EML-517

Environmental Measurements Laboratory

PERSONAL COMPUTER PROGRAMS FOR THE USE IN RADON\THORON PROGENY MEASUREMENTS

Earl O Knutson

March 1989

DEPARTMENT OF ENERGY

NEW YORK, N. Y. 10014

PERSONAL COMPUTER PROGRAMS FOR USE IN RADON/THORON PROGENY MEASUREMENTS

EML--517

DE89 011072

Earl O. Knutson

Environmental Measurements Laboratory
U.S. Department of Energy
376 Hudson Street, New York, NY 10014-3621

March 1989

Printed in the United States of America

Available from

National Technical Information Service
U. S. Department of Commerce
5285 Port Royal Road
Springfield, VA 22161

NTS price codes: A05
Printed copy: $8.00
Microfiche copy: A01

[illegible]

ABSTRACT

Source listings and program notes are given for five programs for reducing data from measurements on radon and thoron progeny. Three of these programs provide for calculating radon and thoron progeny concentrations from gross alpha counting of deposits on filters, by three different methods. The remaining two programs are for calculating aerosol particle size (or diffusion coefficient) distributions from diffusion battery (or graded screen) apparatus.

One program is written in GW BASIC and the remainder are written in Borland's Turbo Pascal. The target machine for these programs is the IBM (or compatible) personal computer (PC). Portable versions of the PC also make it feasible to run these programs in the field.

CONTENTS

PREFACE

This report presents the source listings of five personal computer programs developed at the Environmental Measurements Laboratory (EML) for the reduction of data from radon/thoron progeny measurements. The report does not cover apparatus, sampling strategy or sampling procedure; these topics will be covered in the new edition of the EML Procedures Manual (in press). Neither does it cover the mathematical underpinning of the five programs; this information is given as reference citations.

In spite of a determined effort to modularize the programs and to embed comments, reading the listings themselves is still a chore. Therefore, the listings are placed in a series of eleven appendices to this report, and the main part of the report provides a narrative background and a guide to the programs. Also included in the main part are examples of input and output files.

Some of the terminology in this report is likely to appear cryptic because the programs were developed in support of specific apparatus. Unavoidably, terminology (such as screens and disks) related to specific apparatus appears in the programs. All we can do is to ask the reader to be alert to this, and to watch the context in which the terms appear.

Systeme Internationale (SI) units have been used. Also, these programs use the latest published values for the decay constants and decay energies of alpha radiations from the progeny.

Although the programs published here are quite new, they have already had considerable use and testing at EML. To the best of our knowledge they produce correct results, but it is still possible that some errors have gone undetected. Please inform the author of any errors you may find in these programs.

All five programs are available from EML in the form of MS-DOS files on diskette.

Thanks to Edward F. Maher of Brooks Air Force Base who kindly supplied a FORTRAN listing, dated December 1983, which was very helpful in developing three of these programs. I am grateful to my EML colleagues Andreas C. George and Keng W. Tu for helping to test these programs, and to Ferenc Hajnal for his careful review of this work.

INTRODUCTION

Concentration and particle size are the two aspects of airborne radon progeny that are most important to health effects. Measurement of concentration entails drawing an air sample through a suitable filter, counting the alpha activity from the front face of this filter for selected time intervals, then making an appropriate calculation. A number of algorithms, ranging widely in complexity, are available for the calculation. Until recently, only simple algorithms could be used in field work. The Kusnetz method and the modified Tsivoglou method, described in the EML Procedures Manual (in press), are two of the most commonly used.

Measurement of radon progeny particle size entails several (usually five) simultaneous measurements of concentration using size-selective samplers, followed by an "unfolding" step in which the size distribution is inferred from these concentrations. Again, various algorithms are available, all of which require some form of computer.

New, portable computers make it feasible to do complex data reduction calculations in the field, which previously had to be done in the laboratory on mainframe or mini-computers. However, feasibility becomes reality only when reliable, debugged, documented computer programs are available that will run on these machines. To this end we have invested considerable effort in transcribing old programs and programming new algorithms for personal computers.

Five different programs have resulted from this effort, one written in GW BASIC and the others in Borland's Turbo Pascal. The five programs are:

WWN.Pas - for use in calculating radon decay product (^{218}Po, ^{214}Pb, ^{214}Bi) concentrations from gross alpha counts from any three non-overlapping time intervals;

RWRENNGW.BAS - for calculating radon decay product and, optionally, thoron decay product concentration from gross alpha counts taken at equal time intervals (the algorithm is entirely different from that in the above program);

ExMaxDP.Pas - for the same purpose as RWRENNGW, but based on yet another algorithm (uses the same input data as RWRENNGW.BAS);

ExMaxDB.Pas - for unfolding diffusion battery data to yield aerosol particle size spectra;

ExMaxGS.Pas - for unfolding data from the "graded wire screen" apparatus to yield diffusion coefficient spectra.

MATHEMATICAL COMMENTS

The first program, WWN.Pas, is relatively straightforward in its logic; it is based on the concise set of equations given by Nazaroff (1984). The second program is an implementation of the weighted least squares procedure, clearly described in the already-classic paper of Raabe and Wrenn (1969).

The last three programs are based on an iterative procedure called the expectation maximization algorithm. Maher and Laird (1985) were the first to apply this algorithm to aerosol measurements, after which it has steadily gained followers in the aerosol and radon/thoron progeny measurement community.

It is not within the scope of this report to go into the mathematical background of the individual programs. For example, in two of the programs there is a preliminary step in which integrals are replaced by summations. [In these programs we use the simplest method - the midpoint rule of numerical integration. See Maher and Laird (1985) for a full explanation.] Suffice it here to say that, after these preliminary steps, there is a great deal of similarity among the five programs.

All five programs hinge on solving a matrix "equation"

TRANSFORM*DATA "=" KERNEL*SPECTRUM

in which

DATA is a vector of measured values,
TRANSFORM is matrix which reformats the DATA,
KERNEL is the "response" matrix, and
SPECTRUM is the vector to be determined.

The asterisks indicate matrix product. The two vectors do not necessarily have the same dimensions, and the two matrices are not necessarily square. The equals sign in the equation is placed in quotation marks as a reminder that, due to random and systematic errors, the two sides can never be exactly equal.

In each program, KERNEL is calculated at the outset from first principles and from the known characteristics of the measuring device. In three of the programs, TRANSFORM is an identity matrix, so it could have been left out of the equation.

The expectation maximization algorithm will be described briefly since it is relatively new in aerosol science and radon/thoron progeny measurement technology. This algorithm, EM for short, consists of the iterative use of the equation:

NEWITERATE = EXMAXMATRIX*OLDITERATE

where

OLDITERATE and NEWITERATE (both vectors) are successive iterates in a sequence that will hopefully converge to the desired vector, SPECTRUM;

EXMAXMATRIX is a diagonal matrix whose j-th diagonal element is given by

$$\frac{\sum_i (D_i/C_i) K_{ij}}{\sum_i K_{ij}}$$

D_i is the i-th component of the vector TRANSFORM*DATA;

C_i is the i-th component of the vector KERNEL*OLDITERATE;

K_{ij} is the i,j element of KERNEL.

Thus, the j-th component of OLDITERATE is multiplied by a weighted average of the ratios D_i/C_i to produce the new estimate of that component. The weighting factors are the elements in the j-th column of KERNEL. This equation is applied repeatedly until a certain stopping criterion is met.

EM is an intuitively appealing algorithm which has appeared in various contexts. In his book on inversion mathematics, Twomey (1977) lists it as equation 7.20 and attributes it to Cahine, whose application was radiative transfer in the atmosphere. Doroshenko et al. (1977) showed how it can be derived from Bayes theorem and applied it to neutron energy spectrometry. The algorithm was also examined in a Ph.D. thesis on aerosol measurements by Kapadia (1980), who called it the nonlinear iteration algorithm II.

EM was given a much broader significance in 1977 by Dempster et al. (1977), who showed that EM converges to the solution of a certain class of statistical problems. Specifically, if the components of TRANSFORM*DATA are Poisson random variables (and certain other conditions are met), then EM yields that SPECTRUM which maximizes the likelihood of the observed TRANSFORM*DATA.

The EM programs given here were inspired by the paper of Maher and Laird (1985). Maher kindly supplied a FORTRAN listing which was very helpful in developing these programs.

SOURCE LANGUAGE AND EXECUTABLE FILES

Table 1 shows the name, size and date of the pertinent source files. The five programs already mentioned are also included, and are described in more detail later in this report. One file is an ordinary text file, and the five remaining files are "units" (a Turbo Pascal construct which can be used to modularize programs) which are shared among the four Pascal programs. The files are:

Aerosol.Pas - contains several procedures for calculating aerosol penetration through screens or diffusion battery stages.

Algorith.Pas - contains four procedures: ForwardCalc, which performs the matrix multiplication KERNEL*SPECTRUM; ExpectMax, which performs the EM algorithm; StandardErr, which does the error propagation step following the EM calculation; and TwomeyAlg. The latter performs a somewhat enhanced version of Twomey's nonlinear iteration algorithm (Twomey, 1977; Twomey, 1975), another widely used method for inverting the basic matrix equation. It is included primarily for easy comparison.

DataMess.Pas - contains a prompting message concerning the format of input data for both ExMaxDB and ExMaxGS.

Globals.Pas - definition of certain variables used globally in programs and other units;

Mtrx.Pas - a procedure for inverting square matrices up to 10 x 10. This procedure, which was transcribed from a BASIC program by Flynn (1981), features positioning for size of both rows and columns.

RWRENN.TXT - The purpose of this file, which contains ordinary text rather than the source code, is to give a brief explanation of the program RWRENNGW.BAS. It can be shown from within the latter program provided that the MSDOS PATH statement is properly set up.

In addition, several of the programs and units make use of the standard Turbo Pascal units Dos and Crt.

Table 2 shows the names, sizes and dates of the five executable files. Included is the file RWRENNGW.EXE, which was generated by applying the compiler Turbo Basic to RWRENNGW.BAS. Once you have these programs on your active disk drive, they can be started by simply typing the name of the file (".EXE" need not be included).

PROGRAM NOTES

WWN.Pas

This program is a direct (non-iterative) procedure based on the concise set of equations given by Nazaroff (1984). The vectors DATA and SPECTRUM have three components and KERNEL is a 3 x 3 square matrix. TRANSFORM is a 3 x 3 identity matrix. SPECTRUM is obtained directly by inverting the KERNEL and premultiplying DATA by this inverse. As a final step, uncertainties in the components of SPECTRUM are calculated by propagating the estimated uncertainties in DATA through the inverted KERNEL.

The program prompts the user for all the necessary input, which must be entered via the keyboard. The program is capable of dealing with the case where counting is started during sampling. Therefore, when requested to enter the start and stop times of the count intervals, the user must enter these as measured from the beginning of sampling. For example, if counting is started simultaneously with sampling, the start time for the first count interval would be entered as 0.

Figure 1 shows an input screen for WWN.Pas. The underlines indicate where user responses are required. "Thomas protocol" refers to the case where counting is done 2-5, 6-20, and 21-30 min after the end of sampling. The calculation results are shown on the screen and, optionally, can be printed out.

RWRENNGW.BAS

This program is an implementation of the weighted least squares procedure described in the paper of Raabe and Wrenn (1969). Although more complicated than WWN.Pas, it is also a non-iterative procedure. The experimental uncertainties in DATA are embedded in the matrix TRANSFORM and used as the least squares weighting factors. The uncertainties in SPECTRUM are developed as an integral part of the calculation.

The logic, even the nomenclature, of this program closely follows the paper by Raabe and Wrenn (1969). However, the very last step of the calculation - propagating the error into the air concentrations - was not spelled out in the paper. The equations that were used to do this are described in lines 1720-1800 of the listing.

Figure 2 shows how to organize the input data, including the 19 input parameters, that are needed to run this program. Figure 3 shows the printout of a disk file containing two blocks of data ready for processing.

When the program has been started, the user is asked if he needs instructions. By answering "Y", the information in Appendix J will be shown on the screen [provided that the file RWRENN.TXT is on the disk, and that the proper PATH statements have been set up in the computer - if there is any trouble, just consult Appendix J or Figure 2]. Next the program asks for the name of the file containing the input data, and for another file name for the output of the calculation. As the programs runs, the results are shown on the screen as well as written to the output file.

Figure 4 shows a printout of the file which was produced by running the data shown in Figure 3.

ExMaxDP.Pas

Since radioactive decay, to a good approximation, follows Poisson statistics, the EM algorithm should be well suited for the following problem: "given a set of gross alpha counts obtained in equal time intervals following sampling, find the combination of radon progeny concentrations that maximizes the likelihood of having observed those counts." That is what ExMaxDP.Pas does.

ExMaxDP is designed to accept input data in the same format as RWRENNGW.BAS, already discussed. However, the six single-digit parameters that follow the counter efficiencies are ignored by ExMaxDP. Instead, ExMaxDP calculates for thoron progeny if and only if the count data spans at least 300 min. It always calculates for radon progeny.

Although the RWRENNGW and ExMaxDP are entirely different, experience shows that they produce nearly the same results. Typically, the largest differences, often about 10%, are found for ^{218}Po. The uncertainty terms in the two calculations show similar agreement. One difference is that ExMaxDP never produces negative concentrations for the three (or five) nuclides.

ExMaxDB.Pas

ExMaxDB is for reducing data from the diffusion battery, yielding particle size spectra for the aerosol sampled. More specifically, the program is written for use with any of four EML diffusion batteries. The spectra obtained are most reliable in the size range from 10 to 200 nm.

In our application, the input data for ExMaxDB come from a condensation nucleus counter, or from applying one of the three preceding programs to data from radioactivity measurements. Poisson statistics does not govern these input data,* so the underlying premise of the algorithm is not met. We use the program anyway, confident that it will generate something close to the maximum likelihood solution for our input data.

Since the Poisson distribution does not govern the input data in our case, we think it is necessary to radically alter the way in which random errors are estimated and propagated. The following rules are used:

1. If during keyboard input, the vector SPECTRUM is given more components than the vector TRANSFORM*DATA, error propagation is suspended; and

2. If the uncertainties in the input data are known and are included in the input file, they are used in propagating errors.

Beyond these rules, the user is given three choices for propagating errors:

1. The error propagation may be omitted altogether;

2. The input data can be simply accepted as Poisson variables; and

3. The input data may be rescaled into an "equivalent number of counts" (no larger than 32767), which are then used as though they are actual Poisson-distributed count data.

Input data for this program are taken in part from the keyboard and in part from a disk file. Figure 5 shows the structure required for the disk file. This information will appear on the screen if the first prompt is answered y or Y.

Figure 6 shows the stream of prompts that appear after starting the program. Figure 7 shows an example input data file, and Figure 8 gives an example of the output.

No provision is made to correct for losses between diffusion battery stages, or to correct for the efficiency characteristics of the aerosol detector. If needed, such corrections must be made separately.

**Suppose that a measurement with a nucleus counter gave the result 10,000 particles cm^{-3}. If we apply the "square root of n" rule from Poisson statistics, we would calculate the uncertainty in this measurement due to random error to be 100 particles cm^{-3}, or 1%.. Our experience tells us that the estimate is too small, proving that Poisson is not the correct statistics for the condensation nucleus counter.*

In the case of radioactivity measurements, Poisson statistics applies only to the raw count data. For the refined data that are used as input to ExMaxDB, uncertainty estimates must take into account the volume of air sampled, and other factors. This is done in each of the three preceding programs, so we prefer to use those uncertainty estimates in place of the "square root of n" approach.

ExMaxGS.Pas

The program deals with a technique called the "graded screen" method, which is aimed at measuring the diffusion coefficient spectrum of radon/thoron progeny particles in the regime below about 25 nm. [We could also say size spectrum, but the diffusion coefficient is a more fundamental quantity in this regime than is particle size.] In brief, the technique involves drawing an air sample through different grades of wire screen, then alpha-counting these screens as though they were filters. The number of screens is normally 3 or 4, and during sampling, they may be arranged either in series or side-by-side. The apparent concentration detected by each screen is used as input for the unfolding calculation. For further information, consult Holub and Knutson (1987) and Holub et al. (1987).

Like ExMaxDB, input data for ExMaxGS is taken partly from the keyboard and partly from a disk file. Figure 9 shows that part of the prompt sequence that is different from ExMaxDB.

It should be mentioned that no provision is made in ExMaxGS for losses between screens (if in series) or in the inlet section before the screens.

RELATIONSHIPS AMONG PROGRAMS

As already pointed out, the first three of the programs presented here have the same purpose: to calculate radon decay product concentrations from sequential gross alpha counts of particle deposits on air-sampling filters. Explanations are given below as to why three separate programs are necessary.

The first program, **WWN.Pas**, is for use with the three-count method of radon progeny measurements. This method is particularly useful in field sampling since it can be done with relatively simple apparatus and the data can be recorded manually. WWN can be used for Thomas-modified Tsivoglou data, which is the most common of the three-count protocols. In addition, WWN can be used when the Thomas counting intervals have been changed either by accident or by plan.

The second program, **RWRENNGW.BAS**, has a better basis in statistical theory than does WWN, and is the one we use most frequently in research applications. The disadvantage of this program is that it requires more elaborate input data (we commonly use 40 consecutive 1-min counts), and therefore more elaborate data recording equipment. Another difference between RWRENNGW and WWN is that the former can be used to analyze for thoron progeny. This is a user option which we normally use when we have more than 5 hours worth of count data.

The third program, **ExMaxDP.Pas**, is even better in terms of the theory of statistics. It is based on an algorithm that we regard as the ultimate for calculating radon/thoron decay product concentrations. As a penalty, it is an iterative calculation that can consume more time than RWRENNGW. For easy comparison, it is designed to read the same data files as RWRENNGW. The program is quite new and comparisons done so far show the two yield results differing by as much as 10%.

The last two programs, **ExMaxDB.Pas** and **ExMaxGS.Pas**, are closely related. These two are candidates for a merger, when and if the the graded screen method has been proven to be useful.

There is also a serial relationship between certain pairs of these programs in that output from one becomes input data for another. For example, in our measurements using the diffusion battery, we commonly use RWRENNGW followed by ExMaxDB. The former is applied to five blocks of data, yielding five blocks of output in a particular format. This must be reorganized into four blocks, in a different format, for input to ExMaxDB. To do this reorganization, we rely on a standard text editor and on short "reformatting" programs.

REFERENCES

Dempster, A. P., N. M. Laird, and D. B. Rubin
"Maximum Likelihood from Incomplete Data via the EM Algorithm"
J. of the Royal. Statistical Society, 39, 1-38 (1977)

Doroshenko, J. J., S. N. Kraitor, T. V. Kuznetsova, K. K. Kushnereva, and E. S. Leonov
"New Methods for Measuring Neutron Spectra with Energy from 0.4 eV to 10 MeV"
Nuclear Technology, 13, 296-304 (1977)

EML Procedures Manual
USDOE Report EML-300, 27th Edition, Section 2 (in press)

Flynn, B. J.
"Inverting an Matrix"
Compute!, 3, 66-70 (1981)

George, A. C. and E. O. Knutson
"Measurement of Radon and Thoron Progeny"
in: ***EML Procedures Manual***
USDOE Report EML-300, 27th Edition, Section 2 (in press)

Holub, R. F. and E. O. Knutson
P. K. Hopke, Editor
"Measurement of ^{218}Po Diffusion Coefficient Spectra Using Multiple Wire Screens"
Proceedings of the Symposium on Radon and Its Decay Products: Occurrence, Properties and Health Effects, Symposium Series 331, American Chemical Society, Washington DC, pp. 340-356 (1987)

Holub, R. F., E. O. Knutson, and S. B. Solomon
"Tests of the Graded Wire Screen Technique for Measuring the Amount and Size of 'Unattached' Radon Progeny"
Rad. Prot. Dosim., accepted for publication (1988)

Kapadia, A.
"Data Reduction Methods for Aerosol Size Distribution Measuring Techniques"
Ph.D. Thesis, University of Minnesota (1980)

Maher, E. F. and N. M. Laird
"EM Algorithm Reconstruction of Particle Size Distributions from Diffusion Battery Data"
J. Aerosol Science, 16, 557-570 (1985)

Nazaroff, W. W.
"Optimizing the Total-Alpha Three-Count Technique for Measuring Concentrations of Radon Progeny in Residences"
Health Physics, 46, 395-405 (1984)

Raabe, O. G. and M. E. Wrenn
"Analysis of the Activity of Radon Daughter Samples by Weighted Least Squares"
Health Physics, 17, 593-605 (1969)

Twomey, S.
"Comparison of Constrained Linear Inversion and an Iterative Nonlinear Algorithm Applied to the Indirect Estimation of Particle Size Distributions"
J. Comp. Phys., 18, 188-200 (1975)

Twomey, S.
"Introduction to the Mathematics of Inversion in Remote Sensing and Indirect Measurements"
Elsevier Scientific Publishing, New York (1977)

TABLE 1

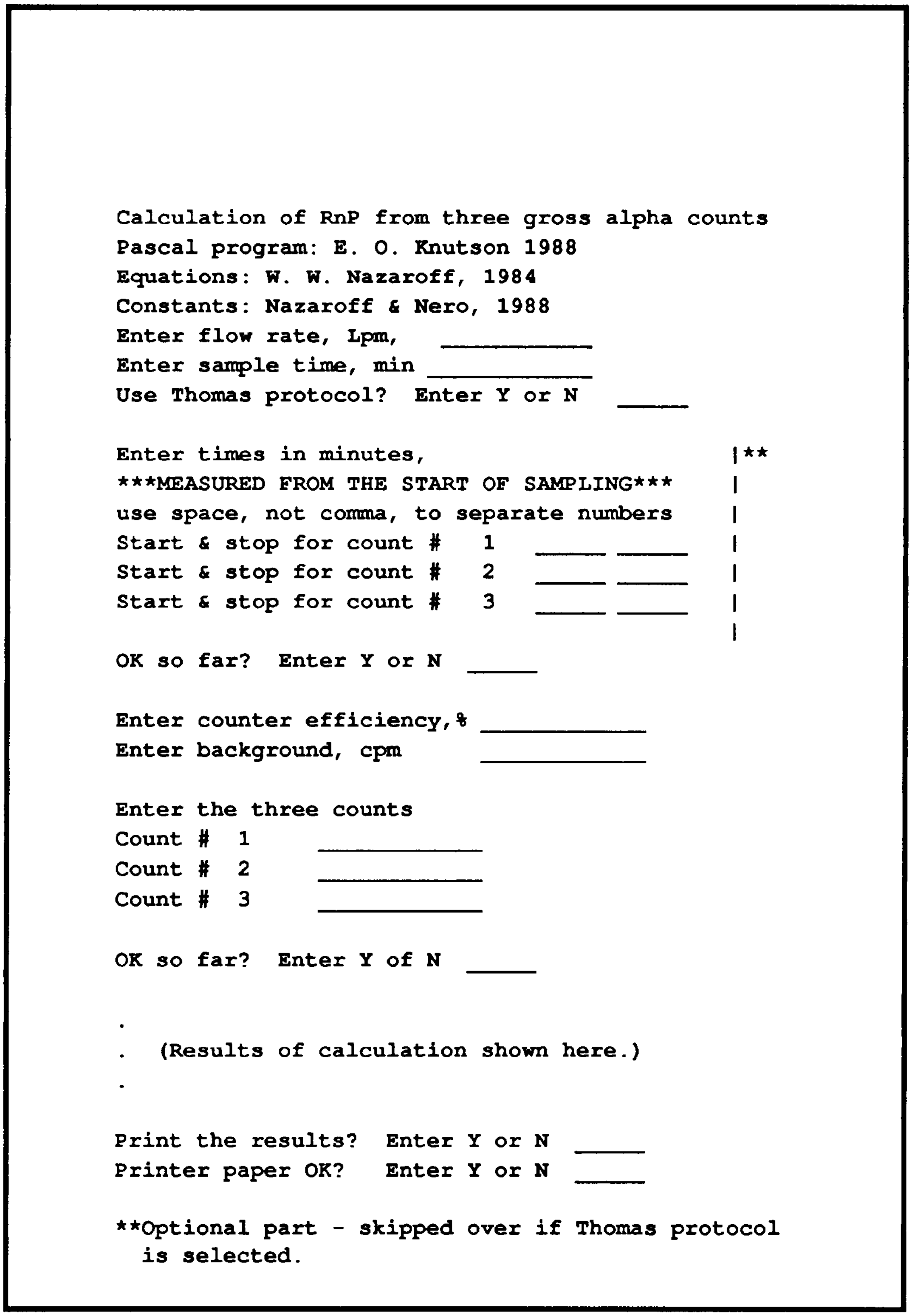
SOURCE LANGUAGE FILES

Name	Size	Date
Aerosol.Pas	2212	01-21-89
Algorith.Pas	7999	02-01-89
DataMess.Pas	1084	10-03-88
ExMaxDB.Pas	14772	10-28-88
ExMaxDP.Pas	16069	02-03-89
ExMaxGS.Pas	15320	02-01-89
Globals.Pas	936	06-08-88
Mtrx.Pas	4001	06-01-88
RWRENN.TXT	2779	02-01-89
RWRENNGW.BAS	16873	02-01-89
WWN.Pas	7470	01-27-89

TABLE 2

EXECUTABLE FILES

Name	Size	Date
ExMaxDB.EXE	30576	02-01-89
ExMaxDP.EXE	28192	02-03-89
ExMaxGS.EXE	28480	02-01-89
RWRENNGW.EXE	59363	02-21-89
WWN.EXE	18192	01-27-89



```
Calculation of RnP from three gross alpha counts
Pascal program: E. O. Knutson 1988
Equations: W. W. Nazaroff, 1984
Constants: Nazaroff & Nero, 1988
Enter flow rate, Lpm,    ___________
Enter sample time, min  ____________
Use Thomas protocol?  Enter Y or N   _____

Enter times in minutes,                      |**
***MEASURED FROM THE START OF SAMPLING***    |
use space, not comma, to separate numbers    |
Start & stop for count #  1  _____ _____     |
Start & stop for count #  2  _____ _____     |
Start & stop for count #  3  _____ _____     |
                                             |
OK so far?  Enter Y or N  _____

Enter counter efficiency,% ____________
Enter background, cpm      ____________

Enter the three counts
Count #  1          ____________
Count #  2          ____________
Count #  3          ____________

OK so far?  Enter Y of N  _____

.
.  (Results of calculation shown here.)
.

Print the results?  Enter Y or N  _____
Printer paper OK?   Enter Y or N  _____

**Optional part - skipped over if Thomas protocol
  is selected.
```

Figure 1. Example input screen for WWN.Pas

The input data is assumed to be in an ASCII disk file, and you will be prompted for its name. The first line must be a title line <80 characters long (avoid using "*" or a "blank space" as a first character). The second line specifies the source of the nineteen input parameters needed to do the calculation:

> if the second line consists of a file name, the parameters will be read from that file (which must contain exactly 19 numbers);
>
> if the second line consists of numbers the first of which is a fraction, the first 19 numbers will be used as the parameters;
>
> if the first number is an integer, all input is stored as count data; you will be prompted to enter parameters from the keyboard.

The actual count data begins (or continues) on the third line, one or more integers per line. Within each line of numeric data, numbers are demarked by commas or - if no comma is found - by spaces. The final line of each block of data must be a nul line, or an end-of-file mark.

The 19 parameters are:

1-5
Counter efficiency (fractional, not percent) for -
^{218}Po, ^{214}Po, ^{212}Bi, ^{212}Po, Nuclide X.

6-11
Which nuclides to analyze for - (enter 1 to analyze, 0 to skip)
^{218}Po, ^{214}Pb, ^{214}Po, ^{212}Bi, ^{212}Po, Nuclide X

12-19
background count time, min,
background counts,
counter dead time, microsec.,
flow rate, L min^{-1},
sampling time, min,
sample transfer time, sec,
length of each count, sec,
down time between counts, sec,

If the first count datum is negative, it is discarded and the TTRANS corrected by adding TCOUNT and TDOWN to it. This allows the user to delete the first count by prefixing a minus sign.

Figure 2. Description of format for RWRENNGW input data.

```
ui4-13-88-10:28-60-100-200-635 filt  04-13-1988  10:41:29   60  chan 1
.516 .516 .516 .516 .516 1 1 1 0 0 0 30 3 53 3.0 10 120 60 0.07
      66      65      54      43      46      36      34      35
      29      22      24      24      23      35      23      20
      16      20      25      22      21      17      23      24
      25      22      28      27      22      26      27      23
      38      22      22      22      27      22      25      23
      22      22      19      23      22

614881510BASAIR      CHNL 1   120   200
hptta
  1741  1540  1449  1309  1342  1274  1214  1207  1166  1164
  1141  1148  1122  1042  1078   981  1008   961   939   938
   907   866   809   779   766   827   690   713   642   686
   669   624   601   571   614   546   489   477   483   515
   456   421   394   402   443   374   345   353   301   306
   298   291   279   254   247   216   204   212   206   203
   172   170   169   155   149   149   151   128   121   118
   108   113   114   115   111   113    91    84    96    74
    82    78    66    65    72    58    63    42    70    64
    47    36    49    35    35    46    42    29    38    42
    34    32    35    21    25    32    23    22    16    28
    22    18    15    24    20    24    15    16    14    15
    17    16    10    11    16     8     6    11    15    11
     4    10    10    11    12     8     6     8     7     7
    12     6     9    10    11    11     4    10     7     9
     7     7     8     8     2     9     5     2     8     5
    14     3     8     4     6     6     4     6     1    10
     5     4     6     7     4     6     3     2     8     9
     4     9     3     3     3     6     3     5     6     3
     6     7     5     5     2     4     5     6     7     8
```

Figure 3. Printout of a file containing two input data blocks for RWRENNGW.

```
ui4-13-88-10:28-60-100-200-635 filt  04-13-1988  10:41:29   60  chan 1
   0.516   0.516   0.516   0.516   0.516  1  1  1  0  0  0
  30.000   3.000  53.000   3.000  10.000 120.000  60.000   0.070

Results in terms of bequerels and PAEC. # cnts = 45

Nuclide  Bq/fltr  Std. Err.  Bq m-3  Std. Err.  PAEC, nJ m-3  PAEC, mWL
218Po     3.78      0.34     314.44   28.07       185.35         8.91
214Pb     1.25      0.10      24.72    3.31        70.37         3.38
214Bi     0.13      0.12      -1.90    5.19        -3.98        -0.19
212Pb     0.00      0.00       0.00    0.00         0.00         0.00
212Bi     0.00      0.00       0.00    0.00         0.00         0.00
Nucl-X    0.00      0.00       0.00    0.00         0.00         0.00

Potential alpha energy conc.  =                   251.73        12.10
PAEC standard error =                               8.63         0.41
Variance of the fit =                               0.57
Analyzed on  09-28-1988  via Basic


614881510BASAIR      CHNL 1   120    200
   0.480   0.480   0.480   0.480   0.480  1  1  1  1  1  0
 100.000  10.000  53.000  14.500  10.000 120.000 120.000   1.000

Results in terms of bequerels and PAEC. # cnts = 200

Nuclide  Bq/fltr  Std. Err.  Bq m-3  Std. Err.  PAEC, nJ m-3  PAEC, mWL
218Po    20.26      1.73     348.84   29.84       205.62         9.89
214Pb    25.39      0.51     173.69    4.12       494.44        23.77
214Bi    19.27      0.49     124.69    4.12       261.16        12.56
212Pb     0.08      0.01       0.53    0.08        37.08         1.78
212Bi     0.09      0.27       0.63    2.01         4.13         0.20
Nucl-X    0.00      0.00       0.00    0.00         0.00         0.00

Potential alpha energy conc.  =                  1002.43        48.19
PAEC standard error =                               7.31         0.35
Variance of the fit =                               1.10
Analyzed on  09-28-1988  via Basic
```

Figure 4. Example printout from RWRENNGW.

```
You will be prompted for the name of an
ASCII file containing the input data,
which may contain four types of lines:
   blank lines,        comment lines,
   title lines,        data lines.

Comment lines must start with "*".
 They will be echoed to the output.
Title lines must be 1 to 80 char.
 & the first cannot be "*" or" "
Data lines (real numbers) must immed-
 eately follow the Title line.

The data line must have the penetrations
or catches in a standard, known sequence,
e.g., monotone decreasing for DB.  (Use
spaces, not commas, to separate the numbers.)

Option: the line may include 1-sigma errors,
as follows: data-0, error-0, data-1, error-1...
```

Figure 5. Structure for the ExMaxDB input data file.

```
 .
 .  (Sign on banner showing program credits)
 .
Need a reminder on data format? Enter Y or N ____

Type of diffusion battery:
  0  - Series Screen
  1  - Parallel Screen
  2  - Series Disk
  3  - Parallel Disk
Enter 1-digit code (2 not available) ____

Enter flow rate, Lpm   ____
OK so far? Enter Y or N  ____

Enter particle size classes:
  smallest diameter, nm        _____
  largest diameter, nm         _____
No. of size classes,  = 16  _____
 .
 . (optional message concerning error calculation)
 .
Enter file names -
  - of input            ____________
  - for output          ____________
Does input file include error terms? Enter Y or N  ___
OK so far? Enter Y or N _____
 .
 .  (another optional message)
 .
Enter max number of iterations, => 0
  - for Twomey                       ____
  - for expect. max.                 ____
Enter Twomey speed factor            ____      |**
Enter EM conv. crit.                 ____      |
 .                                              |
 .                                              |
 .  Enter error scaling number, 0-32127        |
      (0 = autoscaling                   ____  |
Include the matrices in output: Enter Y or N ____
OK so far? Enter Y or N ____


**Optional, depending on previous responses.
```

Figure 6. Sequence of prompts and replies for ExMaxDB. (Underlines indicate user replies.)

```
*Multichannel screen diffusion battery msmts at Lawrence
*Berkeley Laboratory
*May 7-10, 1984
*Dates are in the sequence: filter, DB-1, DB-2, DB-3, DB-4
*Flow rate was 25 lpm

LBL 5/7/2   1501     -   A
 1156.0    843.0     781.0     546.0     242.0
LBL 5/7/2   1501     -   B
 1109.0   1018.0     919.0     644.0     292.0
```

Figure 7. Example input data file for ExMaxDB.

```
*Multichannel screen diffusion battery msmts at Lawrence Berkeley
*Laboratory
*May 7-10, 1984
*Dates are in the sequence: filter, DB-1, DB-2, DB-3, DB-4
*Flow rate was 25 Lpm

LBL 5/7/2   1501     -   A
Parallel screen battery @  25.00 Lpm
Calculated yr,mo,dy:  1988  6 17      Error scale factor,  10000

Diam.,nm  DA/DlogD  1-sigma
   1.00     320.37  225.38  *++++*++++*++++*+
   4.73      49.98  289.76  *+++
  22.36      79.35  101.47  *++++
 105.74     994.50   81.07  *++++*++++*++++*++++*++++*++++*++++*++
 500.00     267.33   45.89  *++++*++++*+++

TAP      RAW      1-sigma     FIT
  1   1156.00      -1.00   1154.84  Twomey iterations:       0
  2    843.00      -1.00    873.85         speedfactor:   5.0E-01
  3    781.00      -1.00    745.04  ExMax iterations:       270
  4    546.00      -1.00    536.58        conv. crit.:    5.0E-04
  5    242.00      -1.00    257.69  test of matrix inver.: 1.0E-09

LBL 5/7/2   1501     -   B
Parallel screen battery @  25.00 Lpm
Calculated yr,mo,dy:  1988  6 17      Error scale factor,  10000

Diam.,nm  DA/DlogD  1-sigma
   1.00      10.01  240.46  *
   4.73       7.38  309.27  *
  22.36     128.87  108.19  *++++*+
 105.74    1168.10   86.40  *++++*++++*++++*++++*++++*++++*++++*++
 500.00     323.51   48.92  *++++*++++*++++

TAP      RAW      1-sigma     FIT
  1   1109.00      -1.00   1105.14  Twomey iterations:       0
  2   1018.00      -1.00   1045.61         speedfactor:  5.0E-01
  3    919.00      -1.00    888.85  ExMax iterations:     287
  4    644.00      -1.00    636.19        concriterion:  5.0E-04
  5    292.00      -1.00    306.20  test of matrix inver 1.3E-09
```

Figure 8. Example of the output from ExMaxDB.

```
.
.    (Sign-on banner and program credits.)
.
Standard Screens:  A - EML 20 mesh
                   B - EML 60 mesh
                   C - EML 100 mesh
                   D - EML 200 mesh
                   E - EML 635 mesh
                   F - filter

Are there any other screens? Enter Y or N  ___
.
.    [you will be prompted to supply the wire diameter,
.    thickness, and solid fraction of each other screen]
.
Enter the screens used (from the set A to [F]), and show their
configuration.  Use contiguous letters (no spaces) to indicate
"arranged in series"; use spaces to indicate "arranged in
parallel."  For series arrangement, assume air flow is left to
right.  Example: BCD F means filter in parallel with the series
group BCD.
-+-+-+- Enter configuration -->  ________
.
.    [echo of input after last prompt]
.
```

Figure 9. Sequence of Prompts from ExMaxGS (where they differ from ExMaxDB). Underlines show where user input is required.

Appendix A
File: Aerosol.pas

```
UNIT aerosol;

INTERFACE
USES globals;
FUNCTION DiffCoef (partdiam : real) : real;
FUNCTION LogChengKK (facevel, partdiam : real) : real;
FUNCTION GormleyKen (mu : real) : real;

IMPLEMENTATION

FUNCTION DiffCoef (partdiam : real) : real;

CONST  mnfreepath = 65.3E-7; (* centimeters *)
       boltz      = 1.38E-16;
       abstemp    = 298;
       viscosity  = 1.81E-4;
       pi         = 3.14159;

VAR    slipcor, x : real;

BEGIN
(*  Change particle diameter to centimeters *)
    partdiam := partdiam*1.0E-7;

(*  Hinds equation 3.20 for slipcor *)
    x := 2.514 + 0.800*EXP(-0.55*partdiam/mnfreepath);
    slipcor := 1.0 + (mnfreepath/partdiam)*x;

    diffcoef := boltz*abstemp*slipcor/
             (3.0*pi*viscosity*partdiam);

END;

FUNCTION LogChengKK (facevel, partdiam : real) : real;

(*
       Cheng-Keating-Kanipilly equation for the transport of
       aerosol particles through 635-mesh wire screens.
       Equation and constants taken from Chapter 73 of
       AEROSOLS, edited by Marple and Liu (1983)
       The value reported out is the negative common log
       of ChengKK.
*)
CONST  wirediam   = 0.0020; (* centimeters *)
```

```
        thickness   = 0.0050; (* centimeters *)
        solidfrac   = 0.345;
        a0          = 1.96;
        a1          = 3.37;
        a2          = 1.94;

VAR peclet, x : real;

BEGIN
    peclet := facevel*wirediam/diffcoef(partdiam);

    x := EXP(2*LN(peclet)/3);

(*  Change particle diameter to centimeters *)
    partdiam := partdiam*1.0E-7;

    LogChengKK := a0/x + a1*SQR(partdiam/wirediam) +
         a2*EXP(2*LN(partdiam/wirediam)/3)/SQRT(peclet);

END;

FUNCTION GormleyKen (mu : real) : real;
(*  Note that we are using the classical
    Gormley & Kennedy equation circa 1949.
    See Fuchs, p. 205                         *)

VAR  p,x : real;
BEGIN
  IF mu < 0.02 THEN
  BEGIN
    x := EXP(LN(mu)/3.0);
    p := 1.0 - 2.56*x*x + 1.2*mu + 0.177*mu*x
  END
  ELSE BEGIN
    p := 0.0;
    x := 3.657*mu;
    IF x < 20.0 THEN p := p + 0.819*EXP(-x);
    x := 22.3*mu;
    IF x < 20.0 THEN p := p + 0.097*EXP(-x);
    x := 57.0*mu;
    IF x < 20.0 THEN p := p + 0.032*EXP(-x)
  END;
  GormleyKen := p;
END;
END.
```

Appendix B
File Algorith.pas

```
UNIT algorithm;
INTERFACE
USES Crt, globals, mtrx;

PROCEDURE forwardcalc(amtperclass : rsltvector;
                numdatapts, numclasses : integer;
                VAR kernel : kernelmatrix;
                VAR fitdata : datavector);

PROCEDURE TwomeyAlg (rawdata : datavector;
             numdatapts, numclasses, maxiter : integer;
                   twmyspeed : real;
             VAR kernel : kernelmatrix;
             VAR amtperclass : rsltvector;
             VAR ok : boolean);

PROCEDURE ExpectMax (rawdata : datavector;
             numdatapts, numclasses, maxiter : integer;
             epsilon : real;
             VAR kernel : kernelmatrix;
             VAR amtperclass : rsltvector;
             VAR lastiter : integer;
             VAR fitdata  : datavector;
             VAR ok : boolean);

PROCEDURE StandardErr(datavariance : datavector;
                   numdatapts, numclasses : integer;
                   VAR kernel : kernelmatrix;
                   VAR stderr : rsltvector;
                   VAR ok : boolean);

IMPLEMENTATION

PROCEDURE forwardcalc;

VAR   i, j    : integer;

BEGIN
  FOR i := 1 TO numdatapts DO
  BEGIN
    fitdata[i] := 0.0;
    FOR j := 1 TO numclasses DO
      fitdata[i] := fitdata[i] +
          kernel[i,j]*amtperclass[j];
  END;
END;

PROCEDURE TwomeyAlg;

VAR a, b, sum, big: real;
    i, j, m        : integer;
    prior, prod    : ARRAY [1..16] of real;
    converged      : boolean;
    twmykernel     : ARRAY [1..12,1..16] OF real;
    weight         : ARRAY [1..12] OF real;
    monotonic      : boolean;

BEGIN
  IF maxiter < 1 THEN EXIT;

  GoToXY(1,24);
  WRITE('TWOMEY ALGORITHM');

  FOR j := 1 TO numclasses DO
    IF (amtperclass[j] < 0) OR (amtperclass[j] > 1.0E10) THEN
    BEGIN
      GoToXY(1,24);
      WRITE('INVALID STARTING SIZEDIST - ABORTING TWOMEYALG');
      converged := false;
      EXIT;
    END;

(*  test for monotone decreasing data and kernel; if montoinc,
    take first differences, else use w/o differencing.  *)

  monotonic := true;
  FOR i :=2 TO numdatapts DO
    monotonic := monotonic AND (rawdata[i-1] >= rawdata[i]);

  j := 1;
  WHILE monotonic AND (j  <= numclasses) DO
  BEGIN
    FOR i := 2 TO numdatapts DO
      monotonic := monotonic AND (kernel[i-1,j] >= kernel[i,j]);
      j := j + 1;
   END;

  FOR j := 1 TO numclasses DO
  BEGIN
    a := 0.0;
    FOR i := numdatapts DOWNTO 1 DO
    BEGIN
      twmykernel[i,j] := kernel[i,j] - a;
      IF monotonic THEN a := kernel[i,j];
    END;
  END;
  a := 0.0;
  IF monotonic THEN
    FOR i := numdatapts DOWNTO 1 DO
    BEGIN
      b           := rawdata[i];
      rawdata[i] := b - a;
      a           := b;
     END;

  If monotonic THEN WRITE('  - monotonic')
  ELSE WRITE('  - nonmonotonic');

  FOR i := 1 TO numdatapts DO
  BEGIN
    big := 0.0;
    FOR j := 1 TO numclasses DO
      IF twmykernel[i,j] > big THEN big := twmykernel[i,j];
    weight[i] := twmyspeed/big;
  END;

  m := 1;
  REPEAT

    FOR j := 1 TO numclasses DO
      prior[j] := amtperclass[j];

    i := Random(numdatapts) + 1;

      sum := 0.0;
      FOR j := 1 TO numclasses DO
      BEGIN
        prod[j] := twmykernel[i,j]*amtperclass[j];
        sum := sum + prod[j];
      END;

      a := weight[i]*(rawdata[i]/sum - 1.0);

      FOR j := 1 TO numclasses DO
        amtperclass[j] := amtperclass[j] + a*prod[j];

      GoToXY(1,24);
      WRITE('TWOMEY COUNTER ',m:3,'       INPUT DATA #    ',i:3);

  m := m + 1;
  UNTIL m > maxiter*numdatapts; (*   *)

  ok := true;

END;


PROCEDURE ExpectMax;
```

```
VAR bp, sum, bsum, likelihood : real;
    i, j, l, k, m    : integer;
    pp, prior        : ARRAY [1..16] of real;
    prodmatrix       : ARRAY [1..12,1..16] OF real;
    converged        : boolean;

BEGIN
  IF maxiter < 1 THEN EXIT;

  GoToXY(1,24);
  WRITE('EXPECTATION MAXIMIZATION

  FOR j := 1 TO numclasses DO
    IF (amtperclass[j] < 0) OR (amtperclass[j] > 1.0E10) THEN
    BEGIN
      GoToXY(1,24);
      WRITE('INVALID STARTING SIZEDIST  - ABORTING EXPECTMAX');
      converged := false;
      EXIT;
    END;

  FOR l := 1 to numclasses DO
  BEGIN
    bp := 0.0;
    FOR i := 1 TO numdatapts DO
    BEGIN
      prodmatrix[i,l] := rawdata[i]*kernel[i,l];
      pp[l] := kernel[i,l] + bp;
      bp := pp[l];
    END;
  END;

  forwardcalc(amtperclass,numdatapts,numclasses,kernel,fitdata);

  m := 1;
  REPEAT

    FOR l := 1 TO numclasses DO
    BEGIN
      prior[l] := amtperclass[l];

      bsum := 0.0;
      FOR k := 1 TO numdatapts DO
        bsum := prodmatrix[k,l]/fitdata[k] + bsum;

      amtperclass[l] := prior[l]*bsum/pp[l];

    END;

    converged := true;
(*  note that the convergence test is based on the ABSOLUTE
    difference between iterates, not the relative difference
    tha Ed Maher used.  *)
    FOR j := 1 TO numclasses DO
      converged := converged AND
      (ABS(prior[j] - amtperclass[j]) < epsilon);

    forwardcalc(amtperclass,numdatapts,numclasses,kernel,fitdata);

    bp := 0.0;
    FOR i := 1 TO numdatapts DO bp := bp +
      rawdata[i]*LN(fitdata[i]) - fitdata[i];
    likelihood := bp;

    GoToXY(1,24);
    WRITE('EXMAX ITERATION #',m:3,'      LIKELIHOOD:   ',likelihood:16);

  m := m + 1;
  UNTIL (m > maxiter) OR converged;

  lastiter := m;
  ok := converged;

END;

PROCEDURE StandardErr;
(*=============================================================
    This part of Maher's program is based on the notion that the
    input data are drawn from independent Poisson populations.
    Therefore, his matrix D is a diagonal matrix using the inverse
    of the fitted data points as the diagonal elements.  That is,
    the fitted data are used as the Poisson estimates of the
    variance.

    In fact, the Poisson assumption is seldom correct (e.g.,
    it is not the case for "counts" obtained with a condensation
    nucleus counter).  In order to make the calculation meaningful,
    we will improvise: in place of the vector fitdata, we input a
    vector data variance, comprising the estimated variance of the
    data.  Thus, it is up to the calling program to supply meaning-
    ful estimates of the variances.
=============================================================*)

VAR diag    : ARRAY [1..12] of real;
    b       : ARRAY [1..16,1..16] OF real;
    i, j, k, l      : integer;
    fmax, sumjl, a : real;

BEGIN
  GoToXY(1,24);
  WRITE('PROCEDURE stderr                                          ');

(*
    Again we deviate a little from Maher.  His matrix B, which
    calculates the additional correlations among size class
    populations due to forcing normalization on them, will be
    omitted.  We prefer not to normalize, therefore, we have no
    need for matrix B.

    Construct the "diagonal" matrix, using the estimated var-
    iances as discussed above.
*)

  FOR i := 1 TO numdatapts DO diag[i] := 1.0/datavariance[i];

(*
  Form the matrix product [p-transpose][diag][p] and store into
  matrix1.  Note that the innermost sum collapses into a single
  term, since diag is a diagonal matrix.  Note also, on the line
  marked <-aaaa, that we used kernel[j,i] in place of the i,j
  element of the matrix [p-transpose].
*)

  FOR i := 1 TO numclasses DO
    FOR l := 1 TO numclasses DO
    BEGIN
      matrix1[i,l] := 0.0;
      FOR j := 1 TO numdatapts DO
      BEGIN
        sumjl := diag[j]*kernel[j,l];   (* <----"sum" *)
        matrix1[i,l] := matrix1[i,l]
          + kernel[j,i]*sumjl;          (* <---- aaaa *)
      END;
    END;

(*
  Call matinvert; inverse of matrix1 will appear in matrix2
*)

  GoToXY(1,24);
  WRITE('PROCEDURE matinvert
  matrixsize := numclasses;
  matinvert;

  ok := NOT matsingular;

  IF ok THEN FOR j := 1 TO numclasses DO
      stderr[j] := SQRT(matrix2[j,j]);

END;
END.
```

Appendix C
File Datamess.pas

```
UNIT datamessage;
INTERFACE
USES Crt;
PROCEDURE dataformat;
IMPLEMENTATION
PROCEDURE dataformat;
BEGIN
    ClrScr;
    WRITELN('You will be prompted for the name of an ');
    WRITELN('ASCII file containing the input data, ');
    WRITELN('which may contain four types of lines: ');
    WRITELN('   blank lines,       comment lines,');
    WRITELN('   title lines,       data lines.');
    WRITELN;
    WRITELN('Comment lines must start with "*".');
    WRITELN(' They will be echoed to the output.');
    WRITELN('Title lines must be 1 to 80 char.');
    WRITELN(' & the first cannot be "*" or" "');
    WRITELN('Data lines (real numbers) must immed- ');
    WRITELN(' ately follow the Title line.');
    WRITELN;
    WRITELN('The data line must have the penetrations ');
    WRITELN('or catches in a standard, known sequence,');
    WRITELN('e.g., monotone decreasing for DB.  (Use');
    WRITELN('spaces, not commas, to separate the numbers.)');
    WRITELN;
    WRITELN('Option: the line may include 1-sigma errors,');
    WRITELN('as follows: data-0, error-0, data-1, error-1...');
    WRITELN;
END;
END.
```

Appendix D
File ExMaxDB.Pas

```
PROGRAM ExMaxDiffBatt (input, output);
(*==========================================================
    This Pascal program implements the Expectation-Maximization
    algorithm for calculating particle size spectra from
    diffusion battery data.  (The Twomey algorithm is also
    coded, and may be used alone or as a starter for the
    Ex-Max calculation.)  The Ex-Max coding draws heavily on
    the FORTRAN code kindly provided by Edward Maher.  Major
    changes are explained along the way.

    Original Maher program : December, 1983
    This Pascal program    : January, 1988
    Refinements            : March, May 1988

==========================================================*)
Uses  Crt, Dos, mtrx, algorithm, globals, aerosol, datamessage;

TYPE   DiffBatType = (SS, PS, SD, PD, PC);
       arrytyp9 = ARRAY [1..10,1..10] OF real;

VAR titleoftest                : string;
    results                    : text;

    pdiam                      : rsltvector;
    penmtx                     : kernelmatrix;
    infilename, outfilename    : string[20];
    maxmaher,maxtwmy,lastmaher : integer;
    errorscale                 : integer;

    i, numsizes, numdatapts    : integer;
    oksofar, errorterms, matprint : boolean;
    sizedist, stderr           : rsltvector;
    rawdata, fitdata, rawerror : datavector;
    dbdata                     : text;

    DBType     : DiffBatType;
    DBI        : integer;

    flowrate, concriterion, twmyspeed    : real;

    cap, rawmax  : real;
    datavariance : datavector;
    j            : integer;
```

```
FUNCTION kbquery(msg : string) : boolean;
VAR  query : char;
BEGIN
  REPEAT
  WRITE(msg,' Enter Y or N ');
  READLN(query)
  UNTIL query IN ['y','Y','n','N'];
  kbquery := (query IN ['y','Y']);
END;

PROCEDURE ScanForTitle (VAR inputfile : text);
VAR validtitle : boolean;
BEGIN
    WINDOW(1,1,80,24);
    REPEAT
      READLN(inputfile, titleoftest);
      validtitle := (length(titleoftest) > 0) AND (titleoftest[1] <> '*')
        AND (titleoftest[1] <> ' ');
      IF NOT validtitle THEN WRITELN(results,titleoftest);
      GoToXY(1,23); WRITELN(' ':72);
      GoToXY(1,23); WRITELN (titleoftest);
    UNTIL validtitle OR EOF(inputfile);
END;

PROCEDURE SignOn;
VAR gstep, minsize, maxsize : real;
    j : integer;
BEGIN
    ClrScr;
    Window(15,2,80,24);
    WRITELN('Calculation of Aerosol Size Distributions');
    WRITELN('from Diffusion Battery Data');
    WRITELN;
    WRITELN('Using the Expectation - Maximization');
    WRITELN('Algorithm.  (Twomey also available)');
    WRITELN;
    WRITELN('Pascal Program by Earl O. Knutson,');
    WRITELN('USDOE/EML,           January, 1988');
    WRITELN;
    WRITELN('Ex-Max code based on FORTRAN Program by');
    WRITELN('Edward F. Maher,                 13Dec83');
    WRITELN;
    WRITELN;
    IF kbquery('Need a reminder on data format?') THEN
    BEGIN
     dataformat;
     IF NOT kbquery('Are you ready to continue?') THEN HALT;
    END;

  REPEAT
    ClrScr;
    WRITELN('Type of diffusion battery:');
    WRITELN('  0  - Series Screen');
    WRITELN('  1  - Parallel Screen');
    WRITELN('  2  - Series Disk');
    WRITELN('  3  - Parallel Disk');
(*    WRITELN('  4  - Parallel Carbon');*)
    WRITELN;
    REPEAT
      WRITE('  Enter 1-digit code (2 not available) ');
      READLN(DBI);
      DBType := DiffBatType(DBI);
    UNTIL DBType IN [SS, PS, PD];
    WRITELN;
    WRITE('Enter flowrate, Lpm   ');
    READLN(flowrate);
  UNTIL kbquery('OK so far?');

    CASE DBType OF
      PS,PD,PC   : numdatapts := 5;
      SS         : numdatapts := 11;
      SD         : numdatapts := 12
    END;

  REPEAT
    ClrScr;
    WRITELN('Enter particle size classes:');
    WRITE('  smallest diameter, nm      ');
    READLN(minsize);
    WRITE('  largest diameter,  nm      ');
    READLN(maxsize);
    WRITE('  no. of size classes, < = 16 ');
    READLN(numsizes);
    IF numsizes > numdatapts THEN
    BEGIN
      WRITELN('....Since numsizes > numdatapts, Ex-Max');
      WRITELN('....error calc will not be done.');
    END;
    WRITELN;
    WRITELN('enter filenames - ');
    WRITE('  - of input   ');
    READLN(infilename);
    WRITE('  - for output ');
    READLN(outfilename);
    errorterms := kbquery('Does input file include error terms?');
    IF errorterms THEN WRITELN('OK. These will be used in the stderr calc');
  UNTIL kbquery('OK so far?');

  REPEAT
    ClrScr;
    WRITELN('enter max # of iterations,=>0');
    WRITE('  - for twomey        ');
    READLN(maxtwmy);
    WRITE('  - for expect. max.  ');
    READLN(maxmaher);
    IF (maxtwmy > 0) THEN
    BEGIN
      WRITE('enter twomey speed factor ');
      READLN(twmyspeed);
    END
    ELSE twmyspeed := -1.0;

    IF (maxmaher > 0) THEN
    BEGIN
      WRITE('enter exmax conv. crit. ');
      READLN(concriterion);
    END
    ELSE concriterion := -1.0;

    IF (numsizes > numdatapts) OR errorterms THEN
    BEGIN
      errorscale := -1;
      matproof := -1.0;
    END
    ELSE
    BEGIN
      WRITELN('enter error scaling number, 0-32768');
      WRITE('   (0 = autoscaling     ');
      READLN(errorscale);
    END;
    WRITELN;
    matprint := kbquery('Include the matrices in output?');
  UNTIL kbquery('OK so far?');

    pdiam[1] := minsize;
    gstep := ln(maxsize/minsize)/(numsizes - 1);
    gstep := exp(gstep);

    FOR j := 2 TO numsizes DO pdiam[j] := gstep*pdiam[j-1];

    Randomize;
    Window(1,1,80,24);
END;

PROCEDURE GetPenMatrix;

VAR flowarea,facevel,partdiam,logscrnpen,x,f,p : real;
    i, j, k  : integer;

BEGIN
  ClrScr;

  CASE DBType OF
    SS : flowarea := 11.40; (* diameter = 3.81 cm
                               per Cheng and Yeh *)
    PS : flowarea := 81.07  (* diameter = 10.16 cm *)
  END;
```

```
FOR j := 1 TO numsizes DO
BEGIN
  GoToXY(1,24);
  WRITE('BUILD PENMATRIX, SIZE CLASS:',j:3);
  penmtx[1,j] := 1.0;

  CASE DBType OF

    SS, PS : logscrnpen :=
              LogChengKK(1000*flowrate/flowarea/60, pdiam[j]);

    SD, PD : f := 3.14159*2.54*14500.0/(flowrate*1000.0/60)
           (*  14500 = the # of cylindrical holes in each of
           Sinclair's "CHS" disks (careful here - the newer
           batteries had a small number of holes);
           1000/60 converts from Lpm to cm3 s-1; *)

  END;

  CASE DBType OF

    SS : FOR i := 2 TO numdatapts DO
         BEGIN
           x := i*(i - 1)*logscrnpen/2.0;
           IF x > 25 THEN penmtx[i,j] := 0.0
           ELSE penmtx[i,j] := EXP(-x*LN(10.0))
         END;

    PS : FOR i := 2 TO numdatapts DO
         BEGIN
           CASE i OF
              2  : x := 1*logscrnpen;
              3  : x := 5*logscrnpen;
              4  : x := 15*logscrnpen;
              5  : x := 40*logscrnpen
           END;
           IF x > 25 THEN penmtx[i,j] := 0.0
           ELSE penmtx[i,j] := EXP(-x*LN(10.0))
         END;

    PD : BEGIN
         x := f*DiffCoef(pdiam[j]);
         penmtx[1,j] := 1.0;
         penmtx[2,j] := GormleyKen(0.135*x);
         penmtx[3,j] := GormleyKen(0.252*x)*GormleyKen(0.122*x);

         IF x > 5.0 THEN penmtx[4,j] := 0.0 ELSE
         penmtx[4,j] := GormleyKen(0.517*x)*GormleyKen(0.405*x)*
                 GormleyKen(0.740*x)*GormleyKen(0.507*x)*
                 GormleyKen(1.013*x);

         IF x > 2.0 THEN penmtx[5,j] := 0.0 ELSE
         penmtx[5,j] := GormleyKen(1.005*x)*GormleyKen(1.030*x)*
```

```
                    GormleyKen(1.009*x)*GormleyKen(1.015*x)*
                    GormleyKen(1.006*x)*GormleyKen(1.015*x)*
                    GormleyKen(1.022*x)*GormleyKen(1.003*x)*
                    GormleyKen(1.008*x)*GormleyKen(1.027*x);

             END
      END;
    END;
  END;

PROCEDURE PrintResults (sizedist, stderr : rsltvector;
                        rawdata : datavector;
                        lastiter : integer;
                        VAR outpath : text;
                        ok : boolean);

VAR i, j, k      : integer;
    bigg, logstep, chisqr        : real;
    year, month, day, dayofweek : word;

BEGIN

  GetDate(year,month,day,dayofweek);
  logstep := Ln(pdiam[2]/pdiam[1])/Ln(10);
  forwardcalc(sizedist, numdatapts, numsizes, penmtx, fitdata);

  WRITELN(outpath, titleoftest);
  CASE DBType of
    SS : WRITE(outpath,'Series Screen ');
    PS : WRITE(outpath,'Parallel Screen ');
    SD : WRITE(outpath,'Series Disk ');
    PD : WRITE(outpath,'Parallel Disk ');
    PC : WRITE(outpath,'Parallel Carbon ')
  END;
  WRITELN(outpath,'battery @ ',flowrate:6:2,' Lpm');
  WRITELN(outpath,'Calculated yr,mo,dy: ',year:5,month:3,day:3,
                '       Error scale factor, ',errorscale:6);

  IF NOT ok THEN
    WRITELN(outpath,'  Calculation failed - no results to report');

    bigg := 0.0;
(*  convert to DA/DlogD & find peak value   *)
    FOR j := 1 TO numsizes DO
    BEGIN
      sizedist[j] := sizedist[j]/logstep;
      IF stderr[j] > 0 THEN stderr[j] := stderr[j]/logstep;
      IF sizedist[j] > bigg THEN bigg := sizedist[j];
    END;

    WRITELN(outpath);
    WRITELN(outpath,'Diam.,nm  DA/DlogD  1-sigma');
    FOR j := 1 TO numsizes DO
    BEGIN
      WRITE(outpath,pdiam[j]:7:2, sizedist[j]:10:2,
            stderr[j]:8:2, '   *');

      k := ROUND(50.0*sizedist[j]/bigg);
      i := 1;
      WHILE i  <= k DO
      BEGIN
        IF (i MOD 5) = 0 THEN WRITE(outpath,'*')
        ELSE WRITE(outpath, '+');
        i := i + 1;
      END;
      WRITELN(outpath);
    END;

    WRITELN(outpath);
    WRITELN(outpath,'TAP      RAW      1-sigma      FIT');
    FOR i := 1 TO numdatapts DO
    BEGIN
      WRITE(outpath,i:3,rawdata[i]:10:2,rawerror[i]:10:2,fitdata[i]:10:2);
        CASE i OF
        1 : WRITE(outpath,'  Twomey iterations:   ',maxtwmy:6);
        2 : WRITE(outpath,'         speedfactor: ',twmyspeed:6);
        3 : WRITE(outpath,'  ExMax iterations:    ',lastiter:6);
        4 : WRITE(outpath,'        concriterion: ',concriterion:6);
        5 : WRITE(outpath,'  test of matrix inver',matproof:8)
        END;
        WRITELN(outpath)
    END;

    IF errorterms THEN
    BEGIN
      chisqr := 0.0;
      FOR i := 1 TO numdatapts DO chisqr := chisqr +
        SQR((rawdata[i] - fitdata[i])/rawerror[i]);
      WRITELN(outpath, ' ':13,' ChiSqr = ',chisqr:10:2);
    END;

    FOR j := 1 to 78 DO WRITE(outpath,'-'); WRITELN(outpath);
    WRITELN(outpath);

END;


PROCEDURE PrintPenMatrix;
VAR  i,j : integer;
BEGIN

    FOR i := 1 TO numdatapts DO
    BEGIN
      FOR j := 1 TO numsizes DO
        WRITE(results, penmtx[i,j]:12);
      WRITELN(results);
    END;
    WRITELN(results);

END;

PROCEDURE PrintErrMatrices;
VAR  i,j : integer;
BEGIN

    FOR i := 1 TO numsizes DO
    BEGIN
      FOR j := 1 TO numsizes DO
        WRITE(results, matrix1[i,j]:12);
      WRITELN(results);
    END;
    WRITELN(results);

    FOR i := 1 TO numsizes DO
    BEGIN
      FOR j := 1 TO numsizes DO
        WRITE(results, matrix2[i,j]:12);
      WRITELN(results);
    END;
    WRITELN(results);
    WRITELN(results);

END;
BEGIN  (*  PROGRAM DiffBatCrunch   *)

  SignOn;

  GetPenMatrix;

  Assign(dbdata, infilename);
  RESET(dbdata);
  Assign(results, outfilename);
  REWRITE(results);
  ClrScr;

  ScanForTitle(dbdata);

  WHILE NOT EOF (dbdata) DO
  BEGIN

    i := 0;
    WHILE NOT EOLN (dbdata) DO
    BEGIN
      i := i + 1;
      READ (dbdata, rawdata[i]);
      IF errorterms THEN READ(dbdata, rawerror[i])
      ELSE rawerror[i] := -1.0;
    END;
    READLN (dbdata);
    oksofar := (i = numdatapts);

    rawmax := 0.0;
    FOR i := 1 TO numdatapts DO
      IF rawmax < rawdata[i] THEN rawmax := rawdata[i];

    IF oksofar THEN
      FOR i := 1 TO numsizes DO
      BEGIN
        sizedist[i] := rawmax/numsizes;
        stderr[i] := -1.0;
      END;

    IF oksofar THEN
      TwomeyAlg (rawdata, numdatapts, numsizes, maxtwmy,
                 twmyspeed, penmtx, sizedist, oksofar);

    lastmaher := 0;
    IF oksofar THEN
      ExpectMax (rawdata, numdatapts, numsizes, maxmaher,
                 concriterion*rawmax, penmtx, sizedist,
                 lastmaher, fitdata, oksofar);

    IF oksofar AND (numsizes  <= numdatapts) AND (lastmaher > 0) THEN
    BEGIN
      cap := 0.0;
      FOR i := 1 TO numdatapts DO
        IF fitdata[i] > cap THEN cap := fitdata[i];

(*============================================================
      As seen below, we provide three different ways to estimate the
      input data variances needed to calculate standard errors:

        First, if error terms are available in the input data stream,
        we use the square of those terms as the variance vector for
        the input data;

        Second, if error terms are not available and we have specified
        a zero error scale factor in the keyboard input, we use the
        fitted data vector as the estimate of variance;

        Third, if the error scale factor set positive, the fitdata vector
        is rescaled so that its maximum component is equal to the error
        scale factor, and this is used as the variance estimate.

      The second method - the one used by Ed Maher - is valid only if
      the input data are actual raw counts.  This is a necessary condition
      for the Poisson distribution to be valid.  (Even here, there are
      probably other sources of variability that swamp the Poisson
      variability.)

      Options one and three are provided because, in our opinion, the
      Poisson assumption is rarely valid for data from diffusion
      batteries.  We realize that there is a logical inconsistency
      in the idea of maximizing the Poisson-based likelihood function
      but using a non-Poisson estimate of variance.  So be it - half
      a loaf is better than none.
========================================================================*)


      IF errorterms THEN
        FOR i := 1 TO numdatapts DO datavariance[i] := SQR(rawerror[i]);

      IF NOT errorterms AND (errorscale = 0) THEN
        FOR i := 1 TO numdatapts DO datavariance[i] := fitdata[i];

      IF NOT errorterms AND (errorscale > 0) THEN
        FOR i := 1 TO numdatapts DO datavariance[i]
                                   := fitdata[i]*errorscale/cap;

      StandardErr (datavariance, numdatapts, numsizes, penmtx,
                   stderr, oksofar);

      IF NOT errorterms AND (errorscale > 0) THEN
        FOR j := 1 TO numsizes DO stderr[j] := stderr[j]*cap/errorscale;

    END;

    WINDOW(1,1,80,22); ClrScr;
    PrintResults (sizedist, stderr, rawdata, lastmaher, output,oksofar);
    PrintResults (sizedist, stderr, rawdata, lastmaher, results,oksofar);

    IF matprint THEN
      BEGIN
        PrintPenMatrix;
        IF (lastmaher > 0) AND (numsizes  <= numdatapts) THEN
          PrintErrMatrices;
      END;

    ScanForTitle(dbdata);
  END;
  CLOSE(results);
  WINDOW(1,1,80,24);
  GoToXY(1,23); WRITELN(' ':72);
  WRITE('***   F I N I    ***');

END.
```

Appendix E
File ExMaxDP.Pas

```
PROGRAM ExMaxDecayProd;
(*============================================================
    This is a program that had to be written because the
    problem is tailor-made for the EM algorithm.  The
    input data are raw radioactivity counts, widely
    accepted as conforming to Poisson statistics as
    required for EM.

    This program is designed to accept the same input files
    as RWRENNGW.BAS.  Handling the three-fold option for
    input of parameters was more difficult in Pascal than
    in BASIC, as can be seen from the PROCEDURE initialize.

    The concise equations given by Nazaroff (Health Phys.
    46, 395) are used in building the kernel matrix for the
    algorithm.  In fact, the set has been expanded to include
    three equations for thoron progeny; these were written
    down by analogy with the radon progeny equations.

    Although the input data are in terms of counts in equal
    time intervals, we have chosen to collect the counts into
    5, 6 or 7 time brackets (depending on the number of input
    data points) for this calculation.  Analysis for thoron
    progeny is attempted if and only if the total count
    interval spans at least 300 minutes.

       Earl O. Knutson
       USDOE Environmental Measurements Laboratory
       May, 1988

    A very important correction was made on 2 Feb 1989.
    Prior to that time, we had wrongly included a function G44,
    pertaining to the "alpha" from Pb-212.  The potential alpha
    energy table was also wrong prior to that time.
============================================================*)
USES Crt, Dos, Globals, mtrx, algorithm;

VAR decayconst, alphaenergy,
    dpconc, stderr, eweight             : rsltvector;
    kernel                              : kernelmatrix;
    rawdata, fitdata                    : datavector;
    i,j, k, maxiter, lastiter, halfpage     : integer;

     infilename, outfilename, titleoftest  : string;
     indata, results, parameters       : text;
     bkgdcount, numdatapts, numcounts   : integer;
     inthebag, nextblock, numnuclides   : integer;
     counts                             : ARRAY [1..2000] OF integer;
     cntsperblk                         : ARRAY [1..12] OF integer;
     cnteffic                           : ARRAY [1..5] OF real;

     bkgdtime, deadtime, flowrate, sum  : real;
     initguess, sampletime, temp,
                         transfertime   : real;
     timepercount, timebtwncnts         : real;
     ta, tb, t0, concriterion, chisqr   : real;
     ok, dothoron                       : boolean;

     year, month, day, dayofweek        : word;

FUNCTION f(i,j : integer) : real;
BEGIN
  f := decayconst[i]/(decayconst[i] - decayconst[j]);
END;

FUNCTION r(i : integer; t : real) : real;
BEGIN
  IF decayconst[i]*t > 80 THEN r := 1.0
  ELSE r := 1.0 - exp(-decayconst[i]*t);
END;

FUNCTION s(i : integer; t : real) : real;
BEGIN
  IF decayconst[i]*t > 80 THEN s := 0.0
  ELSE s := exp(-decayconst[i]*t);
END;

(*
   As explained by Nazaroff, Gij is the accumulated number
   of alphas emitted from nuclide i on the filter, due to
   collecting the j-th nuclide at a rate of 1 Bq per min.
   The factor 60, which is dpm per Bq, replaces Nazaroff's
   2.22, which is dpm per pCi. The units of Gij are min per Bq.
*)

FUNCTION G11(t,t0 : real) : real;
VAR G : real;
BEGIN
  IF t < t0 THEN
    G := t - r(1,t)/decayconst[1]
  ELSE
    G := t0 - r(1,t0)*s(1,(t - t0))/decayconst[1];
  G11 := 60*G/decayconst[1];
END;

FUNCTION G31(t, t0 : real) : real;
VAR G : real;
```

```
BEGIN
  IF t < t0 THEN
    G := t
         - f(2,1)*f(3,1)*r(1,t)/decayconst[1]
         - f(1,2)*f(3,2)*r(2,t)/decayconst[2]
         - f(1,3)*f(2,3)*r(3,t)/decayconst[3]
  ELSE
    G := t0
         - f(2,1)*f(3,1)*r(1,t0)*s(1,(t - t0))/decayconst[1]
         - f(1,2)*f(3,2)*r(2,t0)*s(2,(t - t0))/decayconst[2]
         - f(1,3)*f(2,3)*r(3,t0)*s(3,(t - t0))/decayconst[3];
  G31 := 60*G/decayconst[1];
END;

FUNCTION G32(t, t0 : real) : real;
VAR G : real;
BEGIN
  IF t < t0 THEN
    G := t
         - f(3,2)*r(2,t)/decayconst[2]
         - f(2,3)*r(3,t)/decayconst[3]
  ELSE
    G := t0
         - f(3,2)*r(2,t0)*s(2,(t - t0))/decayconst[2]
         - f(2,3)*r(3,t0)*s(3,(t - t0))/decayconst[3];
  G32 := 60*G/decayconst[2];
END;

FUNCTION G33(t, t0 : real) : real;
VAR G : real;
BEGIN
  IF t < t0 THEN
    G := t - r(3,t)/decayconst[3]
  ELSE
    G := t0 - r(3,t0)*s(3,(t - t0))/decayconst[3];
  G33 := 60*G/decayconst[3];
END;

(*
    To permit including thoron progeny in the analysis, the
    functions G54 and G55 - shown below - have been
    added to Nazaroff's list.  They were written down by
    analogy: G54 from G32; G55 from G33.  (This was changed
    on 2 Feb 1989.  Prior to that time, we had wrongly
    included a function G44, pertaining to the "alpha"
    from Pb-212.  The alpha energy table was also wrong
    prior to that time.)
*)

FUNCTION G54(t, t0 : real) : real;
VAR G : real;
BEGIN
  IF t < t0 THEN
```

```
      G := t
           - f(5,4)*r(4,t)/decayconst[4]
           - f(4,5)*r(5,t)/decayconst[5]
    ELSE
      G := t0
           - f(5,4)*r(4,t0)*s(4,(t - t0))/decayconst[4]
           - f(4,5)*r(5,t0)*s(5,(t - t0))/decayconst[5];
    G54 := 60*G/decayconst[4];
  END;

  FUNCTION G55(t, t0 : real) : real;
  VAR G : real;
  BEGIN
    IF t < t0 THEN
      G := t - r(5,t)/decayconst[5]
    ELSE
      G := t0 - r(5,t0)*s(5,(t - t0))/decayconst[5];
    G55 := 60*G/decayconst[5];
  END;

  FUNCTION kbquery(msg : string) : boolean;
  VAR  query : char;
  BEGIN
    REPEAT
    WRITE(msg,' Enter Y or N ');
    READLN(query)
    UNTIL query IN ['y','Y','n','N'];
    kbquery := (query IN ['y','Y']);
  END;

  PROCEDURE ScanForTitle (VAR inputfile : text);
  VAR validtitle : boolean;
  BEGIN
      WINDOW(1,1,80,24);
      REPEAT
        READLN(inputfile, titleoftest);
        validtitle := (length(titleoftest) > 0) AND (titleoftest[1] <> '*')
          AND (titleoftest[1] <> ' ');
        IF NOT validtitle THEN WRITELN(results,titleoftest);
        GoToXY(1,23); WRITELN(' ':72);
        GoToXY(1,23); WRITELN (titleoftest);
      UNTIL validtitle OR EOF(inputfile);
  END;

  PROCEDURE transcribe(msg : string; nmbr : integer);
  (* Take a number from a file or kybd, write into a new file *)
  VAR x : real;
  BEGIN
    GoToXY (1,24);
    WRITE(msg, nmbr,' :');
    READ(parameters,x );
    IF EOLN(parameters) THEN READLN(parameters);
    WRITE(results,x:8:3);
```

```
END;

PROCEDURE FixTheString(VAR scratch : string);
VAR i : integer;
BEGIN
  IF Pos('.',scratch) = 1 THEN Insert('0',scratch,1);
  i := Pos(' .',scratch);
  WHILE i > 0 DO
  BEGIN
    Insert('0',scratch,i+1);
    i := Pos(' .',scratch);
  END;
  i := Pos('. ',scratch);
  WHILE i > 0 DO
  BEGIN
    insert('0',scratch,i+1);
    i := Pos('. ',scratch);
  END;
  i := length(scratch);
  IF copy(scratch,i,1) = '.' THEN scratch := scratch + '0';
END;

PROCEDURE initialize;
VAR i : integer;
    scratch : string;
    thisfile, otherfile, fromkbd : boolean;
BEGIN

(*decay constants in inverse minutes *)
  decayconst[1] := LN(2)/3.11;
  decayconst[2] := LN(2)/26.8;
  decayconst[3] := LN(2)/19.9;
  decayconst[4] := LN(2)/638.4;
  decayconst[5] := LN(2)/60.5;

(*alphaenergies in nano joules *)
  alphaenergy[1] := 13.69*1.6021E-4;
  alphaenergy[2] :=  7.69*1.6021E-4;
  alphaenergy[3] :=  7.69*1.6021E-4;
  alphaenergy[4] :=  7.79*1.6021E-4;
  alphaenergy[5] :=  7.79*1.6021E-4;

  halfpage := 0;
  FOR j := 1 to 5 DO eweight[j] := alphaenergy[j]*60.0/decayconst[j];

(*
    This code rewrites files prepared for RWRENNGW.BAS into a
    standard form 'tempfile.dat' for use by the present Pascal
    program.
*)
  ClrScr;
  WRITE('Name of RWRENNGW-compatable data file ');
  READLN(infilename);
  ASSIGN(indata, infilename);  RESET(indata);
  ASSIGN(results,'tempfile.dat');  REWRITE(results);

  ScanForTitle(indata);

  WHILE NOT EOF (indata) DO
  BEGIN

    WRITELN(results,titleoftest);

(*  Find source of parameters - thisfile, otherfile, fromkbd *)
    READLN(indata,scratch);
    thisfile := (Pos('.',scratch) > 0) AND (scratch[1] < 'A') ;
    otherfile  := (scratch[1] >= 'A');
    fromkbd := NOT (thisfile OR otherfile);

    IF otherfile THEN
    BEGIN
      ASSIGN(parameters,scratch);
      RESET(parameters);
      READLN(parameters,scratch);
      CLOSE(parameters);
    END;

    IF thisfile OR otherfile THEN
    BEGIN
      (*  fix the 'naked decimal points' that TURBO doesn't like *)
      FixTheString(scratch);
      WRITELN(results, scratch);
    END;

    IF fromkbd THEN
    BEGIN
      (*  construct and insert the parameters line. *)
      ASSIGN(parameters,'CON');
      RESET(parameters);
      FOR i := 1 TO 5 DO transcribe('count effic',i);
      WRITE(results, 1,1,1,0,0,0);
      transcribe('bkgd count time          ',0);
      transcribe('bkgd count               ',0);
      transcribe('dead time each pulse, us',0);
      transcribe('flowrate, Lpm            ',0);
      transcribe('sampling time, min       ',0);
      transcribe('transfer time, s         ',0);
      transcribe('time per count           ',0);
      transcribe('time between counts      ',0);
      WRITELN(results);
      CLOSE(parameters);
    END;

    REPEAT
      READLN(indata,scratch);
      WRITELN(results,scratch);
    UNTIL scratch[0] = CHR(0);

    ScanForTitle(indata);

  END;
  CLOSE(indata); CLOSE(results);

  ClrScr;
  WRITE('Enter limit on number of EM iterations ');
  READLN(maxiter);
  WRITE('Enter EM convergence criterion ( << 1) ');
  READLN(concriterion);
  WRITELN;
  WRITELN('Data was taken from file      ',infilename);
  WRITE('Enter name of file for output ');
  READLN(outfilename);

  ASSIGN(indata,'tempfile.dat');  RESET(indata);
  ASSIGN(results, outfilename);  REWRITE(results);

END;

PROCEDURE GetNextData;
(* Reads the next block of data from 'tempfile.dat'. *)
VAR i : integer;
    x : real;
BEGIN
  ScanForTitle(indata);

  IF NOT EOF(indata) THEN
  BEGIN
    GoToXY (1,23);
    i := 0;
    WHILE i < 19 DO
    BEGIN
      i := i + 1;
      IF EOLN(indata) THEN READLN(indata);
      READ (indata, x);
      CASE i OF
        1,2,3,4,5       : cnteffic[i]  := x;
(*
        discard the next 6 numbers
*)
        12      : bkgdtime     := x;
        13      : bkgdcount    := TRUNC(x);
        14      : deadtime     := x*1.0E-6/60.0;
        15      : flowrate     := x;
        16      : sampletime   := x;
        17      : transfertime := x/60.0;
        18      : timepercount := x/60.0;
        19      : timebtwncnts := x/60.0
      END;
    END;
    i := 0;
    REPEAT
      WHILE NOT EOLN (indata) DO
      BEGIN
        i := i + 1;
        IF i  <= 2000 THEN READ (indata, counts[i]);
      END;
      READLN(indata);
    UNTIL EOLN(indata);
    IF i < = 2000 THEN numcounts := i ELSE i := 2000;

    IF (counts[1] < 0) THEN
    BEGIN
      FOR i := 2 TO numcounts DO counts[i-1] := counts[i];
      numcounts := numcounts - 1;
      transfertime := transfertime + timepercount + timebtwncnts;
    END;

  END;
END;

PROCEDURE buildthekernel;
BEGIN
    i := 1;
    t0 := sampletime;
    ta := t0 + transfertime;
    inthebag := 0;
    dothoron := (timepercount*numcounts > 300.0);
    IF dothoron THEN numnuclides := 5
    ELSE numnuclides := 3;

    REPEAT
      nextblock := numcounts - inthebag;
      IF nextblock > inthebag THEN nextblock := inthebag;
      IF nextblock = 0 THEN nextblock := 1;

      cntsperblk[i] := 0;
      FOR j := 1 TO nextblock DO
        cntsperblk[i] := cntsperblk[i] + counts[inthebag + j];

      rawdata[i] := (cntsperblk[i] - nextblock*timepercount*bkgdcount/bkgdtime)*
       (1.0 + (nextblock -1)*timebtwncnts/(nextblock*timepercount));

      tb := ta + nextblock*timepercount
          + (nextblock -1)*timebtwncnts;

      kernel[i,1] := G11(tb,t0) - G11(ta,t0)
                   + G31(tb,t0) - G31(ta,t0);
      kernel[i,2] := G32(tb,t0) - G32(ta,t0);
      kernel[i,3] := G33(tb,t0) - G33(ta,t0);
      IF dothoron THEN
      BEGIN
        kernel[i,4] := G54(tb,t0) - G54(ta,t0);
        kernel[i,5] := G55(tb,t0) - G55(ta,t0);
      END;

      FOR j := 1 to numnuclides DO
        kernel[i,j] := kernel[i,j]*cnteffic[j]*flowrate/1000.0;

      GoToXY (1,24);
      WRITE('rawdata[',i:2,'] = ',rawdata[i]:12:3);
      numdatapts := i;

      i := i + 1;
      ta := tb + timebtwncnts;
      inthebag := inthebag + nextblock;
    UNTIL (inthebag = numcounts) OR (i = 12);
END;

PROCEDURE printresults (VAR outpath : text);
BEGIN
  WRITELN(outpath,'*EX-MAX CALCULATION of DECAY PRODUCT CONCENTRATION ... for
      data set:');
  WRITELN(outpath,titleoftest);
  WRITELN(outpath,
      '-------------------------------------------------------------------');


  WRITELN(outpath,' ':33,
          '   Nuclide   Concent.  1-sigma');
  WRITELN(outpath,'Flowrate, Lpm    ', flowrate:14:2,'    ',
          '    Po-218 ',dpconc[1]:9:3,stderr[1]:9:3,' Bq/m3');
  WRITELN(outpath,'Sample time, min', sampletime:14:2,'    ',
          '    Pb-214 ',dpconc[2]:9:3,stderr[2]:9:3,' Bq/m3');
  WRITELN(outpath,'Transfer time, s',ROUND(60*transfertime):14,'
          '    Bi-214 ',dpconc[3]:9:3,stderr[3]:9:3,' Bq/m3');


  IF dothoron THEN
  BEGIN
    WRITELN(outpath,' ':33,
            '    Pb-212 ',dpconc[4]:9:3,stderr[4]:9:3,' Bq/m3');
    WRITELN(outpath,' ':33,
            '    Bi-212 ',dpconc[5]:9:3,stderr[5]:9:3,' Bq/m3');
  END;


  WRITELN(outpath,' ':33,
          '    WtdAve ',dpconc[6]:9:3,stderr[6]:9:3,' Bq/m3');
  WRITELN(outpath);
  WRITELN(outpath,' ':33,
          '    PAEC   ',dpconc[7]:9:3,stderr[7]:9:3,' nJ/m3');
  WRITELN(outpath,' ':33,
          '    PAEC   ',dpconc[8]:9:3,stderr[8]:9:3,' mWL');
  WRITELN(outpath);
  WRITELN(outpath,' ':33,' Block  Counts  RawData   FitData');
  FOR i := 1 TO numdatapts DO
  BEGIN
    CASE i OF
      1 : WRITE(outpath,'Number of count intervals',numcounts:5,'   ');
      2 : WRITE(outpath,'Calculation done (Yr,Mo,Dy)      ');
      3 : WRITE(outpath,' ':17,year:5,month:4,day:4,'   ');
      4 : WRITE(outpath,'Convergence criterion',concriterion:9:5,'   ');
      5 : WRITE(outpath,'Iterations (max',maxiter:6,')',lastiter:8,'   ');
    ELSE
      WRITE(outpath,' ':33);
    END;
    WRITELN(outpath,i:5, cntsperblk[i]:8, rawdata[i]:10:2, fitdata[i]:10:2);
  END;

  chisqr := 0.0;
  FOR i := 1 TO numdatapts DO
    chisqr := chisqr + SQR(rawdata[i] - fitdata[i])/fitdata[i];
  WRITELN(outpath,' ':43,'     ChiSqr =',chisqr:10:2);


  WRITELN(outpath,
      '-------------------------------------------------------------------');
END;

BEGIN  (* MAIN PROGRAM *)

  ClrScr;
  Window(14,10,80,25);
  GetDate(year, month, day, dayofweek);
  WRITELN('EM Calculation of Rn-Th Decay Product Concentration');
  WRITELN;
  WRITELN('      Pascal Program by E.O. Knutson, 1988');
  WRITELN;
  WRITELN(' Patterned after Maher and Laird''s 1985 paper on');
  WRITELN('     unfolding data from diffusion batteries.');
  WRITELN;
  IF NOT kbquery('Ready to start?') THEN EXIT;

  Window(1,1,80,25);
  initialize;

  GetNextData;
  REPEAT

    buildthekernel;

    initguess := counts[1]*(1000.0/flowrate/sampletime)
                 /(cnteffic[1]*timepercount*60.0);
    FOR j := 1 TO 8 DO
    BEGIN
      dpconc[j] := initguess;
      IF j > numnuclides THEN dpconc[j] := 0.0;
      stderr[j] := -1.0;
    END;

    IF dothoron THEN  (* do 20 iterations to get better start on Bi-212 *)
    BEGIN
       ExpectMax (rawdata, numdatapts, numnuclides, 20,
       0.00001, kernel, dpconc, lastiter, fitdata, ok);
       dpconc[5] := dpconc[4];
    END;

    ExpectMax (rawdata, numdatapts, numnuclides, maxiter,
               concriterion*initguess, kernel, dpconc, lastiter, fitdata, ok);
(*
    In this case, for sure, the data itself is a good estimate
    of the variance; hence 'fitdata' in the PROC call below.
 *)

    StandardErr(fitdata, numdatapts,numnuclides, kernel, stderr, ok);

    sum := 0.0;
    dpconc[7] := 0.0;
    FOR j := 1 TO numnuclides DO
    BEGIN
      dpconc[7] := dpconc[7] + dpconc[j]*eweight[j];
      sum := sum + eweight[j];
    END;
    dpconc[6] := dpconc[7]/sum;
    dpconc[8] := dpconc[7]/20.8;

    temp := 0.0;
    FOR j := 1 TO numnuclides DO
      FOR k := 1 TO numnuclides DO
        temp := temp + eweight[j]*eweight[k]*matrix2[j,k];

    stderr[7] := SQRT(temp);
    stderr[6] := stderr[7]/sum;
    stderr[8] := stderr[7]/20.8;

    ClrScr;
    printresults(output);
    halfpage := halfpage + 1;
    IF (halfpage MOD 2) = 1  THEN WRITELN(results, '*n*');
    printresults(results);

    getnextdata;

  UNTIL EOF(indata);
  CLOSE(results);

END.
```

Appendix F
File ExMaxGS.Pas

```
PROGRAM ExMaxGradedScreen;          (* file name ExMaxGS.PAS *)
(*==============================================================
    This program is for use in unfolding Graded Wire Screen data.
    There is a choice of using the Twomey algorithm or the
    Expectation-Maximization algorithm (which code is based on
    Edward Maher's Fortran program for diffusion battery data).
    The whole package is very similar to my 1/88 Pascal program
    ExMaxDB.

    Original Maher program : December, 1983
    This Pascal program    : April, 1988
    Refinements            : June, 1988

==============================================================*)
Uses  Crt, Dos, mtrx, algorithm, globals, datamessage;

TYPE   arrytyp4 = ARRAY ['A'..'J'] OF real;

VAR titleoftest                : string;
    scrnsdata, results         : text;

    rawdata, fitdata, rawerror,
                datavariance : datavector;

    spectrumpt, spectrum,
              stderr           : rsltvector;

    kernel                     : kernelmatrix;

    wirediam, thickness,
                   solidfrac : arrytyp4;

    scrnconfig, infilename,
              outfilename    : string;

    maxmaher,maxtwmy,
        lastmaher, errorscale  : integer;

    i,j, numspecpts, numdatapts: integer;

    oksofar, errorterms,
                  savematrix : boolean;

    flowrate, flowarea,
      concriterion, twmyspeed,
      cap, rawmax              : real;

    gsident                    : char;

FUNCTION kbquery(msg : string) : boolean;
VAR  query : char;
BEGIN
  REPEAT
  WRITE(msg,' enter Y or N ');
  READLN(query)
  UNTIL query IN ['y','Y','n','N'];
  kbquery := (query IN ['y','Y']);
END;

PROCEDURE ScanForTitle (VAR inputfile : text);
VAR validtitle : boolean;
BEGIN
    WINDOW(1,1,80,24);

    REPEAT
      READLN(inputfile, titleoftest);
      validtitle := (length(titleoftest) > 0) AND (titleoftest[1] <> '*')
        AND (titleoftest[1] <> ' ');
      IF NOT validtitle THEN WRITELN(results,titleoftest);
      GoToXY(1,23); WRITELN(' ':72);
      GoToXY(1,23); WRITELN (titleoftest);
    UNTIL validtitle OR EOF(inputfile);
END;

PROCEDURE StartGradScrn;

(*  set largest and smallest diffusion coefficient, cm2 s-1  *)
CONST  maxspecpt = 0.08;
       minspecpt = 8E-5;

VAR gstep : real;
    j : integer;

PROCEDURE countandcnvt (VAR strng : string; Var count : integer);
VAR i : integer;
BEGIN
  count := 0;
  FOR i := 1 TO ORD(strng[0]) DO
    BEGIN
      IF strng[i] in ['a'..'z'] THEN
         strng[i] := CHR(ORD(strng[i]) - 32);
      IF strng[i] in ['A'..'Z'] THEN
         count := SUCC(count);
    END;
END;

PROCEDURE emlscreens; (*  wirediam, thickness incm  *)
```

```
BEGIN
    wirediam['A'] := 0.039;  thickness['A'] := 0.0838;  solidfrac['A'] := 0.215;

    wirediam['B'] := 0.016;  thickness['B'] := 0.0356;  solidfrac['B'] := 0.36;

    wirediam['C'] := 0.010;  thickness['C'] := 0.0249;  solidfrac['C'] := 0.308;

    wirediam['D'] := 0.0040; thickness['D'] := 0.0135;  solidfrac['D'] := 0.275;

    wirediam['E'] := 0.0020; thickness['E'] := 0.0050;  solidfrac['E'] := 0.345;

END;

BEGIN

    ClrScr;
    Window(16,1,80,25);
    WRITELN('Analysis of Data from Graded Wire Screens');
    WRITELN('Based on Cheng-Yeh equation for screen efficiency');
    WRITELN('and on Holub-Knutson-ACS front to back ratios');
    WRITELN('Choice of Twomey or Expectation-Maximization');
    WRITELN('iteration algorthms - or both!');
    WRITELN;
    WRITELN('Pascal program by Earl O. Knutson');
    WRITELN('  EML/USDOE,  April, 1988');
    WRITELN;
    IF kbquery('Need a reminder on data format?') THEN
    BEGIN
     dataformat;
     IF NOT kbquery('Are you ready to continue?') THEN HALT;
    END;
    Window(1,1,80,25);
    ClrScr;
    emlscreens;
    WRITELN('Standard Screens:  A - EML 20 mesh');
    WRITELN('                   B - EML 60 mesh');
    WRITELN('                   C - EML 100 mesh');
    WRITELN('                   D - EML 200 mesh');
    WRITELN('                   E - EML 635 mesh');
    WRITELN('                   F - filter');
    WRITELN;
    gsident := 'F';
    WHILE kbquery('Are there any other screens') DO
      BEGIN
        gsident := SUCC(gsident);
        WRITE('screen ',gsident,'  diameter of wires in cm ');
        READLN(wirediam[gsident]);
        WRITE('          thickness of screen in cm ');
        READLN(thickness[gsident]);
        WRITE('          solid fraction            ');
        READLN(solidfrac[gsident]);
      END;
```

```
REPEAT
WRITELN;
WRITELN('Enter the screens used (from the set A to ',gsident,'), and show');

WRITELN('their configuration.  Use contiguous letters (no spaces) to');
WRITELN('indicate "arranged in series";  use spaces to indicate "arranged');

WRITELN('in parallel."  For series arrangement, assume airflow is left to
  right.');
WRITELN('Example: BCD F means filter in parallel with the series group
  BCD.');
WRITELN;
WRITE('-+-+-+- Enter configuration --> ');
READLN(scrnconfig);
countandcnvt (scrnconfig, numdatapts);
WRITELN(scrnconfig,'       Number of collectors = ',numdatapts:3);
WRITELN;
UNTIL NOT kbquery('Would you like to reconsider?');

ClrScr;
WRITELN('enter filenames - ');
WRITE('  - of input   ');
READLN(infilename);
WRITE('  - for output ');
READLN(outfilename);
errorterms := kbquery('error terms in input file?');
savematrix := kbquery('save matrices with output');
WRITELN;
WRITE('Enter flowrate, Lpm   ');
READLN(flowrate);
WRITE('Enter flow area of screens in cm2 (e.g., 2.78) ');
READLN(flowarea);
WRITELN;
WRITE('no. of spectrum points for the integration  ');
READLN(numspecpts);
IF numspecpts > numdatapts THEN
  BEGIN
    WRITELN('....Since numspecpts > numdatapts, ExMax');
    WRITELN('....error calculation will not be done.');
    WRITELN;
  END;

WRITELN('enter max # of iterations,=>0');
WRITE('  - for twomey        ');
READLN(maxtwmy);
WRITE('  - for expect. max.  ');
READLN(maxmaher);

twmyspeed := 0.0;
IF maxtwmy > 0 THEN
  BEGIN
    WRITE('enter twomey speed factor ');
    READLN(twmyspeed);
```

```
      END;

    concriterion := 0.0;
    IF maxmaher > 0 THEN
      BEGIN
        WRITE('enter exmax conv. crit. ');
        READLN(concriterion);
      END;

    IF (maxmaher > 0) AND (numspecpts < = numdatapts) AND
      (NOT errorterms) THEN
        BEGIN
          WRITELN;
          WRITELN('enter error scaling number, < 32768');
          WRITE('  (0 = autoscaling;  negative = skip error calc   ');
          READLN(errorscale);
         END
       ELSE errorscale := -1;

    randomize;

    spectrumpt[1] := minspecpt;
    gstep := ln(maxspecpt/minspecpt)/(numspecpts - 1);
    gstep := exp(gstep);

    FOR j := 2 TO numspecpts DO spectrumpt[j] := gstep*spectrumpt[j-1];
    WRITELN;

END;

PROCEDURE GetGSMatrix;

VAR nextscrn : char;
    i, j, k    : integer;
    penproduct : ARRAY [1..16] OF real;
    velocity, forwardalphas, pen : real;

FUNCTION ChengYeh (diffcoef, facevel, wirediam,
                   thickness, solidfrac : real) : real;
CONST pi = 3.14159;
VAR   B, Pe  : real;
BEGIN
  B := 4*solidfrac*thickness / pi / (1 - solidfrac) / wirediam;
  Pe := facevel*wirediam / diffcoef;
  ChengYeh := EXP(-2.7*B / EXP(2*LN(Pe)/3));
END;

BEGIN
  velocity := 1000*flowrate/60/flowarea;
  INSERT(' ', scrnconfig, 1);      (* make sure we lead with a space *)
  i := 0;  k := 0;
  WHILE i < ORD(scrnconfig[0]) DO
  BEGIN
```

```
    i := i + 1;
    nextscrn := scrnconfig[i];
    If nextscrn in ['A'..'J'] THEN k := k + 1;
    CASE nextscrn OF
     ' ' : FOR j := 1 TO numspecpts DO penproduct[j] := 1;
            (* means first collector in a series *)
     'F' : FOR j := 1 TO numspecpts DO
           BEGIN
             kernel[k,j] := penproduct[j];
             penproduct[j] := 0.0
           END
    ELSE
    BEGIN
      FOR j := 1 TO numspecpts DO
      BEGIN
        pen := ChengYeh(spectrumpt[j], velocity, wirediam[nextscrn],
                        thickness[nextscrn], solidfrac[nextscrn]);
        forwardalphas := 0.85*(1 - pen);
        IF forwardalphas < 0.67 THEN forwardalphas := 0.67;
(*
        Note: 'forwardalphas' refers to that fraction of the screen alpha
        activity which impacts on the detector.  More precisely, it is the
        ratio of the count rate from the deposit on the screen to the count
        rate for the same activity deposited on the face of a filter.

        The equation just above for forwardalphas is derived from the front to
        back ratios in the Holub-Knutson ACS paper.  The 1.07 loss correction
        is incorporated.
*)
        kernel[k, j] := penproduct[j]*forwardalphas*(1 - pen);
        penproduct[j] := penproduct[j]*pen
      END;
      END;
    END;
  END;
END;


PROCEDURE PrintResults (spectrum, stderr : rsltvector;
                        rawdata : datavector;
                        lastiter : integer;
                        VAR outpath : text;
                        ok : boolean);

VAR i, j, k      : integer;
    bigg, logstep, chisqr       : real;
    year, month, day, dayofweek : word;

BEGIN

  GetDate(year,month,day,dayofweek);
  logstep := Ln(spectrumpt[2]/spectrumpt[1])/Ln(10);
  forwardcalc(spectrum, numdatapts, numspecpts, kernel, fitdata);
  WRITELN(outpath, titleoftest);
  WRITELN(outpath,'Screens Used: ',scrnconfig,
               ';                  Flowrate: ', flowrate:6:2,' Lpm');
  WRITELN(outpath,'Calculated yr,mo,dy: ',year:5,month:3,day:3,
                '      Error scale factor: ',errorscale:6);

  IF NOT ok THEN
    WRITELN(outpath,'  Calculation did not converge as it should');

    bigg := 0.0;
(*  convert to DA/DlogDC & find peak value  *)
    FOR j := 1 TO numspecpts DO
    BEGIN
      spectrum[j] := spectrum[j]/logstep;
      IF stderr[j] > 0 THEN stderr[j] := stderr[j]/logstep;
      IF spectrum[j] > bigg THEN bigg := spectrum[j];
    END;

    WRITELN(outpath);
    WRITELN(outpath,'DC,cm2/s DA/DlogDC  1-sigma');


    FOR j := 1 TO numspecpts DO
    BEGIN
      WRITE(outpath,spectrumpt[j]:7, spectrum[j]:10:2,
            stderr[j]:8:2, '  *');

      k := ROUND(50.0*spectrum[j]/bigg);
      i := 1;
      WHILE i  <= k DO
      BEGIN
        IF (i MOD 5) = 0 THEN WRITE(outpath,'*')
        ELSE WRITE(outpath, '+');
        i := i + 1;
      END;
      WRITELN(outpath);
    END;

    WRITELN(outpath);
    WRITELN(outpath,'TAP      RAW      1-sigma      FIT');
    FOR i := 1 TO numdatapts DO
    BEGIN
      WRITE(outpath,i:3,rawdata[i]:10:2,rawerror[i]:10:2,fitdata[i]:10:2);
        CASE i OF
        1 : WRITE(outpath,'  Twomey iterations:   ',maxtwmy:6);
        2 : WRITE(outpath,'           speedfactor: ',twmyspeed:6);
        3 : WRITE(outpath,'  ExMax iterations:    ',lastiter:6);
        4 : WRITE(outpath,'          concriterion: ',concriterion:6);
        5 : WRITE(outpath,'  test of matrix inver',matproof:8)
        END;
        WRITELN(outpath)
    END;
    IF errorterms THEN
    BEGIN
      chisqr := 0.0;
      FOR i := 1 TO numdatapts DO chisqr := chisqr +
        SQR((rawdata[i] - fitdata[i])/rawerror[i]);
      WRITELN(outpath, ' ':13,' ChiSqr = ',chisqr:10:2);
    END;

    FOR j := 1 to 78 DO WRITE(outpath,'-'); WRITELN(outpath);
    WRITELN(outpath);

END;


PROCEDURE PrintPenMatrix;
VAR  i,j : integer;
BEGIN

    FOR i := 1 TO numdatapts DO
    BEGIN
      FOR j := 1 TO numspecpts DO
        WRITE(results, kernel[i,j]:12);
      WRITELN(results);
    END;
    WRITELN(results);

END;

PROCEDURE PrintErrMatrices;
VAR  i,j : integer;
BEGIN

    FOR i := 1 TO numspecpts DO
    BEGIN
      FOR j := 1 TO numspecpts DO
        WRITE(results, matrix1[i,j]:12);
      WRITELN(results);
    END;
    WRITELN(results);

    FOR i := 1 TO numspecpts DO
    BEGIN
      FOR j := 1 TO numspecpts DO
        WRITE(results, matrix2[i,j]:12);
      WRITELN(results);
    END;
    WRITELN(results);
    WRITELN(results);

END;
BEGIN  (*  PROGRAM ExMaxGradedScreen  *)

  StartGradScrn;
```

```
WRITELN('Start finished');
GetGSMatrix;

Assign(scrnsdata, infilename);
RESET(scrnsdata);
Assign(results, outfilename);
REWRITE(results);
ClrScr;

ScanForTitle(scrnsdata);

WHILE NOT EOF (scrnsdata) DO
BEGIN

  i := 0;
  WHILE NOT EOLN (scrnsdata) DO
  BEGIN
    i := i + 1;
    READ (scrnsdata, rawdata[i]);
    IF errorterms THEN READ(scrnsdata, rawerror[i])
    ELSE rawerror[i] := -1.0;
  END;

  READLN (scrnsdata);
  oksofar := (i = numdatapts);

  rawmax := 0.0;
  FOR i := 1 TO numdatapts DO
    IF rawmax < rawdata[i] THEN rawmax := rawdata[i];

  IF oksofar THEN
    FOR i := 1 TO numspecpts DO
    BEGIN
      spectrum[i] := rawmax/numspecpts;
      stderr[i] := -1.0;
    END;

  IF oksofar THEN
    TwomeyAlg (rawdata, numdatapts, numspecpts, maxtwmy,
               twmyspeed, kernel, spectrum, oksofar);

  lastmaher := 0;
  IF oksofar THEN
    ExpectMax (rawdata, numdatapts, numspecpts, maxmaher,
               concriterion*rawmax, kernel, spectrum,
               lastmaher, fitdata, oksofar);

  matproof := 0.0;
  IF oksofar AND (numspecpts < = numdatapts) AND (lastmaher > 0) THEN
  BEGIN
    cap := 0.0;
    FOR i := 1 TO numdatapts DO
      IF fitdata[i] > cap THEN cap := fitdata[i];
```

```
(*==================================================================
      As seen below, we provide three different ways to estimate the
      input data variances needed to calculate standard errors:

        First, if error terms are available in the input data stream
        we use the square of those terms as the variance vector for
        the input data;

        Second, if error terms are not available and we have specified
        a zero error scale factor in the keyboard input, we use the
        fitted data vector as the estimate of variance;

        Third, if the error scale factor set positive, the fitdata vector
        is rescaled so that its maximum component is equal to the error
        scale factor, and this is used as the variance estimate.

      The second method - the one used by Ed Maher - is valid only if
      the input data are actual raw counts.  This is a necessary condition
      for the Poisson distribution to be valid.  (Even here, there are
      probably other sources of variability that swamp the Poisson
      variability.)

      Options one and three are provided because, in our opinion, the
      Poisson assumption is rarely valid for data from diffusion
      batteries.  We realize that there is a logical inconsistency
      in the idea of maximizing the Poisson-based likelihood function
      but using a non-Poisson estimate of variance.  So be it - half
      a loaf is better than none.
===================================================================*)


      IF errorterms THEN
        FOR i := 1 TO numdatapts DO datavariance[i] := SQR(rawerror[i]);

      IF NOT errorterms AND (errorscale = 0) THEN
        FOR i := 1 TO numdatapts DO datavariance[i] := fitdata[i];

      IF NOT errorterms AND (errorscale > 0) THEN
        FOR i := 1 TO numdatapts DO datavariance[i]
                                  := fitdata[i]*errorscale/cap;

      StandardErr (datavariance, numdatapts, numspecpts, kernel,
                   stderr, oksofar);

      IF NOT errorterms AND (errorscale > 0) THEN
        FOR j := 1 TO numspecpts DO stderr[j] := stderr[j]*cap/errorscale;

    END;

    WINDOW(1,1,80,22); ClrScr;
    PrintResults (spectrum, stderr, rawdata, lastmaher, output,oksofar);
    PrintResults (spectrum, stderr, rawdata, lastmaher, results,oksofar);
```

```
      IF savematrix THEN
        BEGIN
          PrintPenMatrix;
          IF (lastmaher > 0) AND (numspecpts < = numdatapts) THEN
            PrintErrMatrices;
        END;

      ScanForTitle(scrnsdata);

    END;
    CLOSE(results);
    WINDOW(1,1,80,24);
    GoToXY(1,23); WRITELN(' ':72);
    WRITE('***   F I N I    ***');

  END.
```

Appendix G
File Globals.Pas

```
UNIT globals;
INTERFACE
TYPE   rsltvector = ARRAY [1..16] OF real;
       datavector = ARRAY [1..12] OF real;
       kernelmatrix = ARRAY [1..12,1..16] OF real;
IMPLEMENTATION
BEGIN
END.
```

Appendix H
File Mtrx.Pas

```
UNIT mtrx;
INTERFACE

VAR matrix1, matrix2 : ARRAY [1..10,1..10] OF real;
    matrixsize : integer;
    matproof   : real;
    matsingular: boolean;

PROCEDURE MatInvert;

IMPLEMENTATION

PROCEDURE MatInvert;

(*
       MATRIX INVERSION WITH FULL PIVOTING FOR SIZE.

       From a BASIC program by Brian J. Flynn in the
           October, 1981 issue of COMPUTE!.
       Transcribed into Pascal by Earl O. Knutson,
           January, 1988.
*)

CONST  epsilon1 = 1.0E-20;

TYPE arrytyp1 =  ARRAY [1..10,1..10] OF real;

VAR    i, j, pivotpoint, col, row, l, last : integer;
       switchmap : ARRAY [1..10] of integer;
       matrix3   : arrytyp1;
       pivot, t  : real;


BEGIN

  FOR i := 1 TO matrixsize DO
  BEGIN
    FOR j := 1 TO matrixsize DO
    BEGIN
      matrix2[i,j] := matrix1[i,j];
      matrix3[i,j]  := 0;
    END;
    matrix3[i,i] := 1;
    switchmap[i] := i;
```

```
   END;

  pivotpoint := 1;
  REPEAT
    pivot := matrix2[pivotpoint,pivotpoint];

    IF pivotpoint < matrixsize THEN
    BEGIN
(*
      Find pivot element, then reposition row or column.
*)
      col := pivotpoint;
      row := pivotpoint;

      FOR i := pivotpoint + 1 TO matrixsize DO
      BEGIN
        IF ABS(matrix2[i,pivotpoint]) > ABS(pivot) THEN
        BEGIN
          pivot := matrix2[i,pivotpoint];
          row := i; col := pivotpoint;
        END;
        IF ABS(matrix2[pivotpoint,i]) > ABS(pivot) THEN
        BEGIN
          pivot := matrix2[pivotpoint,i];
          row := pivotpoint; col := i;
        END;
      END;

      IF row > pivotpoint THEN
      BEGIN
        FOR i := 1 TO matrixsize DO
        BEGIN
          t := matrix2[pivotpoint,i];
          matrix2[pivotpoint,i] := matrix2[row,i];
          matrix2[row,i] := t;
          t := matrix3[pivotpoint,i];
          matrix3[pivotpoint,i] := matrix3[row,i];
          matrix3[row,i] := t;
        END;
      END;


      IF col > pivotpoint THEN
      BEGIN
        FOR i := 1 TO matrixsize DO
        BEGIN
          t := matrix2[i,pivotpoint];
          matrix2[i,pivotpoint] := matrix2[i,col];
          matrix2[i,col] := t;
        END;
        l := switchmap[pivotpoint];
        switchmap[pivotpoint] := switchmap[col];
        switchmap[col] := l;
      END;
    END;  (* END OF IF pivotpoint < matrixsize *)

 (*
    Pivot point found and row/column repostioned.
    Now do the Gauss-Jordan row arithmetic.
 *)
    matsingular := (ABS(pivot) < epsilon1);
    IF matsingular THEN
    BEGIN
      WRITELN('MATINVERT ABORTED - INPUT WAS SING.');
      EXIT;
    END
    ELSE
    BEGIN
      FOR j := pivotpoint TO matrixsize DO
        matrix2[pivotpoint,j] := matrix2[pivotpoint,j]/pivot;
      FOR j := 1 to matrixsize DO
        matrix3[pivotpoint,j] := matrix3[pivotpoint,j]/pivot;

      IF pivotpoint < matrixsize THEN last := matrixsize
      ELSE last := matrixsize - 1;

      FOR l := 1 TO last DO
      BEGIN
        IF l <> pivotpoint THEN
        BEGIN
          t := matrix2[l,pivotpoint];
          FOR j := pivotpoint TO matrixsize DO
            matrix2[l,j] := matrix2[l,j] - t*matrix2[pivotpoint,j];
          FOR j := 1 TO matrixsize DO
            matrix3[l,j] := matrix3[l,j] - t*matrix3[pivotpoint,j];
        END;
      END;
    END;

  pivotpoint := pivotpoint + 1;
  UNTIL (pivotpoint > matrixsize);

 (*
  unscramble rows of the inverted
  matrix and store the result in matrix2.
 *)

    FOR i := 1 TO matrixsize DO
      FOR j := 1 TO matrixsize DO
        matrix2[switchmap[i],j] := matrix3[i,j];

 (*
   Test the inverse matrix by forming the product with the
   original matrix, and checking for clean "0"s and "1"s.
 *)
    matproof := 0.0;
    FOR i := 1 TO matrixsize DO
    BEGIN
      FOR j := 1 TO matrixsize DO
      BEGIN
        IF i = j THEN t := -1 ELSE t := 0;
        FOR l := 1 TO matrixsize DO
          t := t + matrix1[i,l]*matrix2[l,j];

        IF ABS(t) > matproof THEN matproof := ABS(t);
      END;
    END;
END;
END.
```

Appendix I
File RWRENN.TXT

This is a new rendering of the classical Raabe & Wrenn least squares program for calculating radon and thoron daughter concentrations from sequential gross alpha counts after sampling on filters. The reference is Health Physics, 17, 593-605, 1969.

This program closely follows the R-W paper, including the naming of variables. However, the automatic inclusion/dropping of nuclides has not been implemented since we think this requires judgement. Also, R-W are not very specific about how to propagate the error terms for N-bar's back through the equations to get the errors in the air concentrations, C. This program follows a suggestion by C. V. Gogolak of EML that the N-bar covariances, as well as their variances, need to be propagated.

The input data is assumed to be in an ASCII disk file, and you will be prompted for its name. The first line must be a title line <80 characters long (avoid using "*" or a "blank space" as the first character). The second line specifies the source of the nineteen input parameters needed to do the calculation:

> if the second line consists of a file name, the parameters will be read from that file (which must contain exactly 19 numbers);
>
> if the second line consists of numbers the first of which is a fraction, the first 19 numbers will be used as the parameters;
>
> if the first number is an integer, all input is stored as count data; you will be prompted to enter parameters from the keyboard.

The actual count data begins (or continues) on the third line, one or more integers per line. Within each line of numeric data, numbers are demarked by commas or - if no comma is found - by spaces. The final line of each block of data must be a nul line, or an end-of-file mark.

The nineteen parameters are:

1-5
Counter efficiency (fractional, not percent) for -
^{218}Po, ^{214}Po, ^{212}Bi, ^{212}Po, Nuclide X.

6-11
Which nuclides to analyze for - (enter 1 to analyze, 0 to skip)
^{218}Po, ^{214}Pb, ^{214}Po, ^{212}Bi, ^{212}Po, Nuclide X

12-19

background count time, min,	TBG
counter dead time, microsec.,	TAU : TAU = .000001*TAU/60
flow rate, liters/min,	FLOWR
sampling time, min,	TSAMP
sample transfer time, sec,	TTRANS : TTRANS = TTRANS/60
length of each count, sec,	TCOUNT : TCOUNT = TCOUNT/60
down time between counts, sec,	TDOWN : TDOWN = TDOWN/60

In the last eight lines, the BASIC variables are shown; note that all times are stored as minutes.

If the first count datum is negative, it is discarded and the TTRANS corrected by adding TCOUNT and TDOWN to it. This allows the user to delete the first count by prefixing a minus sign.

Appendix J
File RWRENNGW.BAS

```
10 '=============== E.O.Knutson, USDOE/EML, 9/86 ==================
20 'CLEAR 2000 : '      (minor changes made 11/87, 3/88)
30 GOSUB 3360 : '     Sign-on banner and formalities
40 GOSUB 3180 : '     Define variables and set dimensions
50 GOSUB 2490 : '     Input next set of count data
60 GOSUB 2940 : '     Input counter effic's & do prelim. cals's
70 GOSUB 2770 : '     Input count timing and count bkgd
80 GOSUB 390 : '      Correct for deadtime & bkgd; get weight factors
90 GOSUB 240 : '      Compute S-matrix
100 GOSUB 530 : '     Compute A-matrix and B vector
110 GOSUB 670 : '     Format A-matrix for Gauss-Jordan
120 GOSUB 770 : '     Gauss-Jordan inversion routine
130 IF AMAX >         .0001 THEN 210 :'inversion failed, abort calc
140 GOSUB 1230 : '    Compute NBAR and its std error
150 GOSUB 1480 : '    Show NBAR, SE, AND variance-covariance matrix
160 GOSUB 1560 : '    Build matrix for NBAR-to-C step
170 GOSUB 770 : '     Call Gauss-Jordan again
180 GOSUB 1730 : '    Compute air conc's & their std errors
190 GOSUB 2080 : '    Send results to screen and disk
200 GOTO 50
210 GOSUB 2430 : '    Print bad news message to screen & disk
220 GOTO 50
230 '======================================================
240 DEF FNX(J) = EXP(-LAMBDA(J)*BETA(I)) - EXP(-LAMBDA(J)*ALPHA(I))
250 FOR I = 1 TO N%
260 LOCATE 3,12 : PRINT "Compute S-matrix, step";I
270 SA = - (F1 + F2*R2/LAMBDA(1))*FNX(1)
280 IF KY(1) = 0 THEN 320
290 SB = - F2*R1/LAMBDA(2)*FNX(2)
300 SC = F2*(R1+R2)/LAMBDA(3)*FNX(3)
310 S(1,I) = SA + SB + SC
320 IF KY(2) = 1 THEN S(2,I) = - F2*R3/LAMBDA(2)*FNX(2) + F2*R3/LAMBDA(3)*FNX(3)
330 IF KY(3) = 1 THEN S(3,I)  = - F2*FNX(3)
340 IF KY(4) = 1 THEN S(4,I) = F3*R4/LAMBDA(4)*FNX(4) - F3*R4/LAMBDA(5)*FNX(5)
350 IF KY(5) = 1 THEN S(5,I) = - F3*FNX(5)
360 IF KY(6) = 1 THEN S(6,I) = - F4*FNX(6)
370 NEXT I : RETURN
380 '======================================================
390 IF CO(1) >= 0 THEN 430
400 FOR I = 1 TO N%-1 : CO(I) = CO(I+1) : NEXT I
410 N% = N% -1
420 TTRANS = TTRANS + TCOUNT + TDOWN
430 FOR I = 1 TO N%
440 LOCATE 2,12 : PRINT "Compute weights & correct for bgd, step";I
```

```
450 CT = CO(I)/(1 - CO(I)*TAU/(BETA(I) - ALPHA(I)))
460 BG = CBG/TBG*(BETA(I) - ALPHA(I))
470 D(I) = CT - BG
480 SBG2 = ((BETA(I) - ALPHA(I))/TBG)^2*CBG
490 IF CO(I) <> 0 THEN SD2 = CT*CT/CO(I) + SBG2
500 W(I) = 1/SD2
510 NEXT I : RETURN
520 '=====================================================
530 FOR J = 1 TO 6 : FOR JP = J TO 6
540 A(J,JP) = 0 : IF KY(J)*KY(JP) = 0 THEN 570
550 FOR I = 1 TO N%
560 A(J,JP) = A(J,JP) + W(I)*S(J,I)*S(JP,I) : NEXT I
570 A(JP,J) = A( J,JP)
580 LOCATE 4,12 : PRINT "Compute A-matrix, element";J;JP;
590 PRINT USING" #.##^^^^";A(J,JP) : NEXT JP : NEXT J
600 '=====================================================
610 FOR J = 1 TO 6
620 B(J) = 0 : IF KY(J) = 0 THEN 650
630 FOR I = 1 TO N%
640 B(J) = B(J) + W(I)*D(I)*S(J,I) : NEXT I
650 NEXT J : RETURN
660 '=====================================================
670 K = 0 : FOR J = 1 TO 6
680 IF KY(J) = 1 THEN K = K+1 : JM(K) = J
690 NEXT J : J9 = K
700 FOR J = 1 TO 6 : FOR JP = 1 TO 6
710 AX(J,JP) = 0 : NEXT JP,J
720 FOR J = 1 TO K : FOR JP = 1 TO K
730 AX(J,JP) = A(JM(J),JM(JP))
740 NEXT JP, J
750 RETURN
760 '=====================================================
770 FOR J = 1 TO K : FOR JP = 1 TO K
780 X(J,JP) = AX(J,JP) : X(J,K+JP) = -(J=JP)
790 NEXT JP : M%(J) = J : NEXT J
800 'INVERT MATRIX, PER ARTICLE IN COMPUTE! MAGAZINE, 10/81
810 FOR Q% = 1 TO K : LOCATE 5,12 : PRINT "Inverting matrix, step";Q%
820 IF Q% = K THEN 910
830 HE = ABS(X(Q%,Q%)) : HR = 0 : HC = 0
840 FOR I = 1 TO K - Q%
850 DV = ABS(X(Q% + I,Q%)) : IF DV>HR THEN HR = DV : R% = Q% + I
860 DV = ABS(X(Q%,Q% + I)) : IF DV>HC THEN HC = DV : C% = Q% + I
870 NEXT I
880 IF HE >= HR AND HE >= HC THEN 910
890 IF HR >= HC THEN FOR J = 1 TO 2*K : HO = X(R%,J) : X(R%,J) = X(Q%,J) :
      X(Q%,J) = HO : NEXT J
900 IF HR<HC THEN FOR J = 1 TO K : HO = X(J,C%) : X(J,C%) = X(J,Q%) : X(J,Q%) =
      HO : NEXT J : H1% = M%(Q%) : M%(Q%) = M%(C%) : M%(C%) = H1%
910 B = X(Q%,Q%) : IF B = 0 THEN PRINT"SINGULAR MATRIX" : STOP
920 FOR J = Q% TO 2*K
930 X(Q%,J) = X(Q%,J)/B
940 NEXT J
950 FOR L% = 1 TO K
```

```
960 IF L% = K AND  K = Q% THEN 1020
970 IF L% = Q% THEN L% = L% + 1
980 D = X(L%,Q%)
990 FOR J = 1 TO 2*K
1000 X(L%,J) = X(L%,J)-D*X(Q%,J)
1010 NEXT J
1020 NEXT L%,Q%
1030 FOR I = 1 TO K
1040 C% = 0
1050 FOR J = 1 TO K
1060 IF M%(J) = I THEN C% = J
1070 NEXT J
1080 IF C<>I THEN FOR L% = 1 TO K : HO = X(I,K + L%) : X(I,K + L%) = X(C%,K +
      L%) : X(C%,K + L%) = HO : NEXT L% : H1% = M%(I) : M%(I) = M%(C%) : M%(C%)
      = H1%
1090 NEXT I
1100 'CHECK QUALITY OF INVERSE
1110 AMAX = 0
1120 FOR I = 1 TO K
1130 FOR J = 1 TO K : A = (I=J)
1140 FOR L% = 1 TO K
1150 A = A + X(I,K + L%)*AX(L%, J) : NEXT L%
1160 IF ABS(A) > AMAX THEN AMAX = ABS(A)
1170 LOCATE 6,12
1180 PRINT USING "Check quality of inverse  # # #.########"; I, J, AMAX;
1190 NEXT J,I
1200 RETURN
1210 '=================================================
1220 '    make a copy of the variance-covariance matrix
1230 FOR J = 1 TO 6 : FOR JP = 1 TO 6
1240 AI(J,JP) = 0 : NEXT JP,J
1250 FOR J = 1 TO J9 : FOR JP = 1 TO J9
1260 AI(JM(J),JM(JP)) = X(J, J9+JP) :NEXT JP,J
1270 '====now calculate the nbar's =========================
1280 FOR J = 1 TO 6
1290 NBAR(J) = 0 : IF KY(J) = 0 THEN 1320
1300 FOR JP = 1 TO 6
1310 NBAR(J) = NBAR(J) + AI(J,JP)*B(JP) : NEXT JP
1320 NEXT J
1330 '====and the overall variance =========================
1340 S2 = 0 : FOR I = 1 TO N%
1350 S1 = 0 : FOR J = 1 TO 6
1360 S1 = S1 + NBAR(J)*S(J,I) : NEXT J
1370 S2 = S2 + W(I)*(S1 - D(I))^2 : NEXT I
1380 S2 = S2/(N% - J9)
1390 '====and the std errors of the nbar's ==================
1400 'This exactly follows the Raabe-Wrenn paper.  Note that the
1410 'std errors are calculated from a combination of: 1) the fit of
1420 'the data points to the regression curve, 2) the magnitude of
1430 'the input counts (as reflected in the elements of the var-cov
1440 'matrix AI).
1450 FOR J=1 TO 6 : SE(J) = SQR(AI(J,J)*S2) : NEXT
1460 RETURN
```

```
1470 '==== show nbar, se, & variance-covariance matrix ========
1480 CLS : LOCATE 2,1 : PRINT TI$
1490 LOCATE 7,12 : PRINT TAB(27)"Variance = ";S2
1500 PRINT"Nuclide    Nbar  Std.Err.   = Covariance Matrix ="
1510 FOR J = 1 TO 6 : PRINT NUCL$(J);
1520 PRINT USING " ########"; NBAR(J); SE(J);
1530 FOR J1 = 1 TO 6 : PRINT USING" ########"; AI(J,J1);
1540 NEXT J1 : PRINT "" : NEXT J : RETURN
1550 '=================================================
1560 DEF FNY(J) = (1 - EXP(-LAMBDA(J)*TSAMP))/LAMBDA(J)
1570 DEF FNZ(J1) = (EXP(-LAMBDA(J1)*TSAMP) - EXP(-LAMBDA(J2)*TSAMP))/(LAMBDA(J2)
     - LAMBDA(J1))
1580 LOCATE 3,12 : PRINT"Calculate back to concentrations in air"
1590 '       Build lower diagonal matrix which relates nbar to c
1600 K = 6 : FOR J = 1 TO K : FOR JP = 1 TO K
1610 AX(J,JP) = 0 : NEXT JP,J
1620 AX(1,1) = FNY(1) : AX(2,2) = FNY(2)
1630 J2 = 1 : AX(2,1) = AX(2,2) - FNZ(2)
1640 AX(3,3) = FNY(3) : J2 = 3 : AX(3,2) = AX(3,3) - FNZ(2)
1650 LM = LAMBDA(2)/(LAMBDA(1) - LAMBDA(2))
1660 AX(3,1) = AX(3,2) - LM*FNZ(2)
1670 J2 = 1 : AX(3,1) = AX(3,1) + LM*FNZ(3)
1680 AX(4,4) = FNY(4) : AX(5,5) = FNY(5)
1690 J2 = 4 : AX(5,4) = AX(5,5) - FNZ(5) : AX(6,6) = FNY(6)
1700 RETURN : 'MATRIX READY FOR INVERTING
1710 '==================================================
1720 'Compute the air concentrations
1730 FOR J = 1 TO K : C(J) = 0
1740 FOR JP = 1 TO K
1750 C(J) = C(J) + X(J,JP+K)*NBAR(JP)
1760 NEXT JP,J
1770 FOR J = 1 TO K : C(J) = C(J)/FLOWR : NEXT
1780 'Now the std errors for the air concentrations.
1790 'The Raabe-Wrenn paper is unclear about this step, so we
1800 'improvise.  We follow CVG/EML who notes that C(3), for
1810 'example, is calculated from nbar(1)...nbar(3), and so
1820 'has error contributions from all three.  Furthermore,
1830 'the errors in the three are correlated.  We have to
1840 'propagate not only the variance of the three, but also
1850 'their covariance.  Hence, the use of the full variance-
1860 'covariance matrix, AI(J1,J2)*S2.
1870 FOR J = 1 TO K : CE(J) = 0
1880 FOR J1 = 1 TO K : FOR J2 = 1 TO K
1890 CE(J) = CE(J) + X(J,J1+K)*X(J,J2+K)*AI(J1,J2)*S2
1900 NEXT J2,J1 : CE(J) = SQR(CE(J)) : NEXT J
1910 FOR J = 1 TO K : CE(J) = CE(J)/FLOWR : NEXT
1920 '====now the PAEC ===================================
1930 'Note: the PAEC error calc was completely revised 11/87
1940 'Prior to that, we used "adding at right angles" which
1950 'gives answers too large.  See below for further change, 3/88
1960 PAEC = 0 : PERR = 0
1970 FOR J = 1 TO K : PP(J) = 0
1980 PAEC = PAEC + ENERGY(J)*C(J)
```

```
1990 FOR J1 = 1 TO K
2000 PP(J) = PP(J) + ENERGY(J1)*X(J1,J+ K)  : NEXT J1,J
2010 FOR J1 = 1 TO K : FOR J2 = 1 TO K
2020 PERR = PERR + PP(J1)*PP(J2)*AI(J1,J2)*S2
2030 '/FLOWR was added two lines down in late 3/88.
2040 'This corrects an oversight in the 11/88 revisions.
2050 NEXT J2,J1 : PERR = SQR(PERR)/FLOWR
2060 RETURN
2070 '=======================================================
2080 CLS : PRINT TI$ : PRINT"Results in terms of atoms" : PRINT
2090 PRINT"Nuclide    atoms/liter  std. error"
2100 FOR J = 1 TO 6
2110 PRINT USING"\       \       ####.##       ####.##";NUCL$(J),C(J),CE(J)
2120 NEXT
2130 PRINT : PRINT"Results in terms of bequerels" : PRINT
2140 PRINT"Nuclide       Bq/m3     std. error"
2150 FOR J = 1 TO 6
2160 PRINT USING"\       \       ####.##
      ####.##";NUCL$(J),C(J)*LAMBDA(J)*1000/60,CE(J)*LAMBDA(J)*1000/60
2170 NEXT : TSHOW = TIMER
2180 IF FO$ = "" THEN 2410
2190 PRINT#2,"" : PRINT#2,"" : PRINT#2, TI$
2200 FOR I = 1 TO 5 : PRINT#2, USING" ###.###";PHI(I); : NEXT
2210 FOR I = 1 TO 6 : PRINT#2, USING"  #";KY(I); : NEXT : PRINT#2,""
2220 PRINT#2, USING" ###.###";TBG; CBG; TAU*1000000!*60; FLOWR; TSAMP;
      TTRANS*60; TCOUNT*60; TDOWN*60
2230 IF Q2$ = "Y" OR Q2$ = "y" THEN 2240 ELSE GOTO 2290
2240 PRINT#2,"" : PRINT#2,"Nuclide   Nbar Std.Err.   == Covariance Matrix =="
2250 FOR J = 1 TO 6 : PRINT#2, NUCL$(J);
2260 PRINT#2, USING " #######"; NBAR(J); SE(J);
2270 FOR JP = 1 TO 6 : PRINT#2, USING" #########";AI(J,JP);
2280 NEXT JP : PRINT#2,"" : NEXT J
2290 PRINT#2,"": PRINT#2, "Results in terms of bequerels and PAEC. # cnts =";N%
      : PRINT#2, ""
2300 PRINT#2, "Nuclide   Bq/fltr  Std.Err.       Bq/m3      Std.Err.   PAEC,nJ m-
      3    PAEC,mWL"
2310 FOR J = 1 TO 6 : PRINT#2, NUCL$(J);
2320 PRINT#2, USING"#######.##"; NBAR(J)*LAMBDA(J)/60; SE(J)*LAMBDA(J)/60;
2330 PRINT#2,  USING"   #######.##";
      C(J)*LAMBDA(J)*1000/60,CE(J)*LAMBDA(J)*1000/60, C(J)*ENERGY(J)*.16,
      C(J)*ENERGY(J)/130
2340 NEXT : PRINT#2,""
2350 PRINT#2,"Potential alpha energy conc.  =                    ";
2360 PRINT#2, USING"   #######.##";PAEC*.16, PAEC/130
2370 PRINT#2,"PAEC standard error =                              ";
2380 PRINT#2, USING"   #######.##";PERR*.16, PERR/130
2390 PRINT#2, USING"Variance of the fit =
      #######.##";S2
2400 PRINT#2, "Analyzed on  ";DATE$;"  via Basic"
2410 WHILE TIMER < TSHOW + 3 : WEND : RETURN
2420 '------------------print bad news
2430 CLS : PRINT TI$ : PRINT"Sorry, no results.  Matrix inversion failed."
2440 TSHOW = TIMER : IF FO$ = "" THEN 2470
```

```
2450 PRINT#2,"" : PRINT#2,"" : PRINT#2, TI$
2460 PRINT#2,"Sorry, no results.  Matrix inversion failed."
2470 WHILE TIMER < TSHOW + 5 : WEND : RETURN
2480 '=====================================================
2490 IF EOF(1) THEN CLOSE #2 : CLOSE #1 : PRINT : PRINT" F I N I : file ";FI$ :
      END
2500 LINE INPUT #1, TI$ : IF TI$ = "" THEN 2490
2510 SOURCE$ = "" : N% = 0 : CLS : PRINT TI$
2520 LINE INPUT #1, CO$ : IF LEFT$(CO$,1) < "A" THEN GOSUB 2610 : GOTO 2580
2530 OPEN CO$ FOR INPUT AS #3
2540 WHILE NOT EOF(3)
2550 LINE INPUT #3, CO$ : GOSUB 2610
2560 WEND : CLOSE #3
2570 IF N% <> 19 THEN PRINT"wrong number of parameters " : STOP
2580 LINE INPUT #1, CO$ : IF CO$ = "" THEN 2690
2590 GOSUB 2610 : IF NOT EOF(1) THEN 2580 ELSE 2690
2600 '--------------decompose one data line
2610 :I1 = 0 : IF INSTR(1,CO$,",") THEN FS$ = "," ELSE FS$ = " "
2620 :I1 = I1+1 : IF I1 > LEN(CO$) THEN RETURN
2630 :IF MID$(CO$,I1,1) = FS$ THEN 2620 ELSE I0 = I1
2640 :I1 = I1+1 : IF I1 > LEN(CO$) THEN 2660
2650 :IF MID$(CO$,I1,1) <> FS$ THEN 2640
2660 :N% = N%+1 : CO(N%) = VAL(MID$(CO$,I0,I1-I0))
2670 :PRINT N%;CO(N%) : GOTO 2620
2680 RETURN : '---------------------------
2690 IF CO(1) => 1 OR CO(1) < 0 THEN SOURCE$ = "keyboard" : GOTO 2750
2700 FOR I = 1 TO 5 : PHI(I) = CO(I) : NEXT
2710 FOR I = 1 TO 6 : KY(I) = CO(I+5) : NEXT
2720 TBG = CO(12) : CBG = CO(13) : TAU = CO(14)*.000001/60 : FLOWR = CO(15)
2730 TSAMP = CO(16) : TTRANS = CO(17)/60 : TCOUNT = CO(18)/60 : TDOWN =
      CO(19)/60
2740 FOR I = 20 TO N% : CO(I-19) = CO(I) : NEXT : N% = N% - 19
2750 LOCATE 1,1 : PRINT TI$ : RETURN
2760 '=====================================================
2770 IF SOURCE$ <> "keyboard" THEN 2860
2780 CLS : INPUT"background count time, min";TBG
2790 INPUT"background counts";CBG
2800 INPUT"counter dead time, microsec.";TAU : TAU = .000001*TAU/60
2810 INPUT"flow rate, liters/min";FLOWR
2820 INPUT"sampling time, min";TSAMP
2830 INPUT"sample transfer time, sec";TTRANS : TTRANS = TTRANS/60
2840 INPUT"length of each count, sec";TCOUNT : TCOUNT = TCOUNT/60
2850 INPUT"down time between counts, sec";TDOWN : TDOWN = TDOWN/60 : CLS
2860 ALPHA(1) =TTRANS
2870 FOR I = 1 TO NN% - 1
2880 BETA(I) = ALPHA(I) + TCOUNT
2890 ALPHA(I+1) = BETA(I) + TDOWN
2900 'NEXT I : BETA(I) = ALPHA(I) = TCOUNT : 'incorrect, caught 10/88
2910 NEXT I : BETA(I) = ALPHA(I) + TCOUNT : 'new line, 10/88
2920 RETURN
2930 '=====================================================
2940 IF SOURCE$ <> "keyboard" THEN 3090
2950 CLS : PRINT TI$ : PRINT"Input counter efficiency for -"
```

```
2960 INPUT"Po-218";PHI(1)
2970 INPUT"Po-214";PHI(2)
2980 INPUT"Bi-212";PHI(3)
2990 INPUT"Po-212";PHI(4)
3000 INPUT"Nuclide X";PHI(5)
3010 CLS : PRINT"Input which nuclides to analyze for -"
3020 PRINT"enter 1 to analyze, 0 to skip"
3030 INPUT"Po-218";KY(1)
3040 INPUT"Pb-214";KY(2)
3050 INPUT"Po-214";KY(3)
3060 INPUT"Bi-212";KY(4)
3070 INPUT"Po-212";KY(5)
3080 INPUT"Nuclide X";KY(6)
3090 F1 = PHI(1) : F2 = PHI(2) : F4 = PHI(5)
3100 F3 = .337*PHI(3) + .663*PHI(4)
3110 R0 = LAMBDA(1)*LAMBDA(2)*LAMBDA(3)
3120 R1 = R0/(LAMBDA(1) - LAMBDA(2))/(LAMBDA(3) - LAMBDA(2))
3130 R2 = R0/(LAMBDA(1) - LAMBDA(2))/(LAMBDA(1) - LAMBDA(3))
3140 R3 = LAMBDA(2)*LAMBDA(3)/(LAMBDA(3) - LAMBDA(2))
3150 R4 = LAMBDA(4)*LAMBDA(5)/(LAMBDA(4) - LAMBDA(5))
3160 RETURN
3170 '=====================================================
3180 NN% = 520 : DEFINT I,J,K : ' DEFDBL A,B,D,X
3190 DIM ALPHA(NN%), BETA(NN%) : 'beginning and end of each count
3200 '                            measured from the end of sampling
3210 DIM CO(NN%) : '              counts straight from the scaler
3220 DIM D(NN%) : '               counts corrected for dead time and bkgd
3230 DIM W(NN%) : '               weight factor for each count, from statistics
3240 DIM S(6,NN%) : '             Raabe & Wrenn's S-matrix
3250 DIM A(6,6), AI(6,6) : '      A-matrix and its inverse
3260 DIM X(6,12), AX(6,6) : '     scratch matrix used for inverting
3270 '
3280 DATA 0.2228, 0.0259, 0.0352,0.001086, 0.01146, 0.0001
3290 FOR I = 1 TO 6 : READ LAMBDA(I) : NEXT
3300 DATA Po-218, Pb-214, Bi-214, Pb-212, Bi-212, Nucl-X
3310 FOR J = 1 TO 6 : READ NUCL$(J) : NEXT
3320 DATA 13.68, 7.68, 7.68, 7.86, 7.86, 0
3330 FOR J = 1 TO 6 : READ ENERGY(J) : NEXT
3340 RETURN
3350 '==========================================
3360 CLS : PRINT : PRINT : PRINT : N% = 15
3370 PRINT TAB(N%)"Raabe - Wrenn Least Squares Calculation"
3380 PRINT TAB(N%)"for Radon and Thoron Progeny from Count Data"
3390 PRINT : PRINT TAB(N%)"programmed in Microsoft GW-Basic by"
3400 PRINT
3410 PRINT TAB(N%+4)"Earl O. Knutson, Ph.D."
3420 PRINT TAB(N%+4)"Environmental Measurements Laboratory"
3430 PRINT TAB(N%+4)"U.S. Department of Energy"
3440 PRINT
3450 PRINT TAB(N%+4)"August 86; November 87; March 88" : PRINT
3460 PRINT TAB(N%)"The count data are assumed to be in a disk file, "
3470 PRINT TAB(N%)"each block starting with a one-line title and"
3480 PRINT TAB(N%)"ending with a null line or end-of-file mark."
```

```
3490 PRINT
3500 PRINT TAB(N%)"Nineteen parameters are needed in the calculation;"
3510 PRINT TAB(N%)"their source is coded into the first data line."
3520 PRINT
3530 PRINT TAB(N%)"Need further instructions, Y/N";
3540 INPUT Q0$ : IF Q0$ = "Y" OR Q0$ = "y" THEN GOSUB 3630
3550 CLS : LOCATE 3,N% : PRINT TAB(N%)"Enter name of input file
3560 INPUT FI$ : OPEN "I", #1, FI$
3570 PRINT
3580 PRINT TAB(N%)"File name for output - no name, no save"
3590 PRINT TAB(N%)" (if file exists, output will append)  "; : INPUT FO$
3600 IF FO$ = "" THEN 3620 ELSE OPEN "a", #2, FO$
3610 PRINT TAB(N%)"Include covariance matrix in disk file, Y/N"; : INPUT Q2$
3620 CLS : RETURN
3630 CLS : SHELL "type rwrenn.txt | more"
3640 LOCATE 24,1 : PRINT "Want to see it again, Y/N";
3650 B$ = INKEY$ : IF B$ = "" THEN 3650
3660 IF B$ = "Y" OR B$ = "y" THEN RUN ELSE CLS : RETURN
```

Appendix K
File WWN.Pas

```
PROGRAM nazaroff (input, output);

(*****************************************************
*
*  This Pascal program does the calculations needed to
*  determine RnP concentrations from three gross alpha
*  counts. The equations used are those given by W.W.
*  Nazaroff in The article in Health Physics 46,395-405
*  (1984).
*
*  The half-lives used are: 3.11, 26.8 and 19.9 min.
*  These are taken from page 6 of the new book by
*  Nazaroff and Nero (1988).
*
*  The equations have been changed to SI units.
*  Mainly, this involved substituting 60 for 2.22
*  everywhere, and expressing decay energies in nono
*  joules.
*
*  The program has been checked (2/13/88) by reverting
*  to Nazaroff's original values for the constants
*  and running the Thomas protocol.  The largest
*  difference in the coefficient matrix for the three
*  nuclide equations was 5 parts in 2000 and in the
*  PAEC coefficients, 3 parts in 900.
*
*
*  Earl O. Knutson
*  USDOE/Environmental Measurements Laboratory
*  New York, NY  10014
*
*  2/88
*
*****************************************************)
USES Crt, Printer, Dos, mtrx;

TYPE  abindex = (a,b);

VAR decayconst      : ARRAY [1..3] OF real;
    alphaenergy     : ARRAY [1..3] OF real;
    i,j             : integer;
    time0,flowrate  : real;
    efficiency      : real;
    background      : real;
    time            : ARRAY [1..3,abindex] OF real;
    counts          : ARRAY [1..3] OF integer;
    rnpconc,stderror : ARRAY [1..4] OF real;
    ab              : abindex;
    firsttime       : boolean;
    yr, mo, dy, dow : word;

FUNCTION f(i,j : integer) : real;
BEGIN
  f := decayconst[i]/(decayconst[i] - decayconst[j]);
END;

FUNCTION r(i : integer; t : real) : real;
BEGIN
  r := 1 - exp(-decayconst[i]*t);
END;

FUNCTION s(i : integer; t : real) : real;
BEGIN
  s := exp(-decayconst[i]*t);
END;

(*
   As explained by Nazaroff, Gij is the accumilated number
   of alphas emitted from nuclide i on the filter, due to
   collecting the j-th nuclide at a rate of 1 Bq per min.
   The factor 60, which is dpm per Bq, replaces Nazaroff's
   2.22, which is dpm per pCi. The units of Gij are min per Bq.

*)

FUNCTION G11(t,t0 : real) : real;
VAR G : real;
BEGIN
  IF t < t0 THEN
    G := t - r(1,t)/decayconst[1]
  ELSE
    G := t0 - r(1,t0)*s(1,(t - t0))/decayconst[1];
  G11 := 60*G/decayconst[1];
END;

FUNCTION G31(t, t0 : real) : real;
VAR G : real;
BEGIN
  IF t < t0 THEN
    G := t
        - f(2,1)*f(3,1)*r(1,t)/decayconst[1]
        - f(1,2)*f(3,2)*r(2,t)/decayconst[2]
        - f(1,3)*f(2,3)*r(3,t)/decayconst[3]
  ELSE
    G := t0
        - f(2,1)*f(3,1)*r(1,t0)*s(1,(t - t0))/decayconst[1]
        - f(1,2)*f(3,2)*r(2,t0)*s(2,(t - t0))/decayconst[2]
```

```
          - f(1,3)*f(2,3)*r(3,t0)*s(3,(t - t0))/decayconst[3];
  G31 := 60*G/decayconst[1];
END;

FUNCTION G32(t, t0 : real) : real;
VAR G : real;
BEGIN
  IF t < t0 THEN
    G := t
          - f(3,2)*r(2,t)/decayconst[2]
          - f(2,3)*r(3,t)/decayconst[3]
  ELSE
    G := t0
          - f(3,2)*r(2,t0)*s(2,(t - t0))/decayconst[2]
          - f(2,3)*r(3,t0)*s(3,(t - t0))/decayconst[3];
  G32 := 60*G/decayconst[2];
END;

FUNCTION G33(t, t0 : real) : real;
VAR G : real;
BEGIN
  IF t < t0 THEN
    G := t - r(3,t)/decayconst[3]
  ELSE
    G := t0 - r(3,t0)*s(3,(t - t0))/decayconst[3];
  G33 := 60*G/decayconst[3];
END;

FUNCTION kbquery(msg : string) : boolean;
VAR  query : char;
BEGIN
  REPEAT
  WRITE(msg,' ENTER Y OR N ');
  READLN(query)
  UNTIL query IN ['y','Y','n','N'];
  kbquery := (query IN ['y','Y']);
END;


PROCEDURE initialize;
BEGIN

(*decayconstants in inverse minutes *)
  decayconst[1] := LN(2)/3.11;
  decayconst[2] := LN(2)/26.8;
  decayconst[3] := LN(2)/19.9;

(*alphaenergies in nano joules *)
  alphaenergy[1] := 13.69*1.6021E-4;
  alphaenergy[2] :=  7.69*1.6021E-4;
  alphaenergy[3] :=  7.69*1.6021E-4;

  REPEAT
  WRITE('Enter flowrate, Lpm,   ');
  READLN(flowrate);
  WRITE('Enter sample time, min ');
  READLN(time0);
  IF kbquery('Use Thomas protocol?') THEN
    BEGIN
(*
      Thomas Protocol, times in minutes.
*)
      time[1,a] := 2.0 ;    time[1,b] := 5.0;
      time[2,a] := 6.0 ;    time[2,b] := 20.0;
      time[3,a] := 21.0;    time[3,b] := 30.0;

      FOR i := 1 TO 3 DO
        FOR ab := a TO b DO
          time[i,ab] := time[i,ab] + time0;
    END
  ELSE
    BEGIN
      WRITELN('Enter times in minutes,');
      WRITELN('***MEASURED FROM THE START OF SAMPLING***');
      WRITELN('use space, not comma, to separate numbers');
      FOR i := 1 TO 3 DO
      BEGIN
        WRITE('Start & stop for count #',i:3);
        READLN(time[i,a], time[i,b]);
      END;
    END
  UNTIL kbquery('OK so far?');

  GoToXY (1,24);
  WRITE('Building H-matrix');

  FOR i := 1 TO 3 DO
  BEGIN

    matrix1[i,1] := G11(time[i,b],time0) - G11(time[i,a],time0)
                  + G31(time[i,b],time0) - G31(time[i,a],time0);

    matrix1[i,2] := G32(time[i,b],time0) - G32(time[i,a],time0);

    matrix1[i,3] := G33(time[i,b],time0) - G33(time[i,a],time0);

  END;

  GoToXY (1,24);
  WRITE('Inverting H-matrix');
  matrixsize := 3;
  matinvert;

  ClrScr;
  WRITELN('Nazaroff''s K-matrix, Bq/min');
  FOR i := 1 TO 3 DO
  BEGIN
    FOR j := 1 TO 3 DO WRITE(matrix2[i,j]:10:6);
    WRITELN;
  END;

  WRITELN('Nazaroff''s L-matrix, nJ/min');
  FOR j := 1 TO 3 DO
  BEGIN
    matrix2[4,j] := 0.0;
    FOR i := 1 TO 3 DO
      matrix2[4,j] := matrix2[4,j]
                    + 60*alphaenergy[i]*matrix2[i,j]
                    /decayconst[i];
    WRITE(matrix2[4,j]:10:6);
  END;
  WRITELN;
END;


PROCEDURE GetCounts;

BEGIN
  REPEAT
  WRITE('Enter counter efficiency,% ');
  READLN(efficiency);
  efficiency := efficiency/100.0;
  WRITE('Enter background, cpm       ');
  READLN(background);

  WRITELN('Enter the three counts ');
    FOR i := 1 TO 3 DO
    BEGIN
      WRITE('Count #',i:3,' ');
      READLN(counts[i]);
    END

  UNTIL kbquery('OK so far?');

END;

BEGIN  (* MAIN PROGRAM *)

  ClrScr;
  WINDOW(16,4,64,20);
  firsttime := TRUE;

  REPEAT
    ClrScr; highvideo;
    WRITELN('Calculation of RnP from three gross alpha counts');
    lowvideo;
    WRITELN('Pascal program: E.O. Knutson 1988');
    WRITELN('Equations: W.W. Nazaroff, 1984');
    WRITELN('Constants: Nazaroff & Nero, 1988');
    IF firsttime THEN initialize
    ELSE
    IF kbquery('Enter new flow and times?') THEN initialize;

    firsttime := FALSE;

    GetDate(yr,mo,dy,dow);
    GetCounts;

    FOR i := 1 TO 3 DO
      counts[i] := counts[i] -
        ROUND(background*(time[i,b] - time[i,b]));

    FOR j := 1 TO 4 DO
    BEGIN
      rnpconc[j] := 0.0;
      stderror[j] := 0.0;
      FOR i := 1 TO 3 DO
      BEGIN
        rnpconc[j] := rnpconc[j] + matrix2[j,i]*counts[i];
        stderror[j] := stderror[j] + matrix2[j,i]
          *matrix2[j,i]*counts[i];
      END;
      rnpconc[j] := 1000*rnpconc[j]/efficiency/flowrate;
      stderror[j] :=
             1000*SQRT(stderror[j])/efficiency/flowrate;
(*                  1000 = liters per cubic meter *)
    END;

    WRITELN('Results - ');
    WRITELN('Nuclide Concent.  1-sigma');
    WRITELN('Po-218 ',rnpconc[1]:9:3,stderror[1]:9:3,' Bq/m3');
    WRITELN('Pb-214 ',rnpconc[2]:9:3,stderror[2]:9:3,' Bq/m3');
    WRITELN('Bi-214 ',rnpconc[3]:9:3,stderror[3]:9:3,' Bq/m3');
    WRITELN('PAEC   ',rnpconc[4]:9:3,stderror[4]:9:3,' nJ/m3');
    WRITELN;

    IF kbquery('Print the results?') THEN
    IF kbquery('Printer paper OK? ') THEN
    BEGIN
      WRITELN(Lst);
      WRITE(Lst,'         Three-count Radon Progeny calc');
      WRITELN(Lst,'ulation (Nazaroff''s equations )');
      WRITE(Lst,'         Input data...........         ');
      WRITELN(Lst,'Results (calculated',yr:5,mo:3,dy:3,')');
      WRITE(Lst,'         Cnt# Start Stop  Count        ');
      WRITELN(Lst,'Nuclide Concent.  1-sigma');
      WRITE(Lst,'           1 ',time[1,a]:6:1,time[1,b]:6:1,counts[1]:6,' ':8);
      WRITELN(Lst,'Po-218 ',rnpconc[1]:9:3,stderror[1]:9:3,' Bq/m3');
      WRITE(Lst,'           2 ',time[2,a]:6:1,time[2,b]:6:1,counts[2]:6,' ':8);
      WRITELN(Lst,'Pb-214 ',rnpconc[2]:9:3,stderror[2]:9:3,' Bq/m3');
      WRITE(Lst,'           3 ',time[3,a]:6:1,time[3,b]:6:1,counts[3]:6,' ':8);
      WRITELN(Lst,'Bi-214 ',rnpconc[3]:9:3,stderror[3]:9:3,' Bq/m3');
      WRITE(Lst,'         Samp.time,min =',time0:7:2,' ':8);
      WRITELN(Lst);
      WRITE(Lst,'         Flowrate, Lpm =',flowrate:7:2,' ':8);
      WRITELN(Lst,'PAEC    ',rnpconc[4]:9:3,stderror[4]:9:3,' nJ/m3');
      WRITE(Lst,'         Efficiency, % =',100*efficiency:7:1,'    ');
      WRITELN(Lst);
      WRITE(Lst,'         Backgrnd, cpm =',background:7:2,'    ');
      WRITELN(Lst);
    END

  UNTIL NOT kbquery('Do another? ');
END.
```