5215

СООБЩЕНИЯ
ОБЪЕДИНЕННОГО
ИНСТИТУТА
ЯДЕРНЫХ
ИССЛЕДОВАНИЙ
Дубна

Р5 - 5215

Л. Александров

К АНАЛИЗУ ЭКСПОНЕНЦИАЛЬНЫХ ЗАВИСИМОСТЕЙ (С ПРИЛОЖЕНИЕМ ПРОГРАММЫ SHE)

ЛАБОРАТОРИЯ ВЫЧИСЛИТЕЛЬНОЙ ТЕХНИКИ
И АВТОМАТИЗАЦИИ

1970

Р5 - 5215

Л. Александров

К АНАЛИЗУ ЭКСПОНЕНЦИАЛЬНЫХ ЗАВИСИМОСТЕЙ (С ПРИЛОЖЕНИЕМ ПРОГРАММЫ SHE)

В физике и других естественных науках очень часто ставится задача: на основе данного ряда наблюдений найти те законы, суммарное воздействие которых вызывает наблюдаемое явление. Одна из первых задач, относящихся к указанному классу,- задача о нахождении скрытых экспонент. Ее аналитическая формулировка довольно проста. Предполагается, что функция $f(t)$ имеет вид

$$f(t) = p + \sum_{n=1}^{N} a_n \exp(-e_n t) \quad , \qquad 0 \le t < \infty .$$

Пусть известен ряд наблюдений $b_m = f(t_m)$, $m = 0, 1, \ldots, M$. Ищем следующие величины: "фон" p, амплитуды a_n, показатели e_n, а также число экспонент N.

I. Корректность поставленной задачи

Пусть $t_m = m$, $m = 0, 1, \ldots, M$ и $2N = M$. Поставленная задача приводит к решению трансцендентной системы

$$(1) \quad \begin{cases} p + a_1 + a_2 + \ldots + a_N = b_0 \\ p + a_1 \exp(-e_1) + a_2 \exp(-e_2) + \ldots + a_N \exp(-e_N) = b_1 \\ p + a_1 \exp(-2e_1) + a_2 \exp(-2e_2) + \ldots + a_N \exp(-2e_N) = b_2 \\ \vdots \\ p + a_1 \exp(-Me_1) + a_2 \exp(-Me_2) + \ldots + a_N \exp(-Me_N) = b_M \end{cases}$$

В векторной форме система (I) имеет вид

$$f(x) = y \quad , \qquad (\mathrm{I}')$$

где $x = (p, a_1, \ldots, a_N, e_1, \ldots, e_N) \in R^{2N+1}$, $y = (b_1, b_2, \ldots, b_M) \in R^M$,

$f(x) = (f_1(x), f_2(x), \ldots, f_M(x)) \in R^M$ $\qquad f_m(x) = p + \sum_{n=1}^{N} a_n \exp(-m e_n)$,

$m = 0, 1, \ldots, M$.

Обозначим через H объединение всех подпространств $p = 0$, $a_i = 0$, $e_r = e_q$, $i, r, q = 1, 2, \ldots, N$, $r \neq q$, а через X - множество всех $x \in R^{2N+1}$, для которых $\det J(x) \neq 0$. Здесь через $J(x)$ обозначена матрица Якоби функции $f(x) - y$ относительно x. Пусть еще $Y = f(X) \subset R^M$.

Нетрудно проверить, что $X \cap H = \emptyset$.

В каждой точке $(x, y) \in X \times Y$ к функции $f(x) - y$ можно применить теорему о неявной функции, откуда следует, что для каждого $y \in Y$ существует единственное решение уравнения (I') $x \in X$. Теорема о неявной функции дает вдобавок и то, что через (I') $x \in X$ непрерывно зависит от $y \in Y$. Все это показывает, что нахождение решения уравнения (I') для каждого $y \in Y$ (в случае, когда число N известно!) есть корректно поставленная задача.

Несмотря на сказанное, когда точка $x \in X$ находится близко к множеству H, очень трудно удается построить устойчивый и хорошо сходящийся алгоритм для решения уравнения (I'). На практике этот случай встречается часто. Например, когда в анализируемой кривой есть хотя бы две экспоненты с мало отличающимися показателями, или в случае, когда существуют амплитуды и показатели, близкие по величине к нулю.

Когда число экспонент N неизвестно, решение данной задачи можно искать следующим образом.

В системе (I) для N берется число, большее, чем предполагаемое число экспонент. Тогда в решении ожидаются экспоненты с равными показателями или экспоненты с нулевыми амплитудами или нулевыми показателями. Очевидно, что число различающихся экспонент с ненулевыми амплитудами и показателями и будет искомое число N.

На основе того, что в этом случае $x \in H$, заключаем, что теперь задача о решении системы (I) является некорректно поставленной /2/.

Приводимая далее программа базируется на алгоритме, построенном при предположении, что задача о нахождении скрытых экспонент некорректно поставлена вообще. При этом в каждом конкретном случае вместе с построением сглаженного квазирешения отыскивается и число N.

2. Алгоритм

Употребленный алгоритм итеративен и основан на одной специфической для экспоненциальных зависимостей линеаризации системы (I).

Вводятся следующие обозначения:

$e^s = \mathrm{colon}\,[e_1^s, e_2^s, \ldots, e_N^s]$,

$$A_K(e^s) = \begin{bmatrix} 1 & 1 & 1 & \ldots & 1 \\ 1 & \exp(-e_1^s) & \exp(-e_2^s) & \ldots & \exp(-e_N^s) \\ 1 & \exp(-2e_1^s) & \exp(-2e_2^s) & \ldots & \exp(-2e_N^s) \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & \exp(-Ke_1^s) & \exp(-Ke_2^s) & \ldots & \exp(-Ke_N^s) \end{bmatrix}$$

$a = \mathrm{colon}\,[p, a_1, a_2, \ldots, a_N]$, $b = \mathrm{colon}\,[b_1, b_2, \ldots, b_N]$, $d = \mathrm{colon}\,[b_2, b_3, \ldots, b_N]$,

$v(e^s) = \mathrm{colon}\,[p, a_1 \exp(-e_1^s), a_2 \exp(-e_2^s), \ldots, a_N \exp(-e_N^s)]$.

Приближенное решение уравнения (I) можно построить посредством следующего итерационного процесса

I: / i_0 / $e := e^0$ / $s = 0$/.

/ i / решается система $A_M(e^s)a = b$ относительно a.

/ ii / решается система $A_{M-1}(e^s)v(e^s) = d$ относительно $v(e^s)$.

/ iii / определяется вектор

$$e^{s+1} = colon\left[\frac{v_n(e^s)}{a_n}\right] n=0,1,2,\ldots, N.$$

$s = 1, 2, \ldots$

Здесь решение линейных задач / i / и / ii / ищем в виде[I,4]

$$a = (A_M' A_M + \alpha_1 E)^{-1} A_M' b \tag{2}$$

$$v = (A_{M-1}' A_{M-1} + \alpha_2 E)^{-1} A_{M-1}' d, \quad \alpha_1, \alpha_2 \in [0, \infty). \tag{3}$$

Случай $\alpha_1 = \alpha_2 = 0$ эквивалентен решению задач / i / и / ii / по методу наименьших квадратов. Параметры $\alpha_1 \neq 0$ и $\alpha_2 \neq 0$ дают посредством формул (2) и (3) возможность построить решение системы (I) даже когда системы / i / и / ii / – вырожденные или близки к вырожденным в условиях разброса наблюдаемых величин $\{b_m\}$ $m = 0, 1, 2, \ldots, M$ и неизвестного N.

Здесь употребляется вычислительный процесс I_α с формулами (2) и (3) при $\alpha \equiv \alpha_1 = \alpha_2 = const$ для каждого s.

Введенный параметр α определяется следующим критерием

P: α – такое положительное число, для которого выполнены условия: I/ неравенства $v_n(e^s) > 0$, $a_n > 0$, $n = 0, 1, 2, \ldots N$ выполняется для каждого s ; 2/ для фиксированного начального вектора итерационный процесс I_α сходится; 3/ найденное решение $x_{\alpha p}$ посредством I_α удовлетворяет равенству $\|f(x_{\alpha_p}) - y\| = \min_{\alpha \in \Omega} \|f(x_\alpha) - y\|$, /3/

где Ω – множество всех α, для которых выполнены условия 2/ и 3/ P, и $\| \;\|$ – подходящее расстояние в R^M.

Программа построена по следующему алгоритму L :

I/ задается подходящий начальный вектор e^0 ; 2/ задается предполагаемое множество $\bar{\Omega}$; 3/ для фиксированного $\alpha \in \bar{\Omega}$ находится посредством I_α решение x_α ; 4/ если не выполнено хотя бы одно из условий I/, 2/ P, процесс I_α прерывается и шаг 3/ повторяется для следующего $\alpha \in \bar{\Omega}$. В противном случае переходят к шагу 5/; 5/ вычисляется число $K(x_\alpha) = \|f(x_\alpha) - y\|$; 6/ шаги 3/, 4/ и 5/ повторяются для каждого $\alpha \in \bar{\Omega}$; 7/ выбирается то α_p и соответствующее $x_{\alpha p}$, для которого выполнено условие $K(x_{\alpha p}) = \min_{\alpha \in \bar{\Omega}} K(x_\alpha)$.

3. Примеры применения программы SHE

3.I. О точности построенного посредством программы решения можно судить на основе искусственных численных примеров, максимально приближенных к условиям естественной задачи.

Пусть, например

$$f(t) = 2 + 1.00\exp/-0.10t/ + 1.00\exp/-0.15t/.$$

Если возьмем для t сетку $\{1, 2, \ldots\ldots, 10\}$, для M число 3, для Δe_1, Δe_2 и Δe_3 допустимое отклонение в 0.00I и для $e_1^0 = e_2^0 = e_3^0 = 0.125$ и, если $b = \{3.7655, 3.555, 3.379, 3.218, 3.079, 2.955, 2.846, 2.751\}$, где элементы этого множества имеют альтернативный разброс не более 0.0I, то посредством программы SHE находится следующая кривая:

$$f(t) = 1.985+0.812\, exp/-0.903t/+0.596\, exp/-0.149t/ + +0.607\, exp/-0.144t/. \qquad (4)$$

Видно, что в этом случае фон распознается программой с относительной ошибкой 0,8% амплитуды a_1, a_2 и a_3 соответственно с ошибками в 19,8%, 19,2% и 21,4%, а показатели ℓ_1, ℓ_2 и ℓ_3 соответственно с 18,8%, 0,67% и 4,0%.

Из (4) заключается, что искомое число N =2.

3.2. Очень часто в практике экспоненциальных экспериментов недооценивается влияние экспонент с высшими показателями. Это объясняется не только отсутствием в лабораториях надежных программ для нахождения скрытых экспонент, но и постоянным стремлением некоторых исследователей упрощать наблюдаемые явления. На следующем примере показывается недопустимость игнорирования высших экспонент в некоторых случаях.

Пусть дан ряд наблюдений экспоненциального эксперимента $b = \{1100,922,780,674,573,521,470,396,384,375,317,273,263,227,243,197,167,200,194,168\}$. При предположениях N =1 и N =3, посредством программы SHE находятся следующие кривые

$$f(t) = 177.806+1110.21\, exp/-0.20267\, t/, \quad \alpha_p = 0.08 \qquad (5)$$

$$f(t) = 114.27+621.93\, exp/-0.12033\, t/+390.54\, exp/-0.2453t/+ +208.92\, exp/-0.50305\, t/. \quad \alpha_p = 0.01 \qquad (6)$$

В этом случае $\Delta\ell_1 = \Delta\ell_2 = \Delta\ell_3 = 0.00001$ и $\ell_1^{\circ} = 0.18$, $\ell_2^{\circ} = 0.28$, $\ell_3^{\circ} = 0.38$.

Из рис.1 видна ошибочная тенденция кривой 2 в конце интервала t. Представление об ошибке предположения, что в наблюдаемой кривой (1) есть только одна экспонента вместо 3, можно получить в этом случае из относительной ошибки отклонения асимптоты кривой (2) относительно кривой (3). Легко заметить, что эта ошибка имеет величину 55.3%.

3.3 Пример из нейтронной физики

При изучении термализационных процессов в кристаллических модераторах экспериментальным путем возникает задача о нахождении скрытых экспонент[4,5,6]. Показатель второй экспоненты для графита, бериллия и др. имеет важное значение (из него получается время термализации).

Посредством методов, употребляемых до сих пор ($I_{\alpha=0}$, градиентные методы и др.), довольно трудно найти хорошую экспериментальную кривую, где можно выделить и вторую экспоненту. В то же время вопрос о решении задачи при неизвестном числе N вообще не рассматривался.

Пусть дана кривая, измеренная посредством селектора "Агнешка" в опытной постановке Графштейна и др.[6], относящаяся к кубу из графита с ребром в 40 см (при комнатной температуре)

$b = \{1632,1270,1041,876,748,661,596,551,523,493,474,447,442,427,419,426,404,411,409,405\}$.

С помощью программы SHE доказывается, что в этой кривой есть две экспоненты. Найдена следующая аппроксимация[4]

$$f(t) = 406.15+848.73\, exp/-0.13201\, t/+595.05\, exp/-0.2112\, t/. \quad \alpha_p = 0.001$$

Задача решена для $\Delta\ell_1 = \Delta\ell_2 = \Delta\ell_3 = \Delta\ell_4 = 0.00002$,

$e_1^o = 0.I$, $e_2^o = 2$, $e_3^o = 3$, $e_4^o = 4$, $M = 4$.

В таблице I дается зависимость решения $x_{\alpha\rho}$ от параметров N, M, *alpha* и S (S - число необходимых итераций для достижения заданной точности).

В частности, из колонки 4 и 7 таблицы I делается вывод о возможности экономии экспериментальных данных при употреблении алгоритма L и программы SHE .

Литература

I. Тихонов А.Н. ДАН, №3, I63, (I965).

2. Тихонов А.Н. ДАН, №5, ~~I6I~~, (I965).

3. Philipps D.L., J.Assoc.Comput.Machinery, 1962,v.9, Nol.

4. Александров Л. Доклад 5RC/127r на У конференции по физике и технике ядерных реакторов. Варшава (I968).

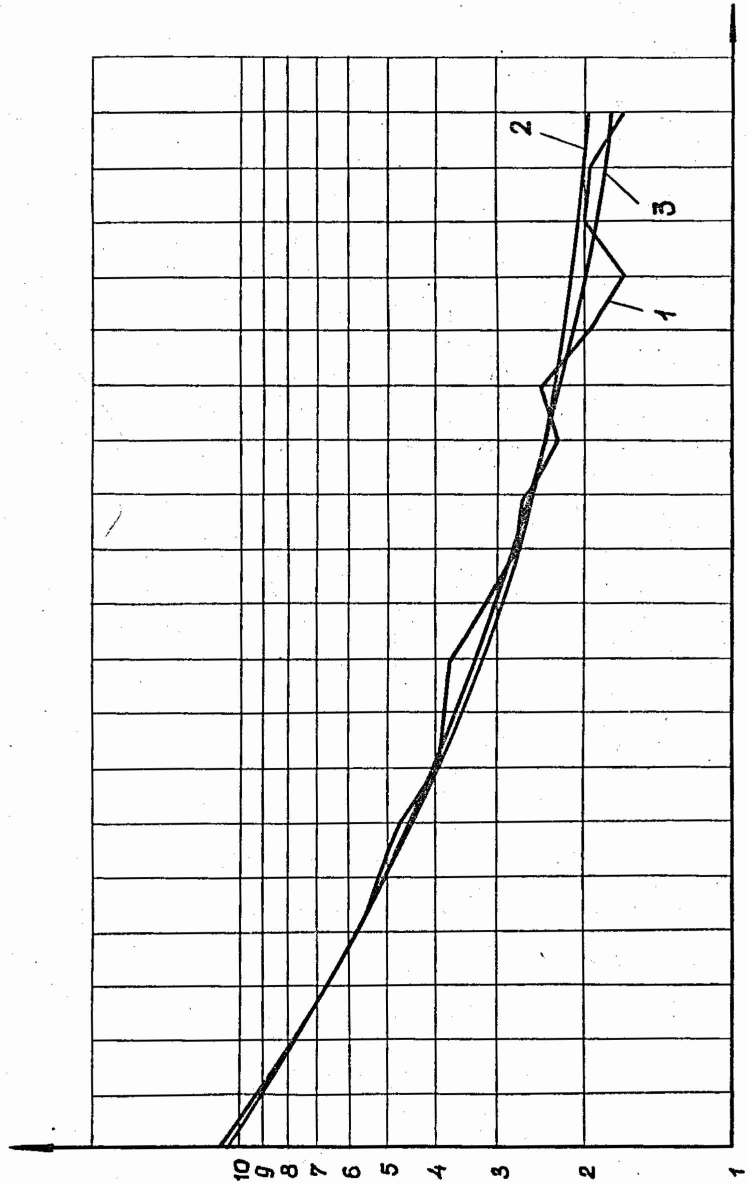
5. Alexandrov L. Compt.rend de l'Academi Bulgare, v.21, No 12, 1968.

6. Graffstein A. et al. Nukleonika 11, 1966.

Рукопись поступила в издательский отдел
2 июля I970 года.

Таблица I
Зависимость найденного квазирешения программой из параметров
N, M, *alpha*, S.

Параметры	I	2	3	4	5	6	7
N	I	I	2	2	2	4	4
M	40	40	40	40	40	20	20
alpha	0_ρ	0,000I	0	$0{,}00I_\rho$	0,004	0	0,042 ρ
S	5	I	3I	24	II	2	25
Р Е Ш Е Н И Е	ρ =4I4.687 a_1=I429.97 e_1=0.I6I6	ρ =4I4.688 a_1=I429.9 e_1=0.I6I0	итерация не сходящаяся	ρ =406.I55 a_1=848.725 a_2=595.054 e_1=0.I32009 e_2=0.2II2I	ρ =407.77 a_1=854.243 a_2=587.203 e_1=0.I3347 e_2=0.20927	$det(A'_M A_M + \alpha E)$, $det(A'_{M-1} A_{M-1} + \alpha E)$ на втором цикле одна из детерминант аннулировалась	ρ=406.03 a_1=283.85 a_2=283.82 a_3=283.82 a_4=595.25 e_1=0.I320I e_2=0.I320I e_3=0.I320I e_4=0.2II22



(I) - наблюдаемая функция,
(2) - аппроксимация одной экспонентой,
(3) - аппроксимация тремя экспонентами.

Рис. 1

4. Программа

```
begin
comment Gieralgol3 PROGRAM S.H.E.(SEARCH TO HIDDEN EXPONENTS);
 integer k,l,m,n,u,w,t,K,L,M,N,ALPHA,test;
 real p,q,r;
 end code;
real procedure det Ganss(n,m,A,exit);
comment Solving of linear eqvations system by
        the Gauss method(from Gier library);
procedure mult(x,y,z,I,J,K);
value I,J,K;
integer I,J,K;
array x,y,z;
     begin
          integer i,j,k;
         for i:=1 step 1 until Ido
         for k:=1 step 1 until Kdo
        begin
              z[i,k]:=0;
              for j:=1 step 1 until J do
             z[i,k]:=z[i,k]+x[i,j]xy[j,k]
           end
end mult procedure;
 procedure hemus(x,alpha,I);
valueI,alpha;
integer I,J;
real alpha;
 array x;
      begin
      integer i,j;
      for i:=1 step 1 until J do
      x[i,i]:=x[i,i]+alpha
      end hemus procedure;
procedure outtest(text,y,I1,I2);
 value I1,I2;
integer I1,I2;
string text;
array y;
        begin
        integer i1,i2;
        if test=1 then
         begin outcr;
            outtext(⪡<outtest(⪢); outtext(text); outtext(⪡<)⪢);
            outcr;
              if I2=0 then
            begin outcr;
              for i1:=1 step 1 until I1 do
             output(⪡dddd.ddd10-d⪢,y[I1],outsp(2))
        end;
```

```
        else
            begin outcr;
            for i1:=1 step 1 until I1 do
            for i2:=1 step 1 until I2 do
               output(⋠dddd.ddd₁₀-d⋡,y(I1),outsp(2))
        end
      end
end outtest procedure;
 BODY:
input(L);
comment L consecutive problems on finding hidden exponents
        can be solved by the program;
for l:=1 step 1 until L do
   begin
      input(M,N,test,w,ALPHA); outcr;
     comment test can have the values of 0 or I.When test=I the
             outtest procedure may be used.ALPHA is the number
             of tested values of alpha;

      begin
     array eo,e,epsilons,solve2[1:N],solve1[1:2xN],
        A[1:M,1:N],AT[1:N 1:M] AA[1:N 1:n+1]
         alphas[1:S,1:1],  b,bI,b2 [I:M,I:I]   ;
    input(eo b alphas epsilons):
comment the input vectors are: eo - the given initial guess,
        b - as noticed in the algorithm description, alphas -
        the tested values of ALPHAS, epsilons - the admissible
        errors of the solution vector x;

for m:=1 step 1 until M  do
 begin
    b1[n.1]:=b[n+1,1];
 end;
b1:   for n:=1 step 1 until N do e[n]:=eo[n];
      for k:=1 step 1 until ALPHA do
b2:   begin
        t:=t+1;
        if t>40 then goto bI0 else
b3:     begin
        for n:=1 step 1 until N-1 do
         A[1:n]:=exp(-wxe[n]);
        A[1,N]:=1;
       for m:=2 step 1 until M do
        for n:=1 step 1 until N do
       A[m,n]:=A[m-1,n]xA[1,n];
b4:     for n:=1 step 1 until N do
        for m:=1 step 1 until M do
        AT[n,m]:=A[m,n];
b15:    mult(AT,A,AA,N,M,N);
        outtest(⋠block5,AA⋡,N,N);
        alpha:=alphas[k];
         hemus(AA,alpha,N);
bi6:   mult(AA,b,b2,N,M,1);
         outtest(⋠blockbi,b2⋡,b2,N,0);
        for m:=1 step 1 until N do
       AA[m,N+1]:=b2[m,1];
bi7:    det Gauss(N,1,AA,b11);
        for n:=1 step 1 until N do
```



```
              begin
                 solve1[n]:=e[n];
                solve1[N+n]:=AA[n,N+1]
              end;
              outtest(⋠block71,solve1⋡,solve1,2xN,0);
bi15:         mult(AT,A,AA,N,M-1,N);
               hemus(AA,alpha,N);
bi16:         mult(AA,b1,b2,N,M-1,1);
                for m:=1 step 1 until N do
                AA[m,N+1]:=b2[m,1];
bi17:          det Gauss(N,1,AA,b11);
               outtest(⋠block bi17,AA⋡,AA,N,N);
b8:            for n:=1 step 1 until N do
               begin
               r:=AA[n,N+1]/solve1[N+1];
               if r≤0then
              begin outcr;
              outtext(⋠Logoritm⋡);
               goto b10;
              end;
                e[u]:=ln(r);
               solve2[n]:=e[n]
         end
       end block3;
b9:         for n:=1 step 1 until N do
             begin
             if abs(solve1[n]-solve2[n])>epsilon[n] then
             goto b2;
         end;
b10:          outcr;q:=0;
             outtext(⋠alpha=⋡);
             output(⋠-ddd.dddddd⋡,alpha);
              p:=0; outsp(5):
             outtext( ⋠Number of the iterations =⋡);
              output(⋠-ddd⋡,t); outcr;
             outtext(⋠Powers,amplitudes,background⋡outsp(5));
             for n:=1 step 1 until 2xN do
              begin
             q:=q+1;
              if q>8 then
             begin
                  q:=0; outcr;
              end;
             output(⋠-dddd.ddd₁₀-d⋡,solve1[n],outsp(2))
              end;
              outtext(⋠Points on the reconstruction curve⋡);
             outcr; u:=0;
            for m:=1 step 1 until M do
              begin
              r:=0; u:=u+1;
             for n:=1 step 1 until N-1 do

              r:=r+solve1[N+n]xexp(-mxsolve1[n]);
              r:=r+solve1[2xN];
              if u>12 then
```

```
          begin
          u:=0,outcr;
          end;
           p:=p+(r-b[n,1])↑2;
           output(≮dddddd.ddd≯,r,outsp(1))
      end; outcr;
        outtext(≮<Reconstruction criterium=≯);
      output(≮dddd.dddddd₁₀-d≯,p);
      outcr; outcr;
     t:=0;
end block2;
goto block12;
b11:         outcr;
             outtext(≮<Degenerated system≯);
             goto b10;
            end;
b12:         end body of the program;
end S.H.E.program;
```