KFKI
14/1968




# Code for the Calculation of Multigroup Constants from a 26-Group System of Micro-Cross-Sections

P. Vértes

HUNGARIAN ACADEMY OF SCIENCES
CENTRAL RESEARCH INSTITUTE FOR PHYSICS

BUDAPEST

# CODE FOR THE CALCULATION OF MULTIGROUP CONSTANTS FROM A 26-GROUP SYSTEM OF MICRO-CROSS-SECTIONS

Péter Vértes

Central Research Institute for Physics
Budapest, Hungary

ABSTRACT

An algol procedure to calculate the multigroup constants from 26-group micro-cross-sections is described. The tabulated micro-cross-sections are written into magnetic tape backing store by special codes which are also discussed.

The procedure is as simple as possible and needs a relatively small fast store. It has a short and a long version. In the latter corrections are made for resonance self-shielding.

## 1. INTRODUCTION

Micro-cross-sections of elements which are of interest in reactor physical calculations have been already published in tabulated from [1], [2], [3]. The tables of [3] have been used as a cross-section library for the calculation of the multigroup constants in homogeneous mixtures.

A special code /referred to as RFGB/ is used for writing the tabulated micro-cross-sections punched on tape in a convenient form into a file, called RFT1, on the magnetic tape.

By the algol procedure, referred to as "group", the data on this file are used for calculating the macro-multigroup constants of a given mixture.

The guiding principle in the writing of this procedure has been to minimize the occupied fast store and the running, time, in order to facilitate its application as a subroutine of a larger reactor code. Therefore the lion's share of the calculations is performed by the code RPGB and only a small part of the arithmetics is left to be done by the "group" program.

The "group" procedure is written in two versions. The first version is much simpler than the second one since the resonance self-shielding corrections recommended by [3] are not performed.

In 2. the construction of cross-section library RFT1 and the first version of the group procedure is discussed. The method of correction for resonance self-shielding is presented in 3.

## 2. CONNECTION BETWEEN MICRO-CROSS-SECTIONS AND MULTIGROUP CONSTANTS AND THE CONSTRUCTION OF CROSS-SECTION LIBRARY

From the micro-cross-sections tabulated in [3] the scattering matrix can be calculated up to the linear term. This is sufficient in most of the practical cases. The following multigroup constants are to be calculated in this way.

$\sum^{j}$ the total cross section in group j,

$\sum_{o}^{j}$ the total removal cross section in group j,

$\sum_{o}^{i\to j}$ the transfer cross-section from i to group j,

$$\sum_{1}^{i\to j} = \mu^{i\to j} \sum_{o}^{i\to j}$$

where

$\mu^{i\to j}$ is the average cosine of the scattering angle with transfer from the group i to the group j, i.e. the transport cross section

$$\Sigma_1^j = \Sigma^j - \mu^{j\to j}\,\Sigma_o^{j\to j}$$

$\left(\nu\,\Sigma_f\right)^j$ is the product of the fission cross section and the average number of fission neutrons in the group j.

The micro-cross sections taken from the tables [3] are

$\sigma_t^j$ the total cross section,

$\sigma_e^j$ the scattering cross section,

$\sigma_{s(e)}^j$ the slowing down cross section,

$\sigma_{in}^{i\to i+k}$ transfer cross section by inelastic scattering from the group i to be group i+k.

These cross sections are used for each isotope in the mixture.

To specify the other data of a given isotope, three constants, A, B, C, have been introduced as follows.

For hydrogen and deuterium /A = 1/ the elastic transfer cross section $\sigma_e^{i\to i+k}$ and the average cosine $\mu_e^{i\to i+k}$ are also given.

For elements with atomic mass $6 \le M < 20$ /A = 0, B = 1/ the average cosine of the scattering angle is given for slowing down from group j - $\mu_{s(e)}^{\delta}$ - and for inelastic scattering from group i to group i + k, $\mu_{in}^{i\to i+k}$

For elements with atomic mass $6 \le M$ /A = 0/ the average cosine of the elastic scattering $\mu_e^j$ is used.

For fissionable isotopes C = 1 /otherwise C = 0/, then the fission cross section $\sigma_f^j$ and the average number of fission neutrons $\nu^j$ are given.

The constants A, B, C punched before the cross sections of each isotope in order to control the process of reading and evaluation.

As known, each of the macroconstants of a homogeneous mixture can be obtained from the formula

$$\Sigma = \sum_{i=1}^{n} \sigma^{(i)} q_i \qquad /1/$$

where $\rho_i$ is the number of nuclei/cm$^3$ of the isotope i. Since $\sigma^{(i)}$ is given in barns, $\rho_i$ should be taken in units of $10^{24}$.

After the reading-in of the micro cross sections from the paper tape, the microconstants to be used in /1/ are calculated and written on the file RFT1 by the RPGB program. Thus the formula /1/ describes the only arithmetics to be done by the first version of the "group" procedure.

The micro-cross sections and the micro-multigroup constants are related by the formulae

$$\sigma_o^j = \sigma_t^j - \sigma_e^j + \sigma_{s(e)}^j - \sigma_{in}^{j\to j} \qquad /2/$$

$$\sigma_1^j = \sigma_t^j - \mu_e^j \sigma_e^j (1 - A) + \mu_{s(e)}^j \sigma_{s(e)}^j (1 - A)B$$

$$-\sigma_e^{j\to j} \mu_e^{j\to j} A - \mu_{in}^{j\to j} \sigma_{in}^{j\to j} B \qquad /3/$$

$$\sigma_o^{\ell\to j} = A\sigma_e^{\ell\to \ell+(j-\ell)} + \sigma_{s(e)}^{j-1}(1-A)\,\delta_{\ell,j-1} + \sigma_{in}^{\ell\to \ell+(j-\ell)} \qquad /4/$$

$$\sigma_1^{\ell\to j} = \mu_e^{\ell\to \ell+(j-\ell)} \sigma_e^{\ell\to \ell+(j-\ell)} A + \left(\mu_{s(e)}^{j-1} \sigma_{s(e)}^{j-1} (1 - A)\delta_{\ell,j-1}\right.$$

$$\left. + \mu_{in}^{\ell\to \ell+(j-\ell)} \sigma_{in}^{\ell\to \ell+(j-\ell)}\right) B \qquad /5/$$

$$(\nu\sigma_f)^j = \nu^j \sigma_f^j \qquad /6/$$

$$\sigma_t^j = \sigma_t^j \qquad /7/$$

In order to accelerate the work with RFT1, the above constants are arranged in three arrays, a follows

| | | |
|---|---|---|
| stfO [j] = $\sigma_t^j$ | atm [j, $\ell$] = $\sigma_o^{j-\ell\to j}$ | res [j] = $\sigma_t^j$ |
| | | res [j + 26] = $\sigma_{fj}^j$ |
| stfO [j+26] = $(\nu\sigma_f)^j$ | atm [j+26, $\ell$] = $\sigma_1^{j-\ell\to j}$ | res [j + 52] = $\nu$ |
| | | res [j + 78] = $\sigma_e^j$ |
| stfO [j+52] = $\sigma_o^j$ | $(\ell<\min(11,j))$ | res [j + 104] = $\sigma_{s(e)}^j$ |
| stfO [j+78] = $\sigma_1^j$ | | |

The third array is to be used for resonance self-shielding calculation.

Thus, each isotope is represented on the file RFT1 by these three arrays. The only difference between the isotopes is their sequence on RFT1, specified by a serial, or identification number which has to be referred to when the data of any isotope are needed. The sequence of the isotopes is the same as that on the punched tape input of RPGB.

Besides the micro-cross sections, the RFT1 contains five fission spectra /taken from [3]/ and the width of the lethargy intervals for the

26-group system. These data are written into the RFT1 prior to the cross section data.

The order of punching can be read from the listing of the RPGB program. This order has been chosen to simplify the punching operation as much as possible.

The given order of isotopes in the mixture to be calculated is rearranged so that the magnetic tape should move only in the forward direction while searching. The reading-in of the data and the calculation by Eq. /1/ are going on simultaneously, thus the required fast store capacity is practically independent of the number of isotopes in a mixture. Finally, here is the declaration of the algol "group" procedure

```
'procedure' group (r,Pk,n,s,ng0,sg1,nsg,sgt,khi,du,t);
'value' n,s,t; 'integer' n,s; 'real' t;
'array' sg0, sg1, nsg, sgt, khi, r, du;
'integer' 'array' Pk;
'algol';
```

n = the number of isotopes in the mixture
r[1:n]= the density vector of nuclei
Pk[1:n]= the identification numbers of the system components /the order must be the same as in the r[1:n]/;

s = identification number of the fission spectrum to be used /1 - 5/ if s = 0, then no fission spectrum is used.

sg0 → $\Sigma_0$ sg1 → $\Sigma_1$ nsg → $\nu\Sigma_f$ sgt → $\Sigma_t$

khi = is the fission spectrum
du = the width of lethargy intervals
t = is the temperature in Kelvin's. If t = 0, no resonance correction is made.

This declaration is valid in both versions of group.

## 3. RESONANCE SELF-SHIELDING CORRECTION

As known, the flux depression due to resonance absorption gives rise to a change in the group averaged cross sections. This effect depends on the quantity [3] (the group index is dropped)

$$\overline{\sigma}^{(i)} = \frac{1}{\rho_i} \sum_{j=1}^{n} \rho_j \sigma_t^{(j)} - \sigma_t^{(i)} \qquad /8/$$

/when the cross sections of the i-th isotope are to be calculated/. Thus, the resonance-self-shielding correction has to be evaluated separately for each mixture.

For the calculation of resonance self-shielding, correction coefficients are given in the tables [3] as functions of $\bar{\sigma}$ and temperature. The corrected cross sections are obtained as the product of the correction factor f and the tabulated group averaged cross sections. The factor f is given for a few values of $\bar{\sigma}$ /denoted by $\sigma_1 \cdots \sigma_q \cdots$/and at the temperatures 300°, 900° and 2100°K in those cases where Doppler corrections have to be considered.

The interpolation formulae used in the calculation are taken from [4].

Thus, for temperature dependence we take

$$f(T) = A + B/\sqrt{T} \qquad /9/$$

where A is chosen with the assumption that $f(900) = f_{900}$, and B is chosen in the interval $0 < T \leq 900$ with the assumption that $f(300) = f_{300}$ and in the interval $900 \leq T < \infty$ with the assumption $f(2100) = f_{3100}$.

The rule of interpolation concerning the $\bar{\sigma}$ is the following. Let $\bar{\sigma}_L$ be the largest and $\bar{\sigma}_s$ the smallest value of $\bar{\sigma}_q$-s given in the table [3]. Thus

$$\text{if } \bar{\sigma} < \bar{\sigma}_s \text{ then } f = f_s$$

$$\text{if } \bar{\sigma}_s \leq \bar{\sigma}_q \leq \bar{\sigma} \leq \sigma_{q+1} \leq \bar{\sigma}_L \text{ then}$$

$$f(\bar{\sigma}) = f(\bar{\sigma}_q) + \frac{\left(f(\bar{\sigma}_{q+1}) - f(\bar{\sigma}_q)\right) \ln(\bar{\sigma}/\bar{\sigma}_q)}{\ln \bar{\sigma}_{q+1} / \bar{\sigma}_q} \qquad /10/$$

$$\text{if } 10^7 \geq \bar{\sigma} > \sigma_L \text{ then}$$

$$f(\bar{\sigma}) = 1 + \frac{1 - f(\bar{\sigma}_L)}{\ln(10^7 / \bar{\sigma}_L)} \ln(\bar{\sigma} / 10^7)$$

$$\text{if } \bar{\sigma} > 10^7 \text{ then } f = 1.$$

If $\sigma_s = 0$ then we use linear interpolation in the first interval, rather than /10/.

To get a satisfying $\sigma_s$ for every isotope an iteration process would be needed. However the errors in the coefficients f are greater than that arising from the neglection of this iteration procedure. Thus the value given by Eq. /8/ is used for $\bar{\sigma}$.

Upon the calculation of the multigroup macro-constants, the second version of the group procedure can be used to calculate and to apply, in addition, the above correction, too.

This job is also done by making use of the file RFT1 on magnetic tape.

The correction factors are written into the file by a special code called RPGP. They are written after the cross section data of all isotopes. Since not each isotope has to be resonance corrected, a complementary information is written into the RFT1 between the cross sections and the resonance correction factors. This information is contained in an integer array defined as follows: rkor[i] is the identification number of resonance correction factors of the isotope with cross sections identified by the number i.

If the isotope with cross sections identified by number i has no correction factors, then rkor[i] = 0. On the instructions rkor resonance correction is applied by the "group" to all isotopes which have to be corrected without the need of additional information. The sequence of the isotopes in the resonance correction part of the RFT1 file has to be the same as that in the cross section part and is determined by the input sequence on the punched tape for the RPGP program.

The paper tape of resonance correction factors of each isotope is punched as follows.

rk = is the identification number of the first group in which resonance self-shielding is to be taken into account,

rv = is the identification number of the last resonance corrected group,

teta $\begin{cases} = 1 & \text{if no temperature effect is taken into account} \\ = 3 & \text{if it is,} \end{cases}$

srf = the number of columns with fission corrections /srf $\leq$ 4/. If there are no fission corrections, then srf = 0,

src = the number of columns with capture cross section corrections /src $\leq$ 6/. If there are no such corrections, then src = 0,

srt = the number of columns with total cross section corrections /0 < srt $\leq$ 4/,

sre = the number of columns with elastic scattering correction /0 < sre $\leq$ 4/.

After punching these parameters the columns of correction factors are punched one by one from left to right, including the value of $\sigma_q$ -s, too. /Wherever this value is $\infty$ , the punch is $10^7$./

In order to accelerate the operation "group", some transformations of the input data are performed through the code RPGP.

It is advantageous to perform the correction group by group to spare considerable fast store. Therefore, the resonance correction data of any isotope are written by the code RPGP in rv-rk+2 records.

The first record called PAR contains rk, rv, teta, srf, src, srt, sre. The second record called PTR contains $\sigma_q'$ s for $f_f$, $f_c$, $f_t$, $f_e$, /i.e. the second row of the heading of the table of correction factors/. The PAR record has 7 and the PTR record has thus srf+src+srt+sre elements.

The j-rk+1 -th of the other rv-rk+1 records, all called REZKOR contains the following

$$\sigma_t^j,\ \nu^j,\ \sigma_f^j,\ \sigma_e^j,\ \sigma_{s(e)}^j,\ 0.00$$

and if teta = 1, then REZKOR is the j-rk+1 -th row of the table of vorrection factors and if teta = 3, then the three f-s /$f_{2100}$, $f_{900}$, $f_{300}$/ belong to every group. Thus the elements of REZKOR will be tripled.

The first five elements of the j+1-rk -th record are res [j], res [j+26], res [j+52], res [j+78], res [j+104] of the record RES, thus they can be found on the RFT1 and must be used by RPGP program.

The first step in the "group" procedure for resonance correction is to find the records PAR and PTR of all the isotopes involved. Thus we shall know which of the isotopes in a given group must be corrected for resonance self-shielding. Starting with the first group, we find the REZKOR and interpolate, if necessary, according to /9/ and /10/. Then self-shielding corrections for the isotope i can be carried out by the formulae

$$\Sigma_{t\,(corr)}^j = \Sigma_t^j - \rho_i\, \sigma_t^{(i)j} \left(1 - f_t^{(i)j}\,(\bar{\sigma},\ T)\right)$$

$$\nu\,\Sigma_{f\,(corr)}^j = \left(\nu\,\Sigma_f\right)^j - \rho_i\, \nu_i^j\, \sigma_f^{(i)j} \left(1 - f_t^{(i)j}(\bar{\sigma},\ T)\right)$$

$$\Sigma_{o\,(corr)}^j = \Sigma_o^j - \rho_i \left[\left(\sigma_t^{(i)j} - \sigma_e^{(i)j} - \sigma_f^{(i)j}\right)\left(1 - f_c^{(i)j}(\bar{\sigma},\ T)\right)\right.$$

$$\left. - f_f^{(i)j}\,(\bar{\sigma},\ T)\sigma_f^{(i)j} - \sigma_{s(e)}^{(i)j}\, f_e^{(i)j}\,(\sigma,\ T)\right]$$

$$\sum^{j}_{1\,(corr)} = \sum^{j}_{1} - \rho_i \, \sigma_t^{(i)\,j} \left(1 - f_t^{(i)\,j} \left(\bar{\sigma}, T\right)\right)$$

$$\sum^{j\to j+1}_{o\,(corr)} = \sum^{j\to j+1}_{o} - \rho_i \, \sigma_{s\,(e)}^{(i)\,j} \left(1 - f_e^{(i)\,j} \left(\bar{\sigma}, T\right)\right)$$

This operation is performed group by group except for the thermal group.

The effect of resonance self-shielding on anisotropic scattering is neglected.

REFERENCES

[1] G.E. Hansen, W.H. Roach, LAMS-2543

[2] Cross-section Tables /Kernforschungszentrum, Karlsruhe, 1962/.

[3] Abagian et al. Group Constants for Reactor Calculations /Atomizdat, 1964/ /in Russian/

[4] Report BWNL-146.

LIST OF RPGB

```
'begin' 'integer' i,k,l,j,A,B,C;
'integer''array' HK1[1:3];
'array' se,st,sf,nu,me,mz,sz[1:26],
sel,mel,snl,mnl[1:26,0:10],stf0[1:104],atm[1:52,1:10],res[1:130];
'procedure' freemt(n);'value'n;'integer'n;
'external';
'procedure' in(se); 'array'se;
'for' i:=1 'step' 1 'until' 26 'do'
se[i]:=read;
'procedure' beolv(sm); 'array' sm;
'for' i:=0 'step' 1 'until' 10 'do'
'for' j:=1 'step' 1 'until' 25 'do' sm[j,i]:=read;
'procedure' use(n,t); 'value' n; 'integer' n;'string' t; 'external';
'procedure' create(n,t); 'value' n; 'integer' n; 'string' t;
          'external';
'procedure' writearray(n,A,t); 'value' n; 'integer' n;
'array' A; 'string' t; 'external';
create(20,'('RFT1')'); use(20,'('RFT1')');
HK1[1]:=26; HK1[2]:=10; HK1[3]:=37;
writearray(20,HK1,'('HK1')');
selectinput(0);
in(se); writearray(20,se,'('DU')');
'for' k:=1 'step' 1 'until' 5 'do' 'begin'
in(se); writearray(20,se,'('HASP')') 'end';
'for' k:=1 'step' 1 'until' HK1[3] 'do' 'begin'
A:=read; B:=read; C:=read;
'for' i:=1 'step' 1 'until' 26 'do' 'begin'
se[i]:=st[i]:=sf[i]:=nu[i]:=me[i]:=mz[i]:=sz[i]:=0;
'for' j:=0 'step' 1 'until' 10 'do'
sel[i,j]:=mel[i,j]:=snl[i,j]:=mnl[i,j]:=0; 'end';
'for' i:=1 'step' 1 'until' 52 'do'
'for' j:=1 'step' 1 'until' 10'do'
atm[i,j]:=0;
in(st);
'if' C>0 'then' 'begin' in(sf); in(nu) 'end';
in(se);
'if' A=0 'then' in(me);
in(sz);
'if' A=0 'and' B>0 'then' in(mz);
'if' A=1 'then' 'begin' beolv(sel); beolv(mel) 'end';
beolv(snl);
'if' B=1 'then' beolv(mnl);
'for' i:=1 'step' 1 'until'26 'do' 'begin'
res[i]:=stf0[i]:=st[i];
res[i+26]:=sf[i];
res[i+52]:=nu[i];
stf0[i+26]:=nu[i]*sf[i];
res[i+78]:=se[i];
res[i+104]:=sz[i];
stf0[i+52]:=st[i]-se[i]+sz[i]-snl[i,0];
stf0[i+78]:=st[i]-me[i]*se[i]*(1-A)+sz[i]*mz[i]*(1-A)*B
          -sel[i,0]*mel[i,0]*A-mnl[i,0]*snl[i,0]*B;
'for' l:=1,l+1 'while' l 'le' 10 'and' l<i 'do' 'begin'
atm[i l]:=atm[i,l]+sel[i-1,l]*A+snl[i-1,l];
'if' A=1 'or' B=1 'then'
atm[i+26,l]:=atm[i+26,l]+mel[i-1,l]*sel[i-1,l]*A+mnl[i-1,l]*snl[i-1,l]*B
 'end';
'if' i<26 'and' A=0 'then' 'begin'
atm[i+1,l]:=atm[i+1,l]+sz[i]; 'if' B=1 'then'
atm[i+27,l]:=atm[i+27,l]+mz[i]*sz[i] 'end' 'end' i;
writearray(20,stf0,'('STF0')');   writearray(20,atm,'('ATM')');
writearray(20,res,'('RES')') 'end' k; freeinput; freemt(20) 'end' RPGB;
```

LIST OF RPGP

```
'begin' 'integer'i,j,k;
'integer' s;
'integer''array' par[1:7],rkor[1:40],dr[0:50];
'array'ptr[1:14],sm[1:14,0:50], res[1:130];
'procedure' dataskip(n); 'value' n;
'integer' n; 'external';
'procedure' freemt(n); 'value' n; 'integer' n;
'external';
'procedure' use(n,t);'value' n;
'integer' n;'string' t; 'external';
'procedure' readarray(n,A,t); 'value' n;
'integer' n; 'array'A; 'string' t; 'external';
'procedure' writearray(n,A,t); 'value' n;
'integer'n; 'string't; 'array'A; 'external';
'procedure' skip(n,m); 'value'n,m;
'integer' n,m; 'external';
'procedure' rewind(n); 'value' n;'integer' n; 'external';

select input(0); s:=read;
'for' i:=1 'step' 1 'until' 37 'do' 'begin'
rkor[1]:=read; dr[rkor[1]]:=1 'end':
use(20,'('RFT1')');
readarray(20,rkor,'('RKOR')');
skip(20,-1);
writearray(20,rkor,'('RKOR')');
'for' k:=1 'step' 1 'until' s 'do' 'begin'
  rewind(20);
'for' i:=1 'step' 1 'until' 7 'do'
par[1]:=read;
'for' i:=1 'step' 1 'until' par[4]+par[5]+par[6]+
par[7] 'do'
'for' j:=0 'step' 1 'until' par[3]*(1+par[2]-par[1]) 'do'
sm[i,j]:=read;
'begin' 'array' rezkor[1:6+par[3]*(par[4]+par[5]+par[6]+par[7])];
'for' j:=1 'step' 1 'until' d[k]'do' 'begin'
skip(20,2);
readarray(20,res,'('RES')') 'end';
'for' i:=1 'step' 1 'until' par[4]+par[5]+par[6]+par[7] 'do'
 ptr[1]:=sm[1,0];
 dataskip(20);
writearray(20,par,'('PAR')');
writearray(20,ptr,'('PTR')');
'for' j:=par[1]'step' 1 'until' par[2] 'do' 'begin'
rezkor[1]:=res[j];
rezkor[2]:=res[j+26];
rezkor[3]:=res[j+52];
rezkor[4]:=res[j+78];
rezkor[5]:=res[j+104];
rezkor[6]:=0;
'for' i:=7 'step' 1 'until' 6+par[3]*(par[4]+par[5]+par[6]
+par[7]) 'do'
rezkor[i]:=sm[(i-7)'/'pa[3]+1,par[3]*(j+1-par[1])-1+7-
par[3]*((7-1)'/'par[3])];
writearray(20,rezkor,'('REZKOR')');
'end'j ; 'end';
'end'; feeinput; freemt(20);
'end' RPGP;
```

LIST OF "GROUP"

```
'procedure' group(r,Pk,n,s,sg0,sg1,nsg,sgt,khi,du,t);
'value' n,s,t; 'integer' n,s; 'real' t;
'array' sg0, sg1,nsg,sgt,khi,du,r; 'integer' 'array' Pk;
'begin' 'integer' g,i,j,k,l,m;
'array' ro[1:n]; 'integer' 'array' P[0:n],HK1[1:3];
'procedure' skip(n,m); 'value' n,m; 'integer' n,m; 'external';
'procedure' readarray(n,A,t); 'value' n; 'integer' n;
'array' A; 'string' t; 'external';
'procedure' rewind(n); 'value'n;
'integer' n; 'external';
'procedure' input(n,s);
   'value' n; 'integer' n; 'string' s; 'external';
input(20,'('RFT1')');
readarray(20,HK1,'('HK1')'); readarray(20,du,'('DU')');
g:=HK1[1]; m:=HK1[2];
'if' s=0 'then'
'for' k:=1 'step' 1 'until' g 'do'
khi[k]:=1-sign(k-1)'else'
'begin' skip(20,s-1);
readarray(20,khi,'('HASP')') 'end'; skip(20,5-s);
'for' i:=1 'step' 1 'until' g 'do' 'begin'
sgt[i]:=nsg[i]:=0;
'for' j:=0 'step' 1 'until' m 'do'
sg0[i,j]:=sg1[i,j]:=0 'end';
'for' k:=n 'step' -1 'until' 1 'do' 'begin'
l:=0;
'for' i:=1 'step' 1 'until' n 'do'
'if'Pk[i]>l 'then''begin' l:=Pk[i]; j:=i 'end';
P[k]:=l; ro[k]:=r[j]; Pk[j]:=0; 'end'; P[0]:=0;
'begin' 'array' stf0[1:4*g],atm[1:2*g,1:m];
'for' k:=1 'step' 1 'until' n 'do' 'begin'
i:=3*(P[k]-P[k-1]-1); skip(20,i);
readarray(20,stf0,'('STF0')');
readarray(20,atm,'('ATM')');
skip(20,1);
'for' i:=1 'step' 1 'until' g 'do' 'begin'
sgt[i]:=sgt[i]+ro[k]*stf0[i];
nsg[i]:=nsg[i]+ro[k]*stf0[i+g];
sg0[i,0]:=sg0[i,0]+Ro[k]*stf0[i+2*g];
sg1[i,0]:=sg1[i,0]+ro[k]*stf0[i+3*g];
'for' l:=1,l+1 'while' l 'le' m 'and' l<i 'do' 'begin'
sg0[i,l]:=sg0[i,l]+ro[k]*atm[i,l];
sg1[i,l]:=sg1[i,l]+ro[k]*atm[i+g,l] 'end' 'end' i;
'end' k; rewind(20); 'end';
'if' t=0 'then' 'goto' VEG;
REZ:'begin''integer''array' par[1:7],rkor[1:40],dr[0:30];
'array' ptr[1:14],inp1,inp2,inp3[1:6];
      'integer' max,mk,lk,q; 'array' rz[1:30];
readarray(20,rkor,'('RKOR')');
'for'i:=37'step'-1'until'2'do'
rkor[i]:=rkor[i-1];
rkor[1]:=
      max:=dr[0]:=l:=0;
'for' i:=1 'step' 1 'until' n 'do'
   'if' rkor[P[i]]>0'then''begin'
l:=l+1; rz[l]:=ro[i];
   dr[l]:=rkor[P[i]] 'end';
RB:'begin''integer''array' rk,rv,srf,src,srt,sre,teta[1:l];
 'array'int[1:20,1:l];
'real' elt;
```

```
'boolean''array' bb[1:l+1];
   bb[l+1]:='false';
  'for' j:=1 'step' 1 'until' l 'do''begin'
  rkor[j]:=0;
  'for' k:=dr[j-1]+1'step' 1 'until' dr[j] 'do''begin'
   R2:readarray(20,par,'('PAR')');
readarray(20,ptr,'('PTR')');
skip(20,1+par[2]-par[1]);
rkor[j]:=rkor[j]+3+par[2]-par[1] 'end' k;
   max:=max+rkor[j];
  rk[j]:=par[1]; rv[j]:=par[2]; teta[j]:=par[3];
  srf[j]:=par[4]; src[j]:=par[5];
srt[j]:=par[6]; sre[j]:=par[7];
  'for' k:=1 'step' 1 'until' 14 'do'
    int[k,j]:=ptr[k];
  'end'j; 'for' i:=1 'step' 1 'until' g-1 'do''begin'
  skip(20,-max);
mk:=1;
'for'lk:=1 'step'1'until'l'do''begin'
bb[lk]:=(1-rk[lk])*(rv[lk]-1) 'ge' 0;
  'if' bb[lk] 'then' mk:=mk+1'end';
RC:'begin''array' ff,ft,fe[1:4,1:mk],fc[1:6,1:mk],
                  nu,st,sc,se,sz,sf[1:mk];
  j:=0;
  'for' lk:=1 'step' 1 'until' l 'do'
  'begin''array' rezkor[1:6+teta[lk]*(srf[lk]+src[lk]+srt[lk]+sre[lk])];
    q:='if' bb[lk] 'then' rkor[lk]-1-rv[lk]+i 'else' rkor[lk];
  RD:skip(20,q);
  'if' bb[lk]'then''begin'
       readarray(20,rezkor,'('REZKOR')');
       j:=j+1;
       'if' teta[lk]=3'then'
'begin'
K2:'for' k:=7 'step' 3 'until' 3*(srf[lk]+src[lk]+srt[lk]+sre[lk]) 'do'
        rezkor[k]:='if' t 'le' 900 'then'
1/(sqrt(3)-1)*(sqrt(3)*rezkor[k+1]-rezkor[k+2]+30/sqrt(t)*
(rezkor[k+2]-rezkor[k+1])) 'else'
 1/(1-sqrt(3/7))*(rezkor[k]-sqrt(3/7)*rezkor[k+1]+30/sqrt(t)*
 (rezkor[k+1]-rezkor[k]));
'end';
sc[j]:=st[j]:=rezkor[1];
        sf[j]:=rezkor[2];
        nu[j]:=rezkor[3];
        se[j]:=rezkor[4];
        sz[j]:=rezkor[5];
        K3:'for' k:=1 'step' 1 'until' srf[lk] 'do'
             ff[k,j]:=rezkor[7+(k-1)*teta[lk]];
       K4:'for' k:=1 'step' 1 'until' src[lk] 'do'
            fc[k,j]:=rezkor[7+teta[lk]*(srf[lk]+k-1)];
      K5:'for' k:=1 'step' 1 'until' srt[lk] 'do'
            ft[k j]:=rezkor[7+teta[lk]*(srf[lk]+src[lk]+k-1)];
      K6:'for' k:=1 'step' 1 'until' sre[lk] 'do'
            fe[k,j]:=rezkor[7+teta[lk]*(srf[lk]+src[lk]+srt[lk]+k-1)];
skip(20,rv[lk]-1)'end';
  'end' lk:
 'begin''real''procedure' insig(sr,kf,inp,sig,j);
   'value'j,sr,sig; 'real' sig:
'integer' j,sr;
'array' inp;
   'array' kf;
   'begin''integer' k; insig:=1;
   'for' k:=1 'step' 1 'until' sr-1 'do'
'begin'
```

```
   'if' sig<inp[k]'and' sig 'ge' inp[k+1]'then'
'begin''if'inp[k+1]>0 'then'
  insig:=kf[k+1,j]+(kf[k,j]-kf[k+1,j])*ln(sig/inp[k+1])/
ln(inp[k]/inp[k+1]) 'else'
insig:=kf[k+1,j]+(kf[k,j]-kf[k+1,j])*sig
/inp[k]
'end' 'else'
 'if' sig>inp[1] 'and' sig<10000000 'then'
 insig:=1+(kf[1,j]-1)*ln(sig/10000000)/ln(inp[1]/10000000)
'else' 'if' sig 'ge' 10000000 'then' insig:=1 'else'
  'if' sig<inp[sr] 'then'
insig:=kf[sr,j] 'end'insig;
'end';
   j:=lk:=0;
'for'lk:=1'step'1'until'l'do'
'if'bb[lk]'then''begin'
  j:=j+1;
        K8: 'for' k:=1 'step' 1 'until' srt[lk] 'do'
            inp1[k]:=int[k+srf[lk]+src[lk],lk];
            sgt[1]:=sgt[1]/rz[lk]-st[j];
            st[j]:=st[j]*insig(srt[lk],ft,inp1,sgt[1],j);
            sgt[1]:=rz[lk]*(sgt[1]+st[j]);
'end';
      ITE: j:=lk:=0;
'for'lk:=1'step'1'until'l'do'
'if'bb[lk]'then''begin'
    j:=j+1;
        K9: 'for' k:=1 'step' 1 'until' 6 'do''begin'
          inp1[k]:=int[k,lk];
          inp2[k]:=int[k+srf[lk],lk];
          inp3[k]:=int[k+srf[lk]+src[lk]+srt[lk],lk]
         'end'k;
      elt:=sgt[1]/rz[lk]-st[j];
      nsg[1]:=nsg[1]-rz[lk]*nu[j]*sf[j]*(1-insig(srf[lk],ff,inp1,elt,j));
      sg0[1,0]:=sg0[1,0]-rz[lk]*((sc[j]-se[j]-sf[j])*(1-insig(src[lk],
fc,inp2,elt,j))+sf[j]*(1-insig(srf[lk],ff,inp1,elt,j))
+sz[j]*(1-insig(sre[lk],fe,inp3,elt,j)));
      sg1[1,0]:=sg1[1,0]-rz[lk]*(sc[j]-st[j]);
      sg0[i+1,1]:=sg0[i+1,1]-rz[lk]*sz[j]*(1-insig(sre[lk],fe,inp3,elt,j));
     'end'lk;'end';'end'RC;'end'1,'end'RB;'end'REZ;
rewind(20);
VEG:'end';
```

Printed in the Central Research Institute for Physics, Budapest
Kiadja a KFKI Könyvtár- és Kiadói Osztály
o.v.: Dr. Farkas Istvánné

Szakmai lektor: Szatmáry Zoltán
Nyelvi lektor: Monori Jenőné

Példányszám, 135 Munkaszám: 3753 Budapest, 1968. junius 11.