ISSN 2071-2898 (Print)
ISSN 2071-2901 (Online)

Л.С. Туровцева

Решение обратных некорректно поставленных задач методом статистической регуляции. (Программа ОБР-23)

Рекомендуемая форма библиографической ссылки: Туровцева Л.С. Решение обратных некорректно поставленных задач методом статистической регуляции. (Программа ОБР-23) // Препринты ИПМ им. М.В.Келдыша. 1975. № 134. с.
https://library.keldysh.ru/preprint.asp?id=1975-134

ОРДЕНА ЛЕНИНА
ИНСТИТУТ ПРИКЛАДНОЙ МАТЕМАТИКИ
АКАДЕМИИ НАУК СССР

Л.С. ТУРОВЦЕВА

РЕШЕНИЕ ОБРАТНЫХ НЕКОРРЕКТНО ПОСТАВЛЕННЫХ ЗАДАЧ МЕТОДОМ СТАТИСТИЧЕСКОЙ РЕГУЛЯРИЗАЦИИ

(ПРОГРАММА ОБР-23)

Москва, 1975 г.

О Г Л А В Л Е Н И Е

Многие задачи самых разных разделов физики, связанные с обработкой и интерпретацией экспериментальных данных, принадлежат к классу обратных задач. В большинстве случаев это некорректно поставленные задачи, сводящиеся к решению уравнения вида

$$\hat{K}\vec{\varphi} = \vec{f} \qquad (1)$$

Без дополнительных априорных предположений относительно искомой функции $\varphi(x)$ такое уравнение, с правой частью функцией $f(y)$, являющейся результатом измерений, не имеет единственного решения.

Важной характеристикой всякой функции является степень ее гладкости, приближенным знанием которой мы нередко обладаем. Именно степень гладкости искомой функции привлекается в качестве дополнительной априорной информации в широко известных работах А.Н.Тихонова [1-4], D.L. Phillips [5], В.Ф.Турчина [6-9].

Довольно часто физические соображения позволяют ввести еще одно дополнительное условие – неотрицательность искомой функции. Многочисленные расчеты, проведенные автором, показали, что это существенно улучшает результат восстановления [10] (рис.1), а в случае сильно негладких функций неотрицательность иногда представляет единственную информацию, позволяющую получить решение [11] (рис. 2).

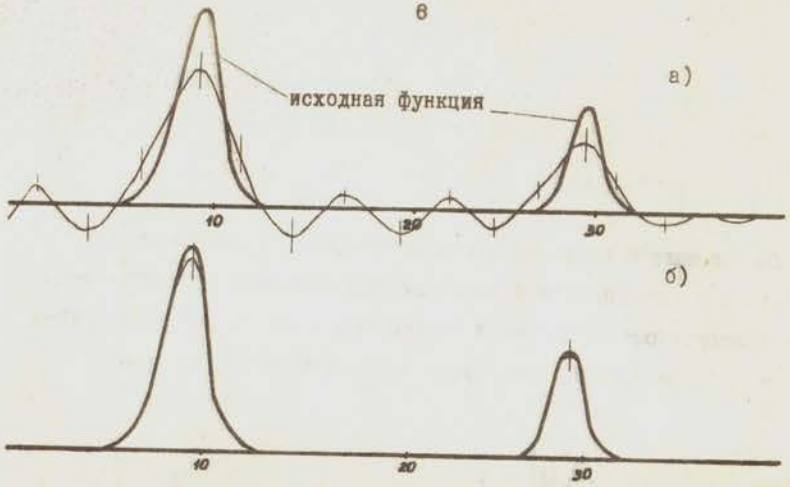


Рис. I. Восстановление функции φ в модельном расчёте. Разностное ядро. Полуширина функции разрешения $\Delta = 5$. Среднеквадратичная величина "экспериментальной" ошибки $S = 0{,}03 f$.

а) восстановление без учёта неотрицательности исходной функции; б) восстановление с учётом неотрицательности.

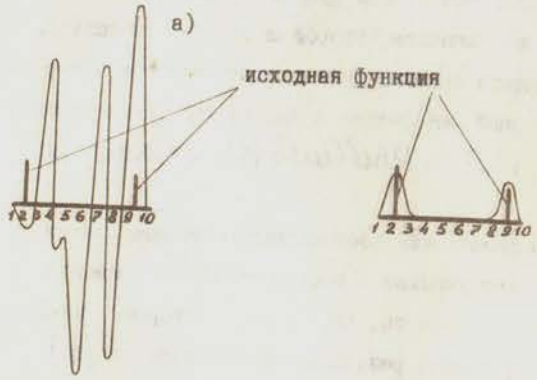


Рис. 2. Восстановление негладкой функции без учёта неотрицательности исходной функции (а) и с учётом неотрицательности исходной функции (б).

В параграфе I работы даётся описание алгоритма учёта дополнительной априорной информации о неотрицательности искомой функции в методе статистической регуляризации. Приведены программы и их описание.

В параграфе 2 приводится описание метода восстановления двухкомпонентной функции. Даны изменения к опубликованным ранее программам [I2] , позволяющие такое восстановление проводить.

В параграфе 3 описана программа WINDOW, осуществляющая выбор спектральных окон осреднения. Программа может быть использована в ряде задач физики атмосферы и в других областях физики при решении обратных задач.

В параграфе 4 приведены тексты программы ОБР-23 на языках АЛГОЛ-60(ГДР) и ФОРТРАН(ДУБНА).

Основные идеи в описанных ниже алгоритмах предложены доктором физико-математических наук В.Ф. Турчиным, под руководством которого были проведены многие расчёты, позволившие создать и довести программы до удобного для пользователей и приемлемого к опубликованию вида.

I. АЛГОРИТМ УЧЕТА ДОПОЛНИТЕЛЬНОЙ АПРИОРНОЙ ИНФОРМАЦИИ О НЕОТРИЦАТЕЛЬНОСТИ ИСКОМОЙ ФУНКЦИИ В МЕТОДЕ СТАТИСТИЧЕСКОЙ РЕГУЛЯРИЗАЦИИ.

В методе статистической регуляризации (см. [6-9]) совокупность всех приближенных решений уравнения (I) данной степени гладкости определяется вероятностно в виде статистического ансамбля

$$P(\vec{\varphi}\,|\,\vec{f},\alpha) = C_1 \cdot exp\left\{-\tfrac{1}{2}(\vec{\varphi},[\hat{B}+\alpha\hat{\Omega}]\vec{\varphi}) + (\vec{B},\vec{\varphi}) - C_2\right\} \qquad (2)$$

где $\hat{\Omega}$ – симметричная, неотрицательно определенная матрица, вводимая таким образом, что квадратичная форма $(\vec{\varphi},\hat{\Omega}\vec{\varphi})$ является нормой производной функции $\vec{\varphi}$ некоторого порядка. Математическое ожидание по этому ансамблю объявляется восстановленной функцией $\varphi(x)$, а дисперсия по тому же ансамблю определяет квадрат ошибки восстановления

$$\langle\vec{\varphi}\rangle_\alpha = \int \vec{\varphi}\, P(\vec{\varphi}\,|\,\vec{f},\alpha)\, d\vec{\varphi} = (\hat{B}+\alpha\hat{\Omega})^{-1}\vec{B} \qquad (3)$$

$$\sigma_i^2 = \int (\varphi_i - \langle\varphi\rangle_i)^2 d\vec{\varphi} = \left([\hat{B}+\alpha\hat{\Omega}]^{-1}\right)_{ii} \qquad (4)$$

Здесь

$$\hat{B} = \hat{K}^*\hat{W}\hat{K}; \quad \vec{B} = \hat{K}^*\hat{W}\vec{f}; \quad W_{ji} = \frac{1}{S_j^2}\delta_{ji}; \quad \delta_{ji} = \begin{cases} 1 & j=i \\ 0 & j\neq i \end{cases}$$

S_j – среднеквадратичная ошибка измерения компоненты правой части уравнения (I) – f_j

Статистические соображения дают алгоритм выбора параметра регуляризации α [8,9,12].

Дополнительная априорная информация о $\varphi(x)$ – неотрицательность – не может быть учтена аналогично информации о гладкости, поскольку она не имеет вероятностного характера. Поэтому её учет (см. [10]) производится введением в априорную информацию о гладкости фактора $P(\vec{\varphi})$, который обращается в нуль, если $\varphi_i < 0$. При этом, однако, интеграл по $\vec{\varphi}$, через который выражается апостериорное распределение $P(\alpha,\vec{f})$ [8,9], для параметра регуляризации α , уже не берется аналитически. Поэтому вначале α находится по максимуму распределения $P(\alpha/\vec{f})$ без учета неотрицательности функции $\varphi(x)$, а затем в качестве априорной вероятности берется априорное распределение $P_\alpha(\vec{\varphi})$, характеризующее ансамбль гладких функций, домноженное на $P(\vec{\varphi})$:

$$\widetilde{P}(\vec{\varphi}) = \widetilde{C}_\alpha \cdot P(\vec{\varphi}) \cdot exp\left\{-\tfrac{\alpha}{2}(\vec{\varphi},\hat{\Omega}\vec{\varphi})\right\} \qquad (5)$$

Множитель $P(\vec{\varphi})$, введенный в априорную вероятность, переходит в апостериорную вероятность

$$\widetilde{P}(\vec{\varphi}|\vec{f}) = \bar{C}\,p(\vec{\varphi})\,P(\vec{\varphi}|\vec{f}) \qquad (6)$$

и оказывается под знаком интегралов в формулах (3) и (4). Эти интегралы теперь также не берутся аналитически. По этой причине в качестве решения уравнения (I) берется не математическое ожидание вектора $\vec{\varphi}$ по апостериорному распределению $\widetilde{P}(\vec{\varphi}|\vec{f})$, а тот вектор $\vec{\varphi}$, при котором апостериорная плотность вероятности $\widetilde{P}(\vec{\varphi}|\vec{f})$ принимает максимальное значение. Новое решение имеет смысл, как наиболее вероятное при дополнительном условии неотрицательности вектора $\vec{\varphi}$.

Поиск решения в области $\varphi_i > 0$ сводится к минимизации квадратичной формы $Z(\vec{\varphi})$, стоящей в экспоненте распределения (2), но в ограниченной области $\varphi_i > 0$.

Если все компоненты вектора $\langle\vec{\varphi}\rangle$ неотрицательны для регуляризованного решения, полученного без учета требования неотрицательности, то решением задачи по апостериорному ансамблю $\widetilde{P}(\vec{\varphi}|\vec{f})$ и будет являться $\langle\vec{\varphi}\rangle$, так как эта точка одновременно является точкой максимума как функции $\widetilde{P}(\vec{\varphi}|\vec{f})$ так и функции $P(\vec{\varphi}|\vec{f})$. Таким образом задача оказывается решенной уже на первом этапе. Можно показать, что во всех остальных случаях точка, соответствующая решению, обязательно лежит на одной из граней гиперктанта, то есть в области, где хотя бы одно значение $\varphi_i = 0$. Геометрически эта точка является точкой касания некоторой линии уровня функционала $Z(\vec{\varphi})$, которая самая первая и, следовательно, с минимальным значением $Z(\vec{\varphi})$ касается некоторой грани.

Линиями уровня $Z(\vec{\varphi})$ являются многомерные эллипсоиды с центром в точке абсолютного минимума $Z(\vec{\varphi})$, дающей решение задачи без требования неотрицательности.

Таким образом задача сводится к поиску грани и точки на ней, где выполняются следующие условия:

$$\left.\frac{\partial Z(\vec{\varphi})}{\partial \varphi_i}\right|_{\Gamma_{\text{искомая}}} = \begin{cases} 0, & \varphi_i^{\Gamma} > 0 \\ \geqslant 0, & \varphi_i^{\Gamma} = 0 \end{cases} \qquad (7)$$

Поиск искомой грани осуществляется путем перебора граней, в котором движение от одной грани к другой сопровождается всегда уменьшением значения функционала $Z(\vec{\varphi})$. Процесс начинается с отыскания точки абсолютного минимума $Z(\vec{\varphi})$ в подпространстве, образованном координатными осями, соответствующими положительным компонентам в регуляризованном решении без учета неотрицательности. Это означает, что за начальную точку поиска мы принимаем точку, полученную из регуляризованного решения путем зайуления отрицательных компонент.

Выражение для функционала $Z(\vec{\varphi})$ в этом подпространстве имеет вид:

$$Z(\vec{\varphi}) = \tfrac{1}{2}\left(\vec{\varphi}, P_1(\hat{B} + \alpha\hat{\Omega})P_1\right)\vec{\varphi} - (P_1\vec{b}, \vec{\varphi}) + C_2 \qquad (8)$$

где P_1 – оператор проектирования, C_2 – константа, не зависящая от $\vec{\varphi}$. Решение $\langle\vec{\varphi}\rangle^S$ запишется по аналогии с выражением (3) в виде

$$\langle\vec{\varphi}\rangle^S = \left(P_1(\hat{B} + \alpha\hat{\Omega})P_1\right)^{-1} P_1\vec{b} \qquad (9)$$

Вообще говоря, могут быть два случая:

(1) $\langle\vec{\varphi}\rangle^S \in \Gamma_S$

(2) $\langle\vec{\varphi}\rangle^S \notin \Gamma_S$

индекс S означает размерность рассматриваемого пространства.

В случае (I) первое из условий (7) выполняется автоматически; если выполняется и второе условие, то $\langle\vec{\varphi}\rangle^S$ - решение поставленной задачи, невыполнение же второго условия означает, что существует грань большей размерности со значением функционала $Z(\vec{\varphi})$ меньшим, чем на Γ_S, и максимальная по абсолютной величине отрицательная производная $\left.\frac{\partial Z}{\partial \varphi_i}\right|_{\vec{\varphi}=\langle\vec{\varphi}\rangle^S}$ указывает, какая компонента $i = i_m$ должна быть включена в число координатных для перехода к новой грани Γ_{S+1} со значением функционала $Z(\vec{\varphi})$ заведомо меньшим. На новой грани Γ_{S+1} проводятся те же операции, что и на Γ_S.

В случае (2) минимум $Z(\vec{\varphi})$ нужно искать на одной из подграней грани Γ_S. Поиск такой подграни Γ_{S-1} производится следующим образом. Из числа координатных исключается та компонента $\vec{\varphi}$, которая раньше других становится отрицательной при движении вдоль прямой, соединяющей точку $\langle\vec{\varphi}\rangle^S$, и точку, лежащую на предыдущей грани. На полученной таким образом грани Γ_{S-1} повторяется весь описанный выше процесс. И так до тех пор, пока на некоторой грани для $\vec{\varphi}$ не выполнятся условия (7).

Процесс поиска граней сходится, так как число граней конечно, и движение происходит всегда в направлении уменьшения функционала.

В качестве ошибки восстановления мы пользуемся соотношением

(4), в соответствии с которым ошибка вычисляется без учета неотрицательности функции $\varphi(\vec{x})$, как оценкой сверху, так как введение обрезающей функции $\rho(\vec{\varphi})$ только уменьшает интеграл в (4).

Следует отметить, что значение параметра гладкости α, вычисленное без учета условия неотрицательности, оказывается несколько завышенным. В приведенной ниже программе дан вариант поправки в α_0, учитывающий неотрицательность $\vec{\varphi}$.

Подправленное α вычисляется по формуле

$$\alpha = \alpha_0 \left(\frac{n'}{n}\right)^3 \qquad (10)$$

полученной из следующих рассуждений:

Величина α_0 есть n/Ω_0, где Ω_0 - статистическая оценка интеграла

$$\int \left(\frac{\partial^2\varphi}{\partial x^2}\right)^2 dx \qquad (11)$$

в предположении, что функция $\varphi(x)$ равномерно распределена во всем интервале x, соответствующем точкам i от I до n. Но мы знаем, что в действительности она отлична от нуля только на множестве точек $i \in \Gamma^m$. Обозначим через n' число этих точек и представим себе, что функция $\varphi(x)$ сжата по оси x, чтобы уместиться в интервале длиной n'. Очевидно, интеграл (II) возрастает при этом в $(n/n')^3$ раз. Поэтому, чтобы правильно учесть характер функции $\varphi(x)$ в точках $i \in \Gamma^m$ вместо α_0 берем величину (I0).

Эффективность такой поправки проверена на математических экспериментах.

I.I СПИСАНИЕ ПРОЦЕДУРЫ POLO, ОСУЩЕСТВЛЯЮЩЕЙ УЧЕТ АПРИОРНОЙ ИНФОРМАЦИИ О НЕОТРИЦАТЕЛЬНОСТИ.

Процедура POLO вычисляет искомую функцию φ с учетом дополнительной априорной информации о её неотрицательности и гладкости с параметром гладкости α, полученным в процедуре STREG (см.стр. 61 и [12]).

Следующие идентификаторы процедуры POLO имеют следующий смысл:

1. fi_i, $i=1,2,\ldots,n$ — восстанавливаемая функция с учетом условия неотрицательности.
2. $fi0_i$, $i=1,2,\ldots,n$ — восстанавливаемая функция без учета условия неотрицательности (вычисляется в процедуре STREG под идентификатором fi).
3. ZZ — функционал $Z(\bar{\varphi})$ (см.стр. 11).
4. Zfi_i, $i=1,2,\ldots,n$ — производная функционала $Z(\bar{\varphi})$ вдоль φ_i.
5. z — число отличных от нуля компонент на данной грани, номер грани.

6. $a1$ — массив для сохранения номеров компонент, отличных от нуля на грани с номером z.
7. $a2$, r — вспомогательные массивы для выявления номера максимальной по абсолютной величине производной функционала (в процедуре expans) и для нахождения минимального коэффициента r (в процедуре narrow).
8. sch — число переходов с одной грани на другую (число расширений плюс число сужений).
9. $fi1$, $fi2$ — вспомогательные массивы для вычисления fi.
10. (dd_{ij}) — квадратная матрица, составленная из элементов матрицы d (см.стр.57, соответствующих отличным от нуля компонентам искомой функции на данной грани.
11. $$Zfi1 = \left.\frac{\partial Z(\bar{\varphi})}{\partial \varphi_i}\right|_{\bar{\varphi}=0}$$

Решение задачи с учетом условия неотрицательности искомой функции осуществляется путем обращения к процедурам START, EXPANS, Deffi, narrow, описанных внутри процедуры POLO, а также к процедуре invers (см. [13], стр.37, алгоритм 66а, "Обращение симметричной матрицы).

Процедуры START, EXPANS, Deffi, narrow, Defzz, имеют следующее содержание:

procedure START — вычисляет функцию fi, с которой начинается поиск решения с учетом условия неотрицательности.

Имеются два варианта процедуры START:

I вар. $fi_i \equiv 0$

2вар. $fi_i = \begin{cases} fi0, & fi0_i \geq 0 \\ 0, & fi0 < 0 \end{cases}$

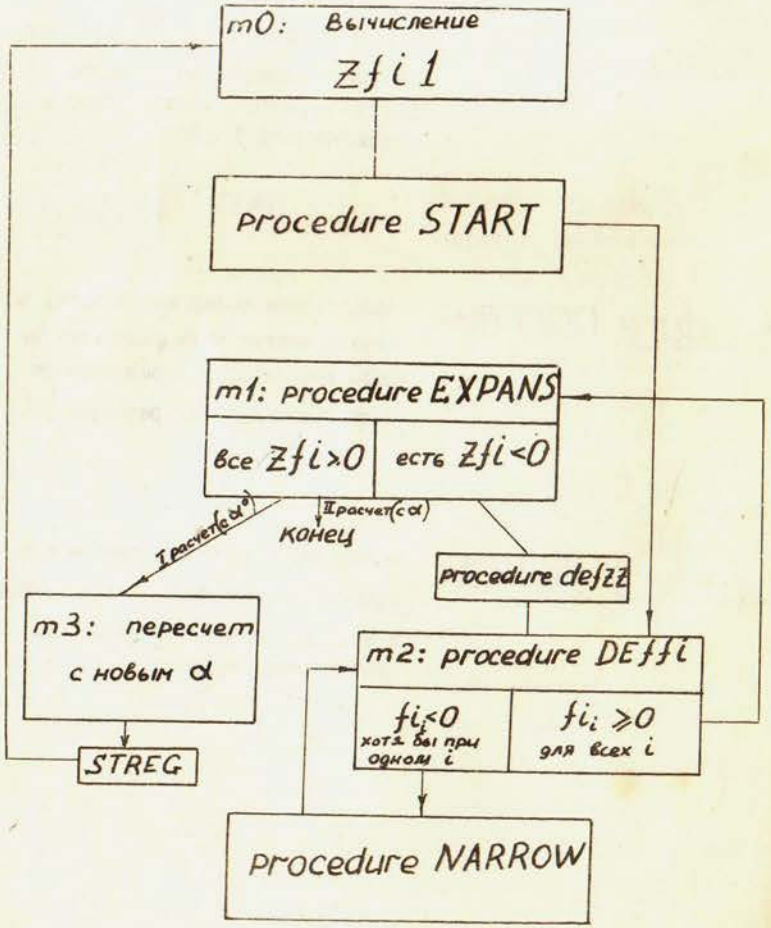
procedure EXPANS — Вдоль компонент функции fi, равных нулю, вычисляется производная функционала zfi; находится максимальная по абсолютной величине отрицательная производная. Компонента, соответствующая этой производной, включается в число ненулевых для образования новой грани большей размерности.

procedure Deffi — Определяется функция fi, соответствующая минимуму функционала в подпространстве, соответствующем рассматриваемой грани.

procedure narrow — Определяется номер компоненты, которая должна быть исключена из числа ненулевых для образования новой грани меньшей размерности.

procedure Defzz — Вычисляется значение функционала $z(\Phi)$ на данной грани (вычисляется и выдается на печать только для контрольной информации).

I.2 Блок-схема процедуры учета неотрицательности искомой функции *POLO*.



I.3 ТЕКСТ ПРОЦЕДУРЫ *POLO* НА ЯЗЫКЕ АЛГОЛ-60(ГДР)

```
procedure   polo(n, alfa, beta, fi);

    value   n;
    integer n;
    real    alfa, beta;
    array   fi;

    begin
        integer  g, z, sch, k1;
        integer array  a1[1:n];
        real  array  zfi1, fi1, fi0 [1:n];
```

```
procedure    start ;                    (вар. 1)

Begin
      for  i:=1  step  1  until  n  do
Begin
      fi0[i]:= fi[i];
      fi[i]:=0;
      fi1[i]:= fi[i];
end;
      z:=0;
end ;  comment  proc. start ;

procedure    start ;                    (вар. 2)

Begin      z:=1 ;
      for  i:=1  step  1  until  n  do
begin fi0[i]:=fi[i];  fi1[i]:=0;
      if  fi[i]≤0  then  go to  m;
      a1[z]:= i;   z:=z+1;  fi1[i]:=fi[i];
m:    fi[i]:=0;   end;  z:=z-1;  go to  m2;
end;  comment  proc. start;
```



```
procedure    defzz (fi);

    array    fi ;
Begin
    real    zz ;
    real  array    gg[1:n];


    zz :=0;
        for  i:=1  step  1  until  n  do
Begin
        gg[i]:=0;
        for  j:=1  step  1  until  n  do

gg[i]:=gg[i] + d[i,j]*(fi[j] - fi0[j]);

zz := zz + gg[i]*(fi[i] - fi0[i]);

    end ;

end;   comment. proc. defzz ;
```

```
    procedure  expans (z,a1, fi);
integer z; integer array a1; array fi;
begin  integer  a, p1, b1;  array  r,a2, zfi [1:n];


  a:=0;   p1:=1;     sch:=sch+1;
          for  i:=1  step  1  until  n  do
begin  if  fi[i]=0  then
begin  zfi[i]:=zfi1[i];  for  j:=1 step 1 until z do
   zfi[i]:=zfi[i]+d[i,a1[j]]*fi[a1[j]];
   if  zfi[i]<0 then begin a:=1; a2[p1]:=i;
                           r[p1]:=abs(zfi[i]);
   if  p1 ≥ 2  then  begin  if r[p1]<r[p1-1]  then
begin  r[p1]:=r[p1-1];  a2[p1]:=a2[p1-1];  end; end;
p1:=p1+1;  end  else  end  end;  if a=0 then go to m3;
      z:=z+1;    a1[z]:=a2[p1-1];
 if   z>1   then   for  i:=z  step  -1  until  2  do
begin    if  a1[i]-a1[i-1]<0  then
begin  b1:=a1[i];  a1[i]:=a1[i-1];  a1[i-1]:=b1;
                   end
end;


end;  comment  proc. expans;
```



```
procedure deffi (z, a1, fi);

      integer  z;  integer array a1,
      real array fi;
begin  real array dd[1:z, 1:z];


  for  i:=1  step  1  until  z  do
  for  j:=i  step  1  until  z  do
begin  dd[i,j]:=d[a1[i], a1[j]];
       if  i ≠ j  then  dd[j,i]:=dd[i,j];
end;     invers(z,dd);
       for  i:=1  step  1  until  z  do
begin
       fi[i]:=0;
       for  j:=1  step  1  until  z  do
   fi[i]:=fi[i] -dd[i,j]*zfi1[a1[j]];
end;


end;  comment  proc.  deffi;
```

```
        procedure   narrow;
begin
            integer  p1;
            real array  r, fi2, a2 [1:n];


    for i:=1 step 1 until n do
     fi2[i]:=0;
     sch:= sch+1;
    for i:=1 step 1 until z do
     fi2[a1[i]]:=fi[i];
     p1:=1;  for i:=1 step 1 until n do
if fi2[i]<0 then begin r[p1]:=fi1[i]/(fi1[i]-fi2[i]);
a2[p1]:=i; if p1≥2 then begin if r[p1]>r[p1-1] then
begin r[p1]:=r[p1-1]; a2[p1]:=a2[p1-1]; end; end;
                p1:=p1+1;  end;
        for i:=1 step 1 until n do
    fi2[i]:=fi1[i]+r[p1-1]*(fi2[i]-fi1[i]);
        for i:=1 step 1 until z-1 do
if a1[i]-a2[p1-1]≥0 then a1[i]:=a1[i+1];
        for i:=1 step 1 until n do fi1[i]:=fi2[i];
            z:=z-1;  go to m2;
end; comment proc. narrow;
```



```
        k1:=0;

m0:     for i:=1 step 1 until n do
        begin
                zfi1[i]:=0;
            for j:=1 step 1 until n do
         zfi1[i]:=zfi1[i]-d[i,j]*fi[j];
        end;
                start; go to m2;
m1:     expans(z, a1, fi);
        defzz(fi);
m2:     deffi(z, a1, fi);

            for i:=1 step 1 until z do
             if fi[i]<0 then narrow;

            for j:=1 step 1 until n do
              fi1[j]:=0;
            for j:=1 step 1 until z do
              fi1[a1[j]]:=fi[j];
            for j:=1 step 1 until n do
              fi[j]:=fi1[j];
               go to m1;
```

```
m3:     k1:=k1+1;
        if k1=1 then begin alfa:=(z/n)**3*alfa;

  'print' "(///5x, 5Halfa=, E18.6)", alfa;

streg (n, m, k, xx, f, s, inf, alfa, beta, fi, sigma);  *)

  'print' "(///60x, 2Hfi///(6F18.6))", fi;
  'print' "(///60x, 5Hsigma///(6F18.6))", sigma;

go to m0;

  end
  end; comment proc. polo;
```

+/ Процедура STREG опубликована в [12] и с необходимыми изменениями приведена в настоящем препринте на стр. 61

I.4 ТЕКСТ ПРОЦЕДУРЫ POLO НА ЯЗЫКЕ ФОРТРАН (ДУБНА)

```
      subroutine polo(n,d,dd,fi)
      common/a3/ alfa
      common/a14/ ich
      common/a15/ k1
      common/a22/ iz
      common/a25/ ia
      common/a29/ fi1(60)
      dimension d(n,n),dd(n,n),fi(n),zfi1(60),ia1(60)
      ich=0
      call reg(n,d)
      do 1 i=1,n
      zfi1(i)=0.
      do 1 j=1,n
1     zfi1(i)=zfi1(i)-d(i,j)*fi(j)
      call start(n,ia1,fi)
      go to 3
2     call expans(n,ia1,zfi1,d,fi)
      if(ia.EQ.0) go to 9
3     call deffi(n,iz,dd,d,ia1,zfi1,fi)
      do 4 i=1,iz
      if(fi(i).LT.0.) go to 8
4     continue
      do 5 i=1,n
5     fi1(i)=0.
      do 6 i=1,iz
      is=ia1(i)
6     fi1(is)=fi(i)
      do 7 i=1,n
7     fi(i)=fi1(i)
      go to 2
8     call narrow(n,ia1,fi)
      go to 3
9     k1=k1+1
      if(k1.GT.1) return
      as=iz
      alfa=(as/n)**3*alfa
      ich=2
      return
```

```
      subroutine  start (n, ia1, fi)
      common/a22/  iz
      common/a23/  fi0(60)
      common/a29/  fi1(60)
      dimension  ia1(n), fi(n)
      iz=1
      do 2 i=1, n
      fi0(i) = fi(i)
      fi1(i)=0.
      if (fi(i).GE.0.)  go to 1
      fi(i) =0.
      go to 2
1     ia1(iz) = i
      iz = iz + 1
2     fi1(i) = fi(i)
      iz = iz-1
      return
      end

      subroutine  deffi(n,iz,dd,d, ia1, zfi1, fi)
      dimension dd(iz,iz), d(n,n), ia1(n), zfi1(n), fi(n)
      do 1  i=1, iz
      do 1  j=i, iz
      ir =ia1(i)
      jd = ia1(j)
      dd(i,j) = d(ir, jd)
      if(i.NE. j)  dd(j, i) = dd(i, j)
1     continue
      call invers (iz, dd)
      do  2  i=1, iz
      fi(i) =0.
      do  2  j=1, iz
      jd = ia1 (j)
2     fi(i) = fi(i)  -  dd(i, j) * zfi1 (jd)
      return
      end
```



```
      subroutine  expans (n, ia1, zfi1, d, fi)

      common/a22/  iz
      common/a25/  ia
      dimension ia1(n), zfi1(n), d(n,n), fi(n),
     c r(60), ia2(60), zfi(60)

      ia =0
      ip1=1
      do  1  i=1, n
      if ( fi(i). NE. 0.)  go to  1
      zfi(i) = zfi1 (i)
      do  2  j=1, iz
      jd = ia1(j)
2     zfi(i) = zfi(i) + d(i, jd) * fi(jd)
      if (zfi(i). GE. 0.)  go to 1
      ia =1
      ia2(ip1) = i
      r(ip1) = abs(zfi(i))
      if( ip1. LT. 2)  go to 3
      if(r(ip1). GE. r(ip1-1))  go to 3
      r(ip1) = r(ip1-1)
      ia2(ip1) = ia2 (ip1-1)
3     ip1 = ip1 +1
1     continue
      iz = iz +1
      ia1 (iz) = ia2(ip1-1)
      if (iz. EQ. 1)  return
      do 4  ir = 2, iz
      i = iz + 2 - ir
      if(ia1(i) - ia1(i-1) . GE. 0)  return
      ib1 = ia1(i)
      ia1(i) = ia1 (i-1)
      ia1(i-1) = ib1
4     continue
      return
      end
```

```
      subroutine narrow (n, ia1, fi)

      common/a22/ iz

      common/a29/ fi1 (60)
      dimension r(60), fi2(60), ia2(60), ia1(n), fi(n)

      do 1 i=1, n
1     fi2(i)=0.
      do 2 i=1, iz
      is=ia1(i)
2     fi2(is)=fi(i)
      ip1=1
      do 4 i=1, n
      if (fi2(i).GE.0.) go to 4
      r(ip1)=fi1(i)/(fi1(i)-fi2(i))
      ia2(ip1)=i
      if(ip1.EQ.1) go to 3
      if(r(ip1).LT.r(ip1-1)) go to 3
      r(ip1)=r(ip1-1)
      ia2(ip1)=ia2(ip1-1)
3     ip1=ip1+1
4     continue
      do 5 i=1, n
5     fi2(i)=fi1(i)+r(ip1-1)*(fi2(i)-fi1(i))
      id=iz-1
      do 6 i=1, id
      if(ia1(i)-ia2(ip1-1).GE.0) ia1(i)=ia1(i+1)
6     continue
      do 7 i=1, n
7     fi1(i)=fi2(i)
      iz=iz-1
      return
      end
```


I.5. ТЕСТ.

МАТЕМАТИЧЕСКИЙ ЭКСПЕРИМЕНТ. НАЧАЛЬНЫЕ ДАННЫЕ И РЕЗУЛЬТАТ РАСЧЕТА. ЭВМ – БЭСМ-6. ЯЗЫК АЛГОЛ-60 (ГДР)

В качестве теста мы приводим результаты математического эксперимента с разностным ядром.

Элемент ядра K_{ji}, правая часть интегрального уравнения (I) f_j и *дисперсия* ошибки измерений S_j вычислялись с использованием следующих процедур *defik*, *defifs*:

```
procedure defik;

begin array as[1:8];
      read(as);

   for j:=1 step 1 until m do
begin real a, b, c, d;
      a:= as[1] + as[2]*j;
      b:= as[3] + as[4]*j;
      c:= as[5] + as[6]*j;
      d:= as[7] + as[8]*j;
   for i:=1 step 1 until n do
 begin if i LE a or i ≥ c then K[j,i]:=0;
if i>a and i≤b then K[j,i]:=(i-a)*d/(b-a);
if i>b and i<c then K[j,i]:=(c-i)*d/(c-b);
 end
end
end; comment proc. defik;
```

```
    procedure defifs;
begin
        integer p; real eps, ksi;
        array rb[0:100];
        procedure normal(x);
        real x; begin integer m1,m2,m3,m4,
        m5, m6; real a;
a:=rb[p]*100; m1:=entier(a); a:=(a-m1)*100;
m2:=entier(a); a:=(a-m2)*100; m3:=entier(a);
a:=rb[p+1]*100; m4:=entier(a); a:=(a-m4)*100;
m5:=entier(a); a:=(a-m5)*100; m6:=entier(a);
x:=(rb[m1]+rb[m2]+rb[m3]+rb[m4]+rb[m5]+rb[m6]
-3)*1.414213562; a:=rb[m1]; rb[m1]:=rb[m4]; rb[m4]
:=rb[m2]; rb[m2]:=rb[m5]; rb[m5]:=rb[m3]; rb[m3]:=
rb[m6]; rb[m6]:=a; p:=p+2; if p>99 then p:=p-100
end; comment proc. normal;
read(eps, fit*), p, rb); for j:=1 step 1 until m do
begin f[j]:=0;
                for i:=1 step 1 until n do
        f[j]:=f[j]+k[j,i]*fit[i];
        s[j]:=f[j]*eps; end;
for j:=1 step 1 until m do begin normal(ksi);
            f[j]:=f[j]+ksi*s[j]; end;
end; comment proc. defifs;
```

*) Функция fit[1:n] должна быть описана в том блоке программы OSP-23, который содержит процедуры defifs и grafik.

Начальные данные:

c = 1

n = 40

m = 40

alfa = -1

beta = 0.75

as
-5
1
0
1
5
1
0.2222222
0

inf
1.05
1.05
0.05

eps = 0.03

p = 96

i	xx	fit (исходная функция)
I	0	0.00I
2	I	0.00I
3	2	0.00I
4	3	0.00I
5	4	0.00I
6	5	0.00I
7	6	0.00I
8	7	0.05
9	8	0.5
I0	9	I.0
II	I0	0.5
I2	II	0.05
I3	I2	0.00I
I4	I3	0.00I
I5	I4	0.00I
I6	I5	0.00I
I7	I6	0.00I
I8	I7	0.00I
I9	I8	0.00I
20	I9	0.00I
2I	20	0.00I
22	2I	0.00I
23	22	0.00I
24	23	0.00I
25	24	0.00I
26	25	0.00I
27	26	0.00I
28	27	0.05
29	28	0.25
30	29	0.5
3I	30	0.25
32	3I	0.05
33	32	0.00I
34	33	0.00I

i	xx	fit
35	34	0.00I
36	35	0.00I
37	36	0.00I
38	37	0.00I
39	38	0.00I
40	39	0.00I

i	rB	i	rB	i	rB
I	0.I368962	25	0.II68408	49	0.I238822
2	0.I9I5I20	26	0.9806987	50	0.I5I296I
3	0.66I2082	27	0.2I83539	5I	0.0553I480
4	0.6649823	28	0.9528008	52	0.3856370
5	0.6599692	29	0.9045457	53	0.I884082
6	0.I5II299	30	0.6I85378	54	0.0485490
7	0.5359362	3I	0.8424513	55	0.6I65097
8	0.54I40II	32	0.488I286	56	0.I234070
9	0.0408359	33	0.484277I	57	0.675961I
I0	0.33679I3	34	0.9579539	58	0.3452257
II	0.7859362	35	0.5387I5I	59	0.4980343
I2	0.65I7226	36	0.4I67028	60	0.9327678
I3	0.9466955	37	0.46I8347	6I	0.3549005
I4	0.3767I85	38	0.09938570	62	0.I676I69
I5	0.992II95	39	0.030I9630	63	0.5270417
I6	0.3375953	40	0.4650034	64	0.I956233
I7	0.6355688	4I	0.40II040	65	0.6426714
I8	0.9888344	42	0.8222476	66	0.8994504
I9	0.5467738	43	0.260643I	67	0.5832420
20	0.2055576	44	0.I498834	68	0.5I96551
2I	0.6386748	45	0.6429870	69	0.I327005
22	0.70I9008	46	0.I0I4314	70	0.408I029
23	0.67I2987	47	0.003694400		
24	0.022528I0	48	0.4823608		

i	rB	i	rB
7I	0.3720402	90	0.442I599
72	0.2994I98	9I.	0.6969485
73	0.08463930	92.	0.6438974
74	0.7398610	93.	0.3440436
75	0.I93I664	94.	0.692242I
76	0.4330381	95.	0.5I02440
77	0.9686567	96.	0.9225887
78	0.05I2920	97.	0.4883II7
79	0.03II2530	98.	0.7930729
80	0.4048785	99.	0.5959049
8I	0.I247378	I00.	0.0I2495I0
82	0.3741578	I0I.	0.6281970
83	0.I9I7830		
84	0.5064878		
85	0.7673984		
86	0.06629850		
87	0.2978427		
88	0.203I844		
89	0.86I9739		

РЕЗУЛЬТАТЫ РАСЧЕТА.

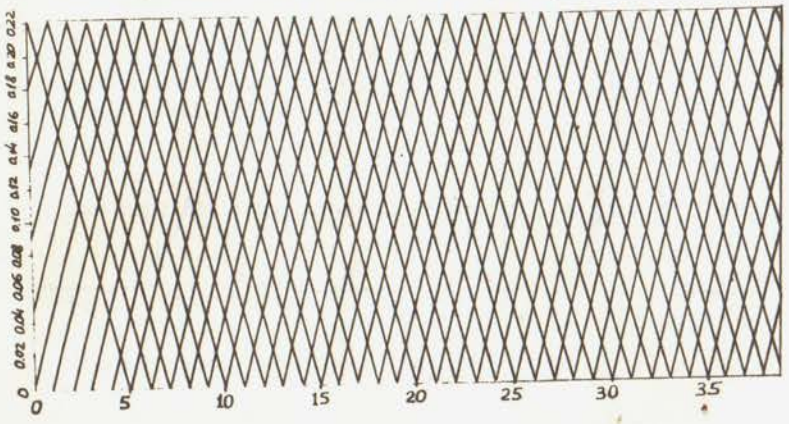
alfa = { 56.5 (без поправки)
13.79395 (с поправкой)

i	fi после STREG alfa =56.5	fi после POLO alfa =I3.79	sigma
I	-0.063I56	0.000000	0.0I8I78
2	0.039501	0.003712	0.02I833
3	0.I02908	0.000000	0.023757
4	-0.009070	0.000000	0.025459
5	-0.II8053	0.000000	0.034479
6	-0.073582	0.00II67	0.043886

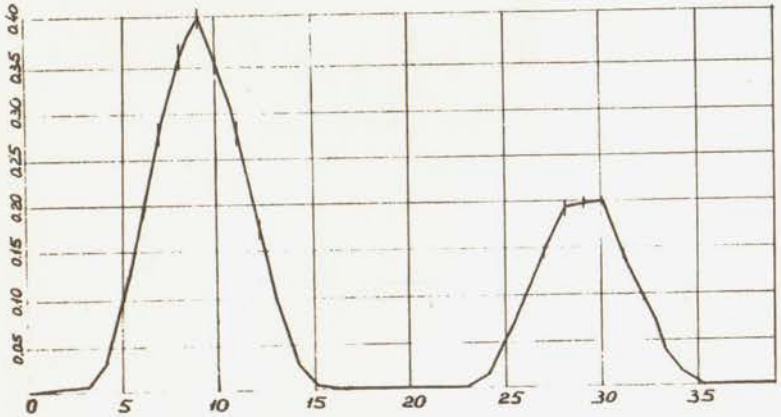
i	fi_i после STREG alfa =56.5	fi_i после POLO alfa = I3.79	sigma
7	0.075234	0.003689	0.043650
8	0.255498	0.049739	0.047465
9	0.503516	0.520355	0.047595
I0	0.673924	0.92071 4	0.047043
II	0.5II425	0.525582	0.046353
I2	0.203866	0.049925	0.045586
I3	0.052593	0.000000	0.044708
I4	-0.044232	0.000000	0.042647
I5	-0.I06864	0.005623	0.039412
I6	-0.082589	0.000000	0.034I6I
I7	0.II2085	0.000000	0.032424
I8	0.064906	0.000226	0.03I8I2
I9	-0.0265I4	0.0005II	0.030923
20	-0.099506	0.002572	0.030I3I
2I	-0.030549	0.000000	0.029978
22	0.073542	0.003092	0.030669
23	0.076625	0.000000	0.03II63
24	-0.009706	0.000000	0.03I904
25	-0.09I804	0.000000	0.035582
26	-0.028428	0.000000	0.039403
27	0.035902	0.007066	0.04I809
28	0.I35II6	0.046639	0.043293
29	0.269906	0.258538	0.043866
30	0.37437	0.493421	0.045034
3I	0.25289	0.233222	0.045369
32	0.095634	0.047009	0.0435I5
33	0.02I779	0.000000	0.042I34
34	-0.0I9706	0.000000	0.037II3
35	-0.036567	0.005730	0.032608
36	0.007I57	0.000000	0.023423
37	0.024280	0.000000	0.02I789
38	0.0II227	0.000634	0.02I080
39	-0.009I68	0.000693	0.0I8570
40	-0.007702	0.002098	0.0I4744

Время счёта I.02 мин.

РЕЗУЛЬТАТЫ РАСЧЕТА, ПРЕДСТАВЛЕННЫЕ ГРАФИЧЕСКИ. ПОЛУЧЕНЫ ПУТЕМ ОБРАЩЕНИЯ К ПРОЦЕДУРАМ graf, graf1, grafik.

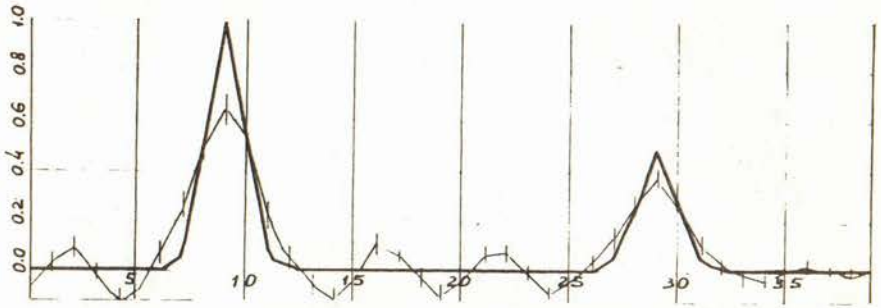


ЯДРО ИНТЕГРАЛЬНОГО УРАВНЕНИЯ

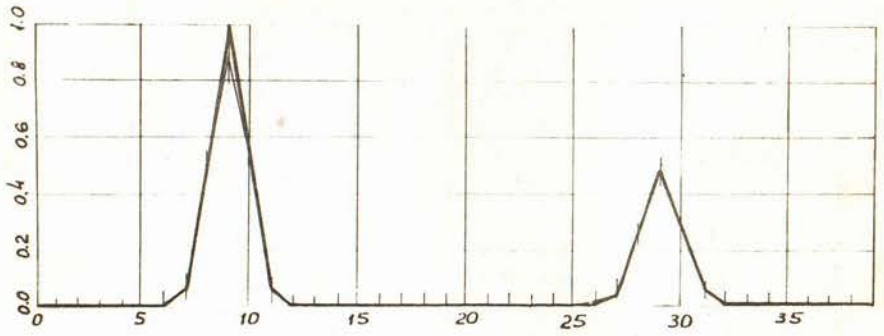


ИЗМЕРЯЕМАЯ ФУНКЦИЯ СВОБОДНЫЙ ЧЛЕН ИНТЕГРАЛЬНОГО УРАВНЕНИЯ

ВОССТАНОВЛЕННАЯ ФУНКЦИЯ БЕЗ УЧЕТА УСЛОВИЯ НЕОТРИЦАТЕЛЬНОСТИ



ВОССТАНОВЛЕННАЯ ФУНКЦИЯ С УЧЕТОМ УСЛОВИЯ НЕОТРИЦАТЕЛЬНОСТИ



2. РЕШЕНИЕ ЗАДАЧИ ДЛЯ СЛУЧАЯ ДВУХ КОМПОНЕНТ ИСКОМОЙ ФУНКЦИИ.

(пример: восстановление распределения частиц по размерам в рассеивающей среде, состоящей из смеси частиц с различными коэффициентами преломления [I4])

Для двух или более компонент левая часть линейного уравнения (I)

$$\hat{K}\,\vec{\varphi} = \vec{f}$$

представляет сумму векторов $\hat{K}^{(r)}\vec{\varphi}^{(r)}$, $r=1,\dots,R$ (R – число компонент, $\hat{K}^{(r)}$ – соответствующий распределению r линейный оператор), а правая часть $\vec{f}$ содержит информацию о всех компонентах. В конечно-разностном виде уравнение запишется

$$\sum_{r=1}^{R}\sum_{i=1}^{n} K_{ji}^{(r)}\varphi_i^{(r)} = f_j \qquad j=1,2,\dots,m \qquad (12)$$

а это эквивалентно уравнению (I), поскольку может быть представлено как

$$\sum_{i=1}^{n\times R} K_{ji}\varphi_i = f_j \qquad j=1,2,\dots,m \qquad (13)$$

где

$$K_{ji} = \begin{cases} K_{ji}^{(1)} & 1\le i\le n \\ K_{j\,i-n}^{(2)} & n+1\le i\le 2n \\ \text{- - - - - - - - - - - -} & \\ K_{j\,i-n(R-1)}^{(R)} & n(R-1)+1\le i\le R\times n \end{cases} \qquad (14)$$

$$\varphi_i = \begin{cases} \varphi_i^{(1)} & 1 \le i \le n \\ \varphi_{i-n}^{(2)} & n+1 \le i \le 2n \\ \text{----------} & \\ \varphi_{i-n(R-1)}^{(R)} & n(R-1)+1 \le i \le R \cdot n \end{cases} \qquad (15)$$

Следовательно, задача может быть решена обычным способом, если в качестве искомого взять вектор, составленный из векторов компонент распределения по правилу (I5), а элементы матрицы оператора $\hat{K}$ составлять из элементов матриц операторов $\hat{K}^{(r)}$ по правилу (I4). Порядок следования компонент r несущественен.

Метод статистической регуляризации предполагает априорную информацию о гладкости в виде нормы некоторой производной искомой функции. Так как составной вектор описывает не одну, а N функций, и его значения в конце одного подвектора никак не связано со значением в начале следующего подвектора, матрица регуляризующего функционала Ω должна быть модифицирована так, чтобы учесть разрывы в точках соединения подвекторов, а именно, регуляризующий функционал должен быть представлен в виде норм производных рассматриваемых подвекторов. Отметим, что в данной работе Ω помножается на один параметр регуляризации α, что предполагает один порядок нормы производной для всех компонент. В случае, если это не так, возможно противоречие между вносимой нами априорной информацией и реальной ситуацией. Например, информативность матрицы более гладкой компоненты может значительно превосходить информативность матрицы менее гладкой

компоненты может значительно превосходить информативность матрицы менее гладкой компоненты, в результате для подвектора с менее информативной матрицей решение получится сильно заглаженным. Вообще говоря, можно ввести несколько параметров регуляризации α и определять их аналогично одному; по-видимому, это улучшит восстановление составного вектора.

Были проведены модельные расчеты для смеси двух сортов частиц ($R = 2$) с показателями преломления I.33 и I.5 + ix0,05 [I4].

Восстановление распределения частиц по размерам в двухкомпонентной рассеивающей среде.
а) компонента с показателем преломления n_1=I.33
б) компонента с показателем преломления n_2=I.5 + i x 0,05

2.I. ДОПОЛНЕНИЯ И ИЗМЕНЕНИЯ, КОТОРЫЕ ДОЛЖНЫ БЫТЬ ВНЕСЕНЫ В ПРОЦЕДУРУ STREG И НАЧАЛЬНЫЕ ДАННЫЕ ДЛЯ РЕШЕНИЯ ЗАДАЧИ С ДВУМЯ КОМПОНЕНТАМИ ИСКОМОЙ ФУНКЦИИ.

I. Число дискретных значений n аргумента восстанавливаемой функции равно сумме соответствующих значений для каждой из компонент.

2. Элементы матрицы оператора $\hat{K}$ должны быть составлены из элементов матриц операторов $\hat{K}^{(1)}$ и $\hat{K}^{(2)}$ по правилу (I4).

3. Элементы матрицы регуляризующего функционала Ω – omo_{ij} ($j=1,2,\ldots,n$; $i=1,2,3,4,5$) вычисляются в процедурах OMOM и OM, приведенных ниже. Эти процедуры должны быть использованы вместо формул для вычисления aa_i, bb_i, cc_i, omo_{ij}, приведенных в [I2] на стр.49 и 54. и в настоящем препринте на стр. 63 и 68.

```
    procedure  omom;
begin  integer  a;
       a:=n;   n:=n/2;   j:=0;  aa[1]:=0; aa[2]:=0;
mm:    for  i:=j+3 step 1 until  j+n  do
       begin  aa[i]:= 1/(xx[i]-xx[i-1]);
              bb[i]:= -aa[i] -aa[i-1];
              cc[i]:= aa[i-1];     end;
aa[n+1]:=0; aa[n+2]:=0; bb[j+2]:=0; bb[j+n+1]:=0;
cc[j+n+1]:=0; cc[j+n+2]:=0;
    for  i:= j+1  step 1  until  j+n  do  begin
omo[1,i]:= aa[i]*cc[i];
omo[2,i]:= aa[i]* bb[i]  + bb[i+1]*cc[i+1];
omo[3,i]:= aa[i]** 2 + bb[i+1]**2 + cc[i+2]**2 ; end;
      j:=j+n;  if j=a/2  then  go to mm  else  n:=2*n;
end;  comment  proc. omom;
```



```
    procedure  om;

begin   integer   a, z, r, v;

        a:=n;   z:=0;   n:=n/2;   omega:=0;
mm:   for  i:=z+1  step 1  until  z+n  do
      begin   if  i=z+1  then  r:=3  else
              if  i=z+2  then  r:=2  else  r:=1;
              if  i=z+n-1  then  begin  v:=4;
                 omo[4,i]:=omo[2,i+1]  end  else

              if  i=z+n  then  v:=3  else  begin
          v:=5;   omo[5,i]:=omo[1,i+2];
                  omo[4,i]:=omo[2,i+1];  end;
    for  j:=r  step 1  until  v  do
omega:=omega + omo[j,i]*d[i+j-3,i];
omega:=omega + omo[3,i]*fi[i]**2;
  if  i≠z+n  and  i≠z+n-1  then
omega:=omega + 2*fi[i]*(omo[2,i+1]*fi[i+1] +
                        omo[1,i+2]*fi[i+2]);
  if  i=z+n-1  then  omega:=omega + 2*fi[i]*fi[i+1]
                        *omo[2,i+1]  end;
z:=z+n;   if  z=a/2  then  go to  mm  else  n:=2*n;
    omega:=(n-2)/alfa - omega;
end;  comment  proc. om;
```

3. ИСПОЛЬЗОВАНИЕ МЕТОДА СТАТИСТИЧЕСКОЙ РЕГУЛЯРИЗАЦИИ В ЗАДАЧЕ ВЫБОРА СПЕКТРАЛЬНЫХ ОКОН ОСРЕДНЕНИЯ [I5].

Во многих физических задачах, сводящихся к решению уравнения (I), функция f есть статистическая характеристика некоторой случайной функции, являющейся результатом измерений. Возникает проблема оценки этой статистической характеристики.

В ряде случаев для получения состоятельной оценки производят свертку периодограммы с различными "спектральными окнами", на которые накладываются некоторые условия, делающие эту оценку состоятельной. Ниже речь будет идти только о прямоугольной форме "спектрального окна", преимущества или недостатки некоторого мы здесь обсуждать не имеем возможности; что же касается одного конкретного случая его применения, то следует обратиться к [I5]. Исследование, проведенное в работе [I5], с использование метода статистической регуляризации позволило сформулировать условия ограничения для прямоугольного окна в следующем виде:

$$q_0 < q < \bar{q}$$

где q – допустимое число интервалов в "окне осреднения", $\bar{q}$ – число интервалов в "окне осреднения", обеспечивающее заданное смещение, то есть отклонение осредненной $\bar{f}(x)$ от значения $f(\bar{x})$ в средней точке "окна", q_0 – минимальное число интервалов осреднения, при котором оценка $f(x)$ имеет еще нормальное распределение (в методе статистической регуляризации функция $f(x)$ обязана быть случайной функцией с нормальным распределением; в упомянутой выше конкретной задаче $q_0 = 15$).

Результаты математических экспериментов, приведенные в [I5], показали, что варьирование величины q в пределах (I6) мало меняет решение задачи (I).

Здесь мы приводим программу, позволяющую получить величину $\bar{q}$. В программе производится выбор "окон осреднения", ширина которых не выводит смещение за пределы допустимых границ. Конечный результат расчета по программе WINDOW дает сетку значений аргумента измеряемой функции f, осредненные значения f_i в точках i сетки и среднеквадратичную ошибку S_i.

3.I. Описание процедуры WINDOW. Блок схема.

В число формальных параметров процедуры WINDOW входят следующие величины:

I.	omega$_i$	аргумент измеряемой функции f.
2.	f_i	измеряемая функция
3.	S_i	случайная ошибка осреднений по "окнам" функции f.
4.	a1, a2	соответственно первое и последнее значения аргумента измеряемой функции.
5.	ch	приблизительное число интервалов, на которые предполагается разбить весь интервал omega.

6. *dm* — интервал измерений по *omega*.

7. *eps* — относительная точность вычисления интеграла осреднения

8. *sm1* — величина максимально возможного смещения (задается)

9. *eps1* — допустимое отклонение от смещения

10. *m* — число "окон" осреднения

Смысл величин, участвующих в процедуре *WINDOW*:

sm — величина смещения, возникающего в процессе нахождения "окон" (в счете изменяется)

dom — величина "окна" осреднения (в процессе счета меняется).

Процедуры *error, simps, defif, SSM*, описанные в теле процедуры *WINDOW*, имеют следующее содержание

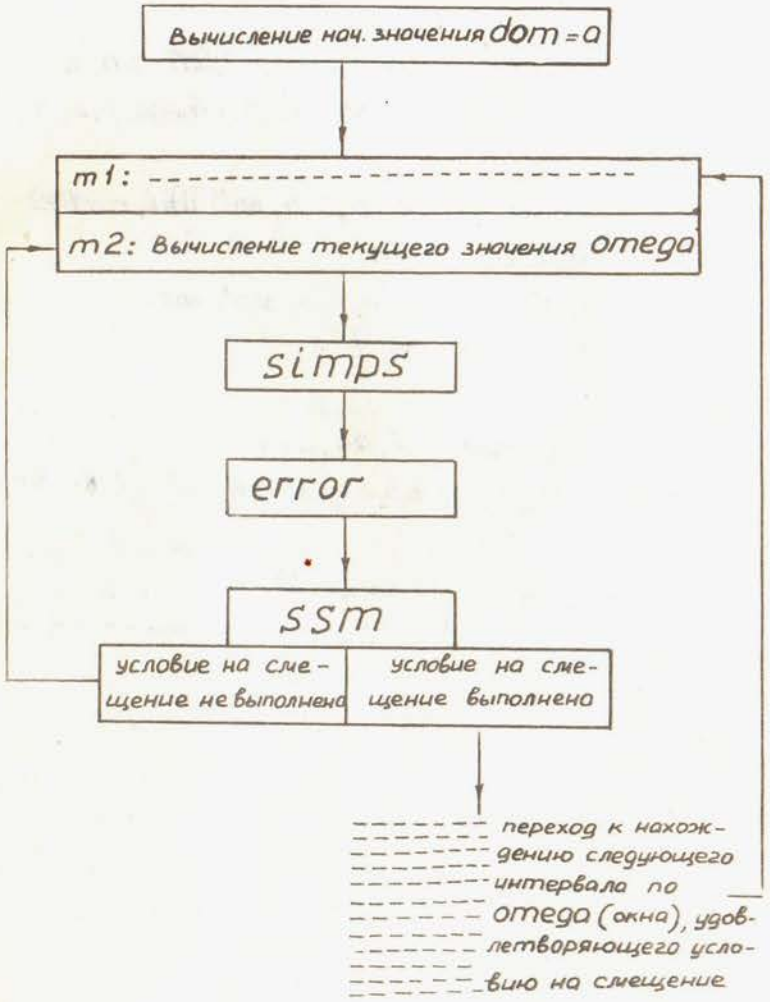
procedure error — вычисляется ошибка *S* осредненной функции *f*

procedure simps — вычисляется определенный интеграл по "прямоугольному окну" от измеряемой функции. *eps* – относительная точность вычисления интеграла.

procedure defif — в процедуре вычисляется или вводится измеряемая функция

procedure SSM — в процедуре вычисляется новое значение величины "окна"

3.2. Блок-схема процедуры *WINDOW*.



3.3 ТЕКСТ ПРОЦЕДУРЫ WINDOW НА ЯЗЫКЕ АЛГОЛ-60(ГДР)

```
procedure  window (a1, a2, ch, dm, eps,
                   sm1, eps1, omega, f, s, m);

    value a1, a2, ch, dm, eps, sm1, eps1;
    integer ch, m;
    real a1, a2, dm, eps, sm1, eps1;
    array omega, f, s;

begin  integer  i, ap, ak, ar;
       real  a, a3, a4, a5, a6, a7, f1, b1, sm, dom;

    procedure  error (a, b);
```

```
    procedure  simps (a, b, eps, f1);

        value  a, b, eps;
        real  a, b, f1, eps;

    begin  integer  mm, nn, kk;
           real  d, h, t, v, y, z, xx, f, b2;

        procedure  defif;
```

```
        h := 0.5*(b-a);  xx := a;  b2 := xx;  defif;
        y := f;  xx := b;  b2 := xx;  defif;
        y := y+f;  xx := 0.5*(a+b);  b2 := xx;  defif;
        t := f;  y := y+4*t;  kk := 0;  nn := 1;
aa:     nn := 2*nn;  kk := kk+1;  d := 0.5*h;  xx := a+d;
        b2 := xx;  defif;  v := f;  mm := 0;
bb:     mm := mm+1;  xx := xx+h;  b2 := xx;  defif;
        v := v+f;  if mm < (nn-1) then go to bb;
        z := y-2*t+4*v;
        if (abs(z-2*y) > abs(z)*eps and kk < 33)
        then begin h := d;  y := z;  t := v;  go to aa;  end;
        f1 := 1/3*d*z;  b2 := 0.5*(a+b);  defif;  b1 := f;
    end;  comment proc. simps;
```

```
procedure   ssm (a7, dom, ar, ap, ak, a3, a4, a5, a6);

    value  a7;
    integer  ar, ap, ak;
    real  a7, dom, a3, a4, a5, a6;

Begin   if a7 < (sm1 - eps1) then
Begin   dom := dom - ar * dm;  ap := ap + 1;
        if ap = 1 then  a3 := dom;
        if ap = 2 then  Begin  a4 := dom;  ap := 0;
        if a3 = a4 then Begin  ar := entier(ar/2);
        if ar = 0 then ar := 1;  end  end;  go to m2;
end;
        if a7 > (sm1 + eps1) then
Begin   dom := dom + ar * dm;  ak := ak + 1;
        if ak = 1 then  a5 := dom;
        if ak = 2 then Begin  a6 := dom;  ak := 0;
        if a5 = a6 then Begin  ar := entier(ar/2);
        if ar = 0 then ar := 1;  end  end;  go to m2;
end;
end;  comment proc. ssm;
```



```
        ap := 0;
        ak := 0;
        a8 := ar;
        dom := (a2 - a1)/ch;
        a := dom;
        i := 1;
        omega[1] := a1;
m1:     if  ar ≤ 1  then  ar := a8;
m2:     omega[i+1] := omega[i] + dom;
        if omega[i+1] ≥ a2 then Begin omega[i+1] := a2;
        go to  m3  end;
        simps (omega[i], omega[i+1], eps, f1);
        f[i] := f1/(omega[i+1] - omega[i]);
        sm := abs (f[i] - B1);
        error (omega[i], omega[i+1]);
        a7 := s[i]/sm;
        ssm (a7, dom, ar, ap, ak, a3, a4, a5, a6);
        dom := a;   i := i+1;   go to  m1;
m3:     m := i;     for  i := 1  step 1  until  m  do
        omega[i] := 0.5 * (omega[i] + omega[i+1]);
'print' "(///5x, 2Hm =, i3)", m;
'print' "(///60x, 5Homega///(6E18.6))", for i := 1 step 1 until m do omega[i];
'print' "(///60x, 4Hf///(6E18.6))", for i := 1 step 1 until m do f[i];
'print' "(///60x, 4Hs///(6E18.6))", for i := 1 step 1 until m do s[i];
    end;  comment proc window;
```

4. ПРИЛОЖЕНИЕ

Программа ОБР -23, использующая процедуры STREG и POLO для реализации статистической регуляризации.

Содержание процедур, описанных в ОБР -23:

procedure defik	вычисление или ввод матрицы K ($\hat{K}$ ядро интегрального уравнения)
procedure defis	вычисление или ввод среднеквадратичной ошибки измерения $\vec{S}$.
procedure defif	вычисление или ввод измеряемой функции $\vec{f}$.
procedure defxx	вычисление или ввод массива xx (см. стр.55)

procedure grafik	Искомая функция fi представляется в графическом виде; В математическом эксперименте для сравнения с fi графически представляется исходная функция.
procedure graf	Ядро интегрального уравнения (I) $\hat{K}$ представляется в графическом виде.
procedure graff	Правая часть уравнения (I) функция f представляется в графическом виде.
C	Вводится. Через каждые C-1счетных точек наносится ошибка восстановления на графике восстановленной функции и ошибка измерения на графике измеряемой функции. C=I — ошибка наносится в каждой точке.

procedure STREG осуществляет статистическую регуляризацию путем автоматического выбора параметра регуляризации (гладкости) α и производит апостериорную оценку ошибки измерения.

В число формальных параметров процедуры **STREG** входят следующие величины:

1. n — число дискретных значений аргумента восстанавливаемой функции (вводится).

2. m — число дискретных значений аргумента измеряемой функции (вводится).

3. K_{ji}, $j=1,2,\ldots,m$, $i=1,2,\ldots,n$ — ядро интегрального уравнения (вводится или вычисляется в процедуре **defik**).

4. XX_i, $i=1,2,\ldots,n$ — дискретные значения аргумента восстанавливаемой функции (вводится).

XX_i могут быть введены с точностью до константы, поскольку они входят только в выражение $\Omega(\bar{\varphi})$ и влияют только на величину α, но при алгебраизации уравнения (I) XX_i учитываются в K_{ji}, и здесь их значения должны соответствовать физике задачи.

5. f_j, $j=1,2,\ldots,m$ — измеряемая функция (вычисляется или вводится в процедуре **defif**).

6. S_j, $j=1,2,\ldots,m$ — среднеквадратичная ошибка измерений (вычисляется или вводится процедурой **defis**).

7. **alfa** — параметр регуляризации (вводится).

alfa>0 — счет с заданным α, равным введенному.

alfa<0 — счет с выбором α, $\alpha_{нач.}=-$**alfa**.

(omo_{ij})
$i = 1,2,3,4,5.$
$j = 1,2,\ldots,n$

прямоугольная матрица, элементы которой являются элементами матрицы Ω. Для удобства программирования *omo* преобразована и не совпадает с Ω, которая является симметричной положительно определенной матрицей.

aa_i, bb_i, cc_i

коэффициенты, входящие в выражение для нормы второй производной функции φ.

alfa, beta, fi, sigma

вычисляются при помощи обращения к процедурам *reg, om, delt, defalf, defbet*, описанных внутри процедуры *STREG*, а также к процедуре *invers* (см. [I3], стр.37, алгоритм 66а, "Обращение симметричной матрицы").

Процедуры *reg, om, delt, defalf, defbet*, описанные в теле процедуры *STREG*, имеют следующее содержание:

procedure reg

вычисляет величины *fi, sigma*, $d = (\beta\tilde{B} + \alpha\Omega)$, $d^{-1} = (\beta\tilde{B} + \alpha\Omega)^{-1}$ при помощи обращения к процедуре *invers*.

8. *beta*

обратный квадрат среднегеометрической ошибки измерения (вводится).

beta>0 — счет с заданным $\beta = \frac{beta}{S_a^2}$

beta<0 — счет с выбором β.

Вводимое *beta* может быть любым отрицательным числом.

9. inf_i
$i = 1,2,3$

вводится

$inf_1 = 1 + \varepsilon_\alpha \quad \varepsilon_\alpha = \frac{\%}{100}$
$inf_2 = 1 + \varepsilon_\beta \quad \varepsilon_\beta = \frac{\%}{100}$
$inf_3 = \varepsilon_\beta$

$\varepsilon_\alpha, \varepsilon_\beta$ точность вычисления α и β.

10. fi_i
$i = 1,2,\ldots,n$

восстанавливаемая функция.

11. $sigma_i$
$i = 1,2,\ldots,n$

ошибка восстановления.

В процедуре *STREG* вычисляются следующие величины:

$$sa = \prod_{j=1}^{m} S_j^{1/m}$$
$$\tilde{B} = K^+ \tilde{W} K$$
$$\tilde{a} = K^+ \tilde{W} \tilde{f}$$

procedure om вычисляет

$$\langle \Omega(\vec{\varphi}) \rangle = Sp(\Omega d^{-1}) + (\vec{\varphi}, \Omega\vec{\varphi})$$

$$omego = \frac{n-2}{\alpha} - \langle \Omega(\vec{\varphi}) \rangle_{\alpha,\beta}$$

procedure delt вычисляет

$$\langle \Delta(\vec{\varphi}) \rangle = Sp(\tilde{B}d^{-1}) + (\vec{\varphi}, \tilde{B}\vec{\varphi}) - 2(\kappa\tilde{W}\vec{f}, \vec{\varphi}) + (\vec{f}, \tilde{W}\vec{f})$$

$$delto = \frac{m}{\beta} - \langle \Delta(\vec{\varphi}) \rangle_{\alpha,\beta}$$

procedure defalf вычисляет α, соответствующее выполнению условия

$$\frac{dP(\alpha,\beta|\vec{f})}{d\alpha} = 0.$$

procedure defbet вычисляет β [+/], соответствующее выполнению условия

$$\frac{dP(\alpha,\beta|\vec{f})}{d\beta} = 0.$$

+/ Относительно β см. статью В.Ф.Турчина, Л.С.Туровцевой "Метод статистической регуляризации с апостериорной оценкой ошибки исходных данных", ДАН АН СССР, 1975, т.212, №3 и [12].

4.1 ТЕКСТ ПРОГРАММЫ ОБР-23 НА ЯЗЫКЕ АЛГОЛ-60(ГДР)

```
begin integer n,m,c,i,j;
      read (n,m,c);
  begin real alfa, beta, a0, p6;
    array f, s [1:m],
          fi, sigma, xx [1:n],
    d[1:n, 1:n], k[1:m, 1:n], inf [1:3];

  procedure invers(n,a1);
. . . . . .
  procedure defik;
. . . . . .
  procedure defis;
. . . . . .
  procedure defif;
. . . . . .
  procedure graf;
. . . . . .
  procedure graf1;
. . . . . .
  procedure grafik;
. . . . . .
  procedure defxx;
. . . . . .
```

```
        procedure   streg(n, m, κ, xx, f, s, inf, alfa,
                          beta, fi, sigma);
. . . . . .
        procedure   polo (n, alfa, beta, fi);
. . . . . .

        a0:=1;   pB:=1;
        defxx;
        defiκ;  graf;   read (alfa, beta, inf);
        defis;
        defif;  graf1;
        streg(n,m,κ,xx,f, s, inf, alfa, beta,fi, sigma);
        grafiκ;  for i:=1 step 1 until n do
        if fi[i]<0 then go to m1;
        go to m2;
m1:     polo(n,alfa, beta,fi); grafiκ;
m2: 'print' "(///60x, 5Halfa =, F18.6)", alfa;
    'print' "(///60x, 2Hfi///(6F18.6))", fi;
    'print' "(///60x, 5Hsigma///(6F18.6))", sigma;

m4:     end;
        end
        eop
```



```
procedure   streg (n, m, κ, xx, f, s, inf, alfa, beta,
                   fi, sigma);

        value     n, m;
        integer   n, m;
        real      alfa, beta;
        array     κ, xx, f, s, inf, fi, sigma;
begin   real      omega, delta, bet, cors, sa;
        integer   r, i, j, ich;
        array     a, aa, bb, cc[1:n+2], B [1:n, 1:n],
                  oma[1:5, 1:n];
```

```
procedure reg;

begin integer r, i, j;
      for i:=1 step 1 until n do
      for j:=1 step 1 until n do
          d[i,j]:=b[i,j]*beta;
      for i:=1 step 1 until n do
begin if i=n-1 then r:=i+1 else if i=n then
      r:=i else r:=i+2; for j:=i step 1 until r do
begin d[i,j]:=d[i,j]+omo[i-j+3,j]*alfa;
      if i≠j then d[j,i]:=d[i,j]
end end; if ich=0 then go to m3;
      invers(n,d);
for i:=1 step 1 until n do begin fi[i]:=0;

      for j:=1 step 1 until n do
      fi[i]:=fi[i]+d[j,i]*o[j]*beta;

end
end; comment proc. reg;
```



```
procedure om;

begin integer i, j, r, v;
      omega:=0; for i:=1 step 1 until n do
begin if i=1 then r:=3 else if i=2 then r:=2 else r:=1;
      if i=n-1 then begin v:=4; omo[4,i]:=omo[2,i+1]
                    end else
      if i=n then v:=3 else begin v:=5; omo[5,i]:=
      omo[1,i+2]; omo[4,i]:=omo[2,i+1]
                    end;
      for j:=r step 1 until v do
omega:=omega+omo[j,i]*d[i+j-3,i];
omega:=omega+omo[3,i]*fi[i]*2;
      if i≠n and i≠n-1 then
omega:=omega+2*fi[i]*(omo[2,i+1]*fi[i+1]+omo[1,i+2]*fi[i+2]);
      if i=n-1 then omega:=omega+2*fi[n-1]*fi[n]*omo[2,n]
end;
      omega:=(n-2)/alfa-omega
end; comment proc. om;
```

```
procedure  delt ;

begin integer  i, j ;  real v, w, dv; array rt, t[1:n];

    delta := 0;   v:=0;  w:=0;  dv:=0;
for i:=1 step 1 until n do begin rt[i]:=0; t[i]:=0;
for j:=1 step 1 until n do begin
        rt[i] := rt[i] + b[i, j] * d[j, i] ;
        t[i] := t[i] + b[i,j] * fi[i] * fi[j]
                end ;
delta := delta + rt[i];  v:=v + t[i];  w:=w + o[i] * fi[i]
                end;
        for i:=1 step 1 until m do
            dv := dv + ( f[i]/s[i])**2 ;
delta := delta + v - 2*w + dv ;   delta := m/beta - delta
end;    comment proc. delt ;
```



```
procedure  defalf ;

begin real   alm , aln , alk , als ;
            alm := 4**sign(omega) ;
            als := omega ;
r1:     alfa := alm * alfa ;  reg ; om ;
        if omega * als > 0 then go to r1 ;
            aln := (alfa + alfa/alm)/5 ;
            alk := 4 * aln ;
r2:     alfa := (aln + alk) * 0.5 ;  reg ; om ;
        if omega < 0 then
            alk := alfa else aln := alfa ;
    if alk > aln * inf[1] then go to r2
end ;   comment proc defalf ;
```

```
procedure defbet ;

begin real betm, betn, betk, bets ;
      betm := 4**sign (delta) ;
      bets := delta ;
r3:   beta := betm * beta ; reg; delt;
      if delta * bets > 0 then go to r3;
      betn := (beta + beta/betm)/5 ;
      betk := 4 * betn ;
r4:   beta := (betn + betk) * 0.5 ; reg; delt;
      if delta < 0 then
      betk := beta else betn := beta ;
      if betk > betn * inf[2] then go to r4
end; comment proc. defbet ;
```



```
sa := 0; for j := 1 step 1 until m do
                sa := sa + ln(s[j]);
sa := exp(sa/m);
for j := 1 step 1 until m do s[j] := s[j]/sa ;
                for i := 1 step 1 until n do
                for r := 1 step 1 until i do
begin B[i,r] := 0; for j := 1 step 1 until m do
B[i,r] := B[i,r] + k[j,i] * k[j,r]/s[j]**2 ;
      if r ≠ i then B[r,i] := B[i,r]
end;
for i := 1 step 1 until n do begin a[i] := 0;
      for j := 1 step 1 until m do
a[i] := a[i] + k[j,i] * f[j]/s[j]**2
      end ;
```

```
aa[1]:=0; aa[2]:=0;
for i:=3 step 1 until n do Begin
aa[i]:= 1/(xx[i]-xx[i-1]); BB[i]:= -aa[i] - aa[i-1];
cc[i]:= aa[i-1]
end;
BB[2]:=0; BB[n+1]:=0;
cc[n+1]:=0; cc[n+2]:=0;
for i:=1 step 1 until n do Begin
omo[1,i]:= aa[i]*cc[i];
omo[2,i]:= aa[i]*BB[i] + BB[i+1]*cc[i+1];
omo[3,i]:= aa[i]**2 + BB[i+1]**2 + cc[i+2]**2
end;
if Beta >0 then Begin Beta:= Beta/sa**2;
if alfa >0 then Begin reg; go to m1
end else Begin alfa:= -alfa;
reg; om; defalf; go to m1
end
end;
```



```
if Beta<0 then Begin Beta:= 1/sa**2;
if alfa >0 then Begin reg; delt; defBet;
go to m1 end else
alfa:= -alfa; Bet:=Beta; reg; om;
m2: defalf; delt; defBet;
if abs(Bet-Beta)<Beta*inf[3]
then go to m1 else Begin om; Bet:=
Beta; go to m2 end
end;
m1: cors:= 1/(sqrt(Beta)*sa);
sa:= sa*cors;
for j:=1 step 1 until m do
s[j]:= s[j]*sa;
'print' "(///5x,5Hcors=,F10.3, 5x, 3Hsa=,E18.6)", cors, sa;
'print' "(60x, Hs///(6E18.6))", s;
```

```
if a0=1 then begin
for i:=1 step 1 until n do
sigma[i]:=sqrt(d[i,i]);
a0:=0 end;
ich:=0; reg;
m3: end; comment proc. streg;
```



```
procedure grafik;

begin real a,b,a1,b1,a2,b2;
array af, ax[1:2];
page(20.0,26.0,0,0,0);
minmax(fi,n,a,b);
minmax(fit,n,a1,b1);
if a1>a then a2:=a else a2:=a1;
if b1>b then b2:=b1 else b2:=b;
limits(xx[1], xx[n], a2, b2);
region(0.5, 3.0, 18.0, 6.0, 0, 0, 1);
lineo(xx, fit, n);
lineo(xx, fi, n);
for j:=1 step 1 until n do
begin ax[1]:=xx[c*j-c+1];
ax[2]:=ax[1];
af[1]:=fi[c*j-c+1]+sigma[c*j-c+1];
af[2]:=fi[c*j-c+1]-sigma[c*j-c+1];
lineno(ax,af,2,0,0); end; axes(0,0,0.0,5,0,0,0.0,5,11);
begin string ay; array ia[1:50];
if pb=1 then ay:="Восстановленная функция без учета
условия неотрицательности" else "Восстановленная функция
с учетом условия неотрицательности";
stoa(ia[1],ia[10],ay); symbol(4.0,10.0,0.3,ia,59,0.0); pb:=0;
end; endpg(0); end; comment proc. grafik;
```

```
procedure graf;

Begin real a, b;
      array xa, k1 [1:n];

      page (24.0, 24.0, 0, 0, 0);
    for j:=1 step 1 until m do
Begin for i:=1 step 1 until n do k1[i]:=k[j,i];
      minmax (k1, n, a, b); xa[j]:=b;
end;
      minmax (xa, m, a, b);
limits (xx[1], xx[n], 0.0, b); region(0.5,8.0,16.0,8.0,0,0,1);
      for j:=1 step 1 until m do Begin
for i:=1 step 1 until n do k1[i]:=k[j,i];
      linemo (xx, k1, n, 0, 5); end;
    axes (0, 0, 0.0, 5, 0, 0, 0.0, 5, 0);
Begin
      string ay;
      array ia [1:5];
ay:="ядро интегрального уравнения";
      stoa (ia[1], ia[5], ay);
   symbol (4.0, 10.0, 0.3, ia, 28, 0.0);
end;
endpg (0);
end; comment proc. graf;
```



```
procedure graf1;

begin real a, b; array af, ax [1:2], yy[1:m];
                 read (yy);
    page (24, 24, 0, 3, 0);
    minmax (f, m, a, b);
  limits (yy [1], yy [m], a, b);
 region (0.5, 12.0, 16.0, 8.0, 0, 0, 1);
  linemo (yy, f, m, 0, 0);
for j:=1 step 1 until m do
begin
    ax[1]:= yy [c*j - c+1];
    ax[2]:= ax [1];
af [1]:= f [c*j - c+1] + s[c*j - c+1];
af [2]:= f [c*j -c+1] - s[c*j - c+1];
      lineo (ax, af, 2);
end;  axes(0, 0, 0.0, 5, 0, 0, 0.0, 5, 11);
    begin
        string ay;
        array ia [1:12];
ay:="измеряемая функция  свободный член
  интегрального уравнения";
      stoa (ia[1], ia[12], ay);
   symbol (4.0, 10.0, 0.3, ia, 59, 0.0);
    end; endpg(0);
end; comment Proc. graf1;
```

4.2 ТЕКСТ ПРОГРАММЫ ОБР-23 НА ЯЗЫКЕ ФОРТРАН(ДУБНА)

```
      program    OBR 23
      common /a1/ n
      common /a2/ m
      common/a3/ alfa
      common/a4/ beta
      common/a5/ ainf
      common/a6/ binf
      common/a7/ fi(60)
      common/a8/ sigma(60)
      common/a32/ ic
      dimension  d(60,60)
9     read 1, n,m, ic
      read 2, alfa, beta, ainf, binf
      print 3, n,m, alfa, beta, ainf, binf
      call koord(n)
      call defik(n,m)
      call defifs(n,m)
      call streg(n,m,d)
      print 4, (fi(i), i=1, n)
      print 5, (sigma(i), i=1, n)
      do 6 i=1, n
      if (fi(i).GT. 1.0E-18) sigma(i)=sigma(i)*100./fi(i)
6     continue
      print 7, (sigma(i), i=1,n)
      print 8, alfa, beta
      go to 9
1     format(i3/i3/i3)
2     format(E10.0)
3     format(///5x,2Hn=,i3, 5x, 2Hm=,i3, 5x, 5Halfa=,
     c F5.2, 5x, 5Hbeta=, F5.2,
     c 5x, 5Hainf=, F5.2, 5x, 5Hbinf=, F5.2)
4     format(///60x, 2Hfi///(6F18.6))
5     format(///60x, 5Hsigma///(6F18.6))
7     format(///60x, 7Hsigmapr///(6F18.6))
8     format(///5x, 5Halfa=, E12.6, 5x, 5Hbeta=, E12.6)
      end
```



```
      subroutine streg (n,m,d)

      common/a3/ alfa
      common /a4/ beta
      common/a6/ binf
      common/a7/ fi(60)
      common/a8/ sigma(60)
      common/a9/ xx(60)
      common/a10/ bbk(60,60)
      common/a11/ f(60)
      common/a12/ s(60)
      common/a13/ omo(5,60)
      common/a14/ ich
      common/a15/ k1
      common/a16/ b(60,60)
      common/a17/ a(60)
      dimension aa(62), bb(62), cc(62), d(n,n),
     c  dd(60,60)

      k1=0
      ich=2
      sa=0.
      do 1 j=1, m
1     sa = sa + alog(s(j))
      sa = exp(sa/m)
      do 2 j=1, m
2     s(j) = s(j)/sa
      do 4 i=1, n
      do 4 k=1, i
      b(i,k)=0.
      do 5 j=1, m
5     b(i,k) = b(i,k) + bbk(j,i)*bbk(j,k)/s(j)**2
      if(k.NE.i) b(k,i) = b(i,k)
4     continue
```

```
      do 6  i=1, n
      a(i)=0.
      do 6  j=1, m
6     a(i)=a(i) + bbk(j,i)*f(j)/s(j)**2
      aa(1)=0.
      aa(2)=0.
      bb(2)=0.
      bb(n+1)=0.
      cc(n+1)=0.
      cc(n+2)=0.
      do 7  i=3, n
      aa(i)= 1./(xx(i) - xx(i-1))
      bb(i) = -aa(i) - aa(i-1)
7     cc(i) = aa(i-1)
      do 8  i=1, n
      omo(1, i) = aa(i) * cc(i)
      omo(2,i) = aa(i) * bb(i) + bb(i+1)*cc(i+1)
8     omo(3, i) = aa(i)**2 + bb(i+1)**2 + cc(i+2)**2
      if(beta.LE.0.) go to 9
      beta = beta/sa**2
10    if(alfa.GE.0.) go to 11
      alfa = -alfa
      call reg(n,d)
      call om (n,d)
      call defalf (n,d)
      go to 12
11    call reg(n,d)
      go to 12
9     beta = 1./sa**2
      if (alfa. GE.0.) go to 13
      alfa = -alfa
      bet = beta
      call reg(n,d)
      call om(n,d)
      go to 14
```



```
13    call reg(n, d)
      call delt(n,m,d)
      call defbet(n,m,d)
      go to 15
14    call defalf(n,d)
      call delt (n,m ,d)
      call defbet (n, m, d)
      if (abs(bet - beta). LT. beta * binf) go to 15
      call om(n,d)
      bet = beta
      go to 14
15    cors = 1. / (sqrt(beta)*sa)
      sa = sa * cors
      do 16 j=1, m
16    s(j) = s(j) * sa
      print 17, cors, sa
17    format (///5x, 5Hcors=,F5.3, 5x, 3Hsa=,E12.6)
      print 18, (s(i), i=1,m)
18    format(///60x, Hs///(6E18.6))
12    if(k1.EQ.0) go to 19
      go to 21
19    do 20 i=1, n
20    sigma(i) = sqrt(d(i,i))
21    do 22 i=1, n
      if ( fi(i).LT.0.) go to 23
22    continue
      return
23    call polo (n, d, dd, fi)
      if(k1. GT. 1)  return
      go to 10
      return
      end
```

```
      subroutine reg (n,d)

      common/a3/ alfa
      common/a4/ beta
      common/a7/ fi(60)
      common/a13/ omo (5,60)
      common/a14/ ich
      common/a16/ b(60,60)
      common/a17/ a(60)
      dimension d(n,n)

      do 1 i=1, n
      do 1 j=1, n
1     d(i,j) = b(i,j) * beta
      do 2 i=1, n
      k=i+2
      if(i.EQ.n-1) k=i+1
      if(i.EQ.n)   k=i
      do 2 j=i, k
      d(i,j) = d(i,j) + omo(i-j+3, j) * alfa
      if(i.NE.j) d(j,i) = d(j,i)
2     continue
      if( ich.EQ.0) return
      call invers(n,d)
      do 3 i=1, n
      fi(i)=0.
      do 3 j=1, n
3     fi(i) = fi(i) + d(i,j) * a(j) * beta
      return
      end
```



```
      subroutine om(n,d)

      common/a3/ alfa
      common/a7/ fi(60)
      common/a13/ omo(5, 60)
      common/a18/ omega
      dimension d(n,n)

      omega=0.
      do 1 i=1, n
      if(i.EQ.1) k=3
      if(i.EQ.2) k=2
      if(i.NE.1.and. i.NE.2) k=1
      if(i.EQ.n-1) go to 2
      if(i.EQ.n)   go to 3
      if(i.NE.n-1.and.i.NE.n) go to 4
2     j=4
      omo(4,i)= omo(2,i+1)
      go to 5
3     j=3
      go to 5
4     j=5
      omo(5,i)=omo(1,i+2)
      omo(4,i) = omo(2,i+1)
5     do 6 jj=k, j
6     omega = omega +omo(jj,i)*d(i+jj-3,i)
      omega = omega +omo(3,i)*fi(i)**2
      if(i.NE.n.and.i.NE.n-1)omega=omega + 2*fi(i)*
     c (omo(2,i+1)*fi(i+1)+ omo(1,i+2)*fi(i+2))
      if(i.EQ.n-1) omega=omega+2*fi(n-1)*fi(n)*omo(2,n)
1     continue
      omega = (n-2)/alfa - omega
      return
      end
```

```
      subroutine  delt (n, m, d)

      common /a4/  beta
      common /a7/  fi (60)
      common/a11/  f (60)
      common/a12/  s (60)
      common/a16/  b (60,60)
      common/a17/  a(60)
      common/a19/  delta
      dimension  d(n,n), rt(60), t(60)

      delta = 0.
      v = 0.
      w = 0.
      dv = 0.
      do  1  i=1, n
      rt(i)=0.
      t(i)=0.
      do  2  j=1, n
      rt(i) = rt(i) + b(i,j) * d(j,i)
2     t(i) = t(i) + b(i,j) * fi(i) * fi(j)
      delta = delta + rt(i)
      v = v + t(i)
1     w = w + a(i) * fi(i)
      do  3  i=1, m
3     dv = dv + (f(i)/s(i)) ** 2
      delta = delta + v - 2 * w + dv
      delta = m/beta  - delta
      return
      end
```



```
      subroutine defalf (n,d)

      common /a3/ alfa
      common/a5/ ainf
      common/a18/ omega
      dimension  d(n,n)

      alm = 4. ** sign (1, omega)
      als = omega
1     alfa = alm * alfa
      call  reg(n,d)
      call  om (n,d)
      if ( omega * als. GT. 0. ) go to 1
      aln = (alfa + alfa/alm)/5
      alk = 4 * aln
2     alfa = (aln + alk) * 0.5
      call reg(n,d)
      call  om (n,d)
      if (omega . LT. 0.)  alk = alfa
      if (omega. GE. 0.)  aln = alfa
      if (alk. GT. aln * ainf) go to 2
      return
      end
```

```
      subroutine defbet (n,m,d)

      common /a4/ beta
      common/a6/ binf
      common/a19/ delta
      dimension d(n,n)

      betm = 4 ** sign (1, delta)
      bets = delta
1     beta = betm * beta
      call reg (n,d)
      call delt (n,m,d)
      if (delta * bets .GT. 0.) go to 1
      betn = (beta + beta/betm)/5
      betk = 4 * betn
2     beta = (betn + betk) * 0.5
      call reg (n,d)
      call delt (n,m,d)
      if (delta .LT. 0.) betk = beta
      betn = beta
      if(betk . GT. betn*(1+binf)) go to 2
      return
      end
```



```
      subroutine grafik

      common /a1/ n
      common/a7/ fi(60)
      common/a8/ sigma(60)
      common/a9/ xx(60)
      common/a31/ fit(60)
      common/a32/ ic
      dimension af(2), x1(2)

      call page(24., 26., 3HOBR, 3, 0)
      call region(0.5, 3.0, 18.0, 6.0, 2Hfi, 2, 1)
      call minmax(fi, n, a, b)
      call minmax(fit, n, a1, b1)
      if (a1.GE.a) a2=a
      if (a1.LE.a) a2=a1
      if (b1.GE.b) b2=b1
      if (b1.LE.b) b2=b
      call limits (xx(1), xx(n), a2, b2)
      call lineo (xx, fit, n)
      call lineo (xx, fi, n)
      do 1 j=1, n
      x1(1) = xx(ic*j - ic + 1)
      x1(2) = x1(1)
      af(1) = fi(ic*j - ic + 1) + sigma(ic*j - ic + 1)
      af(2) = fi(ic*j - ic + 1) - sigma(ic*j - ic + 1)
1     call lineo(x1, af, 2)
      call axes(2Hxx, 2, 0.0, 5, 2Hfi, 2, 0.0, 5, 11)
      call symbol(4.,10.,.3, 58Hвосстановленная функция
     с с учетом условия неотрицательности, 58, 0.)
      call endpg(0)
      return
      end
```

```
      subroutine graf1

      common /a2/ m
      common /a11/ f(60)
      common/a12/ s(60)
      common/a32/ ic
      dimension af(2), x1(2), yy(60)

      read 1, (yy(i), i=1, m)
1     format (7E10.0)
      call page(24.,24.,3HOBR,3,0)
      call region(0.5, 12.0, 16.0, 8.0, Hf, 1, 1)
      call minmax(f, m, a, b)
      call limits(yy(1), yy(m), a, b)
      call linemo(yy, f, m, 0, 0)
      do 2 j=1, m
      x1(1) = yy(ic*j - ic + 1)
      x1(2) = x1(1)
      af(1) = f(ic*j - ic + 1) + s(ic*j - ic + 1)
      af(2) = f(ic*j - ic + 1) - s(ic*j - ic + 1)
2     call lineo(x1, af, 2)
      call axes(2Hyy, 2, 0.0, 5, Hf, 1, 0.0, 5, 11)
      call symbol(4., 10., .3, 59Hизмеряемая функция
     с свободный член интегрального уравнения, 59, 0.)
      call endpg(0)
      return
      end
```



```
      subroutine graf

      common/a1/ n
      common/a2/ m
      common/a9/ xx(60)
      common/a10/ bbk(60,60)
      dimension x1(60), bbk1(60)

      do 1 j=1, m
      do 2 i=1, n
2     bbk1(i) = bbk(j, i)
      call minmax(bbk1, n, a, b)
1     x1(j) = b
      call minmax(x1, n, a, b)
      call page(24., 24., 3HOBR, 3, 0)
      call region(0.5, 12., 16., 8., 0, 0, 0)
      call limits(xx(1), xx(n), 0., b)
      do 3 j=1, m
      do 4 i=1, n
4     bbk1(i) = bbk(j, i)
3     call linemo(xx, bbk1, n, 0, 5)
      call axes(2Hxx, 2, 0., 5, 3Hbbk, 3, 0., 5, 0)
      call symbol(4., 10., .3, 28Hядро интегрального
     с уравнения, 28, 0.)
      call endpg(0)
      return
      end
```

Считаю приятным долгом выразить глубокую признательность В.Ф.Турчину, под руководством которого автору посчастливилось работать в течение последних нескольких лет.

ЛИТЕРАТУРА

I. А.Н.Тихонов. ДАН СССР, I5I, 50I (I963).

2. А.Н.Тихонов. ДАН СССР, I53, 49 (I963).

3. А.Н.Тихонов, В.Б.Гласко. ЖВМ и МФ 4, 564 (I964).

4. А.Н.Тихонов. ЖВМ и МФ 5, 4, 7I8 (I965).

5. D.L. Phillips, J. Associat Comput. Machin. 9, 84 (1962).

6. В.Ф.Турчин. ЖВМ и МФ 7, I270 (I967).

7. В.Ф.Турчин. ЖВМ и МФ 8, 230 (I968).

8. В.Ф.Турчин, В.З.Нозин. Изв. АН СССР, сер. "Физ.атм. и океана" 5, 29 (I969).

9. В.Ф.Турчин, В.П.Козлов, М.С.Малкевич. УФН I02, вып.3, ноябрь I970.

I0. В.Ф.Турчин, Л.С.Туровцева. Оптика и спектроскопия, т.XXXYI, вып.2, I974.

II. И.Е.Константинов, И.Ф.Моисеев, Е.Г.Тихонов, Л.С.Туровцева, В.Ф.Турчин, Г.А.Федоров, препринт № 54, I973 г. ИПМ АН СССР.

I2. Л.С.Туровцева, В.Ф.Турчин, препринт № 30, I97I, ИПМ АН СССР.

I3. Н.И.Агеев, В.П.Аяик, Л.В.Малик, Ю.И.Марков, Алгоритмы (5I-I00), вып.3, Вычисл.центр АН СССР, Москва, I966.

I4. К.С.Шифрин, В.Ф.Турчин, Л.С.Туровцева, В.А.Гашко, Изв.АН СССР, сер.Физика атм. и океана, т.X, 4, 4I7 (I974).

I5. Н.С.Тиме, Л.С.Туровцева, препринт № 89, I973, ИПМ АН СССР.

УДК (5I9.9)

Решение обратных некорректно поставленных задач методом статистической регуляризации (программа ОБР-23).

Л.С. Туровцева, ИПМ АН СССР, Москва, I975. 86 стр., библиогр.: I5 назв.

Основное содержание предлагаемой работы - программа ОБР-23, написанная на языках АЛГОЛ-60 и ФОРТРАН и предназначенная для решения широкого класса обратных задач из разных разделов естественных наук методом статистической регуляризации.

Работа содержит также краткое изложение идей, лежащих в основе включённых в программу алгоритмов, а именно: описание способа учёта неотрицательности восстанавливаемой функции в методе статистической регуляризации, описание способа восстановления двухкомпонентной функции.

Приведена написанная на языке АЛГОЛ-60 программа для выбора спектральных "окон" осреднения.

Ключевые слова: уравнение Фредгольма I рода, обратная задача, некорректно поставленная задача, регуляризация, метод статистической регуляризации.