

Aplikace matematiky

Jaromír Janko

Algoritmy. 16. NEWTON. Řešení systémů nelineárních rovnic

Aplikace matematiky, Vol. 13 (1968), No. 5, 435--439

Persistent URL: <http://dml.cz/dmlcz/103191>

Terms of use:

© Institute of Mathematics AS CR, 1968

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



This paper 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>

16. NEWTON

ŘEŠENÍ SYSTÉMŮ NELINEÁRNÍCH ROVNIC

JAROMÍR JANKO, Výzkumný ústav zvukové a reprodukční techniky, Plzeňská 66, Praha 5

Tato procedura slouží k řešení systémů nelineárních rovnic $f_i(X_1, X_2, \dots, X_n) = 0$ pro $i = 1, 2, \dots, n$, Newtonovou metodou [2], [3]. Je určena pro zpracování na počítači ELLIOT, jehož překladač nepřipouští specifikovat formální parametry jako procedury. Proto byly specifikovány **real** a skutečné parametry jim odpovídající jsou zápis funkcí.

procedure *newton* (*n, a, x, limit, gmax, procrs, procdcr, proclin*);
value *n, limit*; **integer** *n, gmax*; **array** *a, x*;
real *limit, procrs, procdcr, proclin*;

comment parametr *n* označuje počet rovnic, *a* odpovídá identifikátoru dvojrozměrného pole, v němž je umístěna matice typu $n, n + 1$. Prvky jejich *n* prvních sloupců počítá procedura vyvolávaná zápisem funkce odpovídajícím *procdcr*, prvky posledního sloupce počítá procedura vyvolávaná zápisem funkce odpovídajícím *procrs*. *x* odpovídá zprvu *n*-složkovému vektoru počáteční aproximace, po provedení procedury vektoru konečné aproximace řešené soustavy. Jsou-li v některém iteračním kroku všechna residua menší než *limit*, je iterace skončena. Rovněž iterační proces končí, jakmile převyší počet iteračních kroků *gmax*. V tomto případě procedura obsazuje proměnnou odpovídající *gmax* záporným číslem.

Obě funkční procedury vyvolávané zápisem funkce odpovídajícím *procdcr* a *procrs* se programují pro každou úlohu znovu podle zadání funkcí f_i a to tak, aby prvá procedura do *i*-tého řádku matice odpovídající formálnímu parametru *a* počítala hodnoty $\partial f_i / \partial x_i$, druhá pak hodnoty funkcí f_i pro aproximaci kořene danou složkami odpovídajícími *x*. Obě tyto procedury mají formální parametry *A* a *X*, jimž při vyvolání procedury NEWTON musí odpovídat tytéž parametry jako formálními parametry *a* a *x*. Formálnímu parametru *proclin* odpovídá zápis funkce vyvolávající proceduru řešící soustavu lineárních rovnic (např. viz [5]) upravenou uměle na proceduru funkční obdobně jako u procedur vyhodnocujících parciální derivace a residua.

begin integer *i, g*; **real** *p, r*; **boolean** *q*;
g := 0;

```

L1: r := procras;
    comment v tomto místo se doporučuje tisk pořadového čísla aproximace, okamžitých hodnot kořenů a residuí;
    q := true;
    for i := 1 step 1 until n do
        begin p := a[i, n + 1];
              q := if abs(p) < limit then q else false;
              a[i, n + 1] := -p
        end;
    if q then goto L3;
    r := procdcr;
    if g = 0 then begin comment doporučuje se tisk matice parciálních derivací v 1. kroku;
                  end;
L2: r := proclin;
    for i := 1 step 1 until n do
        x[i] := x[i] + a[i, n + 1];
        g := g + 1;
        if g < gmax then goto L1;
        gmax := -1;
L3: end of procedure newton;

```

Použití procedury NEWTON předvedeme kompletním řešením následující úlohy na počítači ELLIOT 4100. Je dán systém rovnic:

$$\begin{aligned}x + x^2 - 2yz &= 0,1 \\ y - y^2 + 3xz &= -0,2 \\ z + z^2 + 2xy &= 0,3\end{aligned}$$

Při volbě počáteční aproximace $(0, 0, 0)$ a $gmax = 10$ a $limit = 10^{-4}$ získáme řešením Newtonovou metodou ve čtyřech krocích řešení $0,01282; -0,17780; 0,24468$. V příkladu jsou příkazy procedur pro vstup a výstup použity z překladače ELLIOT-ALGOL.

```

begin
real procedure residues(a, X); array a, X;
begin
a[1, 4] := X[1] × (1 + X[1]) - 2 × X[2] × X[3] - 0.1;
a[2, 4] := X[2] × (1 - X[2]) + 3 × X[1] × X[3] + 0.2;
a[3, 4] := X[3] × (1 + X[3]) + 2 × X[1] × X[2] - 0.3;
residues := 1;
end;

```

real procedure *derivations* (*a*, *X*); **array** *a*, *X*;

begin

$a[1, 1] := 1 + 2 \times X[1];$

$a[1, 2] := -2 \times X[3];$

$a[1, 3] := -2 \times X[2];$

$a[2, 1] := 3 \times X[3];$

$a[2, 2] := 1 - 2 \times X[2];$

$a[2, 3] := 3 \times X[1];$

$a[3, 1] := 2 \times X[2];$

$a[3, 2] := 2 \times X[1];$

$a[3, 3] := 1 + 2 \times X[3];$

derivations := 1;

end;

real procedure *gauss*(*n*, *a*)

.....

viz [5]

gauss := 1;

end of gauss;

procedure *newton*

.....

L3: **end of procedure** *newton*;

integer *n*, *gmax*; **read** *n*;

begin real *limit*; **integer** *i*;

array *x*[1 : *n*], *a*[1 : *n*, 1 : *n* + 1];

read *limit*; **read** *gmax*;

for *i* := 1 **step** 1 **until** *n* **do** **read** *x*[*i*];

newton (*n*, *a*, *x*, *limit*, *gmax*, *residues* (*a*, *x*), *derivations* (*a*, *x*), *gauss* (*n*, *a*));

Print "L3 `Result:`", *x*[1], *x*[2], *x*[3];

go to *S*;

end;

signal 107: **Print** 'No solution of linear system';

S: **if** *gmax* < 0 **then** **print** 'No exact solution';

end;

[1] J. Janko: Algorithms 431 and 432, Comm. ACN, 1968.

[2] Б. Демидович, И. Марон: Основы вычислительной математики, ГФМЛ, Москва 1968.

[3] T. Saaty, J. Bram: Nonlinear Mathematics, Mc Graw Hill, New York, 1964.

[4] J. Janko: Solution of Nonlinear Equations Systems by Newton's Method and the Gradients Method, Aplikace Matematiky 10, 1965, č. 3, s. 230—234.

[5] Algorithm 126, Comm. ACM, 1962, p. 511.

17. GRADIENT

ŘEŠENÍ SYSTÉMŮ NELINEÁRNÍCH ROVNIC

JAROMÍR JANKO, Výzumný ústav zvukové a reprodukční techniky, Plzeňská 66, Praha 5

Tato procedura slouží k řešení systémů nelineárních rovnic $f_i(X_1, X_2, \dots, X_n) = 0$ pro $i = 1, 2, \dots, n$ metodou gradientů – (viz literatura [2], [3] algoritmu NEWTON), Je určena pro zpracování na počítači ELLIOT, jehož překladač nepřipouští specifikovat formální parametry jako procedury. Proto byly specifikovány **real** a skutečné parametry jim odpovídající jsou zápisy funkcí.

procedure *gradient* (*n, a, x, f, limit, gmax, proceres, procder*);

value *n, limit*; **integer** *n, gmax*; **array** *a, x, f*;

real *limit, procder, proceres*;

comment Parametry této procedury odpovídají parametrům v proceduře NEWTON s těmito změnami: Pole odpovídající formálnímu parametru *a*, jehož složky nabývají hodnot parciálních derivací funkcí f_i má rozměny *n, n*. Vektor residuí je zobrazen v jednorozměrném poli odpovídajícím formálnímu parametru *f*. Procedura pro řešení soustav lineárních rovnic se zde neuzívá.;

begin integer *i, j, k, g*; **real** *alfa, beta, lambda, p*;

array *v*[1 : *n*, 1 : *n*];

g := 0;

L1: *p* := *proceres*;

comment v tomto místě se doporučuje tisk pořadového čísla aproximace, aproximace hodnot kořenů a residuí;

for *i* := 1 **step** 1 **until** *n* **do**

if *abs*(*f*[*i*]) > *limit* **then goto** L2;

goto L3;

L2: *p* := *procder*;

if *g* = 0 **then begin comment** doporučuje se tisk matice parciálních derivací v 1. kroku

end;

for *i* := 1 **step** 1 **until** *n* **do**

for *j* := *i* **step** 1 **until** *n* **do**

begin *p* := 0;

for *k* := 1 **step** 1 **until** *n* **do**

p := *p* + *a*[*i, k*] × *a*[*j, k*];

v[*i, j*] := *v*[*j, i*] := *p*

end;

alfa := *beta* := 0;

for *i* := 1 **step** 1 **until** *n* **do**

```

begin  $p := 0$ ;
    for  $j := 1$  step 1 until  $n$  do
         $p := p + v[i, j] \times f[j]$ ;
         $alfa := alfa + p \times f[i]$ ;
         $beta := beta + p \times p$ 
    end;
     $lambda := alfa / beta$ ;
    for  $i := 1$  step 1 until  $n$  do
        begin  $p := 0$ ;
            for  $j := 1$  step 1 until  $n$  do
                 $p := p + a[j, i] \times f[j]$ ;
                 $x[i] := x[i] - lambda \times p$ ;
            end;
             $g := g + 1$ ;
            if  $g < gmax$  then goto L1;
             $gmax := -1$ ;
        end
L3: end of procedure gradient;

```

Příklad použití této procedury je analogický použití procedury NEWTON.