ZASTOSOWANIA MATEMATYKI
APPLICATIONES MATHEMATICAE
XII, 1 (1971)

ALGORITHM 10

S. PASZKOWSKI (Wroclaw)

DETERMINATION OF THE BEST POLYNOMIAL IN THE SENSE OF UNIFORM APPROXIMATION BY THE SECOND ALGORITHM OF REMEZ

1. Procedure declaration

procedure *Remez2*(n, p, x, f, *maxr*, a, *enf*);
value n, p, *maxr*;
integer n, p;
real *enf*, *maxr*;
array x, f, a;
comment The procedure *Remez2* calculates

(i) the coefficients of the n-th best polynomial (in the sense of uniform approximation) for a given function $f(x)$ on the finite set

$$X = \{x_0, x_1, \ldots, x_p\},$$

i.e. the coefficients of that polynomial $P(x)$ of degree at most n for which the error

$$\max_{x \in X} |f(x) - P(x)| \tag{1}$$

is minimum,

(ii) the n-th error of the best approximation of $f(x)$ on the set X, i.e. the error (1) for the best polynomial $P(x)$.

Data:

n — the degree of the sought best polynomial,
p — the index of the last point of the set X,
$x[0:p]$ — the array of points of X arranged so that either $x_0 < x_1 < \ldots < x_p$ or $x_0 > x_1 > \ldots > x_p$,
$f[0:p]$ — the array of function $f(x)$ values on the set X,
maxr — the maximum allowed number of type real in the computer.

Results:

$a[0:n]$ — the array of coefficients of the n-th best polynomial $P(x)$ ($a[k]$ — coefficient of x^{n-k} for $k = 0, 1, \ldots, n$),
enf — the n-th error of the best approximation, i.e. the value of (1) for the n-th best polynomial $P(x)$.

Remarks:

(i) It is necessary that $p > n$.

(ii) A typical application of the procedure *Remez2* consists in an approximate determination of the n-th best polynomial (in the sense of uniform approximation) for a given function $f(x)$ in the given closed and finite interval $\langle b, c\rangle$, i.e. of such an polynomial $P(x)$ of degree at most n for which the error

$$\max_{b \leqslant x \leqslant c} |f(x) - P(x)|$$

is minimum. To do this in a sufficiently accurate manner one should choose a set X such which would cover in a sufficiently dense manner the whole interval, and, additionally, in such a manner that near the interval ends should be more densely covered (e.g. so that $p > n^2$ and $x_k \approx \frac{1}{2}(b+c) - \frac{1}{2}(c-b)\cos(\pi k/p)$ for $k = 0, 1, \ldots, p$);

```
begin
  integer i, i1, j, k, l, n1, n2;
  real bk, cenf, d, e, exj, pre, r, sk, xk;
  Boolean nl, z;
  integer array ind[0: n+4];
  array b, nx[0: n+1], s[-1: n+4];
  procedure refp;
    begin
      if exj > cenf
        then cenf: = exj;
      s[j]: = s[j]+exj;
      j: = j+1;
      s[j]: = exj;
      ind[j]: = i1;
      if j = n+4
        then compr
    end refp;
  procedure compr;
    begin
      exj: = maxr;
      for k: = j-1 step -1 until 0 do
        if s[k] < exj
          then begin
                 exj: = s[k];
                 i1: = k
               end s[k] < exj, k;
      if i1 = 0
        then begin
```

```
                for k: = 1 step 1 until j do
                  begin
                    s[k−1]: = s[k];
                    ind[k−1]: = ind[k]
                  end k
              end i1 = 0
          else if i1 < j−1 ∨ nl
              then begin
                  s[i1−1]: = s[i1−1]−s[i1]+s[i1+1];
                  for k: = i1+2 step 1 until j do
                    begin
                      s[k−2]: = s[k];
                      ind[k−2]: = ind[k]
                    end k;
                  j: = j−1
                end i1 > 0 ∧ (i1 < j−1 ∨ nl);
    j: = j−1
  end compr;
enf: = maxr;
pre: = −1.0;
n1: = n+1;
n2: = n+2;
l: = p−n1;
ind[0]: = 0;
ind[n1]: = p;
r: = x[0];
d: = x[p];
e: = .5×(r+d);
d: = .5×(d−r);
r: = 3.14159/n1;
i: = 0;
for k: = 1 step 1 until n do
    begin
      xk: = e−d×cos(k×r);
  l1: i: = i+1;
    if d×(x[i]−xk) < .0
      then go to l1;
    if i < k+l
      then ind[k]: = i
      else begin
             for k: = k step 1 until n do ind[k]: = k+l;
             go to l2
           end i ⩾ k+l
```

```
      end k;
  l2: i: = ind[n1];
    nx[n1]: = x[i];
    b[n1]: = f[i];
    s[n1]: = e: = 1.0;
    l: = n1;
    for k: = n step -1 until 0 do
      begin
        i: = ind[k];
        xk: = nx[k]: = x[i];
        bk: = f[i];
        sk: = e: = -e;
        for j: = n1 step -1 until l do
          begin
            r: = nx[j]-xk;
            bk: = (b[j]-bk)/r;
            sk: = (s[j]-sk)/r
          end j;
        b[k]: = bk;
        s[k]: = sk;
        l: = k
      end k;
    e: = bk/sk;
    b[0]: = b[1]-e×s[1];
    for k: = 2 step 1 until n1 do
      begin
        xk: = nx[k];
        bk: = b[k]-e×s[k];
        for j: = k-1 step -1 until 1 do
          begin
            r: = b[j-1];
            b[j]: = bk-r×xk;
            bk: = r
          end j
      end k;
    j: = -1;
    e: = abs(e);
    nl: = z: = true;
    cenf: = exj: = sk: = s[-1]: = .0;
    for i: = 0 step 1 until p do
      begin
        xk: = x[i];
        d: = b[0];
```

```
        for k: = 1 step 1 until n do d: = d × xk + b[k];
        d: = f[i] − d;
        if d ≠ 0 ∧ z
          then begin
                 sk: = sign(d);
                 if i > 0
                   then sk : = −sk;
                 z: = false
               end d ≠ 0 ∧ z;
        bk: = abs(d);
        if sk × d ⩽ 0 ∧ i > 0
          then begin
                 sk: = −sk;
                 refp
               end sk × d ⩽ 0 ∧ i > 0
          else if bk < exj
                 then go to ei;
        exj: = bk;
        i1: = i;
ei: end i;
    nl: = false;
    refp;
    if cenf < enf
      then begin
             enf: = cenf;
             for k: = 0 step 1 until n do a[k]: = b[k]
           end cenf < enf;
    if j > n ∧ e > pre
      then begin
             pre: = e;
             if j > n2
               then compr;
             if j = n2 ∧ s[0] < s[j − 1]
               then for k: = 1 step 1 until j do ind[k − 1]: = ind[k];
             go to l2
           end j > n ∧ e > pre
end Remez2
```

2. Method used. The so-called second algorithm of Remez has been used in the procedure *Remez2* (see e.g. Meinardus [3]). It may be described most shortly as follows. First, one chooses a subset X_0 of the set X consisting of $n+2$ points. Given the subset

$$X_m = \{x_{m0}, x_{m1}, \ldots, x_{m,n+1}\} \qquad (m \geqslant 0)$$

consisting of $n+2$ points, one calculates the coefficients of the n-th best polynomial $P_m(x)$ for the function $f(x)$ on X_m and the error $|e_m|$ of this approximation. To do this one solves the system of linear equations

$$f(x_{mk})-P_m(x_{mk}) = (-1)^k e_m \quad (k = 0, 1, \ldots, n+1).$$

Next, investigating the difference $f(x)-P_m(x)$ on the set X, one chooses the subset X_{m+1}. This should be done so that, theoretically, after a finite number of steps one obtains the polynomial $P_m(x)$ which is already best on the whole set X (the rounding errors may change the behaviour of the sequence $\{P_m(x)\}$).

Now we shall describe how in the procedure *Remez2* the subsets X_0 as well as the next ones should be chosen, and when the calculations should be stopped. The way of choosing subsets in the procedure *Remez2* differs it mostly from all other known to the author procedures having the same application, e.g. from Boothroyd's procedure [2].

I. The choice of the subset X_0. Some of Bernstein's results (see [1], pp. 85 and 88) suggest that a good subset X_0 which would allow a fast calculation of the best polynomial is the subset consisting of $n+2$ points distributed in the interval $\langle x_0, x_p\rangle$ approximately in a similar way as the numbers

$$\cos \pi k/(n+1) \quad (k = 0, 1, \ldots, n+1)$$

in the interval $\langle -1, 1\rangle$. The fragment of procedure *Remez2* from the instruction *l*: = *p*−*n1* to the nearest instruction **for** *k*: = *1* ... calculates such a subset X_0. More precisely, one calculates here the array *ind*[*0*: *n*+*1*] of indices of those points of X which belong to X_0.

II. The choice of the subsets $X_1, X_2, \ldots$ Assume that we have found already the polynomial $P_m(x)$ for a given X_m. The subset X_{m+1} is defined in different ways for different variants of the second Remez algorithm. Generally, the choice of X_{m+1} is the better, the greater is the n-th error $|e_{m+1}|$ of the best approximation on that subset. To guarantee the most great error value we shall use the following theorem of Remez (see e.g. Meinardus [3]):

If, for a given closed set F, there exist a polynomial $Q(x)$ of degree at most n and points $\xi_0 < \xi_1 < \ldots < \xi_{n+1} \in F$ such that the values of $f(\xi_k)-Q(\xi_k)$ $(k = 0, 1, \ldots, n+1)$ are alternatively non-negative and non-positive then the n-th error the best approximation of the function $f(x)$ on F is at least equal to

$$\frac{1}{2}\min_{0\leqslant k\leqslant n}\left(|f(\xi_k)-Q(\xi_k)|+|f(\xi_{k+1})-Q(\xi_{k+1})|\right). \tag{2}$$

This theorem is stronger than the theorem of de la Vallée Poussin in which we have

$$\min_{0\leqslant k\leqslant n+1} |f(\xi_k)-Q(\xi_k)|$$

in place of (2) and which is usually used in the second algorithm of Remez.

According to Remez theorem, we will include into X_{m+1} such points of X which would assure that the assumptions of Remez theorem would be satisfied for $Q(x) = P_m(x)$ and, in addition, as to have (2) as great as possible. Now we shall describe the construction of X_{m+1} in more detail.

We calculate

$$\delta_k = f(x_k)-P_m(x_k) \quad (k = 0, 1, \ldots, p). \tag{3}$$

If $\delta_0 = \delta_1 = \ldots = \delta_p = 0$, then the numbers (3) will be temporarily replaced by the numbers

$$1, -1, \ldots, (-1)^p$$

(essentially, in the procedure *Remez2* there are not transformations of the sequence (3) but equivalent ones). If $\delta_0 = \delta_1 = \ldots = \delta_{r-1} = 0$, $\delta_r \neq 0$ $(r > 0)$, then $\delta_0, \delta_1, \ldots, \delta_{r-1}$ are replaced by

$$(-1)^r \operatorname{sgn}\delta_r, (-1)^{r-1}\operatorname{sgn}\delta_r, \ldots, -\operatorname{sgn}\delta_r.$$

If $\delta_q \neq 0$, $\delta_{q+1} = \ldots = \delta_{r-1} = 0$ and if either $\delta_r \neq 0$ or $r-1 = p$, then $\delta_{q+1}, \ldots, \delta_{r-1}$ are replaced by

$$-\operatorname{sgn}\delta_q, \ldots, (-1)^{r-1-q}\operatorname{sgn}\delta_q.$$

The modified sequence $\delta_0, \delta_1, \ldots, \delta_p$ is now divided into longest possible parts consisting of equally signed numbers. After that we return to the initial sequence (3). In each part of this sequence we find the number with maximum absolute value. Let those be denoted by

$$\varepsilon_0, \varepsilon_1, \ldots \tag{4}$$

and the appropriate indices of the sequence (3) by

$$i_0, i_1, \ldots \tag{5}$$

Example 1. Let $p = 10$ and let the sequence (3) be consisting of the numbers

$$0, -11, 32, 0, 16, 14, -21, 0, 0, -29, 0.$$

After the replacement of the zeros by ± 1 this sequence becomes

$$1, -11, 32, -1, 16, 14, -21, 1, -1, -29, 1,$$

thus the initial sequence is divided into the following parts (separated by semicolons):

$$0;\ -11;\ 32;\ 0;\ 16, 14;\ -21;\ 0;\ 0,\ -29;\ 0.$$

Therefore we have

$$\varepsilon_0 = 0, \varepsilon_1 = 11, \varepsilon_2 = 32, \varepsilon_3 = 0, \varepsilon_4 = 16, \varepsilon_5 = 21, \varepsilon_6 = 0, \tag{6}$$
$$\varepsilon_7 = 29, \varepsilon_8 = 0,$$

$$i_0 = 0, i_1 = 1, i_2 = 2, i_3 = 3, i_4 = 4, i_5 = 6, i_6 = 7, \tag{7}$$
$$i_7 = 9, i_8 = 10.$$

According to Remez theorem from (4) one should choose $n+2$ numbers

$$\varepsilon_{j_0}, \varepsilon_{j_1}, \ldots, \varepsilon_{j_{n+1}} \tag{8}$$

so that (i) the expression

$$\min_{0 \leqslant k \leqslant n} (\varepsilon_{j_k} + \varepsilon_{j_{k+1}}) \tag{9}$$

be maximum, (ii) the indices $k_0, k_1, \ldots, k_{n+1}$ be alternatively even and odd. The set X_{m+1} would consist of the points

$$x_{i_{j_0}}, x_{i_{j_1}}, \ldots, x_{i_{j_{n+1}}}. \tag{10}$$

Condition (ii) says that in the points (10) the difference $f(x) - P_m(x)$ is alternatively non-negative and non-positive. An exact fulfillment of (ii) needs storing of all numbers (4) and (5), thus at most $2p+2$ numbers. We shall describe now a method of investigating the numbers (4) which need not storing of so many quantities and which gives numbers (8) satisfying condition (ii) and yielding the expression (9) maximum or near maximum. First we shall describe the method for paper-and-pencil calculations, and then for computer calculations.

The numbers (4) and (5) may be determined during the construction of (3) and its division into intervals. Suppose we have already found the modules

$$\varepsilon_0, \varepsilon_1, \ldots, \varepsilon_{n+4} \tag{11}$$

and assume we know that (3) is divided into at least $n+6$ intervals so that ε_{n+4} is not the last element of (4). We find now in (11) two neighbouring numbers $\varepsilon_h, \varepsilon_{h+1}$ such that their sum is minimum. If $h = 0$, then we delete ε_0 from (11), if $h > 0$, then we delete ε_h and ε_{h+1}. The indices of the remaining numbers of (11) are now changed into successive numbers $0, 1, \ldots$ Analogously, the sequence $i_0, i_1, \ldots, i_{n+4}$ may be shortened, to fix the dependence between the indices and the numbers ε_j. Next, we

calculate further differences (3) and determine further ε_j (the indices will now be lower than in (4)) up to the moment of obtaining the sequence (11) anew or to the moment of having investigated the whole sequence (3).

Example 2 (continuation of example 1). If $n = 1$, then we find the first six numbers of (6):

$$0, 11, 32, 0, 16, 21.$$

The two first numbers have the minimum sum, the first number is thus deleted. We have now

$$\varepsilon_0 = 11, \varepsilon_1 = 32, \varepsilon_2 = 0, \varepsilon_3 = 16, \varepsilon_4 = 21,$$
$$i_0 = 1, i_1 = 2, i_2 = 3, i_3 = 4, i_4 = 6.$$

We calculate one number each of the sequences (6) and (7) and denote them as $\varepsilon_5 = 0$, $i_5 = 7$. Minimum sum holds now for $\varepsilon_2 + \varepsilon_3$ thus we delete ε_2 and ε_3 and obtain

$$\varepsilon_0 = 11, \varepsilon_1 = 32, \varepsilon_2 = 21, \varepsilon_3 = 0,$$
$$i_0 = 1, i_1 = 2, i_2 = 6, i_3 = 7.$$

We calculate the two last numbers of each of (6) and (7) and obtain

$$\varepsilon_4 = 29, \varepsilon_5 = 0,$$
$$i_4 = 9, \quad i_5 = 10.$$

Assume that we have investigated all differences (3) having shortened the sequences (4) and (5) when they had $n+5$ elements. Let j be the index of the last element of these sequences (in the considered example is $j = 5$ and not $j = 8$). Thus always holds $j \leqslant n+4$.

The inequality $j \leqslant n$ denotes that it is not possible to choose from (3) $n+2$ numbers being alternatively non-positive and non-negative. Theoretically, this inequality cannot appear because the difference $f(x) - P_m(x)$ should assume alternatively the values e_m and $-e_m$ on the set X_m. In practice, however, the inequality $j \leqslant n$ may appear and this indicates that great rouding errors which deform the sequence (3) have occurred. Is that so, further calculations with the second algorithm of Remez have no value, and they are finished.

If $j = n+1$, then the numbers

$$i_0, i_1, \ldots, i_{n+1} \tag{12}$$

which we have found form the indices of those points of X which belong to X_{m+1}.

If $j = n+2$, then we compare the sums $\varepsilon_0 + \varepsilon_1$ and $\varepsilon_{n+1} + \varepsilon_{n+2}$. If

$$\varepsilon_0 + \varepsilon_1 < \varepsilon_{n+1} + \varepsilon_{n+2}, \tag{13}$$

then we delete the numbers ε_0 and i_0, otherwise the numbers ε_{n+2} and i_{n+2} are deleted. In case of (13) a change of the numbers indices is necessary. We obtain now the indices (12) of the points chosen as belonging to X_{m+1}.

If $j = n+3$ or $j = n+4$, then the sequence

$$\varepsilon_0, \varepsilon_1, \ldots, \varepsilon_j \tag{14}$$

undergoes nearly the same procedure as before had the sequence (11). The difference lies in the fact that for the minimum sum of neighbouring numbers $\varepsilon_{j-1}+\varepsilon_j$ we delete only ε_j from the sequence (14). Of course, i_j is also deleted. The shortened sequence (14) is processed according to the new value of j (equal to $n+3$, $n+2$ or $n+1$) up to the moment of obtaining both $j = n+1$ and the final system (12) of indices of X_{m+1}.

Example 3. In example 2 we have obtained the numbers

$$11, 32, 21, 0, 29, 0;$$

thus $j = 5 = n+4$. Minimum sum is $21+0$, therefore (4) and (5) are shortened to the form

$$\varepsilon_0 = 11, \varepsilon_1 = 32, \varepsilon_2 = 29, \varepsilon_3 = 0,$$
$$i_0 = 1, i_1 = 2, i_2 = 9, i_3 = 10.$$

Now $j = 3 = n+2$. Since $11+32 > 29+0$ we delete additionally ε_3 and i_3. Finally, we have obtained the subset

$$X^1_{m+1} = \{x_1, x_2, x_9\}.$$

The difference $f(x)-P_m(x)$ takes on X_{m+1} the alternatively signed values $-11, 32, -29$. Expression (9) is here equal to 43. It is easy to verify that no other subset gives a greater value of that expression (retaining alternative signs).

It is possible that the described method does not lead to the maximum value of (9).

Example 4. If $n = 1$ and the whole sequence (4) is composed of the numbers

$$10, 27, 4, 29, 8, 17, 7, 33,$$

then after calculating the first six numbers, the numbers 8 and 17 are deleted; after calculation of the last two numbers, are deleted, first, 27 and 4, then 10. We obtain

$$29, 7, 33.$$

A better result (9) is possible choosing either 29, 8, 33 or 10, 29, 8.

The situation encountered in Example 4 has a small probability of occurrence. The procedure *Remez2* has been verified on many examples (see § 3) and of 57 investigated constructions of X_{m+1} only one was not optimum (this, however, did not spoil the method of Remez). Therefore it seems not advisable to improve the probability of optimum choice of X_{m+1}, e.g. by investigating the sequence $\varepsilon_0, \varepsilon_1, \ldots, \varepsilon_{n+5}$ instead of (11).

The choice of the subset X_{m+1} has been described for paper-and-pencil calculations. For computer calculations another (equivalent) procedure will be accepted, a procedure in which the sum $\varepsilon_h + \varepsilon_{h+1}$ will not be calculated many times. The numbers (4) are not kept in memory. In the process of their formation the sums

$$s_0 = \varepsilon_0 + \varepsilon_1, \quad s_1 = \varepsilon_1 + \varepsilon_2, \ldots \tag{14}$$

are calculated. If, at some moment of calculations, the number found last is ε_{j-1}, then the sequence (14) is terminated by numbers

$$s_{j-2} = \varepsilon_{j-2} + \varepsilon_{j-1}, \quad s_{j-1} = \varepsilon_{j-1}.$$

The elongation of sequence (14) after having calculated ε_j consists in enlargement of s_{j-1} by ε_j and in putting $s_j = \varepsilon_j$. During the comparison of numbers $s_0, s_1, \ldots, s_{j-1}$ we choose the minimum one. Let is be s_h. Deleting of ε_0 ($h = 0$) or of ε_h and ε_{h+1} ($h > 0$) from (4) corresponds to simple transformations of (14). In the first case the numbers $s_1, s_2, \ldots$ are denoted by $s_0, s_1, \ldots$ (i.e. they are translocated). In the second case the numbers $s_0, s_1, \ldots, s_{h-2}$ do not change. The number s_{h-1} is replaced by the expression

$$s_{h-1} - s_h + s_{h+1} = (\varepsilon_{h-1} + \varepsilon_h) - (\varepsilon_h + \varepsilon_{h+1}) + (\varepsilon_{h+1} + \varepsilon_{h+2}) = \varepsilon_{h-1} + \varepsilon_{h+2}$$

and the numbers $s_{h+2}, s_{h+3}, \ldots$ are denoted by $s_h, s_{h+1}, \ldots$ If only the last of the numbers ε_h were to be deleted, the sequence (14) would be unchanged (j were diminished by 1). The sequence (5) is formed and transformed as described earlier.

III. Termination of calculations. As mentioned, the calculations are terminated if because of great rounding errors the construction of X_{m+1} is impossible. They may be terminated too if $|e_{m+1}| \leqslant |e_m|$ (see Boothroyd [2]) which links the errors of best approximations on X_m and X_{m+1}. This inequality holds, among others, if two successive sets X_m and X_{m+1} are identical, i.e. if the polynomial $P_m(x)$ being best on X_m is also best on the whole set X.

Since the first termination cause does not allow to find the best polynomial on X, procedure *Remez2* calculates

$$\max_{x \in X} |f(x) - P_m(x)| \tag{15}$$

for $m = 0, 1, \ldots$ and stores the coefficients of $P_m(x)$ such for which (15) is minimum (it is known that both in theory and in practice the sequence is not always ascending).

3. Certification. The procedure *Remez2* has been verified on the computer ODRA 1204 for the following functions:

$$f_1(x) = e^x, \tag{16}$$

the values of which had been calculated by the computer with absolute error 2^{-35};

$$f_2(x) = I_0^2(x) - I_1^2(x), \tag{17}$$

where the Bessel functions values had been taken from tables and were accurate only to 5 decimal places after point;

$$f_3(x) = \frac{K_0(x)}{I_0(x)}\left[\frac{I_1(x)}{I_0(x)} - \frac{I_0(x)}{I_1(x)} + \frac{K_0(x)}{K_1(x)} - \frac{K_1(x)}{K_0(x)} + \frac{4}{x}\right]^{-1} \tag{18}$$

($K_0(x)$, $K_1(x)$ — Bessel functions), the values of which were tabulated with 4 decimal places after point accurate;

$$f_4(x) = x^{-2} f_3(x). \tag{19}$$

In the case (16) for small n the errors of the values of the approximated function are very small in comparison with the maximum values of the difference $f(x) - P_m(x)$. This difference behaves regularly and has exactly $n+2$ local extrema. Therefore the set X_{m+1} is in the procedure *Remez2* identical with that occurring in the classical formulation of the second algorithm of Remez.

In the cases (17)-(19) even for small n the rounding errors of the values of the approximated function have a great influence on the difference $f(x) - P_m(x)$. This difference may behave very irregularly and may have very much local extrema (e.g. 33 for $n = 4$). In such a case the choice of X_{m+1}, according with the general requirements of the second algorithm of Remez, is not unique. The practical importance of a realization of this algorithm depends, among others, on the accepted criterion of choosing X_{m+1} (and, of course, on the choice of X_0 and on the carefulness of programming).

The procedure *Remez2* has been thoroughly investigated, printing out not only the final results but also the indices of the points chosen to $X_0, X_1, \ldots$ and also the values of $f(x) - P_m(x)$ on the whole set X. In all investigated cases the procedure produced a polynomial which was

indeed best on X. The following table gives a selection of the investigated examples.

Function	n	p	Iterations number
(17)	2	30	3
	3	30	5
	4	30	7
	4	105	8
	5	30	7
	5	100	6
	6	30	11
	6	200	8
	7	30	5
	8	30	6
(18)	5	50	4
	8	50	4
	8	60	4
(19)	5	50	4
	8	50	5

In all examples the last iteration produced the same polynomial as the preceding iteration, and this was the termination criterion.

The procedure *Remez2* has been compared with the procedure *chebfit* [2] to test, any others, which one is faster. The calculation times in seconds were as follows:

n	Function (16), $p = 50$		Function (17), $p = 200$	
	Remez2	*chebfit*	*Remez2*	*chebfit*
0	1	0	5	2
1	3	2	11	6
2	4	3	14	16
3	5	6	17	26
4	6	11	27	38
6	6	21	57	85
8	11	31	144	279
10	14	58	43	325
12	6	142		

In the procedure *chebfit* the subset X_0 consists of points

$$x_{[pk/(n+1)+.5]} \qquad (k = 0, 1, \ldots, n+1)$$

([a] denotes the integer part of a). Such a subset choice is usually worse than that chosen by the procedure *Remez2*; this follows from the theorems of Bernstein cited in § 2. Only for $n \leqslant 2$ both definitions are identical or

nearly identical. This is probably the reason why for $n \leqslant 2$ the procedure *Remez2* is slower than the procedure *chebfit*. For greater n, however, the advantage of using procedure *Remez2* is obvious.

4. Modifications. Instead of the usual uniform approximation with error (1) it is possible to consider approximations with relative error

$$\max_{x \in X} \left| \frac{f(x)-P(x)}{f(x)} \right| \tag{20}$$

(assuming $f(x) \neq 0$) and with weighted error

$$\max_{x \in X} \left| \frac{f(x)-P(x)}{w(x)} \right| \tag{21}$$

($w(x)$ — a given function such that $w(x) \neq 0$). This last case the most general one: for $w(x) \equiv 1$ one obtains from (21) the error (1) and for $w(x) \equiv f(x)$ the error (20).

In case of (21) the polynomial $P_m(x)$ which is best for $f(x)$ on the subset X_m satisfies the system of equations

$$\frac{f(x_{mk})-P_m(x_{mk})}{w(x_{mk})} = (-1)^k e_m \quad (k = 0, 1, \ldots, n+1),$$

i.e. the system

$$P_m(x_{mk}) = f(x_{mk}) - (-1)^k e_m w(x_{mk}) \quad (k = 0, 1, \ldots, n+1).$$

To solve this system, the auxiliary function $s(x)$ with values

$$s(x_{mk}) = (-1)^k w(x_{mk}) \quad (k = 0, 1, \ldots, n+1)$$

is introduced, the divided differences of the values of $f(x)$ and $s(x)$ on X_m are calculated, and number e_m is chosen in such a manner that the $(n+1)$-th divided difference of the function

$$f(x) - e_m s(x) \tag{22}$$

be zero:

$$e_m = \frac{f(x_{m0}, x_{m1}, \ldots, x_{m,n+1})}{s(x_{m0}, x_{m1}, \ldots, x_{m,n+1})}.$$

Next, lower order divided differences are calculated and the coefficients of $P_m(x)$ are calculated from Newton's interpolation formula.

Let *Remez2rel* (and *Remez2wt*, respectively) be the name of the procedure which is similar to *Remez2* but approximates with error (20) (or (21), respectively). The procedure differences in all cases are very small.

Each of them calculates in different manner the values of $s(x)$ and of $(f(x)-P_m(x))/w(x)$. The full list of differences is as follows:

```
    Remez2:          procedure Remez2(n, p, x, f, maxr, a, enf);
Remez2rel:           procedure Remez2rel(n, p, x, f, maxr, a, enf);
  Remez2wt:          procedure Remez2wt(n, p, x, f, w, maxr, a, enf);
Remez2, Remez2rel:   array x, f, a;
        Remez2wt:    array x, f, w, a;
    Remez2:          b[n1]: = f[i]; s[n1]: = e: = 1.0;
Remez2rel:           b[n1]: = s[n1]: = f[i]; e: = 1.0;
  Remez2wt:          b[n1]: = f[i]; s[n1]: = w[i]; e: = 1.0;
    Remez2:          sk: = e: = -e;
Remez2rel:           e: = -e; sk: = e × bk;
  Remez2wt:          e: = -e; sk: = e × w[i];
    Remez2:          d: = f[i]-d;
Remez2rel:           d: = 1.0-d/f[i];
  Remez2wt:          d: = (f[i]-d)/w[i];
    Remez2:          end Remez2
Remez2rel:           end Remez2rel
  Remez2wt:          end Remez2wt
```

In the procedure *Remez2wt* additional data are provided by the array $w[0:p]$ of values of $w(x)$ on X.

References

[1] S. N. Bernstein (С. Н. Бернштейн), *Экстремальные свойства полиномов и наилучшее приближение непрерывных функций одной вещественной переменной*, ч. 1, Ленинград 1937.

[2] J. Boothroyd, *Algorithm 318. Chebyshev curve-fit (revised)*, Comm. ACM 10 (1967), pp. 801, 803.

[3] G. Meinardus, *Approximation von Funktionen und ihre numerische Behandlung*, Berlin 1964.

COMPUTING CENTRE
UNIVERSITY OF WROCŁAW

Received on 10. 1. 1970

S. PASZKOWSKI (Wrocław) ALGORYTM 10

WYZNACZANIE WIELOMIANU OPTYMALNEGO ZA POMOCĄ DRUGIEGO ALGORYTMU REMEZA

STRESZCZENIE

Procedura *Remez2* oblicza

(i) współczynniki n-tego wielomianu optymalnego (w sensie aproksymacji jednostajnej) dla danej funkcji $f(x)$ na danym zbiorze skończonym

$$X \in \{x_0, x_1, \ldots, x_p\},$$

tj. tego wielomianu $P(x)$ co najwyżej n-tego stopnia, dla którego błąd

$$\max_{x \in X} |f(x) - P(x)| \tag{1}$$

jest najmniejszy;

(ii) n-ty błąd aproksymacji optymalnej funkcji $f(x)$ na zbiorze X, tj. błąd (1) dla wielomianu optymalnego $P(x)$.

Dane:

n — stopień szukanego wielomianu optymalnego,

p — wskaźnik ostatniego punktu zbioru X,

$x[0:p]$ — tablica punktów zbioru X uporządkowanych tak, że $x_0 < x_1 < \ldots < x_p$ albo $x_0 > x_1 > \ldots > x_p$,

$f[0:p]$ — tablica wartości funkcji $f(x)$ na zbiorze X,

maxr — największa dopuszczalna w maszynie cyfrowej liczba typu **real**.

Wyniki:

$a[0:n]$ — tablica współczynników n-tego wielomianu optymalnego $P(x)$ ($a[k]$ — współczynnik przy x^{n-k} dla $k = 0, 1, \ldots, n$),

enf — n-ty błąd aproksymacji optymalnej, tj. wyrażenie (1) dla n-tego wielomianu optymalnego $P(x)$.

Uwagi:

(i) Musi być $p > n$.

(ii) Typowym zastosowaniem procedury *Remez2* jest przybliżone wyznaczanie n-tego wielomianu optymalnego (w sensie aproksymacji jednostajnej) dla danej funkcji $f(x)$ w danym przedziale skończonym domkniętym $\langle b, c\rangle$, tj. takiego wielomianu $P(x)$ co najwyżej n-tego stopnia, dla którego błąd

$$\max_{b \leqslant x \leqslant c} |f(x) - P(x)|$$

jest najmniejszy. Aby ten wielomian wyznaczyć dostatecznie dokładnie, należy wybrać taki zbiór X, który pokrywa dostatecznie gęsto cały przedział $\langle b, c\rangle$, przy czym gęściej w pobliżu końców przedziału (np. tak, że $p > n^2$ i $x_k \approx \frac{1}{2}(b+c) - \frac{1}{2}(c-b)\cos(\pi k/p)$ dla $k = 0, 1, \ldots, p$).

W procedurze *Remez2* zastosowano tzw. drugi algorytm Remeza (zob. np. Meinardus [3]; jest przekład polski), z pewnymi istotnymi elementami, odróżniającymi tę procedurę od innych o tym samym zastosowaniu, np. od procedury Boothroyda [2]. Użytą metodę opisano szczegółowo w § 2. § 3 zawiera omówienie przykładów kontrolnych wykonanych na maszynie cyfrowej ODRA 1204, które potwierdziły wyższość procedury *Remez2* nad procedurą Boothroyda.