UPORABNI PROGRAMI

```
==========================================
=                                        =
=    Prilagodljivo numerično integriranje =
=                                        =
==========================================
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Informatica UP 18                      %
% Adaptive Numerical Integration         %
% oktober 1984                           %
% pripravil Vladimir Batagelj            %
% sistem DEC-10                          %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

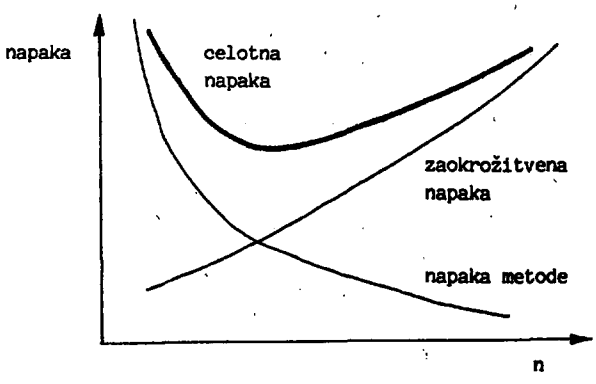
POSTOPKI ZA PRILAGODLJIVO NUMERIČNO INTEGRIRANJE

Vladimir Batagelj
VTOZD Matematika in mehanika
Univerza E. Kardelja v Ljubljani

Namen tega sestavka je opozoriti na postopke za prilagodljivo (adaptivno) numerično integriranje in na vire [2, 4, 5, 6], kjer so podrobneje opisani.

Postopki za prilagodljivo numerično integriranje so področje, na katerem se uspešno prepletata numerična analiza in računalništvo, natančneje, načrtovanje in analiza algoritmov. Tako je na primer v numerični knjižnici NAG podprogram D01AGA/F, ki temelji na Oliverjevi prilagodljivi metodi, priporočen kot najboljši splošni integracijski podprogram.

Če želimo pri običajnih postopkih za numerično integriranje [1] povečati natančnost izračuna, povečujemo število n točk ali vozlov, ki jih upoštevamo pri izračunu. S tem sicer zmanjšamo napako metode, žal pa se nam pri tem zaradi večanja števila operacij večata zaokrožitvena napaka in čas (cena) računanja. Zato se obnaša celotna napaka tako, kot prikazuje slika:



Pri prilagodljivih postopkih pa drobimo interval samo tam, kjer je treba. Zato običajno za dosego željene natačnosti potrebujejo manj korakov - pri (skoraj) isti napaki metode bo manjša zaokrožitvena napaka pa tudi postopek je hitrejši in s tem bolj ekonomičen.

Osnovna zamisel prilagodljivih postopkov numeričnega integriranja je znana v računalništvu kot pristop "deli in vladaj". Recimo, da želimo izračunati vrednost P(a,b,eps) integrala:

$$I(a,b) = \int_a^b f(x)\,dx$$

pri čemer dopuščamo napako eps . Naj bo še S(a,b) ena izmed kvadraturnih formul, ki jih poznamo iz numerične analize [1] , in dajejo oceno za vrednost I(a,b) .

Določimo oceno $S_1 = S(a,b)$ za neznani I(a,b) . Nato razpolovimo interval [a,b] s točko m = (a+b)/2 in izračunamo še oceno $S_2 = S(a,m) + S(m,b)$. Če se oceni razlikujeta za manj kot eps , vzamemo S_2 za P(a,b,eps) . Sicer isti postopek rekurzivno ponovimo na obeh podintervalih [a,m] in [m,b] , pri čemer pa na vsakem podintervalu dopuščamo le pol manjšo napako eps/2 . Ko enkrat poznamo P(a,m,eps/2) in P(m,b,eps/2) , lahko izračunamo tudi

P(a,b,eps) = P(a,m,eps/2) + P(m,b,eps/2)

V praksi se izkaže, da je zmanjšanje dovoljene napake na eps/2 pri rekurzivni ponovitvi postopka preveč pesimistično. Zato v postopek vpeljemo nov parameter q , $1 < q \leq 2$ in dovoljujemo napako eps/q . Izkaže se [3], da je čisto v redu že q = 1.4 .

V tem sestavku je opisani postopek razdelan za primer, ko oceno vrednosti integrala na intervalu [a,b] računamo po Simpsonovem obrazcu:

$$S(a,b) = \frac{b-a}{6}\left(f(a) + 4.f\left(\frac{a+b}{2}\right) + f(b) \right)$$

Treba pa je takoj pripomniti, da podprogrami iz numeričnih knjižnic uporabljajo veliko bolj "zvite" ocene in tudi sam postopek je še nadalje izpopolnjen.

Za primerjavo je dodan še navadni postopek za numerično integriranje po Simpsonu.

V obeh programih se nahajajo spremenljivke, za merjenje posameznih značilnih količin:

- i - zaporedna številka
- countf - število izračunanih funkcijskih vrednosti
- depth - globina rekurzije

Te spremenljivke in stavki, ki jih vsebujejo, ne vplivajo na samo računanje, zato jih lahko tudi izločimo.

Za to, da bodo prednosti prilagodljivega postopka očitne, je bila primerjava narejena na integralu (ploščina četrtine enotskega kroga):

$$\int_0^1 \sqrt{1 - x^2}\,dx$$

ki je za navadni postopek neugoden. Njegova štirikratna vrednost je število pi .

Priloženi tabeli govorita sami zase. Omenimo le pomen posameznih stolpcev:
NAVADNO: zaporedna številka, izračunana vrednost integrala, dejanska napaka, število izračunanih funkcijskih vrednosti;
PRILAGODLJIVO: zaporedna številka, izračunana vrednost integrala, dovoljena napaka, dejanska napaka, število

izračunanih funkcijskih vrednosti, največja globina rekurzije.

Izračuni so bili opravljeni na računalniku DEC-10 , Univerze v Ljubljani.

```
PROGRAM simpson(output);
CONST pi = 3.141592653589793 ;
      errmin = 0.00000005 ;
VAR error, int : real; i, n, countf : integer;
FUNCTION f( x: real ): real ;
BEGIN
   countf := countf + 1 ;
   f := sqrt(abs(1-sqr(x)))
END (* f *) ;
FUNCTION integral
      ( FUNCTION f:real; a,b:real; n:integer ): real;
VAR  h, p, x : real ; c, i : integer ;
BEGIN
   h := (b - a)/(2*n) ; p := f(a) + f(b);
   x := a ; c := 4 ;
   FOR  i := 1  TO  2*n-1  DO BEGIN
      x := x + h ;
      p := p + c*f(x) ;
      c := 6 - c
   END;
   integral := h*p/3
END (* integral *) ;
BEGIN
   writeln(' ':5,'NAVADNO INTEGRIRANJE PO SIMPSONU ');
   writeln;
   n := 1; i := 0 ;
   REPEAT
      countf := 0 ; i := i + 1 ;
      int := 4*integral(f,0,1,n) ;
      error := pi - int;
      writeln( i:7, int, error, countf:8 );
      n := 2*n
   UNTIL abs( error ) <= errmin
END.
```

```
   NAVADNO INTEGRIRANJE PO SIMPSONU

    1   2.976068E+00   1.655249E-01      3
    2   3.083595E+00   5.799752E-02      5
    3   3.121189E+00   2.040347E-02      9
    4   3.134398E+00   7.194966E-03     17
    5   3.139052E+00   2.540439E-03     33
    6   3.140695E+00   8.975565E-04     65
    7   3.141276E+00   3.171861E-04    129
    8   3.141481E+00   1.121163E-04    257
    9   3.141553E+00   3.978610E-05    513
   10   3.141579E+00   1.385808E-05   1025
   11   3.141588E+00   4.917383E-06   2049
   12   3.141591E+00   1.758337E-06   4097
   13   3.141592E+00   5.662441E-07   8193
   14   3.141592E+00   2.980232E-07  16385
   15   3.141592E+00   4.172325E-07  32769
   16   3.141592E+00   5.066395E-07  65537
   17   3.141593E+00  -8.940697E-08 131073
   18   3.141591E+00   1.817942E-06 262145
   19   3.141593E+00  -8.940697E-08 524289
   20   3.141601E+00  -8.434057E-06 1048577
```

```
PROGRAM simpson(output);
CONST pi = 3.141592653589793 ;
      errmin = 0.00000001 ;
VAR   error, int : real; i, countf, depth : integer;
FUNCTION f( x: real ): real ;
BEGIN
   countf := countf + 1 ;
   f := sqrt(abs(1-sqr(x)))
END (* f *) ;
FUNCTION integral
   ( FUNCTION f:real; a,b,error:real ): real;
VAR   m, c, fa, fm, fb : real ; d : integer ;
FUNCTION irec
   ( a, m, b, fa, fm, fb, estimate, error:real ): real;
CONST q = 1.5 ;
VAR   c, ma, mb, fma, fmb, esta, estb, newest : real ;
BEGIN
   d := d + 1 ; IF d > depth THEN depth := d ;
   ma := (a + m)/2 ; fma := f(ma) ;
   mb := (m + b)/2 ; fmb := f(mb) ;
   c  := (b - a)/12 ;
   esta := c * ( fa + 4*fma + fm ) ;
   estb := c * ( fm + 4*fmb + fb ) ;
   newest := esta + estb ;
   IF abs(estimate-newest) > error THEN
      irec := irec(a,ma,m,fa,fma,fm,esta,error/q) +
              irec(m,mb,b,fm,fmb,fb,estb,error/q)
   ELSE
      irec := newest ;
   d := d - 1
END (* irec *) ;
BEGIN
   d := 0 ; m := (a + b)/2 ; c := (b - a)/6 ;
   fa := f(a) ; fm := f(m) ; fb := f(b) ;
   integral := irec(a,m,b,fa,fm,fb,c*(fa+4*fm+fb),error)
END (* integral *) ;
BEGIN
   write(' ':15);
   writeln('PRILAGODLJIVO INTEGRIRANJE PO SIMPSONU');
   writeln ; error := 1 ; i := 0 ;
   WHILE error > errmin DO BEGIN
      countf := 0 ; depth := 0 ; i := i + 1 ;
      int := 4*integral(f,0,1,error) ;
      writeln(i,int,error,pi-int,countf:6,depth:6);
      error := error / 2
   END
END.
```

```
PRILAGODLJIVO INTEGRIRANJE PO SIMPSONU

 1  3.083595E+00  1.000000E+00  5.799752E-02    5   1
 2  3.083595E+00  5.000000E-01  5.799752E-02    5   1
 3  3.083595E+00  2.500000E-01  5.799752E-02    5   1
 4  3.083595E+00  1.250000E-01  5.799752E-02    5   1
 5  3.083595E+00  6.250000E-02  5.799752E-02    5   1
 6  3.083595E+00  3.125000E-02  5.799752E-02    5   1
 7  3.121189E+00  1.562500E-02  2.040350E-02    9   2
 8  3.134383E+00  7.812500E-03  7.209718E-03   13   3
 9  3.139032E+00  3.906250E-03  2.560467E-03   17   4
10  3.141253E+00  1.953125E-03  3.397763E-04   25   6
11  3.141458E+00  9.765625E-04  1.348853E-04   29   7
12  3.141530E+00  4.882813E-04  6.246567E-05   33   8
13  3.141556E+00  2.441406E-04  3.686547E-05   37   9
14  3.141565E+00  1.220703E-04  2.783537E-05   41  10
15  3.141583E+00  6.103516E-05  9.894371E-06   49  11
16  3.141588E+00  3.051758E-05  4.410744E-06   57  12
17  3.141590E+00  1.525879E-05  2.533197E-06   65  13
18  3.141591E+00  7.629395E-06  1.847744E-06   73  14
19  3.141592E+00  3.814697E-06  9.238720E-07   85  15
20  3.141592E+00  1.907349E-06  3.576279E-07  105  17
21  3.141592E+00  9.536743E-07  2.682209E-07  117  18
22  3.141593E+00  4.768372E-07  1.490116E-07  133  19
23  3.141593E+00  2.384186E-07  8.940697E-08  153  20
24  3.141593E+00  1.192093E-07  2.980232E-08  185  21
25  3.141593E+00  5.960464E-08  0.000000E+00  213  22
26  3.141593E+00  2.980232E-08  0.000000E+00  253  23
27  3.141593E+00  1.490116E-08  0.000000E+00  285  24
```

LITERATURA:

[1] Bohte Z.: Numerične metode. (v I. Vidav: Višja matematika III). Ljubljana, 1973

[2] De Boor C.: CADRE: an algorithm for numerical quadrature. (v Mathematical software, J.R.Rice, ed.). Academic Press, New York, 417-449

[3] Dennis J.E. Jr., More J.J.: Computer solution of mathematical problems. (v Conway R., Gries D.: An Introduction to Programming). Winthrop, Cambridge, Mass., 1975, 358-366

[4] Kahaner D.K.: Comparision of numerical quadrature formulas. (v Mathematical software, J.R.Rice, ed.). Academic Press, New York, 229-259

[5] NAG fortran library manual, mark 6. NAG Ltd., 1977

[6] Oliver J.: A doubly-adaptive Clenshaw-Curtis quadrature method. The Computer Journal, 15(1972), 141-147

Vsa literatura je dosegljiva v Matematični knjižnici, Jadranska 19, Ljubljana.

```
==============================================
=                                            =
=    MAGIČNI KVADRATI SODE IN LIHE STOPNJE   =
=                                            =
==============================================


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                             %
%  Informatica UP 19                          %
%  Magic Squares of Even and Odd Degree       %
%  september 1984                             %
%  priredil Anton P. Železnikar               %
%  sistem CP-M, Delta Partner                 %
%  prevajalnik Janus-Ada, verzija 1.5.0       %
%                                             %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

1. Področje uporabe

Problematika magičnih kvadratov sodi v področje matematične in programirne rekreacije. Tu si je mogoče izmišljati različne algoritme za določevanje magičnih kvadratov sode in lihe stopnje.

2. Opis programa

Procedura magiceven v listi 1 določa za kvadratno matriko n krat n elemente z vrednostmi od 1 do n krat n in jih razmešča v leksikografskem zaporedju v polje x(1 .. n; 1 .. n), ko je n sodo celo število, ki ni manjše od 4 in oblikuje na ta način tkim. magični kvadrat.

V magičnem kvadratu so vsote elementov poljubne vrstice, poljubnega stolpca in poljubne diagonale medseboj enake.

Procedura magicodd v listi 1 oblikuje magični kvadrat podobno kot procedura magiceven, le da je stopnja n matrike x liha in ni manjša od 3.

Funkcija magicterm v listi 1 izračuna element s(i,j) magičnega kvadrata s(1 .. n, 1 .. n) za liho vrednost n, ki ni manjša od 3.

Lista 1 vsebuje še procedur za izpis magičnega kvadrata kvadr_izpis in proceduro za preizkušanje magičnosti kvadrata magic_test, ki najprej določi magično vsoto in nato preizkusi na to vsoto vse vrstice, vse stolpce in vse diagonale.

Z glavnim preizkusnim programom liste 1 se uporabijo zadevne procedure in funkcija (magiceven, magicodd in magicterm) z izpisom magičnih matrik in njihovim testiranjem.

3. Izvajanje programa

Lista 2 prikazuje izvajanje programa. Magični kvadrata 1 in 2 sta posledici uporabe procedure magiceven za n = 4 in za n = 6. Magična kvadrata 3 in 4 nastaneta z uporabo procedure magicodd za n = 3 in za n = 5. Magični kvadrat 5 se pojavi z uporabo funkcije magicterm za n = 5. Magični kvadrat 6 je rezultat uporabe procedure magiceven za n = 16 in magični kvadrat 7 je rezultat uporabe procedure magicodd za n = 17.

```
------------------------------------------------
--        Masični kvadrati sode in            --
--              in lihe stopnje               --
------------------------------------------------

WITH util;

PACKAGE BODY magici IS

  n: CONSTANT := 17;
  SUBTYPE d IS integer RANGE 1 .. n;
  TYPE stolpec IS ARRAY (d) OF integer;
  TYPE polje IS ARRAY (d) OF stolpec;

--  Naslednja procedura je predmet naše  pozor-
--     nosti (magični kvadrat sode stopnje)
--*********************************************

  PROCEDURE magiceven ( n: IN integer;
                        x: OUT polje ) IS

    a, b, n2, nn, i: integer;
    p, q, r: Boolean;

    PROCEDURE alpha ( p, q, a: IN integer;
                      h: IN Boolean ) IS
      hh: Boolean;
      BEGIN
        hh := h;
        FOR i IN p .. q LOOP
          hh := NOT hh;
          IF hh THEN
            x(i)(a) := nn - a*n + 1 + n - i;
          ELSE
            x(i)(a) := a*n - n + i;
          END IF;
        END LOOP;
      END alpha;
```

Lista 1. Procedure in programi za generiranje magičnih kvadratov (nadaljevanje na drugi str.)