1), d3, 10^-9)`:
`Function stopped - in this case Jacobi method is divergent.`

$$ans = 1.0e + 09 * \begin{pmatrix} 0.0128 - 0.0047i \\ 1.5608 + 0.7288i \\ 0.0019 + 0.0006i \\ 0.0000 - 0.0000i \end{pmatrix}.$$

Wyniki wywołania: `linsolve(M3, d3)`:

$$ans = \begin{pmatrix} 1.9523 - 0.2883i \\ -1.1118 + 1.5070i \\ 0.5986 + 0.5751i \\ 0.0166 - 0.1261i \end{pmatrix}.$$

# 5 Analiza wyników

Wyniki potwierdzają, że metoda Jacobiego jest zbieżna wtedy i tylko wtedy, gdy macierz iteracji ma promień spektralny mniejszy od 1. Jeżeli mamy zatem pewność, że tak jest, dobrym rozwiązaniem jest użycie tej metody. Zastosowanie funkcji generującej losowe macierze trójdiagonalne o elementach zespolonych i sprawdzającej, czy dla danej macierzy jej macierz iteracji $B$ ma $\rho(B) < 1$ pokazuje jednak, że częściej $\rho(B) > 1$. Funkcja `get_matrix` działa o wiele dłużej, jeżeli wywołamy ją z drugim argumentem równym 0, w przeciwnym przypadku działa szybko nawet dla dużych N. Dla porównania: wywołanie `[M, d] = get_matrix(400, 1)` zajmuje kilka sekund, a wywołania `[M, d] = get_matrix(10, 0)` nie udało się zakończyć przez dwa dni. Zdecydowanie częściej (biorąc pod uwagę wyłącznie pełną losowość) mamy zatem do czynienia z układami, dla których metoda Jacobiego jest rozbieżna.

Oprócz zbieżności metody, istotnym problemem zadania jest fakt, że macierz układu jest trójdiagonalna. Sprawia to, że w rozwiązywaniu układu równań nie musimy zajmować się całą macierzą - dla dużych N większość jej elementów to zera, a jedynymi elementami, które mają znaczenie są elementy z trzech diagonali. Stąd pomysł przekształcenia "tradycyjnej" macierzy iteracji (1) do postaci macierzy $G$. Zauważmy, że macierz (1) jest rozmiaru $2N \times 2N$. Macierz $G$ natomiast ma rozmiar $N \times 5$. Dla małych N takie podejście powoduje wydłużenie działania algorytmu, choć jest ono niezauważalne (bo N jest małe). Dla dużych N zastosowanie macierzy $G$ pozwala na zmniejszenie pamięci zajmowanej przez macierz iteracji (a nawet na obliczenia na bardzo dużych układach - takich, że stworzenie ich macierzy iteracji byłoby niemożliwe) oraz przyspiesza algorytm (unikamy niepotrzebnych ilorazów, których jednym z czynników jest 0). Oczywiste jest zatem, że o wiele bardziej korzystne ze względu oszczędności pamięci oraz wydajności algorytmu jest utworzenie macierzy G i używanie jej w metodzie Jacobiego zamiast macierzy (1).