ZASTOSOWANIA MATEMATYKI
APPLICATIONES MATHEMATICAE
XV, 2 (1976)

S. LEWANOWICZ (Wrocław)

ALGORITHM 47

SOLUTION OF A FIRST-ORDER NON-LINEAR DIFFERENTIAL EQUATION IN CHEBYSHEV SERIES

1. Procedure declaration. Procedure *nldiffeqNort* calculates the approximate values of the coefficients of the Chebyshev series expansion (i.e. the series of Chebyshev polynomials T_k of the first kind) of the solution of the differential equation

$$y' = f(x, y) \quad (-1 \leqslant x \leqslant 1) \tag{1}$$

with the initial condition

$$y(\beta) = \gamma \tag{2}$$

or the boundary condition

$$\alpha y(-1)+\beta y(1) = \gamma, \tag{3}$$

i.e. it calculates the coefficients a_k $(k = 0, 1, \ldots, n;\ n \leqslant N)$ such that

$$y(x) \approx \sum_{k=0}^{n}{}' a_k T_k(x) \quad (-1 \leqslant x \leqslant 1). \tag{4}$$

The symbol $\sum'$ in (4) denotes summation with the first summand halved:

$$\sum_{k=0}^{n}{}' t_k = \frac{1}{2} t_0 + t_1 + \ldots + t_n.$$

Data:

n — integer variable, with value not less than 4 and even, denoting the trial value of the upper summation limit in (4);

a[0: n] — array of trial values of the coefficients in (4) ($a[k] = a_k$ for $k = 0, 1, \ldots, n$) determining a polynomial satisfying either (2) or (3);

nmax — even natural number $N \geqslant 4$, the maximum permissible value of n;

s — non-negative integer number, the degree of the polynomial approximating the function $f_y(x, y(x))$ (see Section 2, formula (13));

init — **true** if the solution satisfies the initial condition (2), and **false** if the solution satisfies the boundary condition (3);

ffy — identifier of the procedure having 4 parameters x, y (called by value), f, fy of type **real**, which for given x and y assigns to f the value of $f(x, y)$ and to fy the value of the partial derivative $f_y(x, y)$;

al, be, ga — numbers α, β and γ from (2) and (3); in case (2) the value of *al* is arbitrary;

eps — positive number ε characterizing the permissible absolute error of a_k (see Section 2, inequalities (39) and (40));

maxit — maximum permissible number of iterations R in the auxiliary iteration process (see Section 2, relation (17));

Chebval — real procedure identifier with parameters n of type **integer**, a of type **array**, and x of type **real**, the value of which for given n, a and x is the sum in (4) (see, e.g., [9]).

Remark. The declaration of the array *a* should be at least [*0* : *nmax*+*1*].

Results:

n — upper summation limit of the right-hand side of (4);

a[*0*: *n*] — array of the coefficients giving a sufficiently good approximation of the solution of equation (1);

maxit — number of calculated approximations of the solution of equation (1) (see Section 2, formula (5)).

Other parameters:

ns — label of the instruction (outside of the procedure body) to which a jump is made in the case where it is not possible to calculate the solution;

mark — integer variable whose value is determined only if a jump to label *ns* has been made; *mark* = 1 indicates that *nmax* was too small to achieve the desired accuracy, and *mark* = 2 indicates that after *maxit* iterations the auxiliary iteration process led to a solution which did not have the desired accuracy.

2. Method used. The following variant of Norton's method [5] has been used. A sequence of approximations $\{y_i\}$ of the solution of either problem (1), (2) or problem (1), (3) is determined. The function y_0 is arbitrary (e.g. a polynomial) provided it satisfies (2) or (3), respectively, and y_i $(i = 1, 2, \ldots)$ is the solution of the linear differential equation

$$\frac{dy_i}{dx} = f(x, y_{i-1}) + (y_i - y_{i-1}) f_y(x, y_{i-1}) \tag{5}$$

```
procedure nldiffeqNort(n,a,nmax,s,init,ffy,al,be,ga,eps,
 maxit,Chebval,ns,mark);
 value nmax,s,al,be,ga;
 integer n,nmax,s,maxit,mark;
 real al,be,ga,eps;
 Boolean init;
 array a;
 label ns;
 real procedure Chebval;
 procedure ffy;
 begin
  integer i,it,i1,i2,i3,i4,k,k1,k2,m,nc,nc4;
  real ab,ak,akm1,akp1,bk,bkm1,bkp1,c0,c1,eps1,ga2,gk,hab,
  k4,qk,rc0,rc4,rm,s1,s2,t,trn,wk,x0,x1,x2,zk;
  Boolean bc0,sgt0;
  array b,d,q,x[0:nmax],c[0:s],g,w,z[1:nmax];
  nc:=n;
  n:=nc+1;
  if init
   then ga2:=-ga-ga
   else
   begin
    ab:=al+be;
    hab:=.5×ab
   end ¬ init;
  for k:=n step 1 until nmax do
   a[k]:=b[k]:=.0;
  eps1:=.1×eps;
  nmax:=nmax-1;
  sgt0:=s>0;
```

```
   it:=0;
iter:
   it:=it+1;
   if n≠nc
    then
    begin
     nc4:=4×nc;
     rn:=1.0/nc;
     trn:=rn+rn;
     t:=3.1415926536×rn;
     x0:=x[0]:=1.0;
     x1:=x[1]:=cos(t);
     t:=x1+x1;
     for k:=2 step 1 until nc do
      begin
       x2:=x[k]:=t×x1-x0;
       x0:=x1;
       x1:=x2
      end k
    end n ≠ nc;
   for k:=nc-1 step -1 until 0 do
    begin
     t:=x[k];
     ffy(t,Chebval(nc,a,t),b[k],d[k])
    end k;
   ffy(-1.0,Chebval(nc,a,-1.0),c1,c0);
   b[nc]:=.5×c1;
   d[nc]:=.5×c0;
   for k:=0 step 1 until s do
    c[k]:=trn×Chebval(nc,d,x[k]);
```

```
c0:=c[0];
bc0:=abs(c0)>eps1;
if bc0
 then
 begin
  rc0:=1.0/c0;
  rc4:=4.0×rc0;
  t:=nc4×rc0
 end bc0;
if sgt0
 then c1:=c[1];
akp1:=bkp1:=wk:=zk:=.0;
ak:=a[nc];
bk:=rn×Chebval(nc,b,-1.0);
for k:=nc step -1 until 1 do
 begin
  k1:=k-1;
  akm1:=a[k1];
  bkm1:=trn×Chebval(nc,b,x[k1]);
  gk:=g[k]:=2.0×(bkp1-bkm1)+c0×(akm1-akp1);
  akp1:=ak;
  ak:=akm1;
  bkp1:=bk;
  bk:=bkm1;
  if bc0
   then
   begin
    wk:=w[k]:=1.0/(wk+t);
    zk:=z[k]:=-wk×(gk×rc0+zk);
    t:=t-rc4
    end bc0
    else z[k]:=gk
  end k:
 if bc0
  then
  begin
   if init
    then
    begin
     q[0]:=1.0;
     qk:=q[1]:=w[1];
     for k:=2 step 1 until nc do
      qk:=q[k]:=w[k]×qk;
     s2:=-1.0/Chebval(nc,q,be)
    end init
    else
    begin
     s2:=ab;
     for k:=nc step -1 until 1 do
      begin
       al:=-al;
       s2:=w[k]×s2+al+be
      end k;
     s2:=1.0/(s2-hab)
    end ¬ init
  end bc0;
 m:=0;
again:
 if bc0
  then
```

```
begin
 if init
  then
  begin
   q[0]:=ga2;
   qk:=q[1]:=z[1];
   for k:=2 step 1 until nc do
    qk:=q[k]:=w[k]×qk+z[k];
   bk:=b[0]:=Chebval(nc,q,be)×s2
  end init
  else
  begin
   s1:=ga;
   t:=.0;
   for k:=1 step 1 until nc do
    begin
     al:=-al;
     t:=t×w[k]+z[k];
     s1:=s1-t×(al+be)
    end k;
   bk:=b[0]:=s1×s2
  end ¬ init;
 for k:=1 step 1 until nc do
  bk:=b[k]:=w[k]×bk+z[k]
end bc0
else
begin
 k4:=-4.0;
 for k:=1 step 1 until nc do
  begin
```

```
              b[k]:=z[k]/k4;
          k4:=k4-4.0
        end k;
      if init
        then
        begin
          b[0]:=.0;
          b[0]:=2.0×(ga-Chebval(nc,b,be))
        end init
        else
        begin
          s1:=s2:=.0;
          for k:=2 step 2 until nc do
            begin
              s1:=s1+b[k-1];
              s2:=s2+b[k]
            end k;
          b[0]:=2.0×(ga-al×(s2-s1)-be×(s1+s2))
        end ¬ init
      end ¬ bc0;
    if sgt0
      then
      begin
        if m>0
          then
          begin
            for k:=0 step 1 until nc do
              if abs(b[k]-d[k])≥eps
                then go to impg;
            go to next
      end m > 0;
impg:
    m:=m+1;
    if m>maxit
      then
      begin
        mark:=2;
        go to ns
      end m > maxit;
    zk:=.0;
    for k:=nc step -1 until 1 do
      begin
        d[k]:=b[k];
        i1:=k+2;
        i2:=abs(k-2);
        t:=g[k]+c1×(b[i1]-b[i2]-a[i1]+a[i2]);
        k1:=k+1;
        k2:=k-1;
        for i:=2 step 1 until s do
          begin
            i1:=k2+i;
            i2:=k1+i;
            i3:=abs(k2-i);
            i4:=abs(k1-i);
            t:=t+c[i]×(b[i2]+b[i4]-b[i1]-b[i3]-a[i2]-a[i4]+a[i1]
                +a[i3])
          end i;
        if bc0
          then zk:=z[k]:=-w[k]×(t×rc0+zk)
          else z[k]:=t
      end k;
      d[0]:=b[0];
      go to again
     end sgt0;
 next:
    n:=nc;
    for k:=0 step 1 until nc do
     if abs(a[k]-b[k])≥eps
      then go to degt;
    for k:=0 step 1 until nc do
     a[k]:=b[k];
    maxit:=it;
    go to endp;
 degt:
    if abs(a[0]-b[0])+abs(a[1]-b[1])<abs(b[nc-3])+abs(b[nc-2])
     then
     begin
      if nc<nmax
       then nc:=nc+2
       else
       begin
        mark:=1;
        go to ns
       end nc ≥ nmax
     end abs;
    for k:=0 step 1 until n do
     a[k]:=b[k];
    go to iter;
 endp:
  end nldiffeqNort
```

with the condition

$$y_i(\beta) = \gamma \tag{6}$$

or

$$\alpha y_i(-1)+\beta y_i(1) = \gamma, \tag{7}$$

in accordance with the condition posed upon the solution of (1).

Assume that all functions appearing in (5) can be expanded into Chebyshev series and let

$$y_i(x) = \sum_{k=0}^{\infty}{}' A_k^{(i)} T_k(x), \tag{8}$$

$$\frac{dy_i(x)}{dx} = \sum_{k=0}^{\infty}{}' A_k'^{(i)} T_k(x), \tag{9}$$

$$f\big(x, y_{i-1}(x)\big) = \sum_{k=0}^{\infty}{}' b_k^{(i)} T_k(x), \tag{10}$$

$$f_y\big(x, y_{i-1}(x)\big) = \sum_{k=0}^{\infty}{}' c_k^{(i)} T_k(x). \tag{11}$$

To facilitate the notation we omit the upper index i in the coefficients of these series and write a_k instead of $A_k^{(i-1)}$. Thus

$$y_{i-1}(x) = \sum_{k=0}^{\infty}{}' a_k T_k(x). \tag{12}$$

Let $s \geqslant 0$. Instead of the exact formula (11) we use the approximation

$$f_y\big(x, y_{i-1}(x)\big) \approx \sum_{k=0}^{s}{}' c_k T_k(x). \tag{13}$$

The approximate values of the coefficients b_k and c_k of expansions (10) and (13), respectively, for the given function $y_{i-1}(x)$ can be obtained by the method described in [2], Section 4.

Replacing in (5) the functions y_i, dy_i/dx, $f(x, y_{i-1})$, y_{i-1} and $f_y(x, y_{i-1})$ by the right-hand sides of (8)-(10), (12) and (13), respectively, and using the relations (see, e.g., [1])

$$2kA_k = A'_{k-1} - A'_{k+1} \quad (k = 1, 2, \ldots),$$

$$2T_p(x)T_q(x) = T_{p+q}(x) + T_{|p-q|}(x) \quad (p, q = 0, 1, \ldots),$$

we obtain the equation

$$c_0 A_{k-1} - 4kA_k - c_0 A_{k+1} = g_k \quad (k = 1, 2, \ldots), \tag{14}$$

where

$$(15)\qquad g_k = 2(b_{k+1}-b_{k-1})+c_0(a_{k-1}-a_{k+1})+$$
$$+\sum_{j=1}^{s} c_j(d_{k+j+1}+d_{k-j+1}-d_{k+j-1}-d_{|k-j-1|}),\qquad d_p = A_p - a_p \quad (p \geqslant 0).$$

By (6) we can obtain the relation

$$(16)\qquad \sum_{k=0}^{\infty}{}' \tau_k A_k = \gamma,$$

where $\tau_k = T_k(\beta)$. In case of condition (7) relation (16) holds also for $\tau_k = (-1)^k\alpha+\beta$.

In [5] an iterative method of solution of problem (14)-(16) is proposed. If we take $A_{0k} = a_k$, we can construct sequences $\{A_{rk}\}$ $(r = 1, 2, \ldots)$ such that

$$(17)\qquad c_0 A_{r,k-1} - 4kA_{rk} - c_0 A_{r,k+1} = g_{rk} \quad (r, k = 1, 2, \ldots),$$

where g_{rk} denotes the right-hand side of the first formula of (15) in which it was assumed $d_p = A_{r-1,p}-a_p$ $(p \geqslant 0)$; in addition, we have

$$(18)\qquad \sum_{k=0}^{\infty}{}' \tau_k A_{rk} = \gamma \quad (r = 1, 2, \ldots).$$

For a fixed r the problem lies in solving the difference equation of the second order

$$(19)\qquad c_0 a_{k-1} - 4ka_k - c_0 a_{k+1} = \gamma_k \quad (k = 1, 2, \ldots),$$

where $\gamma_k = g_{rk}$ with the condition

$$(20)\qquad \sum_{k=0}^{\infty}{}' \tau_k a_k = \gamma.$$

The case $c_0 = 0$ is trivial. Assume now $c_0 \neq 0$. Notice that the homogeneous equation

$$(21)\qquad c_0 a_{k-1} - 4ka_k - c_0 a_{k+1} = 0 \quad (k = 1, 2, \ldots)$$

has a general solution of the form

$$C_1 \sigma^k I_k\left(\frac{|c_0|}{2}\right) + C_2(-\sigma)^k K_k\left(\frac{|c_0|}{2}\right) \quad (\sigma = \operatorname{sign} c_0),$$

where I_k and K_k are modified Bessel functions of the first and second kind, respectively, and where C_1 and C_2 are arbitrary constants.

Since for $k \to \infty$

$$I_k(t) \to 0 \quad \text{and} \quad K_k(t) \to \infty,$$

thus $\sigma^k I_k(|c_0|/2)$ is the minimal solution and $(-\sigma)^k K_k(|c_0|/2)$ is the dominating solution of (21) (see [3], Introduction). Applying the results obtained by Oliver [8] we see that the solution of problem (19), (20) is of the form

$$a_k = C\sigma^k I_k\left(\frac{|c_0|}{2}\right) + \psi_k,$$

where C is an appropriate constant and ψ_k is a particular solution of (19) such that $\psi_k \to 0$ as $k \to \infty$.

In [5] the approximate solution $a_k^{(n)}$ (n sufficiently large) is constructed with the help of the formulas

(22) $$a_k^{(n)} = C\varphi_k^{(n)} + \psi_k^{(n)} \quad (k = 1, 2, \ldots, n),$$

where

(23) $$\varphi_{n+1}^{(n)} = 0, \quad \varphi_n^{(n)} = 1, \quad \varphi_{k-1}^{(n)} = \varphi_{k+1}^{(n)} + 4k\varphi_k^{(n)}/c_0 \quad (k = n, n-1, \ldots, 1)$$

and

(24) $$\psi_{n+1}^{(n)} = \psi_{n+2}^{(n)} = 0, \quad \psi_{k-1}^{(n)} = \psi_{k+1}^{(n)} + (4k\psi_k^{(n)} + \gamma_k)/c_0 \quad (k = n+1, n, \ldots, 1),$$

and the constant C is determined in such a manner that the following condition holds:

$$\sum_{k=0}^{n}{}' \tau_k a_k^{(n)} = \gamma.$$

Hence

(25) $$C = \left(\gamma - \sum_{k=0}^{n}{}' \tau_k \psi_k^{(n)}\right) \Big/ \sum_{k=0}^{n}{}' \tau_k \varphi_k^{(n)}.$$

Formulae (23) describe the algorithm of Miller (backward recurrence algorithm) for equation (21). In view of Theorem 3.1 from [3], for a fixed k we have

$$\varphi^{(n)} \to \varphi_k \quad (\text{as } n \to \infty),$$

where φ_k is the minimal solution of equation (21) which differs from $\sigma^k I_k(|c_0|/2)$ only by a constant.

The algorithm given by (22)-(25) appears to have a strong non-stability when with the increasing k the value of $|a_k|$ decreases significantly slowlier than $I_k(|c_0|/2)$ (see [6], Section 13). Also, it is known ([7] and [8]) that replacing the initial problem (19), (20) by an appropriate boundary problem leads to stability.

Assume that the system of linear equations

$$c_0 a_{k-1}^{(n)} - 4k a_k^{(n)} - c_0 a_{k+1}^{(n)} = \gamma_k \quad (k = 1, 2, \ldots, n), \tag{26}$$

$$a_{n+1}^{(n)} = 0, \tag{27}$$

$$\sum_{k=0}^{n}{}' \tau_k a_k^{(n)} = \gamma \tag{28}$$

has the solution $a_0^{(n)}, a_1^{(n)}, \ldots, a_n^{(n)}$ for all sufficiently large n. One can prove ([8], Section 9) that, provided additional conditions are satisfied, the relation

$$a_k^{(n)} \to a_k \quad \text{for fixed } k \text{ and } n \to \infty$$

holds.

We propose the following method of solving (26)-(28). First, using (27) in the last ($k = n$) equation of system (26), we obtain a relation of the form

$$a_n^{(n)} = W_n^{(n)} a_{n-1}^{(n)} + Z_n^{(n)}, \tag{29}$$

where $W_n^{(n)} = c_0/4n$ and $Z_n^{(n)} = -\gamma_n/4n$. Using the last but one ($k = n-1$) equation of system (26) and also equation (29) to the elimination of unknown $a_n^{(n)}$, we obtain the equation

$$a_{n-1}^{(n)} = W_{n-1}^{(n)} a_{n-2}^{(n)} + Z_{n-2}^{(n)},$$

where

$$W_{n-1}^{(n)} = \frac{c_0}{c_0 W_n^{(n)} + 4(n-1)} \quad \text{and} \quad Z_{n-1}^{(n)} = -\frac{\gamma_{n-1} + c_0 Z_n^{(n)}}{c_0 W_n^{(n)} + 4(n-1)}.$$

Continuing this elimination leads us to a system being equivalent to system (26), (27)

$$a_k^{(n)} = W_k^{(n)} a_{k-1}^{(n)} + Z_k^{(n)} \quad (k = 1, 2, \ldots, n), \tag{30}$$

where

$$W_k^{(n)} = \frac{c_0}{c_0 W_{k+1}^{(n)} + 4k} \quad (k = n,\ n-1, \ldots, 1;\ W_{n+1}^{(n)} = 0), \tag{31}$$

$$Z_k^{(n)} = -\frac{\gamma_k + c_0 Z_{k+1}^{(n)}}{c_0 W_{k+1}^{(n)} + 4k} \quad (k = n,\ n-1, \ldots, 1;\ Z_{n+1}^{(n)} = 0). \tag{32}$$

Using (30) it is possible to derive (by induction) the relation

$$a_k^{(n)} = p_k^{(n)} a_0^{(n)} + q_k^{(n)} \quad (k = 1, 2, \ldots, n), \tag{33}$$

where

$$(34)\qquad \begin{aligned} p_0^{(n)} &= 1, & p_k^{(n)} &= W_k^{(n)} p_{k-1}^{(n)}, \\ q_0^{(n)} &= 0, & q_k^{(n)} &= W_k^{(n)} q_{k-1}^{(n)} + Z_k^{(n)} \end{aligned} \qquad (k = 1, 2, \ldots, n).$$

This and (28) lead to the formula

$$(35)\qquad a_0^{(n)} = \Big(\gamma - \sum_{k=1}^{n} \tau_k q_k^{(n)}\Big) \Big/ \sum_{k=0}^{n}{}' \tau_k p_k.$$

From (30) (or (33)) one can successively calculate $a_1^{(n)}, a_2^{(n)}, \ldots, a_n^{(n)}$.

Let us now investigate the accumulation of the rounding errors during the realization of algorithm (30)-(35). We have by (31) the formula

$$(36)\qquad W_k^{(n)} = \cfrac{c_0}{4k + \cfrac{c_0^2}{4(k+1) + \ldots + \cfrac{c_0^2}{4n}}}.$$

Applying the known theorem of Pincherle (see, e.g., [3], Section 1) for fixed k we have

$$(37)\qquad \lim_{n\to\infty} W_k^{(n)} = \varphi_k / \varphi_{k-1},$$

where φ_k is the minimal solution of (21). Therefore, for sufficiently large n we obtain

$$|W_k^{(n)}| \approx I_k\left(\frac{|c_0|}{2}\right) \Big/ I_{k-1}\left(\frac{|c_0|}{2}\right),$$

from which it follows that $W_k^{(n)}$, whence also the rounding errors arising during the calculation, cannot increase rapidly.

It follows from (36) that $W_k^{(n)}$ and c_0 are of equal sign, and hence the denominators of the fractions in (31) and (32) are sums of two positive summands with the order $4k$. Stability can be lost by the multiple loss of significant places during the calculation of $\gamma_k + c_0 Z_{k+1}^{(n)}$ for $n = n, n-1, \ldots, 1$. This has as yet not been observed in practice.

Now, it follows that the calculation of $p_k^{(n)}$ and $q_k^{(n)}$ is stabile. Loss of accuracy is, however, possible during the calculation of $a_0^{(n)}$ after formula (35) which, in turn, causes a small accuracy of all $a_k^{(n)}$ $(k = 1, 2, \ldots, n)$. This fact can be observed, e.g., when the expression $|\frac{1}{2}\tau_0 + \sigma^{(n)}|$, where

$$(38)\qquad \sigma^{(n)} = \sum_{k=1}^{n} \tau_k p_k^{(n)},$$

is small with respect to $\frac{1}{2}|\tau_0|$. Notice that $p_k^{(n)} = W_1^{(n)} W_2^{(n)} \ldots W_k^{(n)}$ (see the first formula of (34)). Therefore from (37) we obtain

$$\sigma^{(n)} = \sum_{k=1}^{n} \tau_k W_1^{(n)} W_2^{(n)} \ldots W_k^{(n)} \approx \frac{1}{I_0(|c_0|/2)} \sum_{k=1}^{n} \tau_k \sigma^k I_k\left(\frac{|c_0|}{2}\right).$$

In view of

$$e^{cx} = 2\sum_{k=0}^{\infty}{}'(\operatorname{sign} c)^k I_k(|c|)T_k(x) \qquad (-1 \leqslant x \leqslant 1)$$

(see, e.g., [4], p. 32), for the initial problem (1), (2) we have

$$\sigma^{(n)} \approx \frac{1}{2I_0(|c_0|/2)}\left[\exp\left(\frac{1}{2}c_0\beta\right) - I_0\left(\frac{|c_0|}{2}\right)\right],$$

and for the boundary problem (1), (3) we obtain

$$\sigma^{(n)} \approx \frac{1}{2I_0(|c_0|/2)}\left\{\alpha\left[\exp\left(-\frac{1}{2}c_0\right) - I_0\left(\frac{|c_0|}{2}\right)\right] + \beta\left[\exp\left(\frac{1}{2}c_0\right) - I_0\left(\frac{|c_0|}{2}\right)\right]\right\}.$$

Thus $|(\frac{1}{2}\tau_0 + \sigma^{(n)})/\frac{1}{2}\tau_0|$ is small if

$$\left[\exp\left(\frac{1}{2}c_0\beta\right)\right]\Big/2I_0\left(\frac{|c_0|}{2}\right) \quad \text{or} \quad \left|\alpha\exp\left(-\frac{1}{2}c_0\right) + \beta\exp\left(\frac{1}{2}c_0\right)\right|\Big/2I_0\left(\frac{|c_0|}{2}\right),$$

respectively, are small. In practice, however, the value of each of these expressions is of moderate order.

The above-described method of solving the problem (19), (20) is a generalization of the so-called first algorithm of Gautschi ([3], Section 3) of solving a homogeneous difference equation of the order two. In fact, if in (19) $\gamma_k \equiv 0$ holds, then from (32) and from the second line of (34) it follows that $Z_k^{(n)} = q_k^{(n)} = 0$ for $k = 1, 2, \ldots, n$, and formulae (30)-(35) are reduced to the form

$$\begin{aligned}
a_k^{(n)} &= W_k^{(n)} a_{k-1}^{(n)} && (k = 1, 2, \ldots, n),\\
W_k^{(n)} &= \frac{c_0}{c_0 W_{k+1}^{(n)} + 4k} && (k = n, n-1, \ldots, 1;\ W_{n+1}^{(n)} = 0),\\
a_0^{(n)} &= \frac{\gamma}{\frac{1}{2}\tau_0 + \sigma^{(n)}},
\end{aligned}$$

where $\sigma^{(n)}$, defined by (38), can be calculated from the formulae

$$\begin{aligned}
\sigma_k^{(n)} &= W_k^{(n)}(\tau_k + \sigma_{k+1}^{(n)}) \qquad (k = n, n-1, \ldots, 1;\ \sigma_{n+1}^{(n)} = 0),\\
\sigma^{(n)} &= \sigma_1^{(n)}.
\end{aligned}$$

These formulae are to be compared with system (3.9) in [3].

Problem (17), (18) is solved by use of (30)-(35) for $r = 1, 2, \ldots, r_0 \leqslant R$, where r_0 is the smallest natural number such that

$$\max_{0\leqslant k\leqslant n}|A_{r_0 k} - A_{r_0-1,k}| < \varepsilon, \tag{39}$$

where ε is a given positive number. Next, let us take $A_k^{(i)} = A_{r_0 k}$ for $k = 0, 1, \ldots, n$.

If the inequality

$$\max_{0 \leqslant k \leqslant n} |A_k^{(i)} - A_k^{(i-1)}| < \varepsilon \tag{40}$$

holds, then $A_k^{(i)}$ $(k = 0, 1, \ldots, n)$ are the required coefficients of the sum in (4). Otherwise,

1° if the inequality

$$|A_0^{(i)} - A_0^{(i-1)}| + |A_1^{(i)} - A_1^{(i-1)}| < |A_{n-2}^{(i)}| + |A_{n-3}^{(i)}|$$

holds, then we increase n by 2 (see [2]) and

2° we proceed to the calculation of $A_k^{(i+1)}$ $(k = 0, 1, \ldots, n)$, and so forth.

Certification. The procedure has been verified on the Odra 1204 computer for the following problems:

(41) $y' = y^2$, $y(-1) = 0.4$ (solution $y = 2/(3-2x)$);

(42) $y' = x - y^2$, $y(0) = -0.729011132947$

(solution $y = \mathrm{Ai}'(x)/\mathrm{Ai}(x)$, where $\mathrm{Ai}(x) = \frac{1}{\pi}\int_0^\infty \cos\left(\frac{1}{3}t^3 + xt\,dt\right)$);

(43) $y' = \sin y$, $y(-1) = \arccos\tanh 1$
(solution $y = \arccos(-\tanh x)$),

(44) $y' = 1 - \sqrt{|y|} + \cos\pi x$, $y(-1) - y(1) = 0$ (see [5], Section 7).

It was assumed that $y_0(x) \equiv \gamma$ in problems (41)-(43) and that $y_0(x) \equiv 1$ in problem (44). The results for $\varepsilon = 5_{10}-9$, *nmax* $= 100$ and *maxit* $= 50$ were as follows:

Problem	n given	s	Iteration number	n calculated	$10^{10}\Delta$
(41)	20	0	14	24	13
		2	9	22	12
(42)	14	0	10	16	5
		2	5	16	10
(43)	16	0	10	16	7
		1	7	18	11
(44)	22	0	6	22	16
		2	6	22	6
		3	5	22	19

The last column contains the maximum absolute error of the calculated coefficient values, multiplied by 10^{10}.

References

[1] C. W. Clenshaw, *The numerical solution of linear differential equations in Chebysh series*, Proc. Cambridge Phil. Soc. 53 (1957), p. 134-149.

[2] — and H. J. Norton, *The solution of non-linear ordinary differential equatio in Chebyshev series*, Comp. J. 6 (1963), p. 88-92.

[3] W. Gautschi, *Computational aspects of three-term recurrence relations*, SIA Rev. 9 (1967), p. 24-82.

[4] Y. L. Luke, *The special functions and their approximations*, New York 1969.

[5] H. J. Norton, *The iterative solution of non-linear ordinary differential equatio in Chebyshev series*, Comp. J. 7 (1964), p. 76-85.

[6] J. Oliver, *Relative error propagation in the recursive solution of linear recurren relations*, Numer. Math. 9 (1967), p. 323-340.

[7] — *The numerical solution of linear recurrence relations*, ibidem 11 (1968), p. 349-36

[8] F. W. J. Olver, *Numerical solution of second-order linear difference equation* J. Res. NBS 71 B (1967), p. 111-129.

[9] S. Paszkowski, *Procedure Cheval*, in *The library of ALGOL programs and pr cedures of the Odra 1204 computer*, Elwro, Wrocław 1970 (in Polish).

INSTITUTE OF INFORMATICS
UNIVERSITY OF WROCŁAW
50-384 WROCŁAW

Received on 5. 3. 1975

ALGORYTM

S. LEWANOWICZ (Wrocław)

ROZWIĄZANIE NIELINIOWEGO RÓWNANIA RÓŻNICZKOWEGO ZWYCZAJNEGO PIERWSZEGO RZĘDU ZA POMOCĄ SZEREGÓW CZEBYSZEW

STRESZCZENIE

Procedura *nldiffeqNort* oblicza przybliżone wartości współczynników rozw nięcia w szereg Czebyszewa (szereg względem wielomianów Czebyszewa T_k I rodzaj rozwiązania równania różniczkowego (1) z warunkiem początkowym (2) lub z warui kiem brzegowym (3), tj. oblicza współczynniki a_k ($k = 0, 1, \ldots, n$; $n \leqslant N$), tak że zachodzi równość przybliżona (4), w której symbol $\sum'$ oznacza sumę z pierwszy składnikiem podzielonym przez 2.

Dane:

n — zmienna, której wartością jest liczba naturalna parzysta nie mniejsz od 4, próbna wartość granicy sumowania w (4);

a[*0*:*n*] — tablica próbnych wartości współczynników w (4) ($a[k] = a_k$ dl $k = 0, 1, \ldots, n$), określających wielomian, który spełnia odpowiedni warunek (2) lub (3);

nmax — liczba naturalna parzysta $N \geqslant 4$, największa dopuszczalna wartość *r*

s — liczba całkowita nieujemna, stopień wielomianu aproksymująceg funkcję $f_y(x, y(x))$ (zobacz rozdz. 2, wzór (13));

init	—	**true**, gdy rozwiązanie ma spełniać warunek początkowy (2), **false**, gdy ma spełniać warunek brzegowy (3);
ffy	—	nazwa procedury z czterema parametrami x, y (umieszczonymi w zbiorze wartości), f, fy typu **real**, podstawiającej dla danych x, y wartość $f(x, y)$ pod f i wartość pochodnej cząstkowej $f_y(x, y)$ pod fy;
al, be, ga	—	liczby α, β, γ występujące w (2) i (3), w wypadku (2) *al* jest dowolne;
eps	—	liczba dodatnia ε, charakteryzująca dopuszczalny błąd bezwzględny współczynników a_k (zobacz rozdz. 2, nierówności (39) i (40));
maxit	—	zmienna, której wartością jest największa dopuszczalna liczba iteracji w metodzie przybliżonego rozwiązywania układu liniowego spełnianego przez współczynniki a_k (zobacz rozdz. 2, relacja (17));
Chebval	—	nazwa funkcji rzeczywistej z trzema parametrami: n typu **integer**, a typu **array** i x typu **real**, przyjmującej dla danych n, a, x wartość równą wartości sumy występującej w (4) (zobacz np. [9]).

Uwaga. Tablica a musi mieć rozmiar co najmniej [*0*:*nmax*+*1*].

Wyniki:

n	—	górna granica sumowania po prawej stronie (4);
a[*0*:*n*]	—	tablica współczynników dających dostatecznie dobre przybliżenie rozwiązania równania (1);
maxit	—	liczba obliczonych kolejno przybliżeń rozwiązania równania (1) (zobacz rozdz. 2, równanie (5)).

Inne parametry:

ns	—	etykieta instrukcji (poza treścią procedury *nldiffeqNort*), do której następuje skok, gdy nie można otrzymać rozwiązania;
mark	—	zmienna, której wartość określa przyczynę skoku do instrukcji z etykietą *ns*: *mark* = 1, gdy *nmax* jest za małe do osiągnięcia żądanej dokładności, *mark* = 2, gdy dane *maxit* jest zbyt małe.

W procedurze *nldiffeqNort* zastosowano algorytm Nortona [5] z pewnymi modyfikacjami. Użytą metodę szczegółowo opisano w rozdz. 2. Wyniki obliczeń, wykonanych na maszynie cyfrowej Odra 1204, zamieszczone w rozdz. 3, wykazały poprawność algorytmu.