УДК 512.25 + 519.3

С.С. СУРИН

РЕШЕНИЕ ЗАДАЧИ ЛИНЕЙНОГО ПРОГРАММИРОВАНИЯ С БОЛЬШИМ ЧИСЛОМ НУЛЕВЫХ ЭЛЕМЕНТОВ В ИСХОДНОЙ СИМПЛЕКС-ТАБЛИЦЕ

§ I. Постановка вопроса и описание алгорифма

Решение задачи линейного программирования методом последовательного исправления плана [I] связано с необходимостью решать на каждой итерации по две системы линейных алгебраических уравнений, порядок которых равен числу условий. Решение одной из этих систем доставляет значения двойственным переменным, соответствующие текущему допустимому решению задачи. Решение другой определяет коэффициенты разложения по текущему базису вводимого в него вектора.

Различные алгорифмы, основанные на той же схеме, что и метод последовательного исправления плана, решают эти системы разными способами. Например, алгорифм с обратной матрицей [2] хранит и на каждой итерации преобразует обратную матрицу этих систем.

Мы опишем алгорифм, работающий по схеме метода последовательного исправления плана и использующий для решения упомянутых систем свойство "слабой заполненности" -преобладание в базисных матрицах задачи нулевых элементов.

Эта особенность базисных матриц позволяет приводить их к ступенчатому виду, используя для этого только перестановки строк и столбцов. Такую процедуру будем в дальнейшем называть триангуляризацией матрицы. Отметим, что триангуляризованная матрица может оказаться треугольной. Так часто бывает в задачах, содержащих транспортные блоки.

В каждой ступеньке триангуляризованной матрицы отмеча-

ется по одной строке и по одному столбцу. Отмеченный столбец должен быть выбран так, чтоы на пересечении с ним строк более высоких ступенек стояли нули, а на пересечении его со строкой , отмеченной в той же ступеньке, что и он, стоял ненулевой элемент. Уравнения и переменные, соответствующие отмеченным строкам и столбцам базисной матрицы, назовем также отмеченными. В дальнейшем отмеченные переменные будут выражаться и вычисляться с помощью отмеченных уравнений.

Итак, пусть нам нужно решить две системы:

$$yB = c_б \quad , \qquad (I)$$

$$Bx = a \quad . \qquad (2)$$

Система (I) определяет значение компонент вектора двойственных переменных для текущего допустимого решения задачи с базисной матрицей B и вектором базисных цен $c_б$. А решение системы (2) является коэффициентами разложения вектора a ,вводимого в базис B .

Триангуляризуем матрицу B и отметим в ней строки и столбцы, как указывалось выше. Сгруппируем теперь вместе все отмеченные строки и все отмеченные столбцы. Тогда матрицу B можно будет разбить на четыре подматрицы:

$$B = \begin{Vmatrix} B_1 & B_2 \\ B_3 & B_4 \end{Vmatrix} .$$

Подматрица B_1 стоит на пересечении неотмеченных строк и неотмеченных столбцов. Подматрица B_2 стоит на пересечении неотмеченных строк и отмеченных столбцов. Подматрица B_3 стоит на пересечении отмеченных строк и неотмеченных столбцов . Подматрица B_4 стоит на пересечении отмеченных строк и отмеченных столбцов и является треугольной.

Заменим векторы $y, x, c_б$ и a парами векторов $(y^{(1)}, y^{(2)}), (x^{(1)}, x^{(2)}), (c_б^{(1)}, c_б^{(2)})$ и $(a^{(1)}, a^{(2)})$, в которых векторы $y^{(1)}, x^{(1)}, c_б^{(1)}$ и $a^{(1)}$ состоят из компонент векторов $y, x, c_б$ и a , соответствующих неотмеченным строкам и столбцам, а векторы $y^{(2)}, x^{(2)}, c_б^{(2)}$ и $a^{(2)}$ состоят из компонент тех же векторов, соответствующих отмеченным строкам и столбцам матрицы.

Теперь системы (I) и (2) мы можем переписать так:

$$\left.\begin{aligned} y^{(1)}B_1 + y^{(2)}B_3 &= c_*^{(1)}, \\ y^{(1)}B_2 + y^{(2)}B_4 &= c_*^{(2)}, \end{aligned}\right\} \qquad (1')$$

$$\left.\begin{aligned} B_1 x^{(1)} + B_2 x^{(2)} &= a^{(1)}, \\ B_3 x^{(1)} + B_4 x^{(2)} &= a^{(2)}. \end{aligned}\right\} \qquad (2')$$

Чтобы решить систему (1'), умножим справа вторую группу уравнений в ней на $B_4^{-1}B_3$ и добавим к первой

$$y^{(1)}(B_1 - B_2B_4^{-1}B_3) = c_*^{(1)} - c_*^{(2)}B_4^{-1}B_3 ,$$
$$y^{(1)}B_2 + y^{(2)}B_4 = c_*^{(2)} .$$

Отсюда

$$y^{(1)} = (c_*^{(1)} - c_*^{(2)}B_4^{-1}B_3)\cdot(B_1 - B_2B_4^{-1}B_3)^{-1};$$
$$y^{(2)} = (c_*^{(2)} - y^{(1)}B_2)\cdot B_4^{-1} =$$
$$= [c_*^{(2)} - (c_*^{(1)} - c_*^{(2)}B_4^{-1}B_3)(B_1 - B_2B_4^{-1}B_3)^{-1}B_2]\,B_4^{-1}.$$

Аналогичным преобразованием решается система (2') . Из

$$\left.\begin{aligned} (B_1 - B_2B_4^{-1}B_3)x^{(1)} &= a^{(1)} - B_2B_4^{-1}a^{(2)}, \\ B_3 x^{(1)} + B_4 x^{(2)} &= a^{(2)}. \end{aligned}\right\}$$

получаем

$$x^{(1)} = (B_1 - B_2B_4^{-1}B_3)^{-1}(a^{(1)} - B_2B_4^{-1}a^{(2)});$$
$$x^{(2)} = B_4^{-1}(a^{(2)} - B_3x^{(1)}) =$$
$$= B_4^{-1}[a^{(2)} - B_3(B_1 - B_2B_4^{-1}B_3)^{-1}(a^{(1)} - B_2B_4^{-1}a^{(2)})] .$$

Для вычисления векторов y и x по этим формулам нужно обращать только матрицу

$$B_1' = B_1 - B_2B_4^{-1}B_3 ,$$

порядок которой для редко заполненной базисной матрицы B бывает невелик. Обращение же матрицы B_4 можно заменить решением соответствующих систем, которое проводится по рекуррентным формулам, так как матрица B_4 - треугольная.

Таким образом, одна итерация рассматриваемого алгорифма состоит из следующих основных операций:

1. Триангуляризация базисной матрицы. Так как базисные матрицы на соседних итерациях отличаются только одним столбцом и матрица предыдущей итерации имела ступенчатый вид, то на следующей итерации изменит свои места при триангуляризации лишь часть строк и столбцов. В алгорифме это обстоятельство учитывается и на каждой итерации проводится частичная триангуляризация текущей базисной матрицы.

2. Формирование матрицы B_1' и вычисление вектора $c_*^{(1)} - c_*^{(2)}B_4^{-1}B_3$. Эта операция необходима, так как при триангуляризации изменяется разбиение матрицы B на подмат -

рицы B_1, B_2, B_3, B_4.

3. Обращение сформированной матрицы B_1'.
4. Вычисление вектора $y^{(1)}$.
5. Рекуррентное вычисление вектора $y^{(2)}$.
6. Проверка условий оптимальности,
7. Вычисление вектора $a^{(1)} = B_2 B_4^{-1} a^{(1)}$.
8. Вычисление вектора $x^{(1)}$.
9. Рекуррентное вычисление вектора $x^{(2)}$.
10. Исправление плана.

Ниже (§ 4) приводится запись всего алгорифма на языке АЛГОЛ-60, снабженная комментариями.

§ 2. Подготовка исходных данных

Алгорифм решает задачу линейного программирования с максимизацией линейной формы $Z(x)$, записанную в виде:

$$a_{11}x_1 + a_{12}x_2 + \ldots\ldots + a_{1n}x_n = b_1,$$
$$a_{21}x_1 + a_{22}x_2 + \ldots\ldots + a_{2n}x_n = b_2,$$
$$\ldots\ldots\ldots\ldots\ldots\ldots$$
$$a_{m1}x_1 + a_{m2}x_2 + \ldots\ldots + a_{mn}x_n = b_m,$$
$$x_j \geq 0, \quad j = 1, 2, \ldots, n.$$
$$Z(x) = c_1x_1 + c_2x_2 + \ldots + c_nx_n = max.$$

Входная информация готовится следующим образом.

Матрица A дополняется еще одной строкой, состоящей из коэффициентов линейной формы. Ей присваивается номер I. Остальные строки матрицы A нумеруются числами от 2 до $m+1$.

Расширенная таким образом симплекс-таблица записывается в порядке расположения элементов в столбцах. При этом запись элементов каждого нового столбца начинается с соответствующего ему коэффициента линейной формы, а из остальных выписываются только ненулевые элементы. Так формируется массив D.

Одновременно с ним формируется массив ND номеров строк расширенной симплекс-таблицы, содержащих элементы, записанные в массив D.

Порядок формирования массивов D и ND должен быть таким, чтобы на одних и тех же местах в массиве D стоял элемент расширенной симплекс-таблицы, а в массиве ND его порядковый номер в столбце. Таким образом, согласно принятой нумера-

ции строк симплекс-таблицы разделителями столбцов в массиве D служат единицы в массиве ND.

Формируется так же третий массив, -массив x. Он состоит из компонент вектора b, выписанных без пропусков в порядке расположения соответствующих им строк в матрице A.

Вводом присваиваются значения еще трем переменным: m (число условий в задаче линейного программирования), k (количество введенных компонент массива D) и $m1$ (предполагаемый максимальный порядок подматрицы B_t).

Порядок ввода информации:

m ;
K ;
$m1$;
x ;
D ;
ND .

§ 3. Смысл идентификаторов, использованных в записи алгорифма

Переменные

Целые переменные:

m - число условий в задаче линейного программирования;
$m1$ - максимальный порядок матрицы B_t ;
c - счетчик итераций ;
n - число столбцов в симплекс-таблице;
k - количество вводимых компонент в массиве D .

Полная длина массивов D и ND принимается равной

$$k+3m+2 .$$

Компонентам массива D с номерами $k+1$, $k+2,\ldots,k+2m$ присваиваются значения ненулевых (диагональных) элементов начальной (искусственной) базисной матрицы и начального вектора базисных цен. Соответствующие компоненты массива ND равны номерам строк ограничений, в которые добавлены искусственные переменные:

$$md-k+2m+1 .$$

Компоненты массивов D и ND с номерами $md+1$, $md+2,\ldots$ используются для формирования строк базисной матрицы.

p - порядок текущей матрицы B_t ;
j - номер вводимого в базис на данной итерации вектора;
nd - адрес начала информации о j - м столбце матрицы A в массиве D ;
np - адрес начала информации о $j+1$ - м столбце матрицы A в массиве D .

η – переменная, принимающая значения 0 и I, $\eta = 0$, когда ищется допустимый план (I этап), и $\eta = I$, когда ищется оптимальный план (II этап).

Γ – переменная, принимающая значения 0 и I ;

$\Gamma = 0$, когда вычисляются значения двойственных переменных;

$\Gamma = I$, когда вычисляются коэффициенты разложения вводимого в базис вектора;

nu , nv , nw , q , i , s , t – вспомогательные переменные.

Переменные s и t чаще всего используются как параметры цикла.

Вещественные переменные

b – барьер, определяющий выбор вводимого в базис вектора. В начале каждого этапа $b = 10^{10}$.

Если

$$c_j - \sum_{i=1}^{m} a_{ij} y_i \le b , \quad j = 1,2, \ldots, n , \qquad (*)$$

то

$$b := \frac{1}{2} \max_j (c_j - \sum_{i=1}^{m} a_{ij} y_j) .$$

Если выполняется (*) и $b < 10^{-5}$, то данный этап решения считается законченным.

u , v , w – вспомогательные переменные.

Массивы

Целые массивы . Кроме упомянутого выше массива ND , используются еще следующие целые массивы, каждый из которых имеет по m компонент.

$J1$ – номера строк базисной матрицы. Располагаются в массиве в том же порядке, в каком располагаются столбцы в транспонированной триангуляризованной базисной матрице.

J – номера базисных переменных. Располагаются в массиве в том порядке, в каком располагаются соответствующие им строки в транспонированной, триангуляризованной базисной матрице.

AD – адреса базисных цен в массиве D , увеличенные на единицу. Расположены в том же порядке, что и номера базисных переменных в массиве J .

$P3$ – массив, компоненты которого принимают значения 0 или 1 .

$P3[J1[s]] = 0$, если $J1[s]$ – номер отмеченной строки базисной матрицы.

$P3[J1[s]] = 1$, если $J1[s]$ – номер неотме-

ченной строки базисной матрицы.

$J2$. $J2[s] = 0$, если $J1[s]$ - номер отмеченной строки. В противном случае $J2[s]$ определяет для столбца матрицы A с номером $j[s]$ его место в триангуляризованной базисной матрице.

$P1$ и $P2$ - вспомогательные масс..вы.
Каждая их компонента принимает значения 0 или I.

Вещественные массивы

Кроме упомянутого массива D , используется еще 6 вещественных массивов.

x - план, $x[s]$ - интенсивность, с которой в план входит базисная переменная с номером $j[s]$.

Z - в этом массиве помещаются попеременно значения двойственных переменных и коэффициенты разложения вводимого в базис вектора.

$A1$ - квадратная матрица порядка $m1$, имеющая вид:

$B1$	0
0	e

Здесь e - единичная матрица; $A2$ - матрица $A1^{-1}$; $B1$ - массив, предназначенный для хранения векторов

$$c_*^{(1)} - c_*^{(2)} B_*^{-1} B_3 \ ;$$

E - вспомогательный массив, состоящий из m компонент.

Процедуры

Preis -процедура вычисления вектора y ;
Expand -процедура вычисления коэффициентов разложения вводимого в базис вектора;
Recur -процедура рекуррентного решения систем;
St(f,j) -процедура выборки строк и столбцов базисной матрицы в сжатом виде;
New -процедура поиска вектора, подлежащего вводу в базис;
Barrier -процедура, управляющая значениями переменной b и устанавливающая конец очередного этапа вычислений;
Newplan -процедура перевычисления вектора x ;
T -процедура частичной триангуляризации;
Start -процедура, начинающая работу алгорифма. В её функции входит:

ввод массивов;

их обработка;

построение начальной базисной матрицы и начального вектора базисных цен; присвоение начальных значений другим переменным и массивам.

Fin - заключительная процедура алгорифма. Выдает оптимальный план и информацию о нем в следующем порядке:

I) Оценки оптимального плана (в порядке расположения строк в матрице A);

2) оптимальный план;

3) номера базисных переменных оптимального плана ;

2) и 3) выдаются в одном и том же порядке;

4) вычисленные ограничения - выдаются в том же порядке, что и I);

5) значение линейной формы, соответствующее оптимальному плану;

6) значение счетчика итераций.

Эти выдачи производятся дважды в указанном порядке.

В программе предусмотрены остановы с печатью одного числа.

Если печатается число I, то останов означает нарушение неравенства

$$p < m1 .$$

В этом случае надо начинать решение задачи сначала, увеличив $m1$.

Печать числа 2 перед остановом означает, что функционал в задаче неограничен.

В программе используется стандартная процедура обращения матрицы СП 0037.

§ 4. Алгорифм

Алгорифм оформлен в виде программы в соответствии с требованием α - языка [3]:

```
begin integer m,n,m1,k,c,md,p,j,nd,np,7,r,
          nu,nv,nw,q,i,s,t;
          real b,u,v,w;
          ввод (m,k,m1);
    begin integer array I1,J,AD,P1,P2,P3,I2[1:m];
          ND [1:k+3×m+2];
```

```
real array x,z,E[1:m],A1,A2[1:m1,1:m1],
b[1:m1], D[1:k+3×m+2];

procedure st(f,j);
```

comment В зависимости от значения переменной r выбирается одна из двух ветвей работы этой процедуры. Если r=0 (режим вычисления оценок), то st выдает адрес нужного столбца базисной матрицы, засылая в массив z соответствующий элемент в линейной форме. Если r=1 (режим вычисления коэффициентов разложения), то в конце массива D в сжатом виде формируется нужная строка базисной матрицы;

```
value f,j; integer f,j;

begin integer s,t,n;

    if r=0 then

       begin q:=AD[f]; i:=j;

           z[j]:=D[q-1]×(if q-1>k then 1 else 0);

       end

    else

       begin t:=md+1;

       for s:=f step-1 until 1 do

       if I2[s]=0 then goto L 7 k2;

       L 7 k2: for s:=s step 1 until m do

          begin n:=AD[s]-1;

          for n:=n+1 while ND[n]≤ m do

          if ND[n]=j then

              begin D[t]:=D[n];
              ND[t]:=I1[s]; t:=t+1;
              goto L 7 k1
              end

          L 7 k1: end;

          ND [t]:=m+1; q:=md+1; i:=j

       end

end;

          procedure Recur (b,h,e);
```

comment С помощью этой процедуры в системе с (левой) треугольной матрицей отыскивается первое (считая сверху) уравнение, по которому можно вычислить ненулевые зна-

чения соответствующей ему (входящей в это уравнение с ненулевым коэффициентом, стоящим на главной диагонали) переменной и рекуррентно вычисляются значения этой и всех следующих переменных;

```
value b,h,e; integer b,h,e;
begin for s:=b step h until e do
    if z[I1[s]]≠0 then goto L 1 k2
    else if r=0∧D[AD[s]-1]≠0 then
    goto L 1 k2; goto L 1 k3;
L 1 k2: for t:=s step h until e do
    if I2[t]=0 then
    begin st (t,I1[t]); s:=q-1;
        for s:=s+1 while ND[s]≤ m do
            begin if ND[s]≠i then
                z[i]:=z[i]-D[s]×z[ND[s]]
                else u:=D[s]
            end;
            z[i]:=z[i]/u
    end;
L 1 k3: end;
```

```
                    procedure Barrier;
```

comment Если нет ни одной невязки, превышающей барьер, то эта процедура ищет новый барьер. Если он меньше 10^{-5}, то переменная γ получает значение второго этапа или уравнение передается процедуре Fin;

```
begin procedure New;
```

comment New в массиве D отыскивает столбец, подлежащий вводу в базис, и записывает его в развернутом виде в массиве z . Просмотр массива D обычно начинается с того места, на котором он был прерван на предыдущей итерации. Одновременно среди невязок, не превышающих барьер , ищется максимальная.

```
    begin u:=0;
    for s:=np step 1 until k do
        begin if ND [s]>m then
```

```
                    begin j:=j+1; u:=D[s]×η ; nu:=s+1
                    end
                else
                begin u:=u-D[s]×z[ND[s]] ;
                    if ND[s.1]>m  then
                        begin if u>b . then
                                begin  np:=s+1;
                                for t:=1 step 1 until
                                m do  z[t]:=0;
                                for t:=nu step 1
                                 until np-1 do
                                 z[ND[t]] :=D[t] ;
                                 r:=1; nd:=nu;go to L2
                                 end
                            else if v<u  then
                                 begin v:=u; nv:=j;
                                 nw:=nu-1
                                 end
                        end
                end
            end
        end;
        New; j:=0; np:=1; v:=0; New;
        if v < 10^-5  then
           begin if η ≠0  then go to L5
                 else
                 begin η :=1; b:=10^10; go to L4
                 end
           end
        else
        begin b:=v/2; j:=nv-1; np:=nw; New
        end
end;
                            procedure Expand;
```

comment Expand вычисляет вектор $a^{(1)} - B_2 B_4^{-1} a^{(2)}$, умножая его слева на $(B_1')^{-1}$, находит вектор $x^{(1)}$, который записывается в развернутом виде в массив Z. Обращается к процедуре Recur для вычисления вектора $x^{(2)}$;

```
begin if p=0 then go to  L5 k2;
for s:=1 step 1 until m do E[s]:=z[s];
for s:=m step-1 until 1 do
if I2[s]=0 then
    begin st(s,I1[s]); t:=q-1;
    for t:=t+1 while ND[t]≤ m do
        if P3[ND[t]] =0  then
           begin if i≠ND[t]  then
                    E[i] :=E[i]-D[t]×E[ND[t]]
                 else u:=D[t]  end;
        v:=E[i] :=E[i]/u;
    t:=AD[s]-1;
    for t:=t+1 while ND[t]≤ m  do
        if P3[ND[t]] =1  then
            z[ND[t]] :=z[ND[t]] -D[t]× v
    end;
    t:=1; for s:=1 step 1 until m do
       if P3[s]=1  then
          begin E[t]:=z[s]; t:=t+1
          end;
    t:=0; for nu:=1 step 1 until m do
        if  I2[nu]≠0 then
            begin t:=t+1; z[I1[nu]] :=0;
                  for s:=1 step 1 until p do
                      z[I1[nu]] :=z[I1[nu]] +
                       E[s]×A2[s,t].
            end;
        L5 k2:  Recur (m,-1,1)
end;
```

procedure Newplan;

comment Newplan перевычисляет план и делает соответствующие замены в массивах J и AD;

```
 begin u:=10^10;
      for s:=1 step 1 until m do
      if z[I1[s]] > 10^-5  then
```

```
                begin v:=x[s]/z[I1[s]];
                    if v<u then
                        begin u:=v; nu:=s
                        end
                end;
            if u=10^10 then
                begin вывод (2); stop
                end;

    comment обнаружена неограниченность линейной формы;
            if v<10^-5 then u:=10^-5;
            for s:=1 step 1 until m do
                begin v:=x[s]-u*z[I1[s]];
                    x[s]:=if v<10^-5 then 10^-5 else v
                end;
            x[nu]:=u; J[nu]:=j; AD[nu]:=nd;
        end;
                    procedure T;
        begin integer r; if t=1 then
            begin nv:=0 go to L8 k1
            end;
    comment следующие операторы ищут начальную границу и пер-
вое приближение конечной границы ( значение переменной i ) три-
ангуляризации и формируют для неё исходную информацию;
    nw:=1; s:=AD[nu]-1;
    for s:=s+1 while ND[s]≤m do
        begin nv:=ND[s];
        for t:=nw step 1 until m do
            if nv=I1[t] then
                begin i:=t; nw:=t+1; go to L9 k1
```

```
                end;
        L9 k1: end;
    if nu > 1  then;
        begin nv:=nu; nu:=i; i:=nv
        end;
    for s:=nu step -1 until 1 do
        if I2[s]=0  then
            begin nu:=s; go to L9 k5
            end;
    L9 k5: nu:=nu; q:=nu-1;
    for s:=1 step 1 until m do E[s]:=0;
                for s:=1 step 1 until m do
        if  s < nu  then
            begin if I2[s] > nu  then
                  begin P1[s]:=P2[s]:=1; nu:=s
                  end;
            E[I1[s]] :=1
            end
        else
            begin P1[s]:=1; P3[I1[s]] :=0
            end;
    comment  частичная триангуляризация. nu - начальная
граница триангуляризации;
                nu:=nv; nw:=nu;
                L9 k2 : u:=10^10;
          for s:=nw step 1 until m do
              if P1[s]=1 then
                  begin v:=0; t:=AD[s]-1;
                       for t:=t+1 while ND[t]≤ m do
                          if E[ND[t]] =0 then v:=v+1;
                       if v<u  then
```

```
                    begin u:=v; nv:=s;
                      if u=0  then go to L9 k4
                    end
                end;
```

comment следующий оператор корректирует конечную границу;

```
        L9 k4:  if nv >i  then i:=nv;
        if u >0  then
            begin v:=0; s:=AD[nv]-1;
                for s:=s+1  while ND[s]< m do
                if E[ND[s]] =0  then
                    begin if abs (D[s]) >v  then
                            begin v:=abs(D[s]); t:=ND[s]
                            end
                    end;
            q:=q+1; r:=J[q]; J[q]:=J[nv]; J[nv]:=r;
            I2[q]:=0; I1[q]:=t; P1[q]:=0;
            r:=AD[q]; AD[q]:=AD[nv]; AD[nv]:=r;
            v:=x[q]; x[q]:=x[nv]; x[nv]:=v;
             E[t]:=1;
             if u >1  then
                 begin nv:=q+1; s:=AD[q]-1;
                  for s:=s+1 while ND[s]< m do
                      begin if E[ND[s]] =0  then
                              begin E[ND[s]] :=1;
                                I1[nv]:=ND[s];
                                P2[nv]:=P3[ND[s]] :=1;
                                nv:=nv+1
                              end
                      end; q:=nv-1
                end
```

```
        end
           else
             begin for s:=nu  step 1
             until m do
             if P2[s]=1  then
                begin r:=J[s];  J[s]:=
                =J[nv];
                J[nv]:=r;  I2[s]:=q;
                P1[s]:=P2[s]:=0;
                r:=AD[s]; AD[s]:=AD[nv];
                AD[nv]:=r; v:=x[s];
                x[s]:=x[nv]; x[nv]:=v;
                go to L9 k3
                end
             end;
comment проверка конца триангуляризации;
 L9 k3: for s:=nw  step 1 until m do
     if P1[s]=1  then
        begin nw:=s;  go to L9 k6
        end;
 L9 k6: if nw≤i ∧ P1[nw]=1  then go to L9 k2;
 for s:=nw  step 1 until m do
          begin P1[s]:=P2[s]:=0;
          if I2[s]≠0  then P3[I1[s]] :=1
          end;
comment частичное перевычисление матрицы B'_1 и вектора
```

$$C_0^1 - C_0^2 B_4^{-1} B_3 \;;$$

```
nv:=0;
for s:=I step I until m do
if I2[s]≠0  then nv:=nv+I;
```

```
if nv=0  then go to L8 k3;
if nv < p  then
   begin for s:=1 step 1 until p-1 do
                  A1[s,p]:=A1[p,s]:=0;
         A1[p,p]:=1
   end;
         p:=nv; nv:=0;
         if p > m 1 then begin  вывод (1); stop
                          end;
comment      обнаружено нарушение неравенства      p ≤ m 1 ;
         for s:=1 step 1 until nu-1 do
            if I2[s]≠0 ∧ I2[s] < nu  then nv:=nv+1;
         for t:=1  step 1 until m do
            if I2[t] ≥ nu  then go to L8 k1;
         go to L8 k3;
         L8 k1: for s:=t step 1 until m do
      if I2[s]≠0  then
          begin for t:=1 step 1 until m do E[t]:=0;
             t:=AD[s]-1; v:=D[t] × (if t > k then 1 else ?);
             for t:=t+1 while ND[t]<m do E[ND[t]] :=D[t];
             for t:=I2[s]  step-1 until 1 do
             if I2[t]=0 ∧ abs(E[I1[t]] ) > 10-5 then
                 begin nw:=AD[t]-1;
                   for nw:=nw+1 while
                   ND[nw] ≤ m do
                     if ND[nw]=I1[t]  then
                     begin u:=E[I1[t]] /D[nw];
                       go to L8 k2
                     end;
                   L8 k2: nw:=AD[t]-1;
```

```
                                    for nw:=nw+1 while
                                     ND[nw] ≤ m do
                                     E[ND[nw]] :=E[ND[nw]] -
                                      D[nw] × u;
                                     v:=v-D[AD[t]-1] × u ×
                                      (if AD[t]-1 > k then 1 else ? )
                              end;
                         nv:=nv+1; b1[nv] :=v; t:=0;
                    for nw:=1 step 1 until m do
                    if P3[nw]=1 then
                        begin t:=t+1; A1[nv,t]:=E[nw]
                        end
                    end;
                        for s:=1 step 1 until m 1 do
                        for t:=1 step 1 until m1 do
                    A2[s,t]:=A1[s,t];
                       СП  0037(A2[1,1],m1,z[1],E[1]);
               L8 k3: end;
```

procedure Preis;

comment Preis вычисляет вектор $y^{(1)}$, записывает его в развернутом виде в массив Z и обращается к процедуре Recur для вычисления вектора $y^{(2)}$.

```
begin for s:=1 step 1 until m do z[s]:=0;
 r:=0; if p=0 then go to L8 k4;
 nv:=0;
 for s:=1 step 1 until m do
 if P3[s]=1 then
     begin nv:=nv+1; u:=0;
         for t:=1 step 1 until p do
         u:=u+A2[nv,t] × b1[t];
         z[s]:=u
```

```
        end
    L8 k4: Recur (1,1,m); Barrier;
end;
                    procedure start;
begin BBOX (x,D,ND); md:=k+2 × m+1; n:=0;
for s:=1 step 1 until k do
    begin ND[s]:=ND[s]-1;
        if ND[s]=0 then
         begin n:=n+1; ND[s]:=m+1
         end
    end;
for s:=1 step 1 until m do
     begin I1[s]:=s;
           ND[k+2 × s-1]:=m+1;
           ND[k+2 × s]:=s;
           D[k+2 × s-1]:=-1;
           D[k+2 × s]:=if x[s]=0 then 1 else sign(x[s]);
           J[s]:=n+s; AD[s]:=2 × s+k;
           I2[s]:=P1[s]:=P2[s]:=P3[s]:=z[s]:=f[s]:=0;
           x[s]:=abs(x[s])
     end;
b:=10^10; mp:=1; r:=p:=j:=y:=0;
c:=0; ND[md]:=m+1;
for s:=1 step 1 until m1 do
    begin for t:=1 step 1 until m1 do
           A1[s,t]:=0;
           A1[s,s]:=1
    end
end;
                    procedure Fin;
```

```
        begin for s:=1 step 1 until m do E[s]:=0; u:=0
              for s:=1 step 1 until m do
                begin t:=AD[s]-1;u:=u+x[s]× D[t];
                      for t:=t+1 while ND[t]≤m do
                      E [ND[t]]:=E[ND[t]] +D[t]× ∠[s];
                end;
         вывод (s,x,j,E,u,c);
         вывод (s,x,j,E,u,c); stop
        end;
         start;
         L1: Preis;
         L2: Expand;
         Newplan; t:=0;
         L3: T; go to L1;
         L4: t:=1; go to L3;
         L5: Fin
    end
end
```


Литература

1.Л.В.Канторович. Экономический расчет наилучшего использования ресурсов.Изд. АН СССР,1959.

2.С.Гасс. Линейное программирование. Физматгиз. М.,1961.

3.Отчет. Альфа-система. Руководство по использованию.Новосибирск, Вычислительный центр СО АН СССР, 1966.

Поступила в редакцию
15.XI.1967 г.