ZASTOSOWANIA MATEMATYKI
APPLICATIONES MATHEMATICAE
XVII, 1 (1980)

ALGORITHM 79

J. MĄCZYŃSKI (Warszawa)

# AN ALGORITHM FOR FOURIER SERIES EXPANSION IN A SUBINTERVAL OF THE CIRCUMFERENCE

**1. Procedure declarations.** For each of the following three functional bases (controlled by *RED*, *BLUE* and *GREEN*, respectively, in the procedure body):

(i) $\{1.0, x, x^2, x^3, \ldots, x^{rk}\}$,

(ii) $\{\sin x, \cos x, \frac{1}{2}x\cos x, \frac{1}{2}x\sin x\}$,

(iii) $\{e^{\varkappa x}\cos\mu x, -e^{-\varkappa x}\cos\mu x, e^{\varkappa x}\sin\mu x, -e^{-\varkappa x}\sin\mu x\}$,

defined over a subinterval $I_0 = [0, d_{Ls}]$ of the half-circumference $I = [0, d_L]$ and over a set of coefficients $c_r$ ($r = 0, 1, \ldots, r_k$ for basis (i) and $r = 1, 2, 3, 4$ for bases (ii) and (iii)), the procedure *DOFO* produces (truncated to the $k_N$-th harmonic mode) Fourier series expansions valid within $I_0$ and uniformly convergent in $I_0$.

Data:

*select* — for *select* = 0, 1, 2 bases (i), (ii), (iii) are selected, respectively;

*rk* — degree of the polynomial of basis (i), otherwise *rk* = 4;

*p* — $p-1$ is the number of derivatives for which smoothness is preserved on the circumference;

*kN* — number of harmonic modes retained in the Fourier expansion;

*kap*, *mi* — parameters for basis (iii);

*ezk* — length of the "artefact" extension interval $I \setminus I_0$;

*dLs* — length of the subinterval $I_0$;

*c* — coefficient array: *c*[*0* : *rk*] for basis (i), and *c*[*1* : *4*] otherwise.

Results:

*a*[*0* : *kN*], *b*[*1* : *kN*] — arrays of Fourier expansion coefficients.

Other parameter:

*DETGAUSS* — Gaussian elimination subroutine with selection of the pivotal element and equilibration:

```
procedure DETGAUSS(n, m, A, exit);
  value n, m;
```

**integer** $n$, $m$;
**array** $A$;
**label** *exit*;

where:

$n$ — number of equations solved;
$m$ — number of right-hand side vectors;
$A[1:n, 1:n+m]$ — array of coefficients and right-hand side vectors stored as columns;
*exit* — failure escape label.

**2. Method used.** Consider arrays of Fourier coefficients $a[0:kN]$ and $b[1:kN]$ defined by the following two-term integral expressions of Euler-Fourier:

$$a_k = \frac{2}{d_L}\left[\sum_r \int_0^{d_{Ls}} c_r f_r(x)\cos\lambda_k x dx + \sum_r \int_{d_{Ls}}^{d_L} c_r W_r(d_L - x)\cos\lambda_k x dx\right]$$

($r$ even and $k = 0, 1, \ldots, r$),

(1)

$$b_k = \frac{2}{d_L}\left[\sum_r \int_0^{d_{Ls}} c_r f_r(x)\sin\lambda_k x dx + \sum_r \int_{d_{Ls}}^{d_L} c_r W_r(d_L - x)\sin\lambda_k x dx\right]$$

($r$ odd and $k = 1, 2, \ldots, r$),

where $\lambda_k = \pi k/d_L$ is a "wave number", $f_r(x)$ denotes the $r$-th element of the considered basis (cf. (i)-(iii)), $W_r(y)$ is an "artefact" polynomial selected so as to obtain the $C^{(p-1)}$-continuation of $f_r$ into $I \setminus I_0 = [d_{Ls}, d_L]$. The interval $I = [0, d_L]$ is considered as a half of the circumference. A circumference is defined as an interval whose endpoint is made to coincide with its initial point (an additive group of the real axis under the modulus $2d_L$; cf. [3]). The half-circumference is mirrored into the other half with respect to $x = 0$ when the function $f_r$ happens to be even (and we postulate it is definable over at least $[0, d_{Ls}] \cup [-d_{Ls}, 0]$) or mirrored with sign change when $f_r$ is odd. A suitable basis transformation in order to separate even and odd functions was applied in case of basis (iii). Similarly, even and odd polynomials are used accordingly as

(2)
$$W_r(d_L - x) = [a_0 + a_1(d - x)^2 + \ldots + a_{p-1}(d_L - x)^{2p-2}] \cdot \begin{cases} 1.0 & \text{(even)}, \\ (d_L - x) & \text{(odd)}, \end{cases}$$

where the coefficients $a_0, a_1, \ldots, a_{p-1}$ are determined by the $C^{(p-1)}$-smoothness at the junction $x = d_{Ls}$. The second terms in (1) are calculated by

```
procedure DOFO(select,rk,p,kN,kap,mi,ezk,dLs,c,a,b,DETGAUSS);
 value select,rk,p,kN,kap,mi,ezk,dLs;
 integer select,rk,p,kN;
 real kap,mi,ezk,dLs;
 array c,b,a;
 procedure DETGAUSS;
 begin
  integer i,j,k,setout,imax,imin;
  real dL,lambda,lamk,pi,ak,bk,z,dz,W,max,min,p1,p2,p3,p4,W1,
  W2,W3,ilamk,z0,z0p,zk,zkp,IN1,IN2;
  Boolean RED,BLUE,GREEN,Bo1,Bo2;
  RED:=select=0;
  comment basis (i);
  BLUE:=select=1;
  comment basis (ii);
  GREEN:=select=2;
  comment basis (iii);
  pi:=3.1415926535;
  dL:=dLs+ezk;
  begin
   array alf[1:p],F[1:rk,1:p];
   real pow,clam,slam,a0,c2idL,idL2,c12idL,c22idL,c32idL,
   c42idL,ekdLs,p1,p2,p3,p4;
   integer r,k,q;
   Boolean ODD;
   procedure INO(IN1,IN2,N,b,ilam);
    value N,b,ilam;
    integer N;
    real IN1,IN2,b,ilam;
    begin
```

```
integer k;
real gb1,gb2,bk,I1,I2,pom;
gb1:=cos(b/ilam);
gb2:=sin(b/ilam);
bk:=1.0;
I1:=ilam×gb2;
I2:=ilam×(1.0-gb1);
for k:=1 step 1 until N do
 begin
  bk:=bk×b;
  pom:=ilam×(bk×gb2-k×I2);
  I2:=ilam×(-bk×gb1+k×I1);
  I1:=pom
 end k;
IN1:=I1;
IN2:=I2
end IN0;
if BLUE
 then
 begin
  real a,b,c,d,e,f;
  a:=0.5×cos(dLs);
  b:=0.5×sin(dLs);
  e:=b×dLs;
  f:=a×dLs;
  for k:=1 step 1 until p do
   begin
    F[2,k]:=a+a;
    F[1,k]:=b+b;
    F[4,k]:=e;
```

```
      F[3,k]:=f;
      c:=a;
      a:=-b;
      b:=c;
      d:=e;
      e:=f-a;
      f:=b-d
     end k
  end BLUE;
 if GREEN
  then
  begin
   real x,y,z,sig,a,b,c,d,f;
   x:=kap;
   y:=mi;
   z:=sig:=dens;
   b:=y×z;
   a:=exp(x×z);
   c:=cos(b);
   d:=sin(b);
   f:=a×c;
   b:=-c/a;
   c:=a×d;
   d:=-d/a;
   a:=f;
   a:=a-b;
   a:=0.5×a;
   b:=b+a;
   c:=c+d;
   c:=0.5×c;
```

```
d:=c-d;
for q:=1 step 1 until p do
 begin
  F[2,q]:=a;
  F[3,q]:=b;
  F[4,q]:=c;
  F[1,q]:=d;
  f:=x×a-y×c;
  a:=x×c+y×a;
  c:=x×d+y×b;
  d:=x×b-y×d;
  b:=f;
  f:=a;
  a:=d;
  d:=f
 end q
end GREEN;
if BLUEVGREEN
 then
 begin
  idL2:=2.0/dL;
  c12idL:=c[1]×idL2;
  c22idL:=c[2]×idL2;
  c32idL:=c[3]×idL2;
  c42idL:=c[4]×idL2;
  if GREEN
   then
   begin
    p1:=0.25×(c12idL+c22idL);
    c22idL:=0.25×(c12idL-c22idL):
```

```
    c12idL:=0.25×(c32idL-c42idL);
    c42idL:=0.25×(c32idL+c42idL);
    c32idL:=p1
   end GREEN
 end BLUE V GREEN;
a0:=0.0;
if BLUE
 then a0:=sin(dLs)×(c22idL+0.5×c42idL)-cos(dLs)×c42idL×dLs×
          0.5;
if GREEN
 then
 begin
  p1:=exp(kap×dLs);
  p2:=1.0/p1;
  p1:=0.5×(p1+p2);
  comment cosh(kap×dLs);
  p2:=p1-p2;
  comment sinh(kap×dLs);
  p1:=p1×sin(mi×dLs);
  p2:=-p2×cos(mi×dLs);
  a0:=(c22idL×(mi×p1-kap×p2)+c42idL×(kap×p1+mi×p2))/(kap×
     kap+mi×mi);
  a0:=4.0×a0;
  comment c12idL - c42idL formerly divided by 4;
  a0:=a0+a[0]
 end GREEN;
if RED
 then a0:=2×c[0];
z0:=0.0;
z0p:=z0;
```

```
zkp:=zOp+dL;
zk:=zkp-ezk;
if REDVBLUE
 then
 for k:=1 step 1 until kN do
  a[k]:=b[k]:=0.0;
a[0]:=a0;
a0:=0.0;
pow:=dLs;
ODD:=Bo1:=Bo2:=false;
kap:=-kap;
mi:=-mi;
for r:=1 step 1 until rk do
 begin
  integer i,j,k,l;
  real S,T;
  array B[1:p],A[1:p,1:p+1];
  ODD:=¬ODD;
  a0:=0.0;
  if RED
   then
   begin
    B[1]:=dLs××r;
    for j:=1 step 1 until p-1 do
     B[j+1]:=B[j]/dLs×(r+1-j)
   end RED;
  if BLUEVGREEN
   then
   for k:=1 step 1 until p do
    B[k]:=F[r,k];
```

```
      S:=if ODD then -ezk else 1.0;
      T:=ezk×ezk;
      for j:=1 step 1 until p do
       begin
        A[1,j]:=S;
        S:=S×T
       end j;
      if ODD
       then l:=1
       else l:=0;
      for i:=2 step 1 until p do
       for j:=1 step 1 until p do
        A[i,j]:=-(1+2×j-i)×A[i-1,j]/ezk;
      for i:=1 step 1 until p do
       A[i,p+1]:=B[i];
      go to SOL;
EXIT:for i:=1 step 1 until p do
       A[i,i]:=₁₀-5;
SOL: DETGAUSS(p,1,A,EXIT);
      for i:=1 step 1 until p do
       alf[i]:=A[i,p+1];
      c2idL:=2.0×c[r]/dL;
      if GREEN
       then
       begin
        c2idL:=4.0×(if r=1 then c12idL else if r=2 then c22idL
              else if r=3 then c32idL else c42idL);
        Bo1:=¬Bo1;
        if Bo1
         then
```

```
      begin
       Bo2:=¬Bo2;
       mi:=-mi
      end Bo1;
     kap:=-kap;
     ekdLs:=exp(kap×dLs)
    end GREEN;
   for k:=1 step 1 until kN do
    begin
     lamk:=k×pi/dL;
     ilamk:=1.0/lamk;
     ak:=bk:=0.0;
     clam:=cos(lamk×dL);
     slam:=sin(lamk×dL);
     if RED
      then INO(IN1,IN2,r,dLs,ilamk);
     if BLUE∧r=1
      then
      begin
       real fi11,fi22,fi31,fi42,ws,wr,wsins,wsinr,wcoss,
       wcosr,ps,pr,Isl,Irl;
       Isl:=1+lamk;
       Irl:=1-lamk;
       ws:=1.0/Isl;
       wr:=1.0/Irl;
       ps:=dLs×Isl;
       pr:=dLs×Irl;
       wsins:=sin(ps)×ws;
       wcoss:=cos(ps)×ws;
       wsinr:=sin(pr)×wr;
```

```
    wcosr:=cos(pr)×wr;
    fi11:=0.5×(wsins+wsinr);
    fi22:=fi11-wsins;
    fi31:=(wsins-ps×wcoss)/(Isl+Isl);
    fi42:=(wsinr-pr×wcosr)/(Irl+Irl);
    fi31:=0.5×(fi31+fi42);
    fi42:=fi31-fi42;
    ak:=ak+c22idL×fi11+c42idL×fi31;
    bk:=bk+c12idL×fi22+c32idL×fi42
  end BLUE ∧ r=1;
 if GREEN
  then
  begin
    real ro,wspro,sirodLs,corodLs;
    ro:=lamk+mi;
    wspro:=1.0/(ro×ro+kap×kap);
    sirodLs:=sin(ro×dLs);
    corodLs:=cos(ro×dLs);
    IN1:=wspro×(ekdLs×(ro×sirodLs+kap×corodLs)-kap);
    IN2:=wspro×(ekdLs×(kap×sirodLs-ro×corodLs)+ro);
    p1:=c12idL×IN1;
    p2:=c22idL×IN1;
    p3:=c32idL×IN2;
    p4:=c42idL×IN2;
    bk:=bk+(if Bo2 then -p1 else p1)+(if Bo1 then p3 else
        -p3);
    ak:=ak+p2+(if Bo1≡Bo2 then p4 else -p4)
  end GREEN;
 if ODD
  then
```

```
    begin
     if RED
      then bk:=bk+IN2×c2idL;
     for q:=1 step 1 until p do
      begin
       IN0(IN1,IN2,2×q-1,-ezk,ilamk);
       bk:=bk-alf[q]×(clam×IN2+slam×IN1)×c2idL
      end q
    end ODD
    else
    begin
     if RED
      then ak:=ak+IN1×c2idL;
     for q:=1 step 1 until p do
      begin
       IN0(IN1,IN2,2×(q-1),-ezk,ilamk);
       ak:=ak-alf[q]×(IN1×clam-IN2×slam)×c2idL
      end q
    end ¬ODD;
   a[k]:=a[k]+ak;
   b[k]:=b[k]+bk
  end k;
 if ¬ODD
  then
  begin
   S:=c2idL×ezk;
   T:=ezk×ezk;
   for q:=1 step 1 until p do
    begin
     a0:=a0+S×alf[q]/(q+q-1);
```

```
        S:=S×T
      end q
    end ¬ODD;
    pow:=pow×(zk-z0p);
    if RED∧¬ODD
     then a0:=a0+pow×c21dL/(r+1);
    a[0]:=a[0]+a0
  end r
 end alf,F
end DOFO
```

recursive paired formulae (cf. the internal procedure *INO* in *DOFO*):

$$\int_0^b x^r \cos\lambda_k x\,dx = \frac{1}{\lambda_k}\left(b^r \sin\lambda_k b - r\int_0^b x^{r-1}\sin\lambda_k x\,dx\right),$$

$$\int_0^b x^r \sin\lambda_k x\,dx = \frac{1}{\lambda_k}\left(-b^r \cos\lambda_k b + r\int_0^b x^{r-1}\cos\lambda_k x\,dx\right)$$

$$(r = 1, 2, \ldots, r_k).$$

The first terms in (1) are calculated in accordance with the type of basis and accumulated as contributions within the cycle for $r$.

The unknown coefficients $\alpha_0, \alpha_1, \ldots, \alpha_{p-1}$ in (2) are computed from the condition of smoothness at the junction:

$$f_r^{(q)}(d_{Ls}) = (-1)^q W_r^{(q)}(ezk) \qquad (q = 0, 1, \ldots, p-1). \tag{3}$$

The $q$-th right-hand side derivatives in (3) are computed recursively and kept in the array $A[1:p, 1:p+1]$ as coefficients of the unknown $\alpha$'s of equation (2). This is done recursively in two steps.

Step 1:

$$A_{1j} = A_{1,j-1}e_{zk}^2,$$

where according to the function being either odd or even we have

$$A_{1,1} = \begin{cases} -e_{zk} & \text{(odd)}, \\ 1 & \text{(even)}. \end{cases}$$

Step 2:

$$A_{ij} = -(l+2j-i)A_{i-1,j}/e_{zk},$$

where $l = 1$ for an odd function and $l = 0$ for an even one.

The left-hand side derivatives $f_r^{(q)}(d_{Ls})$ are obtained by suitable recursive formulae. These arise since the bases may be visualized as spanning abstract spaces which are closed under differentiation. This means that derivatives are expressed as linear combinations $g_r f_r$ of basis functions $f_r$, the new coefficients $g_r$ depending only on $c_r$ and constant parameters, and not on the function argument $x$.

In the matrix notation, a derivative at $x$ of the element

$$\boldsymbol{f} = \{f_1, \ldots, f_n\}^T$$

is $\boldsymbol{Df}$ or $\boldsymbol{M^{-1}D^*Mf}$ when the basis has previously been transformed with the help of some non-singular $\boldsymbol{M}$. For basis (i), obviously,

$$\boldsymbol{D} = \text{superdiag}[1 \quad 2 \quad 3 \quad \ldots \quad n-1]$$

is used and $\boldsymbol{M} = \boldsymbol{I}$. For basis (ii) derivatives are calculated as $\boldsymbol{M^{-1}D^*Mf}$ with

$$\boldsymbol{D^*} = \begin{bmatrix} 0 & 2 & 0 & 0 \\ -2 & 0 & 0 & 0 \\ 0 & 1 & 0 & -1 \\ 1 & 0 & 1 & 0 \end{bmatrix}$$

(note the "simple" elements) and $\boldsymbol{M} = \text{diag}[\frac{1}{2} \quad \frac{1}{2} \quad 1 \quad 1]$. For basis (iii)

$$\boldsymbol{D^*} = \begin{bmatrix} 0 & \mu & 0 & \varkappa \\ -\mu & 0 & \varkappa & 0 \\ 0 & \varkappa & 0 & -\mu \\ \varkappa & 0 & \mu & 0 \end{bmatrix}$$

is used throughout, since a transformed auxiliary basis with alternating odd and even elements is introduced:

$$\boldsymbol{f^* = Mf} \tag{4}$$

with

$$\boldsymbol{M} = \frac{1}{2}\begin{bmatrix} 0 & 0 & 1 & -1 \\ 1 & -1 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \end{bmatrix}.$$

Therefore, $c$'s are transformed as $\boldsymbol{c^{*T} = c^T M^{-1}}$. Because of the simple form of the matrices, and to enhance the operation speed of the procedure, matrix multiplications have been worked out and the resulting formulae appear in the implementation.

For basis (i) derivatives are put directly into the vector array $B[1:p]$, whereas for bases (ii) and (iii) each row (corresponding to a basis element) of $F[1:rk, 1:p]$ contains successive derivatives. Next, the derivatives

are transferred to the $(p+1)$-st column of the array $A$ being used as input data for the procedure *DETGAUSS* which solves the linear system of equations and produces the result in the last column of $A$. Hence it is used to evaluate the second terms in (1). Further explanation is needed to understand the calculation of the first term integrals in (1). For basis (i) these integrals are computed by using the internal procedure *INO*. Contributions to basis (ii) are obtained by an obvious direct integration and they are produced in the first pass of the $r$-cycle. The corresponding integrals for basis (iii) are calculated as successive contributions in the four passes of the $r$-cycle. For the $M$-transformed basis we write

$$\delta a_k = \frac{2}{d_L} \sum_r c_k^* \int_0^{d_{Ls}} f_r^* \cos \lambda_k x \, dx = \frac{2}{d_L} \sum_r c_r^* I_{ar} \quad (r \text{ even}), \tag{5}$$

$$\delta b_k = \frac{2}{d_L} \sum_r c_r^* \int_0^{d_{Ls}} f_r^* \sin \lambda_k x \, dx = \frac{2}{d_L} \sum_r c_r^* I_{br} \quad (r \text{ odd}). \tag{6}$$

Then, by forming sums and differences, the integrals are simplified:

$$I_{ai} + I_{b,3-i} \operatorname{sign} \mu + (I_{a,5-i} + I_{b,2+i} \operatorname{sign} \mu) \operatorname{sign} \varkappa = (\operatorname{sign} \mu)^i \int_0^{d_{Ls}} e^{\varkappa x} \sin(\lambda_k + \mu) x \, dx = (\operatorname{sign} \mu)^i I_{ni}(\varkappa, \mu, \lambda_k, d_{Ls}), \tag{7}$$

$$\varkappa = |\varkappa|, \ -|\varkappa|; \ \mu = |\mu|, \ -|\mu|; \ i = 1, 2.$$

The integrals $I_{a1}, \ldots, I_{b4}$ of equations (5) and (6) result from (7) by repeated addition and subtraction by pairs of the eight equations. The factor 1/4 is introduced in the procedure at an earlier stage together with the $M$-transformation (equation (4)).

After all the contributions to $a$'s and $b$'s have been accumulated the procedure terminates. Interpreting the work of the procedure *DOFO* as a basis transformation in a Banach space may be useful when devising applications for the algorithm.

**3. Certification.** Test calculations were performed for all basis elements of bases (i)-(iii) taken separately and for some combinations. Tabulation of the input function

$$y^{(0)} = \sum_r c_r f_r$$

and of the Fourier expansion supplied by *DOFO* was performed for $k_N = 10$ at 41 equally spaced points in $I_0$ (including endpoints). A "taxi"

norm was defined by

$$\frac{\sum_i |y_i^{(0)} - y_i^{(1)}|}{n\left(\operatorname{Max}_i(y_i^{(0)}) - \operatorname{Min}_i(y_i^{(1)})\right)}$$

with $n = 41$. Values of the norm are listed in Table 1.

TABLE 1

| $p-1$ | Norm not exceeding |
|---|---|
| 1 | 0.005 |
| 2 | 0.0023 |
| 3 | 0.00088 |
| 4 | 0.00033 |

Each time the interval $I_0$ was [0.0, 1.047] and $e_{zk} = 0.209$. The norm was smaller by a factor of 2 for basis (ii). The parameters $\varkappa$ and $\mu$ were 0.9 and 2.2, respectively. The accuracy is improving with increasing $p$ although no check was made on possible limitations (due to the round-off error or other reasons), since $p = 3$ proved to be sufficient for most applications.

Careful reading of the algorithm shows that further optimization of the performance is possible by some further "interweaving" of instructions. This, however, would further decrease the readability of the procedure text and was not attempted for publication purposes.

**4. Additional remarks.** Continuity on the circumference [3] is a fundamental prerequisite of the Jackson-Bernstein-Zygmund-Korovkin theory of function approximation by Fourier series of trigonometric type (cf. [4] for ample literature). Since in this case known error estimates allow to prescribe the number of terms needed in an expansion for a desired accuracy, limitations to accuracy result theoretically from technical or economical reasons such as computer word length or computing time restrictions.

The above-mentioned condition is however too restrictive for many applications to problems of heat conduction or structural mechanics (see [1] and [2]) and, therefore, devising a method to cover a wider scope of problems seems appropriate. It appears that the periodicity interval of the Fourier representation is sometimes not necessarily identified with some characteristic length in the boundary value problem, unless the independent variable is time, as in problems of vibrations or waves. Early attempts to use Fourier series for discontinuous functions had

enormous heuristic value but failed to provide a reliable tool when anomalies like the Gibbs effect precluded the use of Fourier series for calculation of derivatives at vital endpoints of intervals.

By introducing an artefact external extension of the interval, continuity on the circumference may be reestablished and unwanted effects are made harmless as may be expected from the existing vast theoretical literature on the subject. When considering the basis (iii) it was found that a one-sided extension of the interval may at times be insufficient and another procedure with a two-sided extension was written and successfully tried.

The paper was prepared in partial fulfilment of the Order 109 of the author's Institute research task. The author would like to acknowledge the help of Miss Dunajska who tested and checked the successive versions of the algorithm.

**References**

[1] H. S. Carslaw and J. C. Jaeger, *Conduction of heat in solids*, Russian edition, Moscow 1964.
[2] W. Flügge, *Stresses in shells*, Polish edition, Warszawa 1972.
[3] J.-P. Kahane, *Séries de Fourier absolument convergentes*, Russian edition, Moscow 1976.
[4] G. Meinardus, *Approximation von Funktionen und ihre numerische Behandlung*, Polish edition, Warszawa 1968.

INSTITUTE OF FUNDAMENTAL TECHNICAL RESEARCH
POLISH ACADEMY OF SCIENCES
00-049 WARSZAWA

*Received on 25. 7. 1977*

---

ALGORYTM 79

J. MĄCZYŃSKI (Warszawa)

## ALGORYTM ROZWIJANIA FUNKCJI W SZEREG FOURIERA W PODPRZEDZIALE OKRĘGU

STRESZCZENIE

Procedura *DOFO* wyznacza rozwinięcia w szereg Fouriera dla każdej z baz funkcyjnych (i)-(iii), określonych na podprzedziale $I_0 = [0, d_{Ls}]$ półokręgu $I = [0, d_L]$ i dla danych współczynników $c_r$ ($r = 0, 1, \ldots, r_k$ dla bazy (i) oraz $r = 1, 2, 3, 4$ dla baz (ii) i (iii)).

Dane:

*select* – dla *select* = 0, 1, 2 wybiera się odpowiednio bazę (i), (ii), (iii);
*rk* – stopień wielomianu bazy (i) lub 4 dla baz (ii) i (iii);
*p* – $p-1$ jest liczbą pochodnych, dla których zachowana jest gładkość na okręgu;
*kN* – liczba składowych harmonicznych wchodzących w skład rozwinięcia w szereg Fouriera;
*kap, mi* – parametry bazy (ii);
*ezk* – długość „sztucznego" przedziału przedłużenia $I \setminus I_0$;
*dLs* – długość podprzedziału $I_0$;
*c* – tablica współczynników : $c[0:rk]$ dla bazy (i), a $c[1:4]$ w przeciwnym razie.

Wyniki:

$a[0:kN]$, $b[1:kN]$ – tablice współczynników rozwinięcia w szereg Fouriera.

Inny parametr:

*DETGAUSS* – procedura rozwiązująca układ równań liniowych, mająca nagłówek

**procedure** *DETGAUSS*(*n*, *m*, *A*, *exit*);
**value** *n*, *m*;
**integer** *n*, *m*;
**array** *A*;
**label** *exit*;

gdzie

*n* – liczba równań;
*m* – liczba prawych stron układu;
$A[1:n, 1:n+m]$ – tablica współczynników i prawych stron, zapamiętanych kolumnami;
*exit* – etykieta, do której się skacze w przypadku błędu.