Aplikace matematiky

Evžen Kindler

Algorithms. KINDLER 2. Heuristical algorithm for advanced exponential analysis

Aplikace matematiky, Vol. 20 (1975), No. 1, 73–76

Persistent URL: http://dml.cz/dmlcz/103568

39. KINDLER 2

HEURISTICAL ALGORITHM FOR ADVANCED EXPONENTIAL ANALYSIS

PhDr. RNDr. Evžen Kindler, CSc.,
Matematicko-fysikální fakulta KU, Malostranské nám. 25, 118 00 Praha 1

Let us consider a finite sequence of pairs $\{\langle t_i, y_i\rangle\}_{i=0}^{m}$ where m is an integer greater than 0. We can approximate the values y_i by $z_i = \sum_{j=1}^{n} g_j e^{c_j t}$. The problem to determine the values of g_j and c_j and, if necessary, even n, is called *exponential analysis*; there is usually supposed that all c_j are negative. In [1] an algorithm called *KINDLER 1* has been presented which performs the so-called *simple exponential analysis*: there all values of g_j are supposed to be positive. This algorithm is called *heuristical*, because it determines parallelly the values of n, of all g_j and of all k_j. It performs thus a heuristical work, which cannot be formulated in classical mathematics.

In the present contribution another heuristical algorithm for exponential analysis is presented. It determines the mentioned numerical values under the assumption that one g_j is negative, the others are positive and it obeys the initial conditions for the approximating function, which read that its derivative for $t = 0$ must be equal to zero and that its value for $t = 0$ must be equal to y_0 (naturally, the input data must satisfy the condition that $t_0 = 0$). Such an exponential analysis is called *advanced*. The approximating function must be decreasing and positive for non-negative t and its second derivative must be negative for t from a certain non-empty interval $\langle 0, \tilde{t}\rangle$ and positive for $t > \tilde{t}$.

The present algorithm follows the mathematical theory and the game described in [2]. The properties of data, the functions of keys (buttons), the meaning of output procedures, metaconstants and formal parameters of the present ALGOL description, the development of programming techniques used for the present algorithm and its hardware parameters are exactly the same as those presented in [1] for the algorithm *KINDLER 1*. The argument from the same paper may be used to determine the place of the present algorithm in applied mathematics and cybernetics from the methodological point of view.

```
procedure KINDLER 2 (c, d, k, g, m, n, button, line, licence, newline, print,
      printspace, carriage return, text); value m;
real array c, d, k, g; real licence; integer m, n, line; boolean array button;
procedure newline, print, printspace, carriage return, text;
begin integer i, j, e, f, l; real a, b, p, q, r, s, t, u, w, x, y; array z[0 : m];

procedure h1;
begin if ¬ button [3] then go to L5;
      newline; text ('FROM'); print (f); text ('TO'); print (e);
L5:   if l = 0 then go to L27; r := s := t := u := w := l := 0;
      for i := f step 1 until e do
      begin y := z[i]; if y ≦ licence then go to L6; y := 1/ln(y); t := t + 1;
            s := s + c[i]; w := w + y; x := c[i] × y; r := r + x;
            u := u + c[i] × x;
L6:   end i;
L8:   x := r ↑ 2; if x = 0 then go to L7; x := x − u × w;
      if x = 0 then go to L7; p := (t × r − w × s)/x; q := exp((s − u × p)/r);
      if ¬ button [2] then go to L12; print (q); print (p);
L12:  if ¬ button [1] then go to L10; if ¬ button [9] then go to L14; newline;
      for i := 1 step 1 until line do text ('='); x := z[0]/line;
      for i := 0 step 1 until e do
      begin newline; text ('I'); printspace(abs(q × exp(p × c[i])/x)); text ('O');
            carriage return; printspace(abs(z[i]/x)); text ('X')
      end i; newline;
L14:  if ¬ button [8] then go to L10;
      for i := 0 step 1 until e do
      begin newline; print(q × exp(p × c[i])); y := z[i]; print(y); print(x − y)
      end i; newline; go to L10;
L27:  i := f − 1; y := z[i]; if y ≦ licence then go to L8;
      y := 1/ln(y); t := t − 1; s := s − c[i]; w := w − y; x := c[i] × y;
      r := r − x; u := u − c[i] × x; go to L8;
L7:   newline; text ('IT IS STRANGE, ONLY ZEROES');
      if f = 0 then go to L3; p := a; q := b; f := f − 1;
L10:  end h1;

procedure h2;
      begin x := 0; for j := 0 step 1 until n do x := x + g[j] × exp(k[j] × c[i])
end h2;
```

```
procedure h3;
begin newline; for i := 1 step 1 until line do text ('-'); s := d[0]/line;
      for i := 0 step 1 until m do
      begin newline; text (':'); h2; printspace(abs(x/s)); text ('0'); carriage return;
            printspace(abs(d[i]/s)); text ('+')
      end i
end h3;

procedure h4;
begin if ¬ button [5] then go to L13; newline;
      text ('NEW COMPONENT'); print(g[n]); print(k[n]);
L13: if button [9] then h3; if button [8] then h5
end h4;

procedure h5; for i := 0 step 1 until m do
      begin h2; newline; print(x); y := d[i]; print(y); print(x - y) end i;
comment: beginning of the statements;
      for i := 0 step 1 until m do z[i] := d[i]; n := f := 1; e := m;
L17: if z[f] ≦ 1-licence then go to L18; f := f + 1;
L19: go to if f < m then L17 else L11;
L18: if z[f] > licence then go to L4; f := f + 1; go to L19;
L4:  l := 1; h1;
L9:  f := f + 1; if f = e then go to L3; if p ≧ 0 then go to L17;
      a := p; b := q; h1; if p = a then go to L16; if p > a then go to L20;
      go to if button [6] then L21 else L9;
L3:  go to if p < 0 then L22 else L11;
L20: f := f + 1;
L24: a := p; b := q; h1; if p = a then go to L16;
      go to if p < a then L25 else L21;
L26: f := f + 1; if f < e then go to L24;
L16: f := f - 1; if button [4] then text ('BUTTON 4 OFF');
L22: k[n] := a; g[n] := b; h4; n := n + 1;
      if ¬ button [4] then go to L28; f := f + 1, if f < e then go to L29; text ('TAKE
      OFF BUTTON 4'); f := f - 1;
L28: if button [7] then f := m;
      for i := 0 step 1 until m do z[i] := z[i] - b × exp(a × c[i]); e := f;
L15: if e ≦ 0 then go to L23; if z[e] ≧ licence then go to L30; e := e - 1; go to L15;
L30: f := 0; if z[e - 1] ≧ licence then go to L17; e := e - 2; go to L15;
L21: if ¬ button [6] then go to L26; if p ≧ 0 then go to L16; a := p; b := q;
      newline; text ('I FIX FOR BUTTON 6'); go to L22;
L29: newline; text ('FOR BUTTON 4: I DO NOT FIX'); n := n - 1; go to L24;
```

```
L25: if ¬ button [4] then go to L22; newline;
     text ('FOR BUTTON 4: I DO NOT FIX'); go to L26;
L11: newline; text ('I CANNOT DO IT BETTER');
L23: newline; text ('RESULTS'); x := d[0]; y := 0;
     for i := 0 step 1 until n − 1 do
     begin newline; print(g[i]); print(k[i]); x := x − g[i]; y := y − g[i] × k[i]
     end i;
     newline; print(x); y := y/x; g[n] := x; k[n] := y; print(y);
     for j := 0 step 1 until m do z[j] := z[j] − x × exp(y × c[j])
end KINDLER 2;
```

In Table 1 we can see an example. The first column contains the values of t_i while the second one contains the values of y_i which enter the algorithm. The results are

$$z(t) = 0{\cdot}99416e^{-0.21378t} + 0{\cdot}22925e^{-9.86822t} - 0{\cdot}22340e^{-11.07775t}.$$

The third column of Table 1 presents the computed approximations of y_i and the fourth one contains the differences between them and the corresponding y_i.

Table 1

t_i	y_i	z_i	$y_i - z_i$	i
0·0000	1·000	1·00000000	0·00000000	0
0·0833	1·000	0·98858859	0·01141141	1
0·1667	0·995	0·96835150	0·02664850	2
0·2500	0·973	0·94785834	0·02514166	3
0·3330	0·932	0·92883091	0·00316909	4
0·4167	0·903	0·91096694	−0·00796694	5
0·5000	0·877	0·89414309	−0·01714309	6
1·0000	0·826	0·80281321	0·02318679	7
2·0000	0·691	0·64828470	0·04271530	8
5·0000	0·355	0·34137618	0·01362382	9
6·0000	0·291	0·27566970	0·01533030	10
7·0000	0·210	0·22261008	−0·01261008	11
8·0000	0·135	0·17976315	−0·04476315	12
9·0000	0·091	0·14516319	−0·05416319	13
12·0000	0·088	0·07644057	0·01155943	14
13·0000	0·084	0·06172764	0·02227236	15

References

[1] *E. Kindler:* A heuristical algorithm for simple exponential analysis. Aplikace matematiky 18, No 6, pp. 391—398, ACADEMIA, Prague 1973.

[2] *E. Kindler:* Generalization of pattern recognition method in experiment analysis. Kybernetika 8, No 5, pp. 201—211, ACADEMIA, Prague 1973.