

Algoritmy. 4. JACOBI. Hledání charakteristických kořenů a vektoru čtvercové symetrické matice Jacobiho metodou

Aplikace matematiky, Vol. 12 (1967), No. 2, 149–151

Persistent URL: <http://dml.cz/dmlcz/103082>

Terms of use:

© Institute of Mathematics AS CR, 1967

Institute of Mathematics of the Czech Academy of Sciences provides access to digitized documents strictly for personal use. Each copy of any part of this document must contain these *Terms of use*.



This document has been digitized, optimized for electronic delivery and stamped with digital signature within the project *DML-CZ: The Czech Digital Mathematics Library* <http://dml.cz>

ALGORITHMY

4. JACOBI

HLEDÁNÍ CHARAKTERISTICKÝCH KOŘENŮ A VEKTORŮ
ČTVERCOVÉ SYMETRICKÉ MATICE JACOBIHO METODOU

Tento algoritmus byl převzat z časopisu CACM [3]. Komentářem a kontrolním příkladem jej doplnil Jiří RAICHL, Centrum numerické matematiky, Matematicko-fyzikální fakulta Karlovy university, Praha 1, Malostranské nám. 25.

Tato procedura hledá charakteristické kořeny a vektory čtvercové symetrické matice Jacobiho metodou [1], [2]. Tato metoda spočívá v tom, že vyjdeme od dané matice \mathbf{A}_1 a postupně vytváříme matice

$$(1) \quad \mathbf{T}_k \mathbf{A}_k \mathbf{T}_k = \mathbf{A}_{k+1},$$

kde matice \mathbf{T}_k jsou orthogonální a vybrány tak, aby po tomto násobení byl vždy určitý prvek a_{pq} ($p \neq q$) matice \mathbf{A}_{k+1} roven nule. Posloupnost matic \mathbf{A}_k při tom konverguje k diagonální matici, jejíž i -tý diagonální prvek je i -tým charakteristickým kořenem dané matice. Posloupnost součinů $\mathbf{T}_1 \mathbf{T}_2 \dots \mathbf{T}_k$ konverguje k matici, jejíž i -tý sloupec je i -tým charakteristickým vektorem dané matice.

Nejprve vypočteme $\sqrt{\sum_{i \neq j} a_{ij}^2} \rightarrow thr$ a pak hledáme prvek $|a_{pq}| \geq thr$. Najdeme-li jej, sestavíme matici \mathbf{T}_k a provedeme (1) a hledáme další takový prvek. Když již žádný takový prvek nenajdeme, zjistíme, zda je $thr \leq rho \sqrt{\sum_{i \neq j} a_{ij}^2} / n$ (rho je předem dané číslo charakterisující přesnost, s níž chceme obdržet výsledky). Je-li tomu tak, jsou již všechny nedиагональні prvky matice \mathbf{A}_k dostatečně malé a výpočet skončíme. Ne-li tomu tak, zmenšíme thr n -krát a postup uvedený výše opakujeme.

procedure JACOBI (a, s, n, rho);

comment a na počátku daná matice, po provedení procedury jsou v diagonále aproximace charakteristických kořenů, s matice na počátku libovolná, po provedení procedury jsou jejími sloupci aproximace charakteristických vektorů, n stupeň matice, rho charakterisuje přesnost výsledků.;

value n, rho ;

integer n ; **real** rho ; **real array** a, s ;

begin real $norm1, norm2, thr, mu, omega, sint, cost, int1, v1, v2, v3$;

```

integer i, j, p, q, ind;
comment Dosazení jednotkové matice do pole s.;
for i := 1 step 1 until n do for j := 1 step 1 until i do
  if i = j then s[i, j] := 1 else s[i, j] := s[j, i] := 0;
comment Výpočet součtu čtverců nediagonálních prvků.;
int1 := 0;
for i := 2 step 1 until n do for j := 1 step 1 until i - 1 do
  int1 := int1 + 2 × a[i, j]↑2;
norm1 := sqrt(int1); norm2 := (rho/n) × norm1;
thr := norm1; ind := 0;
main : thr := thr/n;
main1 : for q := 2 step 1 until n do for p := 1 step 1 until q - 1 do
  if abs(a[p, q]) ≥ thr then
    begin ind := 1; v1 := a[p, p]; v2 := a[p, q]; v3 := a[q, q]; mu := 0.5 × (v1 - v3);
      omega := - (if mu = 0 then 1 else sign(mu)) × v2/sqrt(v2↑2 + mu↑2);
      sint := omega/sqrt(2 × (1 + sqrt(1 - omega↑2)));
      cost := sqrt(1 - sint↑2);
      for i := 1 step 1 until n do
        begin if i ≠ p ∧ i ≠ q then
          begin int1 := a[i, p]; mu := a[i, q];
              a[q, i] := a[i, q] := int1 × sint + mu × cost;
              a[p, i] := a[i, p] := int1 × cost - mu × sint
            end;
          int1 := s[i, p]; mu := s[i, q];
          s[i, q] := int1 × sint + mu × cost;
          s[i, p] := int1 × cost - mu × sint
        end Konec výpočtu nediagonálních prvků.;
        a[p, p] := v1 × cost↑2 + v3 × sint↑2 - 2 × v2 × sint × cost;
        a[q, q] := v1 × sint↑2 + v3 × cost↑2 + 2 × v2 × sint × cost;
        a[p, q] := a[q, p] := (v1 - v3) × sint × cost + v2 × (cost↑2 - sint↑2)
      end Nyní testujeme, zda existovalo alespoň jedno  $a_{pq}$  v absolutní hodnotě větší
      nebo rovno thr a pak zda  $thr > rho \times \sqrt{\sum_{i \neq j} a_{ij}^2/n}$ ;
    if ind = 1 then begin ind := 0; goto main1 end
    else if thr > norm2 then goto main
  end
end

```

Kontrolní příklad:

Hilbertova matice 4 stupně (o prvcích $a_{ij} = 1/(i + j - 1)$) má charakteristické kořeny (při $rho = 10^{-5}$):

$$\lambda_1 = 1,50021 \quad \lambda_2 = 0,169141 \quad \lambda_3 = 0,000097 \quad \lambda_4 = 0,006738$$

Charakteristické vektory (sloupce):

0,792608	-0,582075	0,029193	0,179186
0,451923	0,370502	-0,328713	-0,741917
0,322416	0,509579	0,791411	0,100226
0,252161	0,514048	-0,514551	0,638283

Tento algoritmus byl užíván v systému ALGOL-GENIUS na počítači SAAB.

Literatura

- [1] *O. Slaviček a kol.*: Základní numerické metody. SNTL, Praha 1964.
- [2] *A. Ralston, H. S. Wilf*: Mathematical methods for digital computers. New York, Willey.
- [3] *Thomas G. Evans, Bolt, Beranek, Newman*: Algorithm 85. CACM 1962, vol. 5, str. 208.