Poszukiwanie wartości własnych macierzy

Kamil Korolkiewicz

3 grudnia 2009

Temat nr 2:

Metoda potęgowa z normowaniem (przypadek rzeczywisty). Poszukiwanie λ_1 i λ_2 ($\mid \lambda_1 \mid > \mid \lambda_2 \mid > \mid \lambda_j \mid, j \geq 3$). Deflacja przekształceniem $L^{(1)}$.

Opis metody:

Będziemy poszukiwać dwóch największych (co do modułu) wartości własnych λ_1 i λ_2 macierzy diagonalizowalnej A. Za $x^{(0)}$ przyjmijmy wektor - przybliżenie początkowe. Zakładamy, że $\mid \lambda_1 \mid > \mid \lambda_2 \mid > \mid \lambda_j \mid, j \geq 3$, gdzie λ_j to wartości własne macierzy A. Zapis ten oznacza, że co najmniej dwie wartości własne A są dominujące. λ_1 wyznaczymy wykonując iteracyjnie następujący algorytm 1:

$$\begin{cases} y^{(k+1)} = Ax^{(k)} \\ \lambda^{(k)} = \left(y^{(k+1)}, x^{(k)}\right) & k = 0, 1, \dots \\ x^{(k+1)} = \frac{y^{(k+1)}}{\|y^{(k+1)}\|_2} & (*) \end{cases}$$

Operacja (*) to normowanie wektora y, $\| y \|_2 = \sqrt{\sum y_i^2}$

Iloczyn skalarny $\left(y^{(k+1)}, x^{(k)}\right)$ liczymy tradycyjnie jako suma iloczynów współrzędnych wektorów.

Algorytm ma zbieżność liniową: $\mid \lambda^{(k+1)} - \lambda^{(k)} \mid \leq \mid \frac{\lambda_2}{\lambda_1} \mid\mid \lambda^{(k)} - \lambda_1 \mid$

Za kryterium stopu przyjmujemy $\mid \lambda^{(k+1)} - \lambda^{(k)} \mid \leq \varepsilon$

Kolejnym krokiem jest deflacja macierzy A. Algorytm wyznaczył nam największą co do modułu wartość własną. Chcemy się jej pozbyć z macierzy A i w nowej macierzy $\widetilde{A}$ takiej, że $S_p(A) = S_p(\widetilde{A}) \cup \{\lambda_1\}$ odnaleść λ_2 powtarzając algorytm 1 dla macierzy $\widetilde{A}$. W tym celu będziemy szukać takiej macierzy P, że $S_p(PAP^{-1}) = S_p(A)$ oraz PAP^{-1} jest macierzą taką, że

$$PAP^{-1} = \begin{bmatrix} \lambda_1 & a^T \\ 0 & \widetilde{A} \end{bmatrix}$$

Niech

$$L^{(1)} = \begin{bmatrix} 1 & 0 & \dots & 0 \\ -l_{21} & 1 & \dots & 0 \\ \vdots & \vdots & \ddots & 0 \\ -l_{n1} & \cdots & 0 & 1 \end{bmatrix}$$

a zatem

$$(L^{(1)})^{-1} = \begin{bmatrix} 1 & 0 & \dots & 0 \\ l_{21} & 1 & \dots & 0 \\ \vdots & \vdots & \ddots & 0 \\ l_{n1} & \cdots & 0 & 1 \end{bmatrix}$$

gdzie

$$l_{j1} = \frac{x_{1j}}{x_{11}},\ j = 2, \dots, n\,, x_{1k} \neq 0$$

$$k :\mid x_{1k} \mid = \max_{1 \leq j \leq n} \mid x_{1j} \mid$$

$$x_1 = \begin{bmatrix} x_{1k} \\ x_{12} \\ \vdots \\ x_{1k-1} \\ x_{11} \\ x_{1k+1} \\ \vdots \\ x_{1n} \end{bmatrix}$$

x_1 to wektor $x^{(k+1)}$ otrzymany z algorytmu 1 z elementem głównym zamienionym z pierwszym elementem wektora.

Nasze $P = L^{(1)} P_{1,k}$, $PAP^{-1} = L^{(1)} P_{1,k} A P_{1,k} (L^{(1)})^{-1}$, gdzie $P_{1,k}$ to macierz permutacji (w programie nie jest generowana taka macierz, ale zamieniane są tylko kolumny 1 i k macierzy $L^{(1)}$ oraz w między czasie obliczanej macierzy $L^{(1)} P_{1,k} A$).

Po takim przekształceniu otrzymujemy macierz $\widetilde{A}$ i ponownie wykonujemy nasz algorytm 1, z macierzą $\widetilde{A}$ i wyjściowym przybliżeniem $x^{(0)}$ bez ostatniego elementu, znajdując tym samym λ_2.

Przykłady:

Ustalone $\varepsilon = 1e - 6$

1.

$$A = \begin{bmatrix} 3 & 2 & 2 & 1 & 1 \\ 0 & 7 & 3 & 5 & 1 \\ 0 & 0 & -8 & 2 & -1 \\ 0 & 0 & 0 & 9 & 6 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\lambda_1 = 8,99999956815299$$

$$\lambda_2 = -8,00000001349127$$

2.

$$A = \begin{bmatrix} 4 & 6 & 3 \\ 1 & 4 & 1 \\ 3 & 4 & 4 \end{bmatrix}$$

$$\lambda_1 = 9,00000022794242$$

$$\lambda_2 = 2,00000064373035$$