ZASTOSOWANIA MATEMATYKI
APPLICATIONES MATHEMATICAE
XV, 3 (1976)

ALGORITHM 48

S. LEWANOWICZ (Wrocław)

# SOLVING BOUNDARY VALUE PROBLEMS FOR THE BIHARMONIC EQUATION BY THE METHOD OF SUMMARY REPRESENTATIONS

**1. Procedure declaration.** The procedure *biharmeqP* solves the biharmonic equation

$$\Delta\Delta u = \frac{\partial^4 u}{\partial x^4} + 2\frac{\partial^4 u}{\partial x^2 \partial y^2} + \frac{\partial^4 u}{\partial y^4} = f(x, y) \tag{1}$$

over the open rectangle

$$D = \{(x, y) \mid a_1 < x < a_2,\ b_1 < y < b_2\}$$

under the following boundary conditions:

1° on the horizontal sides of $D$ we have

$$u\Big|_{y=b_j} = \varphi_{0j}(x), \qquad \frac{\partial^2 u}{\partial y^2}\Big|_{y=b_j} = \varphi_{1j}(x) \quad (j = 1, 2), \tag{2}$$

2° on each vertical side $x = a_j$ $(j = 1, 2)$ one of the systems of equations

$$u\Big|_{x=a_j} = \psi_{0j}(y), \qquad \frac{\partial^2 u}{\partial x^2}\Big|_{x=a_j} = \psi_{1j}(y) \tag{3}$$

or

$$u\Big|_{x=a_j} = \psi_{0j}(y), \qquad \frac{\partial u}{\partial x}\Big|_{x=a_j} = \psi_{1j}(y) \tag{4}$$

or

$$\left[\frac{\partial^2 u}{\partial x^2} + \sigma\frac{\partial^2 u}{\partial y^2}\right]_{x=a_j} = \psi_{0j}(x), \qquad \left[\frac{\partial^3 u}{\partial x^3} + (2-\sigma)\frac{\partial^3 u}{\partial x \partial y^2}\right]_{x=a_j} = \psi_{1j}(x) \tag{5}$$

holds, where $\varphi_{ij}$ and $\psi_{ij}$ $(i = 0, 1;\ j = 1, 2)$ are given functions, and $\sigma$ is a constant (Poisson's ratio).

Let the uniform mesh domain be defined by

$$(6)\qquad D_h = \{(x_i, y_k) \mid x_i = a_1 + ih,\ y_k = b_1 + kh_1 \quad (i = 0, 1, \ldots, m+1;\ k = 0, 1, \ldots, n+1)\},$$

where $h = (a_2 - a_1)/(m+1)$, and $h_1 = (b_2 - b_1)/(n+1)$. Let $u_{ik}$ denote the approximate value of the solution of (1) at the point $(x_i, y_k)$.

Data:

*f* — functional procedure of type **real**, with parameters $x$ and $y$ of type **real**, i. e., the right-hand side of equation (1);

*a1, a2, b1, b2* — $a_1$, $a_2$, $b_1$, $b_2$, respectively;

*fi* — functional procedure of type **real**, with parameters $i$, $j$ of type **integer** and $x$ of type **real**; the function designator $fi(i, j, x)$ corresponds to the value of $\varphi_{ij}(x)$ for given $i, j$ and $x$;

*psi* — functional procedure of type **real**, with parameters $i, j$ of type **integer** and $y$ of type **real**; the function designator $psi(i, j, y)$ corresponds to the value of $\psi_{ij}(y)$ for given $i, j$ and $y$;

*m, n* — mesh parameters occurring in (6);

*left, right* — *left* (*right*) should be equal to 1, 2 or 3 according to the type of boundary conditions (3), (4) or (5), respectively, prescribed on the left (right) vertical side of the rectangle $D$;

*sigma* — Poisson's ratio $\sigma$.

Results:

*u[1 : m, 1 : n]* — array containing the approximate values of the solution at the nodal points (6); $u[i, k] = u_{ik}$ for $i = 1, 2, \ldots, m$ and $k = 1, 2, \ldots, n$.

**2. Method used.** Strictly speaking, the procedure solves the difference equation

$$\Delta_h \Delta_h u_{ik} = f(x_i, y_k)$$

on the domain $D_h$, where $\Delta_h \Delta_h$ is the finite difference 13-point biharmonic operator. For solving this equation with conditions which are finite difference replacements of given boundary conditions we use the method of

```
procedure biharmeqP(f,a1,a2,b1,b2,fi,psi,m,n,left,right,
 sigma,u);
 value m,n;
 integer m,n,left,right;
 real a1,a2,b1,b2,sigma;
 array u;
 real procedure f,fi,psi;
 begin
  integer hm,i,iz,i1,j,k,mm1,m1,m2,nn1,n1,tm1,type;
  real alm,al1,bem,be1,cc,cga2,ck,ck1,ck2,cm1,c0,den,denb,
  den1,den2,d1a1,d1a2,d2a1,d2a2,fip,fi1,fi1a1,fi1a2,fi1m,
  fi2,fi2a1,fi2a2,fi2m,ga2,ga4,hx,hx2,hx3,hx4,hy,ia,ib,iw,
  lk,mk,ni,ni2,r,r1,r2,sga2,t,tc,tek,thx,tk,t01,t02,t11,t12,
  um1,u0,va,vb,vw,wm,wm1,w0,w1;
  Boolean sped;
  array fk[1:m],nk[-1:m+2],p[1:n,1:n],p01,p02,p11,p12,y[1:n],
  tf[1:m,1:n];
  procedure albe(i,al,be);
   integer i;
   real al,be;
   if i≥iz
    then al:=be:=.0
    else
    begin
     integer p;
     real fi,nki,nkp;
     p:=m1-i;
     nki:=nk[i];
     if p+p≥iz
      then
```

```
        begin
         al:=nki×den×(1.0+i×lk);
         be:=-nki×den1×i
        end p+p≥iz
        else
        begin
         nkp:=nk[p];
         nkp:=nkp×nkp;
         fi:=nkp×den2×(1.0-nki×nki)-i×(1.0+nkp);
         al:=nki×den×(1.0-nkp-lk×fi);
         be:=nki×den1×fi
        end ¬p+p≥iz
      end ¬i≥iz,albe;
procedure W(i,w1,w2);
 integer i;
 real w1,w2;
 begin
  real s1,s2,t;
  integer p,p1,q;
  s1:=s2:=.0;
  q:=m1;
  for p:=1 step 1 until m do
   begin
    q:=q-1;
    p1:=abs(i-p);
    if p1<iz
     then
     begin
      t:=nk[p1]×(p1+lk);
      s1:=s1+t×fk[p];
```

```
      s2:=s2+t×fk[q]
     end p1<iz
   end p;
   w1:=mk×s1;
   w2:=mk×s2
  end W;
procedure modif(i);
 integer i;
 begin
  real y1,yn;
  y1:=y[1];
  yn:=y[n];
  if i=1
   then
   begin
    t01:=hx2×t01-c0-sga2×(y1×fi1a1+yn×fi2a1);
    t11:=-hx3×t11+cga2×(y1×d1a1+yn×d2a1)
   end i=1
   else
   begin
    t02:=hx2×t02-cm1-sga2×(y1×fi1a2+yn×fi2a2);
    t12:=hx3×t12-cga2×(y1×d1a2+yn×d2a2)
   end ¬i=1
 end modif;
switch bcond:=ss,sp,sf,ps,pp,pf,fs,fp,ff;
type:=3×(left-1)+right;
sped:=if left≤2 then right≤2 else false;
m1:=m+1;
tm1:=m1+m1;
m2:=m+2;
```

```
mm1:=m-1;
hm:=entier(.5×m);
if hm+hm≠m
 then hm:=hm+1;
n1:=n+1;
nm1:=n-1;
t:=1.0/n1;
r:=b1;
hy:=(b2-b1)×t;
for i:=1 step 1 until n do
 begin
  y[i]:=r:=r+hy;
  p01[i]:=psi(0,1,r);
  p02[i]:=psi(0,2,r);
  p11[i]:=psi(1,1,r);
  p12[i]:=psi(1,2,r)
 end i;
hx:=(a2-a1)/m1;
hx2:=hx×hx;
hx3:=hx×hx2;
hx4:=hx2×hx2;
thx:=hx+hx;
ga2:=hx2/(hy×hy);
ga4:=ga2×ga2;
cc:=2.0+ga2;
if ¬sped
 then
 begin
  sga2:=sigma×ga2;
  cga2:=ga2+ga2-sga2
```

```
 end ¬sped;
fi1a1:=fi1m:=fi(0,1,a1);
fi2a1:=fi2m:=fi(0,2,a1);
r:=a1+hx;
fi1:=fi(0,1,r);
fi2:=fi(0,2,r);
d1a1:=fi1-fi1m;
d2a1:=fi2-fi2m;
for i:=1 step 1 until m do
 begin
  for k:=1 step 1 until n do
   tf[i,k]:=hx4×f(r,y[k]);
  r1:=r+hx;
  fip:=fi(0,1,r1);
  r2:=cc×fi1-fip-fi1m;
  tf[i,1]:=tf[i,1]+ga2×(r2+r2-hx2×fi(1,1r));
  tf[i,2]:=tf[i,2]-ga4×fi1;
  fi1m:=fi1;
  fi1:=fip;
  fip:=fi(0,2,r1);
  tf[i,nm1]:=tf[i,nm1]-ga4×fi2;
  r2:=cc×fi2-fip-fi2m;
  tf[i,n]:=tf[i,n]+ga2×(r2+r2-hx2×fi(1,2,r));
  fi2m:=fi2;
  fi2:=fip;
  r:=r1
 end i;
fi1a2:=fi1;
d1a2:=fi1-fi1m;
fi2a2:=fi2;
```

```
d2a2:=f12-f12m;
r:=-sqrt(t+t);
t:=3.1415926536×t;
r1:=.0;
r2:=r×sin(t);
ck2:=cos(t);
tc:=ck2+ck2;
nk[0]:=ck1:=1.0;
for k:=1 step 1 until n do
 begin
  ck:=tc×ck1-ck2;
  if k>2
   then
   begin
    for j:=k-1 step -1 until 1 do
     y[j]:=p[k,j]:=p[j,k];
    r1:=y[k-1];
    r2:=y[k-2]
   end k>2
   else
    if k=2
     then
     begin
      r1:=y[1]:=p[2,1]:=p[1,2];
      r2:=.0
     end k=2;
  t:=ck+ck;
  for j:=k step 1 until n do
   begin
    r:=y[j]:=p[k,j]:=t×r1-r2;
```

```
  r2:=r1;
  r1:=r
 end j;
ni:=1.0+ga2×(1.0-ck);
ni:=t:=nk[1]:=ni-sqrt(ni×ni-1.0);
ni2:=ni×ni;
for i:=2 step 1 until m2 do
 nk[i]:=t:=ni×t;
nk[-1]:=1.0/ni;
t01:=t02:=t11:=t12:=.0;
for j:=1 step 1 until n do
 begin
  r:=y[j];
  t01:=t01+r×p01[j];
  t02:=t02+r×p02[j];
  t11:=t11+r×p11[j];
  t12:=t12+r×p12[j]
 end j;
cc:=1.0;
for i:=1 step 1 until m do
 begin
  t:=.0;
  for j:=1 step 1 until n do
   t:=t+y[j]×tf[i,j];
  fk[i]:=t;
  t:=abs(t);
  if t>cc
   then cc:=t
 end i;
iz:=entier(-(ln(m×cc)+27.6310211159)/ln(ni))+1;
```

```
mk:=1.0/(1.0-ni2);
den1:=mk×ni;
if tm1≥iz
 then
 begin
  den:=1.0;
  den2:=tm1
 end tm1≥iz
 else
 begin
  t:=nk[m1];
  den:=1.0/(1-t×t);
  den1:=den×den1;
  den2:=tm1×den
 end ¬tm1≥iz;
lk:=mk×(1.0+ni2);
mk:=ni2×mk×mk;
W(-1,r1,r2);
W(0,w0,wm1);
W(1,w1,wm);
c0:=r1+w1;
cm1:=r2+wm;
albe(1,al1,be1);
denb:=1.0-be1-be1;
albe(m,alm,bem);
if sped
 then
 begin
  u0:=t01-w0;
  um1:=t02-wm1
```

```
        end sped
        else
        begin
         tek:=1.0-ck;
         tk:=cga2×tek;
         tk:=3.0+tk+tk;
         tek:=1.0+sga2×tek;
         tek:=tek+tek;
         t:=-alm-tek×bem;
         W(2,fi1,fi2);
         iw:=-r1-tk×w1+fi1;
         vw:=-r2-tk×wm+fi2;
         albe(-1,fi1,fi2);
         albe(2,r1,r2);
         ia:=-fi1-tk×al1+r1;
         ib:=-fi2-tk×be1+r2;
         albe(m2,fi1,fi2);
         albe(mm1,r1,r2);
         va:=-fi1-tk×alm+r1;
         vb:=-fi2-tk×bem+r2;
         cc:=-ia-tek×ib-tk
        end ¬sped;
       go to bcond[type];
ss:    c0:=hx2×t11+t01+t01-c0;
       cm1:=hx2×t12+t02+t02-cm1;
       go to form;
sp:    c0:=hx2×t11+t01+t01-c0;
       cm176thx×t12-cm1;
       cm1:=cm1+2.0×(alm×u0+al1×um1+bem×c0+be1×cm1+wm)/denb;
       go to form;
```

```
sf: u0:=t01-w0;
    c0:=hx2×t11+t01+t01-c0;
    modif(2);
    um1:=(t12+va×u0-ia×wm1+ib×t02+vb×c0+vw)/cc;
    cm1:=t02+tek×um1;
    um1:=um1-wm1;
    go to form;
ps: c0:=-thx×t11-c0;
    cm1:=hx2×t12+t02+t02-cm1;
    c0:=c0+2.0×(al1×u0+alm×um1+be1×c0+bem×cm1+w1)/denb;
    go to form;
pp: c0:=-thx×t11-c0;
    cm1:=thx×t12-cm1;
    r1:=al1×u0+alm×um1+be1×c0+bem×cm1+w1;
    r2:=alm×u0+al1×um1+bem×c0+be1×cm1+wm;
    bem:=bem+bem;
    r:=2.0/(denb×denb-bem×bem);
    c0:=c0+r×(denb×r1+bem×r2);
    cm1:=cm1+r×(denb×r2+bem×r1);
    go to form;
pf: u0:=t01-w0;
    modif(2);
    t11:=-thx×t11-c0;
    r1:=al1×u0-alm×wm1+bem×t02+be1×t11+w1;
    r2:=t12+va×u0-ia×wm1+ib×t02+vb×t11+vw;
    r:=-1.0/(2.0×vb×t+cc×denb);
    um1:=-r×(2.0×vb×r1+denb×r2);
    cm1:=t02+tek×um1;
    um1:=um1-wm1;
    c0:=t11+2.0×r×(t×r2-cc×r1);
```

```
    go to form;
fs: modif(1);
    um1:=t02-wm1;
    cm1:=hx2×t12+t02+t02-cm1;
    u0:=(t11-ia×w0+ib×t01+va×um1+vb×cm1+iw)/cc;
    c0:=t01+tek×u0;
    u0:=u0-w0;
    go to form;
fp: um1:=t02-wm1;
    modif(1);
    t12:=thx×t12-cm1;
    r1:=-alm×w0+bem×t01+al1×um1+be1×t12+wm;
    r2:=t11-ia×w0+ib×t01+va×um1+vb×t12+iw;
    r:=-1.0/(2.0×vb×t+cc×denb);
    u0:=-r×(2.0×vb×r1+denb×r2);
    c0:=t01+tek×u0;
    u0:=u0-w0;
    cm1:=t12+2.0×r×(t×r2-cc×r1);
    go to form;
ff: modif(1);
    modif(2);
    r1:=t11-ia×w0+ib×t01-va×wm1+vb×t02+iw;
    r2:=t12-va×w0+ib×t02-ia×wm1+vb×t01+vw;
    t:=-va-tek×vb;
    r:=1.0/(cc×cc-t×t);
    u0:=r×(cc×r1-t×r2);
    c0:=t01+tek×u0;
    u0:=u0-w0;
    um1:=r×(cc×r2-t×r1);
    cm1:=t02+tek×um1;
```

```
    um1:=um1-wm1;
form:
    for i:=1 step 1 until hm do
     begin
      albe(i,al1,be1);
      i1:=m1-i;
      albe(i1,alm,bem);
      W(i,w1,wm);
      u[i,k]:=al1×u0+alm×um1+be1×c0+bem×cm1+w1;
      if i1>i
       then u[i1,k]:=alm×u0+al1×um1+bem×c0+be1×cm1+wm
     end i;
    ck2:=ck1;
    ck1:=ck
   end k;
  for i:=1 step 1 until m do
   begin
    for j:=1 step 1 until n do
     y[j]:=u[i,j];
    for k:=1 step 1 until n do
     begin
      t:=.0;
      for j:=1 step 1 until n do
       t:=t+p[k,j]×y[j];
      u[i,k]:=t
     end k
   end i
 end biharmeqP
```

summary representations [1]-[3]. The main formulae used are

$$\boldsymbol{u}_i = \boldsymbol{P}\boldsymbol{v}_i,$$

$$\boldsymbol{v}_i = \boldsymbol{A}_i(\boldsymbol{v}_0-\boldsymbol{w}_0)+\boldsymbol{A}_{m-i+1}(\boldsymbol{v}_{m+1}-\boldsymbol{w}_{m+1})+\boldsymbol{B}_i(\boldsymbol{c}_0-\boldsymbol{w}_{-1}-\boldsymbol{w}_1)+ \\ +\boldsymbol{B}_{m-i+1}(\boldsymbol{c}_{m+1}-\boldsymbol{w}_m-\boldsymbol{w}_{m+2})+\boldsymbol{w}_i \quad (i = 1, 2, \ldots, m),$$

where $\boldsymbol{u}_i$, $\boldsymbol{v}_i$ are $n$-dimensional vectors, $\boldsymbol{u}_i = [u_{i1}, u_{i2}, \ldots, u_{in}]'$, $\boldsymbol{P}$ is the $n$-th order square matrix with elements

$$p_{rs} = \sqrt{\frac{2}{n+1}}\sin\frac{rs\pi}{n+1} \quad (r, s = 1, 2, \ldots, n),$$

$\boldsymbol{A}_i$, $\boldsymbol{B}_i$ are known diagonal matrices of order $n$, and $\boldsymbol{w}_i$ ($i = -1, 0, \ldots, m+2$) are vectors depending on the right-hand side of equation (6) and on the given boundary conditions on the horizontal sides of $D_h$. The vectors $\boldsymbol{v}_0$, $\boldsymbol{v}_{m+1}$, $\boldsymbol{c}_0$ and $\boldsymbol{c}_{m+1}$ are to be determined by using the boundary conditions assumed on the vertical sides of $D_h$.

**3. Certification.** The procedure was applied to many problems. In particular, we consider all (topologically) different combinations of boundary conditions of specified types for the equations

$$\Delta\Delta u = 0 \quad (0 < x, y < 1) \tag{7}$$

with the exact solution $u = \sin\pi x \exp[\pi(y-1)]$ and

$$\Delta\Delta u = 4\pi^4 \sin\pi x \cos\pi y \quad (0 < x, y < 1) \tag{8}$$

with the theoretical solution $u = \sin\pi x \cos\pi y$. The value of the parameter $\sigma$ is always taken equal to 0.25 and we put $m = n = 19$.

The calculations were carried out on the ODRA 1204 computer at the Institute of Informatics of the University of Wrocław. The maximal relative errors of the received solutions are shown in Tables 1 and 2.

TABLE 1. Equation (7)

| | | right 1 | right 2 | right 3 |
|---|---|---|---|---|
| left | 1 | $18_{10}-4$ | $42_{10}-4$ | $41_{10}-4$ |
| left | 2 | | $45_{10}-4$ | $162_{10}-4$ |
| left | 3 | | | $48_{10}-4$ |

TABLE 2. Equation (8)

| | | right 1 | right 2 | right 3 |
|---|---|---|---|---|
| left | 1 | $14_{10}-4$ | $37_{10}-4$ | $76_{10}-4$ |
| left | 2 | | $37_{10}-4$ | $72_{10}-4$ |
| left | 3 | | | $78_{10}-4$ |

**References**

[1] P. I. Chalenko (П. И. Чаленко), *Решение некоторых задач об изгибе прямоугольных пластин со свободными краями*, Вычислительная и прикладная математика 7 (1969), p. 3-16.

[2] — *Об одном варианте расчетных формул решения некоторых краевых задач для бигармонического уравнения методом суммарных представлений*, ibidem 9 (1969), p. 19-32.

[3] G. N. Polozhiĭ, *The method of summary representation for numerical solution of problems of mathematical physics*, London 1965.

INSTITUTE OF INFORMATICS
UNIVERSITY OF WROCŁAW
50-384 WROCŁAW

*Received on 10. 4. 1975*

---

ALGORYTM 48

S. LEWANOWICZ (Wrocław)

# ROZWIĄZYWANIE ZAGADNIEŃ BRZEGOWYCH DLA RÓWNANIA BIHARMONICZNEGO METODĄ REPREZENTACJI SUMARYCZNYCH

STRESZCZENIE

Procedura *biharmeqP* rozwiązuje równanie biharmoniczne (1) w prostokącie

$$D = \{(x, y) \mid a_1 < x < a_2, b_1 < y < b_2\}$$

dla następujących warunków brzegowych: na bokach $y = b_j$ $(j = 1, 2)$ zadane są warunki (2), natomiast na każdym z boków $x = a_j$ $(j = 1, 2)$ — warunki (3), (4) lub (5), gdzie $\varphi_{ij}$ i $\psi_{ij}$ $(i = 0, 1;\ j = 1, 2)$ są danymi funkcjami, a $\sigma$ oznacza współczynnik Poissona. Obliczane są przybliżone wartości rozwiązania w węzłach siatki (6).

Dane:

*f* — funkcja rzeczywista z parametrami $x$ i $y$ typu **real**; prawa strona równania (1);

*a1, a2, b1, b2* — odpowiednio $a_1, a_2, b_1, b_2$;

*fi* — funkcja rzeczywista z parametrami $i, j$ typu **integer** i $x$ typu **real**, której wartością dla danych $i, j, x$ jest $\varphi_{ij}(x)$;

*psi* — funkcja rzeczywista z parametrami $i, j$ typu **integer** i $y$ typu **real**, której wartością dla danych $i, j, y$ jest $\psi_{ij}(y)$;

*m, n* — liczby naturalne nie mniejsze od 2, występujące w (6);

*left, right* — *left* (*right*) jest równe 1, 2 lub 3, gdy na boku $x = a_1$ $(x = a_2)$ prostokąta $D$ zadane są odpowiednio warunki (3), (4) lub (5);

*sigma* — współczynnik Poissona $\sigma$.

Wyniki:

$u[1{:}m, 1{:}n]$ — tablica przybliżonych wartości rozwiązania w węzłach siatki (6); $u[i,k] = u(x_i, y_k)$ dla $i = 1, 2, \ldots, m$ oraz $k = 1, 2, \ldots, n$.

W procedurze *biharmeqP* zastosowano metodę reprezentacji sumarycznych [1]-[3], którą krótko scharakteryzowano w punkcie 2. W punkcie 3 omówiono wyniki przykładów kontrolnych, wykonanych na maszynie cyfrowej ODRA 1204.