Michał Rdzany gr.E3 20.10.2017r.

# Sprawozdanie
# Metody numeryczne 2

# 1. Układy równań liniowych i nieliniowych, zera wielomianów.

Temat 8:

Rozwiązywanie układu równań z macierzą trójdiagonalną w dziedzinie zespolonej. Wyjściowy układ sprowadzamy do układu o elementach rzeczywistych i rozwiązujemy go iteracyjną metodą Gaussa-Seidla (macierz przechowujemy tylko w trzech wektorach).

**Opis problemu:**

Szukamy rozwiązania układu równań $C \cdot cx = cy$, gdzie $C \in M_{n\times n}(\mathbb{C})$ , $cx, cy \in M_{n\times 1}(\mathbb{C})$ i $C$ jest macierzą trójdiagonalną.

Operując na wektorach reprezentujących przekątne zamiast bezpośrednio na macierzy $C$, złożoność naszego algorytmu (czasowa i pamięciowa) jest $O(n)$ zamiast $O(n^2)$. Ma to znaczenie przy bardzo dużych macierzach.

## Opis metody:

Jeśli $A$ będzie miała 0 na głównej przekątnej, wypisujemy komunikat o niezbieżności metody i kończymy funkcję. Jeśli wymiary podanych wektorów się nie zgadzają, wypisujemy odpowiedni komunikat i kończymy funkcję. Jeśli $n = 1$ zwracamy liczbę iteracji 0 i $cx = cy/C$.

W przeciwnym przypadku wyjściowy układ sprowadzamy do układu o elementach rzeczywistych $A \cdot x = v$, gdzie $A \in M_{2n\times 2n}(\mathbb{R})$ , $x, v \in M_{2n\times 1}(\mathbb{R})$.

$$C \cdot cx = cy$$

$$(real(C) + imag(C)i) \cdot (real(cx) + imag(cx)i) = real(cy) + imag(cy)i$$

$$real(C) \cdot real(cx) - imag(C) \cdot imag(cx) + (imag(C) \cdot real(cx) + real(C) \cdot imag(cx))i = real(cy) + imag(cy)i$$

$$\begin{cases} real(C) \cdot real(cx) - imag(C) \cdot imag(cx) = real(cy) \\ imag(C) \cdot real(cx) + real(C) \cdot imag(cx) = imag(cy) \end{cases}$$

$$\begin{bmatrix} real(C) & -imag(C) \\ imag(C) & real(C) \end{bmatrix} \cdot \begin{bmatrix} real(cx) \\ imag(cx) \end{bmatrix} = \begin{bmatrix} real(cy) \\ imag(cy) \end{bmatrix}$$

$$A \cdot x = v$$

$$A = \begin{bmatrix} \begin{vmatrix} b_1 & a_1 & 0 & & & \\ c_1 & b_2 & \ddots & & \mathbf{0} & \\ 0 & \ddots & \ddots & & & \\ & & & \ddots & \ddots & 0 \\ & \mathbf{0} & & \ddots & \ddots & a_{n-1} \\ & & & 0 & c_{n-1} & b_n \end{vmatrix} & \begin{vmatrix} -e_1 & -d_1 & 0 & & & \\ -f_1 & -e_2 & \ddots & & \mathbf{0} & \\ 0 & \ddots & \ddots & & & \\ & & & \ddots & \ddots & 0 \\ & \mathbf{0} & & \ddots & \ddots & -d_{n-1} \\ & & & 0 & -f_{n-1} & -e_n \end{vmatrix} \\ \begin{vmatrix} e_1 & d_1 & 0 & & & \\ f_1 & e_2 & \ddots & & \mathbf{0} & \\ 0 & \ddots & \ddots & & & \\ & & & \ddots & \ddots & 0 \\ & \mathbf{0} & & \ddots & \ddots & d_{n-1} \\ & & & 0 & f_{n-1} & e_n \end{vmatrix} & \begin{vmatrix} b_1 & a_1 & 0 & & & \\ c_1 & b_2 & \ddots & & \mathbf{0} & \\ 0 & \ddots & \ddots & & & \\ & & & \ddots & \ddots & 0 \\ & \mathbf{0} & & \ddots & \ddots & a_{n-1} \\ & & & 0 & c_{n-1} & b_n \end{vmatrix} \end{bmatrix}$$

Wektory $a, b, c$ przedstawiają części rzeczywiste przekątnych macierzy $C$, a wektory $d, e, f$ części urojone.

Otrzymany układ równań rozwiązujemy iteracyjną metodą Gaussa-Seidla, w której kolejne przybliżenia rozwiązania otrzymujemy z równania:

$$x_i^{(k+1)} = \frac{1}{A(i,i)}\left(v_i - \sum_{j=1}^{i-1}\left(A(i,j) \cdot x_j^{(k+1)}\right) - \sum_{j=i+1}^{2\cdot n}\left(A(i,j) \cdot x_j^{(k)}\right)\right)$$

Aby nie musieć iterować po całej macierzy $A$ pomijamy elementy, o których wiemy, że są równe 0. Wyznaczanie wektora $x^{(k+1)}$ wygląda wtedy następująco:

$$x_1^{(k+1)} = \frac{1}{b_1}\left(v_1 - a_1 \cdot x_2^{(k)} + e_1 \cdot x_{n+1}^{(k)} + d_1 \cdot x_{n+2}^{(k)}\right)$$

$$x_2^{(k+1)} = \frac{1}{b_2}\left(v_2 - c_1 \cdot x_1^{(k+1)} - a_2 \cdot x_3^{(k)} + f_1 \cdot x_{n+1}^{(k)} + e_2 \cdot x_{n+2}^{(k)} + d_2 \cdot x_{n+3}^{(k)}\right)$$

.....

.....

$$x_{n-1}^{(k+1)} = \frac{1}{b_{n-1}}\left(v_{n-1} - c_{n-2} \cdot x_{n-2}^{(k+1)} - a_{n-1} \cdot x_n^{(k)} + f_{n-2} \cdot x_{2n-2}^{(k)} + e_{n-1} \cdot x_{2n-1}^{(k)} + d_{n-1} \cdot x_{2n}^{(k)}\right)$$

$$x_n^{(k+1)} = \frac{1}{b_n}\left(v_n - c_{n-1}\cdot x_{n-1}^{(k+1)} + f_{n-1}\cdot x_{2n-1}^{(k)} + e_n \cdot x_{2n}^{(k)}\right)$$

$$x_{n+1}^{(k+1)} = \frac{1}{b_1}\left(v_{n+1} - e_1 \cdot x_1^{(k+1)} - d_1 \cdot x_2^{(k+1)} - a_1 \cdot x_{n+2}^{(k)}\right)$$

$$x_{n+2}^{(k+1)} = \frac{1}{b_2}\left(v_{n+2} - f_1 \cdot x_1^{(k+1)} - e_2 \cdot x_2^{(k+1)} - d_2 \cdot x_3^{(k+1)} - c_1 \cdot x_{n+1}^{(k+1)} - a_2 \cdot x_{n+3}^{(k)}\right)$$

.....

.....

$$x_{2n-1}^{(k+1)} = \frac{1}{b_{n-1}}\left(v_{2n-1} - f_{n-2}\cdot x_{n-2}^{(k+1)} - e_{n-1}\cdot x_{n-1}^{(k+1)} - d_{n-1}\cdot x_n^{(k+1)} - c_{n-2}\cdot x_{2n-2}^{(k+1)} - a_{n-1}\cdot x_{2n}^{(k)}\right)$$

$$x_{2n}^{(k+1)} = \frac{1}{b_n}\left(v_{2n} - f_{n-1}\cdot x_{n-1}^{(k+1)} - e_n \cdot x_n^{(k+1)} - c_{n-1}\cdot x_{2n-1}^{(k+1)}\right)$$

Aż do napotkania warunku stopu powtarzamy wyznaczanie kolejnych wektorów $x^{(k+1)}$, zapamiętując wektory $x^{(k)}$.

Warunki stopu:

1) Nie wszystkie elementy wektora $x^{(k)}$ są skończone.
2) $\left\|x^{(k+1)} - x^{(k)}\right\| < 2 \cdot eps(\left\|x^{(k+1)}\right\|)$
3) Osiągnięto $itMax$(podane przez użytkownika) iteracji (domyślnie$10^4$).

Jeśli nie wszystkie elementy otrzymanego przybliżenia są skończone lub jesteśmy daleko od rozwiązania przy domyślnym $itMax$: wypisujemy komunikat o niezbieżności metody dla podanego przybliżenia początkowego i liczbę wykonanych iteracji, a następnie kończymy funkcję.

Jeśli $n < 20$ wypisujemy wektoy $cx$ i $C \cdot cx - cy$. Wypisujemy obliczony błąd $\|C \cdot cx - cy\|$ i liczbę wykonanych iteracji, zwracamy wektor $cx$ otrzymany z ostatniego przybliżenia.

## Warunki wystarczające zbieżności:

1. Macierz $A$ jest silnie diagonalnie dominująca.
2. Macierz $A$ jest dodatnio określona.

## Przykłady i wnioski:

Skrypty przykładów ustawiają odpowiednie wartości zmiennych w taki sposób, że wystarczy wywołać funkcję $Diag3Cmplx(ca, cb, cc, cy, x0)$; lub skrypt $czas$, który dodatkowo podaje czas wykonania funkcji i czas wykonania funkcji $C\backslash cy$ z użyciem macierzy sparse.
Przykłady 3, 4, 5 używają zmiennej „$n$" jako rozmiaru macierzy (domyślnie $10^4$).
Dla $n < 20$ przykłady dodatkowo wypisują macierz $C$ i wektory $cy, x0$.

Przykład 1

$C = [5 + 2i]$, czyli: $ca = [\,], cb = [5 + 2i], cc = [\,]$

$cy = [2 + 3i], \qquad x0 = [0 + 0i]$

Program podał liczbę iteracji $0$ i $cx = [0.5517 \; + \; 0.3793i]$

$$C \cdot cx - cy = 0$$

Przykład 2

$$C = \begin{bmatrix} 7+i & 3+i & 0 & 0 \\ i & 5+i & 1+i & 0 \\ 0 & 4 & 8 & i \\ 0 & 0 & 2 & 6+i \end{bmatrix}, \text{czyli } ca = \begin{bmatrix} 3+i \\ 1+i \\ i \end{bmatrix}, cb = \begin{bmatrix} 7+i \\ 5+i \\ 8 \\ 6+i \end{bmatrix}, cc = \begin{bmatrix} i \\ 4 \\ 2 \end{bmatrix}$$

$$cy = \begin{bmatrix} 2+i \\ 3+i \\ 4+i \\ 5+i \end{bmatrix}, \qquad x0 = \begin{bmatrix} 1+i \\ 1+i \\ 1+i \\ 1+i \end{bmatrix}$$

Program podał liczbę iteracji $29$ i $cx = \begin{bmatrix} 0.0500 + 0.0414i \\ 0.5735 + 0.0289i \\ 0.2175 + 0.0148i \\ 0.7665 + 0.0340i \end{bmatrix}$

$$C \cdot cx - cy = \vec{0}$$

Przykład 3

$$C \in M_{n\times n}(\mathbb{C}), C = \begin{bmatrix} 5+2i & 1+i & 0 & & & \\ 2+i & \ddots & \ddots & & \mathbf{0} & \\ 0 & \ddots & \ddots & & & \\ & & & \ddots & \ddots & 0 \\ & \mathbf{0} & & \ddots & \ddots & 1+i \\ & & & 0 & 2+i & 5+2i \end{bmatrix}, cy = \begin{bmatrix} 1+2i \\ \vdots \\ 1+2i \end{bmatrix} = x0$$

Dla $n = 10$ program podaje liczbę iteracji $38$ i $\|C \cdot cx - cy\| = 8.382 \cdot 10^{-16}$ w $\sim 0.001s$.
Dla $n = 10^2$ program podaje liczbę iteracji $71$ i $\|C \cdot cx - cy\| = 3.1343 \cdot 10^{-15}$ w $\sim 0.001s$.
Dla $n = 10^3$ program podaje liczbę iteracji $71$ i $\|C \cdot cx - cy\| = 3.2596 \cdot 10^{-15}$ w $\sim 0.005s$.
Dla $n = 10^4$ program podaje liczbę iteracji $71$ i $\|C \cdot cx - cy\| = 3.2596 \cdot 10^{-15}$ w $\sim 0.05s$.
Dla $n = 10^5$ program podaje liczbę iteracji $70$ i $\|C \cdot cx - cy\| = 7.0266 \cdot 10^{-14}$ w $\sim 0.45s$.
Dla $n = 10^6$ program podaje liczbę iteracji $71$ i $\|C \cdot cx - cy\| = 3.2596 \cdot 10^{-15}$ w $\sim 4.5s$.
Dla $n = 10^7$ program podaje liczbę iteracji $71$ i $\|C \cdot cx - cy\| = 3.2596 \cdot 10^{-15}$ w $\sim 40s$.

Przykład 4

$C \in M_{n\times n}(\mathbb{C})$, $C, cy, x0$ są wyznaczone losowo.
(elementy wektorów: $a, c, d, e, f, x, v$ w przedziale $[-10,10]$;
elementy wektora $b$ w przedziale $[50,100]$, co gwarantuje zbieżność.)

Dla $n = 10$ program podaje liczbę iteracji $\sim 18$ i $\|C \cdot cx - cy\| \cong 3.5 \cdot 10^{-15}$ w $\sim 0.001s$.
Dla $n = 10^2$ program podaje liczbę iteracji $\sim 20$ i $\|C \cdot cx - cy\| \cong 1.2 \cdot 10^{-14}$ w $\sim 0.001s$.
Dla $n = 10^3$ program podaje liczbę iteracji $\sim 23$ i $\|C \cdot cx - cy\| \cong 3.9 \cdot 10^{-14}$ w $\sim 0.003s$.
Dla $n = 10^4$ program podaje liczbę iteracji $\sim 25$ i $\|C \cdot cx - cy\| \cong 1.25 \cdot 10^{-13}$ w $\sim 0.02s$.
Dla $n = 10^5$ program podaje liczbę iteracji $\sim 27$ i $\|C \cdot cx - cy\| \cong 4 \cdot 10^{-13}$ w $\sim 0.15s$.
Dla $n = 10^6$ program podaje liczbę iteracji $\sim 30$ i $\|C \cdot cx - cy\| \cong 1.25 \cdot 10^{-12}$ w $\sim 1.7s$.
Dla $n = 10^7$ program podaje liczbę iteracji $\sim 30$ i $\|C \cdot cx - cy\| \cong 4 \cdot 10^{-12}$ w $\sim 17s$.

Na przykładach 3 i 4 widać liniową złożoność czasową użytego algorytmu.
Błąd jest większy dla dłuższych wektorów ponieważ w warunku stopu i przy liczeniu błędu używamy normy-2, a nie $\infty$. Wykonanie operacji $C\backslash cy$ dla $n = [\,10,\ 10^2,\ 10^3,\ 10^4\,]$ zajmuje około $[\,0.01,\ 0.01,\ 0.03,\ 1.6\,]s$. Dla $n = 10^5$ stworzenie macierzy $C$ wymagałoby 149GB pamięci. Wartość ta jest proporcjonalna do $n^2$. Wykonanie operacji $C\backslash cy$, gdy $C$ jest macierzą sparse dla $n = [10, 10^2, 10^3, 10^4, 10^5, 10^6, 10^7]$ zajmuje około $[0.00005, 0.0001, 0.0005, 0.002, 0.02, 0.2, 2]s$ (10 razy szybciej niż nasza funkcja, ale jeśli uwzględnimy czas tworzenia macierzy sparse, który jest znacznie dłuższy niż czas tworzenia wektorów, to już tylko około 2.5 razy szybciej).

Przykład 5

Analogiczny do przykładu 4, z tym że elementy wektora $b$ też są w przedziale $[-10,10]$.

W tym przykładzie nasza metoda rzadko jest zbieżna (częściej dla mniejszych n).

Podsumowując, używanie macierzy sparse wydaje się być lepszym rozwiązaniem, poza wyjątkowymi sytuacjami np. jeśli wiemy, że nasza metoda będzie zbieżna i wykonujemy mało iteracji (mamy dobre przybliżenie początkowe lub nie potrzebujemy zbyt dużej dokładności).