User's Guide for Positive Matrix Factorization programs PMF2 and PMF3, Part 1: tutorial

Last changed on February 25, 2000

Contents

Introduction.. 1

The mathematical models behind PMF2 and PMF3 2

Installation of PMF2 and/or PMF3 3

The programs PMF2 and PMF3 for Positive Matrix Factorization: getting started............................. 3

- The .ini files 3
- Trying out the program PMF2 or PMF3 for the first time on a PC computer 4
- Editing the .INI files and data files 5
- Moving data between PMFx and a spreadsheet program 6
- The first analysis of your own data matrix by PMF2.. 7
- The first analysis of your own data block by PMF3.. 8
- What next? 9

PMF User's Checklist 10

The problem of free rotations in 2-way FA.. 13

- What are the free rotations 13
- After-fit rotations vs. during-fit rotations.. 13
- Different techniques for rotating with PMF214
- Inducing rotations in PMF2 (1): parameter FPEAK.. 14
- Inducing rotations in PMF2 (2): parameter *rotcom* 14
- Inducing rotations in PMF2 (3): pulling individual factor elements towards zero.... 14
- Inducing rotations in PMF2 (4): using target factor shapes 15

A worked-through real example: aerosol measurements in Thailand 17

Requesting support from the author17

Bibliography of PMF-related publications 18

Introduction

This User's Guide covers the two programs PMF2 and PMF3. The reason for combined instructions is that the programs have similar control structures and similar usage patterns. The notation PMFx means either PMF2 or PMF3. The word "matrix" is reserved for two-dimensional arrays. Similarly, the word "block" is used for 3-way arrays.

The User's Guide has been divided into two parts. The first part contains different kinds of tutorial information:

- How to perform first analyses with the programs. This the old "Getting started" document.
- A checklist, helping you in checking if you have dealt properly with the numerous details of performing a PMF analysis. New entries to this list are solicited!
- Discussion of the techniques for obtaining a desired rotation with PMF2.
- A worked-through real environmental example, to be written.

Part 2 of the User's guide contains a complete reference of all details of the programs PMF2 and PMF3. Various details have been moved from the readme file to Part 2. Parts 1 and 2 should be studied in parallel.

Other documents describing PMF are the following:

- Release notes (or a file "readme.txt"), they contain information specific to a certain version of PMF on a certain platform, typically limitations and new features.
- "Introduction to PMF" is intended for distribution to potential new users of PMF, to show them why they might be interested in PMF. You are free to send copies of this document to your colleagues.

The main distribution medium for these documents is the FTP server ftp://rock.helsinki.fi/pub/misc/pmf/. It is assumed that each user is able to download the programs and documentation from this site. It is intended that the documents be mainly available as "Acrobat" files (.pdf). Many of the documents are currently also available as Postscript files (.ps) and/or as Lasejet printer files (.prn). The program files and many of the document files are compressed with the program PKZIP (v. 2.04).

Our scientific papers, e.g. in *Environmetrics* and in *Chemometrics and Intelligent Laboratory Systems* ("Chemolab"), (see list of references) contain general information about the theoretical properties of the two-dimensional Positive Matrix Factorization model, and also some recommendations for applying the model. This information is necessary for efficient utilization of PMF2 and PMF3. Copies of these papers may be supplied by the author. Preprint versions of some papers are also available at the FTP site.

The mathematical models behind PMF2 and PMF3

The program PMF2 solves approximately (in the Least Squares sense) the matrix equation

$$\mathbf{X} = \mathbf{GF} \qquad (1)$$

where **X** is known and **G** and **F** are unknown. Writing in component form and showing the residual matrix explicitly we get

$$x_{ij} = \sum_{h=1}^{p} g_{ih}\, f_{hj} + e_{ij} \qquad (2)$$

where std-dev(**X**) = std-dev(**E**) = **S** and where some or all elements of **G** and **F** are required to be non-negative. There are p "factors" in this model. One column of **G** and the corresponding row of **F** represent one factor. They correspond to the 'scores' and 'loadings' of the customary factor analysis.

When the residual matrix **E** is defined by

$$\mathbf{X} = \mathbf{G}\,\mathbf{F} + \mathbf{E}, \qquad (3)$$

the task of PMF2 may be expressed as minimizing the sum of squares

$$Q = \sum_{i=1}^{n}\sum_{j=1}^{m} (e_{ij} / s_{ij})^2 \qquad (4)$$

In the robust mode, this expression is modified so that the s_{ij} are dynamically readjusted (=iterative reweighting). We are also calling the value of Q by the name "chi2" but this is not quite correct. Strictly speaking, Q is not distributed according to the chi-squared distribution, although the distribution in many cases approximates chi2.

The "PARAFAC" model solved by the program PMF3 is best described in the following component form,

$$x_{ijk} = \sum_{h=1}^{p} a_{ih} b_{jh} c_{kh} + e_{ijk} \qquad (5)$$

Again there are p factors in the model, but in this case there are three entities (columns of **A**,**B**,**C**) forming one factor. And again the model is solved as a non-negatively constrained weighted (iteratively reweighted) Least Squares task.

Notation: we call the array of observed data by the symbol **X** both in PMF2 and in PMF3. Anticipating later needs, we denote the "fit" array by **Y** so that equation (3) may be written as **X**=**Y**+**E**, where the matrix **Y**=**GF**, and equation (5) is also **X**=**Y**+**E**, but now the block **Y** may be written symbolically as **Y**=**ABC**.

Installation of PMF2 and/or PMF3

Currently the PMF programs for 486-Pentium platforms are based on the Fortran 90 compiler LF90 by Lahey Computer Systems. The present instructions are for LF90.

The .EXE file(s) for PMF program(s) and the file lf90.eer should be downloaded from the ftp site and copied into any suitable directory which is included in the PATH. Depending on the version, the .EXE files may have different names, but the most usual names are e.g. PMF2OPT.EXE or PMF3TST.EXE. If you wish, you may rename the files to any other names when copying to your fixed disk. It is recommended that you should mainly use those .exe files whose names contain the letters "tst" or the letter "t". These files are safe to use because they have been compiled so that certain error detecting mechanisms are included. Newest versions are only availble as "tst" files. The files whose names contain the letters "opt" or the letter "o" are slightly faster, because the error detection mechanisms have been omitted. However, they should only be used in situations where repeated high-volume analyses are to be made, and only if achieving the fastest possible running times is critical. Make sure that you do not inadvertently use an obsolete version when using an "opt" file!

The file lf90.eer enables the Fortran system to write plain-language error messages. Also there is the Fortran error message list file rterrmsg.txt which contains the same information in a human-readable text file. You may look at this file with any text editor to determine the meaning of the numerical error codes. You will also need an authorization file or "key file", usually it will be emailed to you. The key file name should be pmf2key.key and/or pmf3key.key. The key file should preferably be copied to the directory C:\PMF or D:\PMF. One key file may contain a license both for PMF2 and for PMF3. Such a file should be copied to the directory twice, under both of these names. Other possible places for the .key files are your working directory and two directory levels above it. The key file contains licensing information, e.g. information about the licensee and about the licensing period. This information is sealed with a numerical check code. Do not change the contents of the key file when copying!

If you wish to run PMFx in Windows 3.11 Dos box, you might perhaps set up the Dos startup command DOSPRMPT.PIF so that PMFx continues to run while you switch away from Dos in order to perform editing or other tasks. Start up the "PIF EDITOR" and open for editing DOSPRMPT.PIF. Put an x mark in the box "Execution: Background", perform a "SAVE", and then "EXIT". If a Dos box is open, make an "EXIT" in it, then reopen the Dos box. Now Windows should continue running PMFx even while you are working elsewhere! But be careful **not to work** on those files which PMFx is either reading or writing to! In the readme file you may find additional information, e.g. about how to configure the dos box of Windows95.

PMFx auxiliary files (for PMF2, e.g. PMF2DEF.INI, GE.INI, GE2.INI, GE.DAT) and your own INI files should be copied into your own working directory. Last-minute information (e.g. warnings for known errors of PMF programs) may be found in the file "readme.txt" or in "release notes". Check for the presence of such a file!

See the disclaimer at the end of Part 2 of User’s Guide for important information!

The programs PMF2 and PMF3 for Positive Matrix Factorization: getting started

The .ini files

The programs PMFx (PMFx means PMF2 and/or PMF3) are controlled by an initialization file or ".INI file". By changing values in this file one may adapt PMF2 and PMF3 to different requirements. In this section we explain how one may use PMF2 or PMF3 with the default .INI file, doing the bare minimum of changes to this file. This means that formats of your data etc. are adapted to the default format of PMF2 or PMF3. The complete instructions in Part 2 of the User’s Guide will explain how to change the default .INI file so that it adapts to the format of your data.

After installing the program PMFx, you should first try it out with a standard test case. For PMF2 we recommend the Gauss-Exponential example: four Gaussian peaks decaying exponentially. This example consists of a data file GE.DAT and a ready-made initialization file GE.INI, thus all you have to do is to proceed according to the following instructions. Note that double quotes in these instructions are not intended to be part of the text to be typed (unless we say so). When we e.g. tell you to type "PMF2.XX" we mean that the characters PMF2.XX should be typed.

The programs PMF2 and PMF3 need an authorization file, PMF2KEY.KEY or PMF3KEY.KEY, respectively. You should have received your personal key file(s) from the author by email. This is a small text file, it identifies the program(s), the user, the platform, and the licensing period. *Do not edit the key file in any way,* if it is changed the program may not accept it any more! One key file may contain licenses for several programs. In such a case you need to make multiple identical copies of the file with the required file names so that each program may recognize the file as belonging to itself. Each KEY file contains typically two or three lines of text with typically 60 to 70 characters on each line. Since 1997, the programs accept a multi-line KEY file, thus there is no need to edit the file in any way.

Trying out the program PMF2 or PMF3 for the first time on a PC computer

These instructions apply when using a Dos box. Instructions for using the 32-bit operating systems will be written shortly.

PMF2.

1. Copy the files GE.INI, GE2.INI and GE.DAT to your working directory (which should also be the current directory). Check that PMF2.EXE is in a directory which is included in your path. (The name of the .exe file may also be something else, e.g. PMF2TST.EXE. You may rename the file to PMF2.EXE if you wish.) Copy the authorization file PMF2KEY.KEY to a suitable directory, perhaps C:\PMF or D:\PMF. Other possibilities are the current directory and two levels above it. If the .key file came to you as the file "pmf2key.nnn", rename it to pmf2key.key.
2. Start PMF2 by typing "PMF2 XX". Now PMF2 should start. First it will write its own name and version information. Then it will complain that the initialization file XX.INI and the default initialization file PMF2DEF.INI are not present. Then it will write the default file PMF2DEF.INI and stop.
3. Inspect PMF2DEF.INI with your favorite text editor. Print the file.
4. Start PMF2 again, this time by typing "PMF2 GE". Now PMF2 should run the analysis of the file GE.DAT as specified by the file GE.INI. The output on the screen will be about 70 lines of text, reporting the process of iteration (it takes about 40 iteration steps).

Inspect the result files: PMF2.LOG and the files G_FACTOR.TXT, F_FACTOR.TXT, GERES.TXT and RESIDUAL.TXT . You will see that in the factor matrix G there are four columns containing "peaks", on most columns there are more than one peak. In the factor matrix F there are smooth curves, most of them decreasing towards zero towards the ends of the rows. In the original model, the columns of G were generated so that there was one Gaussian shape on each column and the rows of F had exponential shape. As you see, the default example GE.INI does not recover these shapes.

On other platforms there may be differences in starting PMF. On DEC Alpha VMS systems it is not simple to transport an argument from the command line to the F90 program. Thus the name of the .ini file is always PMF2.INI or PMF3.INI on DEC systems. You should copy your .ini file to PMFx.ini before starting the program by the command "RUN PMFx". If you are familiar with the operating system, you could write a command file to automate this task.

PMF3.

Trying PMF3 is rather similar as PMF2, and we refer to the preceding paragraphs for instructions.

1. Install PMF3.EXE and PMF3KEY.KEY and load the example files to your working directory.

2. Start PMF3 by typing "PMF3   XX". The program will write its default .ini file "PMF3DEF.INI" in the way described for PMF2. Inspect the .ini file with a text editor.
3. Start PMF3 again by typing PMF3   POISS2. Now the program will read the file POISS2.INI and obey the commands stored in this file. It will read the data file POISS2.DAT which contains a synthetic data array of size 10x8x6. Then it will compute the 2-component factorization of this array and write the results on the disk, as files PMF3.LOG, ABC.TXT, and MISC.TXT.
4. Inspect the results. The file ABC.TXT contains the three factor matrices A, B, and C. The "true values" of the factors (used when the data array was constructed) are simple "step functions". The values in the result file are good approximations of these step functions. In the majority of 3-way models there is no rotational ambiguity and thus the factors may be recovered correctly without any additional tricks. This is true also for this example case. However, this result is not "correct" because the data in POISS2.DAT was in fact constructed with Poisson distribution and in this introductory example it was analyzed neglecting this information, simply assuming that the std-dev of each value equals unity.

Editing the .INI files and data files

For all use of PMFx, you need to edit the initialization files, and often it is also necessary to edit data files. For this purpose, you need a so-called ascii editor. Typical word processors, such as Word-Perfect or Microsoft Word are not good for this (if they must be used, then it is necessary to force them to produce an "ascii file" or "text file"). Suitable editors are e.g. Notepad (in Windows), EDIT (in newer Ms-Dos versions), **PFE** ("Programmers File Editor", freely available, **much recommended**, may be downloaded from http://www.lancs.ac.uk/people/cpaap/pfe/), Qedit, microEmacs, etc. Preferably your editor should be able to handle long lines of text (at least some 500 characters). The author is doing all programming with PFE.

When editing the .INI file, you must not

- change the order of entries in the file
- split or combine the original lines
- change the characters ## on title lines, nor the sequence ##PMF2 or ##PMF3 on the first line.
- change the version stamp unless you also change the file accordingly
- exceed the maximum line length specified for the .INI line (currently 120 characters)
- insert letters or other non-numerical characters in locations where PMFx expects numerics

When editing the .INI file, you are free to

- change the text of title lines after the characters ## . However, we recommend that you only *change the first line* which is reserved to be used as a title for the .INI file. But you must not change the text ##PMF2 or ##PMF3 on the first line.
- make small changes in the layout, e.g. moving a value a few steps to the left or to the right or changing the numerical presentation of a value.
- change all the numerical values, all true/false values and all character strings.
- add annotations at the end of the .INI file. This is especially handy if there are no matrices read from this file. Then one may keep a log of changes at the end of the file. The program never reads them but they help one in remembering why things are as they are.

But of course, you should know what you are doing. The format lines should obey Fortran format rules, which are explained in Fortran textbooks and handbooks. File names must obey the file name rules of the operating system (they may contain path information, too). In locations where the entries in the .INI file mean "yes" or "no", use F or f for "no" (f means "false") and t or T for "yes" (t means "true"). The letters T and F look so similar that they are easily misread in the .ini file. In order to make your changes stand out in the .ini file it may be good to type the changed values so that they are not in line with the original values but one space to the right or to the left. Another possibility is to use more than one letter for the changed values, e.g. "fa" for false and "tr" for true.

Never use the period "." in integer values. It is recommended that you always use the period in non-integer values, although this is not strictly necessary. The codes C1, C2, and C3 are non-integer values.

In most cases, data is input to PMFx in the "list directed" mode, without a format (FMT=0). Then one may use the Fortran repeat notation in data files. Instead of writing repeated values (e.g. 0.0 0.0 0.0) in the file, one may write a repeat count followed by an asterisk and the value, e.g. "3*0.0" is equivalent to "0.0 0.0 0.0". In PMFx, such a repeat is only possible within lines of input, i.e. within rows of an untransposed array, or within columns of a transposed array. A repeat may not continue to another row of matrix.

If you happen to know the Fortran standard for list-directed input, you also know about the possibility of using "a null value", i.e. two consecutive commas without any numerical value between them. PMFx will replace each null value in the input by the special value -999.9. In this way it is possible to input spreadsheet tables where empty cells represent missing values.

Transposed matrices. In the language of mathematics, transposing a matrix means the same as inter-changing the rows and columns, or "flipping around" the matrix about the diagonal which runs "south-east" from the "northwest corner" (top left corner) of the matrix. When PMFx reads or writes matrices in transposed layout, it means that the first column of the matrix is read from or written to the first line (first "record") of the file, second column from/to the second line, and so on. The other alternative, the default, is the "straight" or untransposed layout, where first row of matrix corresponds to first line of file, and so on. Selection between a straight or transposed output is merely a matter of convenience and up to you. The user's guide will explain how to change the .INI file if you wish to adopt the transposed layout for a matrix. In the result file GE.RES, as produced by GE.INI, the matrix of explained variations (F-side) is written in transposed form in order to demonstrate this possibility. For input, one will naturally select straight or transposed layout according to the layout of the existing data matrix.

Moving data between PMFx and a spreadsheet program

Quite often the data to be analyzed by PMFx come from a spreadsheet. Also, the results by PMFx may be plotted by using a spreadsheet. Thus it is essential to be able to move data back and forth between a spreadsheet and PMFx. Instead of a spreadsheet, such programs as "matlab" or "Mathcad" may also be used. A spreadsheet is discussed as a typical example.

Some spreadsheets output numbers so that there are commas between them. In the default setup PMFx reads equally well data without commas or with commas between values.

In the default setup, PMFx writes data as numbers without any special characters. However, some spreadsheets may require that the values be separated by commas for easy importing of data arrays. The default setup of PMFx may be changed so that PMFx will write commas between factor values in the following way:

In the initialization file, find the format line beginning with the number 57. On this line there is the following format string:

```
"((1X,150(G13.5E2,:' ')))  "
```

If you change this to the following:

```
"((1X,150(G13.5E2,:',')))  "
```

there will be a comma between the values. And if you delete the colon, so that the string is as follows:

```
"((1X,150(G13.5E2,',')))   "
```

then a comma will be between the values and also after the last value, as the last character on the line.

Sometimes there seem to be problems in getting long lines written by a spreadsheet. In studying the instructions for your spreadsheet program, please observe that there may be two different ways for producing output: "printing" and "exporting". Try to export your data matrix and check the result with your editor: the rows of the matrix should be complete, not ending early.

Empty cells should be avoided in spreadsheets because an empty cell does not "exist" in a spreadsheet table that has been written as a "text file". Trying to read such a table causes that data values will appear in wrong positions in the table. If you must use a spreadsheet table with empty cells, there are two alternative possibilities: 1. Fill the empty cells with a suitable value. 2. Write the table in "comma-separated" form (.csv). When PMFx reads an empty cell in comma-separated form, it will insert the special value -999.9 in the empty slot. By using suitable commands, this value may be interpreted as a "missing value".

The first analysis of your own data matrix by PMF2

After all those preliminaries, it is time to prepare your own data matrix for analysis. You should have copies of the first two references (our papers which appeared in vol. 5 of Environmetrics) available, they contain useful background information.

Scaling. One should avoid extremely large and extremely small values in the matrix. If any value in a row or column of the matrix is larger than 100000, say, then one should rescale the row or column by using a larger unit for measuring the quantity. Vice versa, if all values on a row or column are less than 0.00001, say, then one should rescale by selecting a smaller physical unit for that row or column. E.g. one could use micrograms instead of milligrams.

Create a file with name "matrix.dat" containing your data matrix. All the values on the first row of the data matrix should form the first line of text in the file. Next row should form the next line, and so on. However, if it is difficult to generate long lines, you could also write the file so that each row of the matrix is contained in several text lines in the file. But it is essential that the first element of each row of the matrix begins a new text line in the file. And generally the first alternative (one matrix row is one text line) is clearer and causes less confusion.

Create another text file with the name "std_dev.dat". This file should have the same number of values as "matrix.dat", organized in the same layout. Each value should represent the standard deviation ("error estimate") of the corresponding matrix element. (In the traditional PCA or Factor Analysis there is no counterpart to this "std_dev" matrix.) Any data value and its standard deviation should be expressed using the same physical unit: if a certain data is expressed in milligrams, it is not permissible to express the corresponding std-dev in micrograms. No value may have the standard deviation equal to zero. Sometimes there are no standard deviations readily available for a matrix of experimental values. In such a case one has to "invent" reasonable values which reflect one’s own view of the reliability of the data.

Make a copy of the default initialization file PMF2DEF.INI with a suitable name, e.g. "MYPMF.INI". Edit the copied file in the following way:

- on the first line, beginning with ##, delete the descriptive text after ##PMF2 and write a short description of your own data or model instead.
- locate the line with numbers 40, 20, and 4.
- as indicated by the title line (located above these numbers) these values represent the number of rows and the number of columns in the data matrix, and the number of factors in the model. Replace these values by the dimensions of your own matrix and by the number of factors in your model. If you don’t have the slightest idea about the correct number of factors, first try three factors. Later on you should then experiment with other values. -- It may be slightly better to have
  number_of_rows > number_of_columns
  in your matrix instead of the opposite relation. (PMF2 may run slightly faster). This is achieved by reading the matrix in transposed layout, if necessary.
- save the edited file mypmf.ini and exit from the editor

Start PMF2 by the command "PMF2 MYPMF". If this produces an error message, write it down for later reference and try to correct what was wrong. It may be helpful to inspect the log file "PMF2.LOG". If the program PMF2 is able to read the matrix, it will write a copy of it in the file "TEMP.TXT". Inspect this file, too.

If there was no error, the program will produce the following four files:

- G_FACTOR.TXT This contains nothing but the left factor matrix G.
- F_FACTOR.TXT contains nothing but the right factor matrix F.
- TEMP.TXT contains copies of the data matrix and of the standard deviations matrix (with titles). While testing a new application, it is good practice to inspect these every now and then.
- MISC.TXT ("MISC" stands for "miscellaneous"). This file contains all the other results (with appropriate titles): standard deviations for the G and F factors, a matrix of explained variation (for F side only), the matrix of scaled residuals of the least squares fit, and a matrix indicating rotational ambiguity of the result. (In the traditional PCA or Factor Analysis there is no counterpart to any of these matrices.)

Start your spreadsheet or other graphics program. Import the G and F factors to the spreadsheet and plot or print them as appropriate.

The default setup of PMF2 is such that it deletes the previous result files G_FACTOR.TXT, F_FACTOR.TXT, and TEMP.TXT if you run the default setup again. Thus you should copy all these results to a "safe place" (possibly changing their names while copying) if you wish to keep them. But note that it is easy to change this behavior. It requires editing of the .INI file and changing some of the definitions of the output files. See the PMF user's guide. On the other hand, the handling of the file MISC.TXT has been arranged so that new results are appended to the end of the existing file. Thus this file will get longer each time the default PMF2 setup is run. It will be necessary to delete this file every now and then.

The first analysis of your own data block by PMF3

First have a look at the preceding section about analyzing your own data with PMF2. Many details are common to both programs.

Create a file with name "matrix.dat" containing your data block. The pages of your data block should be in this file page after page. It is good to have a few empty lines between pages, just for clarity. Each page is written in the file similarly as a "matrix". All the values on the first row of the page should form the first line of text pertaining to this page in the file. Next row should form the next line, and so on. However, if it is difficult to generate long lines, you could also write the file so that each row of the page is contained in several consecutive text lines in the file. But it is essential that the first element of each row of the block begins a new text line in the file. And generally the first alternative (one row is one text line) is clearer and causes less confusion.

Create another text file with the name "std_dev.dat". This file should have the same number of values as "matrix.dat", organized in the same layout. Each value should represent the standard deviation ("error estimate") of the corresponding data value. Any data value and its standard deviation should be expressed using the same physical unit: if a certain data is expressed in milligrams, it is not permissible to express the corresponding std-dev in micrograms. No value may have the standard deviation equal to zero. Sometimes there are no standard deviations readily available for a block of experimental values. In such a case one has to "invent" reasonable values which reflect one's own view of the reliability of the data.

Make a copy of the default initialization file PMF3DEF.INI with a suitable name, e.g. "MYPMF3.INI". Edit the copied file in the following way:

- on the first line, beginning with ##, delete the descriptive text after ##PMF3 and write a short description of your own data or model instead.

- locate the line with numbers 0, 0, 0, 0, and 1 . These are the dimensions of your data block (number of rows on a logical page, number of columns, and number of logical pages in the block), number of factors, and number of repeats to compute. Replace the four zero values by the three dimensions of your own block and by the number of factors in your model. If you don't have the slightest idea about the correct number of factors, first try three factors. Later on you should then experiment with other values. The smallest of the three dimensions should be the number of pages, and usually the largest should be the number of rows. Save the edited file mypmf3.ini and exit from the editor.

Start the program by typing "PMF3 MYPMF3" or by using other name you have selected for the .INI file (you need not type the file name extension ".INI"). If this produces an error message, write the message down for later reference and try to correct what was wrong. It may be helpful to inspect the log file "PMF3.LOG". If the program PMF3 is able to read the data block X, it will write a copy of it in the file "TEMP.TXT". Inspect this file, too. There should also be a copy of the std-dev block in this file. There are titles in the file before each block.

If there was no error, the program will also produce the following four files:

- A_FACTOR.TXT This file contains nothing but the factor matrix A.
- B_FACTOR.TXT contains nothing but the factor matrix B.
- C_FACTOR.TXT contains nothing but the factor matrix C.
- MISC.TXT ("MISC" stands for "miscellaneous"). This file contains all the other results (with appropriate titles): standard deviations for the factors A, B, and C, and the block of scaled residuals of the least-squares fit.

Start your spreadsheet or other graphics program. Import the factors to the spreadsheet and plot or print them as appropriate.

The default setup of PMF3 is such that it deletes the previous result files A_FACTOR.TXT, B_FACTOR.TXT, C_FACTOR.TXT, and TEMP.TXT if you run the default setup again. Thus you should copy all these results to a "safe place" if you wish to keep them. But note that it is easy to change this behavior, as explained for PMF2, see the PMF user’s guide. On the other hand, the handling of the file MISC.TXT has been arranged so that new results are appended to the end of the existing file. Thus this file will get longer each time the default PMF3 setup is run.

What next?

You might go back and have another look at the test example "GE.INI". It writes a different arrangement of result files than your "MYPMF.INI". Also, the name of the matrix file is different, and there is no input file for the standard deviations. All this is due to a different .INI file which governs all these details. You might print the files GE.INI and MYPMF.INI and compare them. Also we wish to remark that PMF2DEF.INI is not intended as a good .INI file for long-term use. On the contrary, this file was formulated to aid in a quick start and also to demonstrate some of the possibilities in using different definitions. For long-term use you should prepare your own version of PMF2DEF.INI, you might call it "MYDEF.INI".

It is well known that there may be rotational ambiguity in factor analytic models. In the default setup of PMF2, the parameter FPEAK has the value zero. FPEAK=0 means that PMF2 will produce the "most central solution" which is equally far from all non-negativity constraints. Setting the parameter FPEAK to a non-zero value squeezes the solution towards one or the other of the constraints: near-zero values will appear either on the F side or on the G side. You might test this in the following way: edit the file GE.INI and change the value of FPEAK to -0.5, say. Run with this new .INI file and plot the results. You will see that these results are "better" than those produced by the original GE.INI: although the chi2 value is somewhat larger, now the "spectra" in the G factors are almost pure single peaks, and the "decay curves" in the F factors are almost pure exponentials. In a real application, this would probably be a more useful solution than the original one.

There is another ready-made test example "GE2.INI" demonstrating several things: running several analyses within one run of PMF2, performing specific rotations (by using the matrix "rotcom"), and changing the layout of results. It reads the same data file GE.DAT as the test example GE.INI.

There are special options in PMF for handling certain distributions, Poisson and lognormal. Among the examples there are files demonstrating how the example POISS2.DAT is analyzed when the information about the distribution is utilized. The results change somewhat from the trivial analysis.

If you feel comfortable with the setup contained in your MYPMF.INI, you may continue using it and never study the details of the initialization file. But if you have a large number of data sets to analyze, and if you wish to explore all the possibilities built into the program, then you should study Part 2 of the User’s Guide and learn the meaning of the details in the .INI files. In environmental applications the most important features are probably robust estimation and modeling the lognormal distribution of data values. For a discussion of a robust analysis of the GE example with outliers, see Paatero(1997).

PMF User’s Checklist

This list contains reminders for using the programs PMF2 and PMF3. The items are organized approximately in decreasing priority. Those questions are discussed first that are most basic and should be checked first, before considering the more difficult and rarely needed details. Before regarding your results ready for publication, you should check all questions in the checklist and have good answers to all of them.

Did you inspect all the "alert messages"?

The programs write important remarks and warnings as "alert messages", framed by rows of asterisks, "********************". The alert messages contain a serial number. The final output of the program mentions how many alert messages have been generated. It is good practice to look at all of the alert messages. If you do not immediately understand some of them, try to find explanations in guide books or from other users! In most cases you should change your run so that the alert messages disappear. There are, however, some cases where an alert message cannot be avoided in a correct and reasonable run.

Did the run converge?

The .ini file contains convergence criteria for the three iteration levels. The third level is crucial for obtaining correct results. The first two levels mainly influence the rate of convergence, i.e. how many iteration steps are needed in order to achieve final convergence. When the program says "convergence achieved" it only means that the convergence criterion defined by you has been met. If this criterion was too loose, it means that there only was an apparent convergence, not a real one. It is sometimes necessary to experiment with different convergence criteria. If the results do not change significantly although many more steps are performed, then a true convergence may be assumed. If the run is interrupted because the maximum iteration count is exceeded, then there obviously is no guarantee of convergence.

For extremely large models, comprising thousands of data points, the default convergence criteria tend to be (much) too tight. Then you lose time if you do not increase the convergence limits. Whereas something like 0.01 may be a good convergence limit for small models, the largest runs might converge well with limits in the range between 10.0 and 1.0, say.

Was the third "lims" value small enough?

Ideally, the models should be solved with an extremely small (non-zero) value of the "lims" parameter. Large values of "lims" distort the results by preventing near-zero values from appearing in the computed factors. Also, large "lims" values tend to pull largest factor values down. Both of these effects are useful in the initial stages of the iteration. A small amount is needed, because of technical reasons, also in the final stage. With new models, you should try different values for the third "lims" setting until you get a "feel" of the needed value. "Too small" hardly ever hurts!

What about outliers in your data?

The final output of the programs tells how many residuals exceeded the outlier threshold distances (=4.0 by default). If there are such residuals, it is better to use the robust mode (it is selected by default). Inspect the output file containing scaled residuals and try to understand why some residuals are so large.

Have you inspected the residuals?

In a thorough data analysis, one should also inspect the residuals for different anomalies. Ideally, the scaled residuals should appear random to the eye. There should be a random pattern of positive and negative values, most of them between -2.0 and 2.0. Groups of mostly positive residuals, or mostly negative, or mostly very small (smaller than 0.3, say) should ideally not occur. If such features do occur, one should try to understand why. One possible reason is that too few factors have been used. Another is that the true situation does not fully obey the mathematical model.

Is the computed Q value (the Chi-2 value) reasonable? Is there need to reconsider the std-dev values for X?

1. If the std-dev values for the data array are theoretically known, then it is possible to judge the quality of the fit based on the Q value. The value of Q should be approximately equal to the number of points in the data matrix minus the total number of elements in the factor matrices.

2. More often, the standard deviations for data points are not well known. Then the obtained Q value depends on the assumed std-dev values, as well as on the quality of the fit. It is reasonable to adjust the std-dev so that a correct value is obtained for Q. *However, then one must not say that the fit is good because a correct* Q *was obtained!* That would be circular reasoning!

3. If the scaled residuals are especially *large* for certain variables, then one may consider that perhaps the std-dev values for these variables have been specified too small. Then one may increase std-dev for these variables.

4. If the scaled residuals are especially *small* for one variable, then two explanations are possible:
4a. Too large std-dev have been specified. You may decrease them.
4b. This variable is explained by a unique factor, i.e. one single factor explains practically all variation of this variable. This situation may occur "naturally", so that it is not an error at all, if one source emits practically only one compound. However, this situation may also appear as the result of specifying too *small* std-dev values for a noisy variable. Then it is essential that the std-dev values are increased to a realistic level, see Paatero and Tapper, Environmetrics (1994).

Have you tested different numbers of factors?

It appears that no mathematical criteria are able to predict the "correct" number of factors. In environmental work, there may be no "correct" number at all, because nature is never fully described with any number of factors. Rather, the question is which number of factors gives the most useful solution. In this connection, one has to consider rotations, too. With certain numbers of factors, there may be more need of rotation before the result may be interpreted as source signatures, say.

Lee *at al.* (1999) observe how the rotational uncertainties (elements of matrix *rotmat*) increase when the number of factors is increased over a certain limit. They interpret this increase as one clue for suggesting the best number of factors. It is not yet known if this approach is ov general validity.

Have you computed multiple results, starting from different pseudorandom starting points?

This question is crucial for PMF3, but also important for PMF2. The factor analytic least squares models often contain local minima in addition to the global minimum. The programs are not guaranteed to find the lowest minimum. Once they get attracted by one (local) minimum, they cannot proceed to another one. By running several times, starting from different starting points (different SEED values), one may explore the different minima. It is recommended that you look at the computed factors corresponding to a few of the deepest local minima, in addition to the global minimum.

In practice, there are two different ways of running several times. First, you may literally run again. Change the value of SEED, then the program will compute a different set of pseudorandom values in the next run. Second, within one single run you may compute several times by increasing the *repeats* code from its default value (=1). In addition, you need to activate the repeated computation of initial values by setting the (R) codes to true on those lines of the I/O table that correspond to the initial values of factor matrices. For large problems, the second alternative is more convenient because such a run may be done overnight, without human supervision.

-- If you have found out that there are no local minima when solving your problem with 4 factors, say, you should *not* automatically assume that the same be true for other numbers of factors, such as 3 or 5!

If the data matrix uses negative values for indicating missing data, did you also remember to specify the optional command "missingneg r" (or "BDLneg r1 r2")?

If you did not, then the negative values will be used "at face value". This will result in large residuals, large Q, and generally distorted and useless results.

Is there rotational freedom in the results?

In 2-way models, there is always some rotational freedom UNLESS the non- negativity constraints restrict the freedom. If all factor values are well above zero, then all constraints are inactive and there is full rotational freedom. Explore the rotational situation by running with the parameter "fpeak" set to both positive and negative values! Observe the increase of Q when fpeak is set to different non-zero values.

In 3-way models, rotational freedom is only present in special cases. The most usual reason for rotational freedom is that in one mode, two factors are (almost) proportional to each other. Example: assume that the two factor vectors C1 and C2 (= the first and second columns of the factor matrix C) are almost identical. Then the model logically reduces to a 2-way model for the factors represented by (A1, B1) and (A2, B2). Between these factors, there will be full rotational freedom unless non-negativity prevents one or both of the possible two rotations. If you cannot exclude the possibility of rotational freedom, you should carefully report your results so that the reader is not mislead to believing that the results are unique if they are not.

Did you obtain a "degenerate" solution from PMF3?

This question is only relevant when non-negativity is not requested for one (or several) modes when running PMF3.

It sometimes happens that the solution of PMF3 diverges towards large positive and negative values. The contributions of different factors cancel each other to a large extent. Such so called "Degenerate solutions" are a property of the 3-way model itself, they are not caused by the special properties of a specific program. A degenerate solution is mathematically correct but not useful for the practical problem. Using increased regularization prevents degenerate solutions at the expense of slightly distorting the "legal" solutions. Use the optional command "minreg r" for specifying an increased level of regularization in PMF3.

Are the computed factors written in a suitable format?

In environmental studies, concentrations of trace elements are very small. Using a fixed-point format, e.g. F10.4, may cause that such concentrations appear as zeros although the values are significantly non-zero. The safest solution is to use a floating-point format, such as (G13.5E2).

What about some extra convenience?

It is inconvenient when the factors appear in random order in the result files. (The columns of factor matrices appear in arbitrary order. However, the columns of matrices G and F appear in the same order, and similarly for A, B, and C.) By using the optional commands sortfactorsg or sortfactorsf (for PMF2) and sortfactorsa, sortfactorsb, or sortfactorsc (for PMF3) you may have the factors appear in approximately the same ordering from run to run. This makes comparing different results much easier!

In order to keep track of different results, it may be convenient to let the programs write the Q value to the headers of all result matrices. You need to take care of two details:

1. append the two characters "Q=" to headers of all result matrices in the .ini file, and
2. make sure that the headers are written together with the matrices (the FIL code must be the same non-zero value for writing the header and for writing the matrix).

In this way it is safe to let the programs write repeatedly to a cumulative file. You will always be able to find out what goes together with what. In order to create a cumulative output file, you must have the file Opening Status = ’UNKNOWN’ instead of ’REPLACE’.

Are you using an obsolete version of PMFx?

From time to time, do check the FTP site ftp://rock.helsinki.fi/pub/misc/pmf/ for newer versions of the programs. If downloading a new version, make sure you do not destroy the previous version before ascertaining that the new one works with your data. Always first use the new "tst" version, which is more safe, although also more slow. After using the "tst" version for some time, you may download the corresponding "opt" version (if it exists), in order to gain a little in program speed. The handbooks and guides keep changing, too. Read the file readme.txt frequently, it tells what features have been changed in the programs.

The problem of free rotations in 2-way FA

What are the free rotations

Basically, all results of 2-way factor analysis possess rotational ambiguity. This means that different factor values may produce exactly the same fit of the data matrix. No statistical criteria may be utilized for choosing among such equivalent solutions. Non-negativity of factor matrices G and F ("scores" and "loadings") limits the domain of possible rotations. Sometimes this limitation is strong enough so that no rotational ambiguity remains in the results of PMF2. More often, some ambiguity remains despite of the non-negativity constraints. In the 3-way PARAFAC models the rotational ambiguity is often entirely eliminated. If two of the factors are similar to each other in one of the modes, then there may be similar ambiguity as in the 2-way case. The discussion in this section is mostly for 2-way models but it also applies to such 3-way cases where rotational ambiguity is present.

The original use of factor analysis was in areas such as psychometrics. There the factors are abstractions in the minds of researchers. Then one may choose freely between different rotations, no rotations may be considered *wrong*. Several techniques have been developed for creating "good" rotations. The best known of these techniques is *varimax*. One should understand that varimax does not find the *correct* rotation, whatever it might mean. The merit of varimax is simply that it finds rotations that tend to be satisfactory in certain areas of research.

The applications of PMFx are mostly in physical sciences, such as chemometrics and environmental sciences. In these areas one looks for such factors which correspond to existing entities, such as sources of pollution or chemical compounds. Then a good rotation reproduces the true compositions of these existing phenomena. Achieving such rotations requires that *a priori* information be utilized. Non-negativity is an example of such *a priori* information. As already mentioned, simply assuming non-negativity is often not sufficient for fully eliminating rotational ambiguity. The two hard questions are: what a priori information can be safely assumed, and how to communicate this information to the program.

After-fit rotations vs. during-fit rotations

It is possible to perform rotations *a posteriori*, after the least squares fit has been computed. (This is what is done in PCA = Principal Component Analysis). Then there is a problem with interpretation of non-negativity constraints. If one decides to obey non-negativity strictly, then the domain of possible rotations becomes restricted in an unrealistic way: a rotation which otherwise appears reasonable may be impossible because it would produce small negative noise-like factor elements (scores or loadings). On the other hand, ignoring the non-negativity constraints is also bad: then arbitrary rotations are allowed and one is back to the full ambiguity which is characteristic of PCA.

In PMF2, rotations may be performed during the least squares fit. This means that the user influences the fitting process in some way so that the result appears rotated. The result is usually not an exact rotated image of the original unrotated result. This is caused by the simultaneous processes of rotation and constrained least squares fitting. If some factor elements try to become slightly negative because of the rotations, the fit changes slightly so that these elements still obey the constraints. This leads to an increase of the Q value.

There remains the question of accepting or rejecting results obtained when enforcing a rotation. The increase of the Q value might be utilized as a criterion for rejecting a result. If the Q increases too much because of a rotation, then one might assume that the result is not valid any more. However, this question appears extremely complicated and no statistical criteria are so far known for judging the allowable increase of Q. Current practical experience suggests that an increase of tens of units is certainly acceptable, while thousands of units is questionable or forbidden. Scientific common sense should be guiding the rotational experiments!

Different techniques for rotating with PMF2

In the following, four different techniques are discussed for inducing rotations with PMF2. The most straightforward is pulling selected factor elements to zero. This technique should be utilized whenever one knows that a certain factor cannot contain a certain chemical compound.

Using the rotational parameter FPEAK has two merits: 1. it explores the whole variability covered by rotations, and 2. using FPEAK is simple because one only chooses one parameter value. In certain chemometrics problems, FPEAK may be able to find the best rotation. However, in typical environmental problems the best solution is not at the extreme ends of the rotational domain. Then FPEAK cannot find the best solution.

Using target factor shapes is the newest and the most complicated of the rotational techniques. For the time being, there is no practical experience about the usefulness or about the problems of this potentially powerful approach. If you try this approach, please tell about your experiences!

Inducing rotations in PMF2 (1): parameter FPEAK

The rotations may be visualized as additions and subtractions of factor vectors, see Paatero and Tapper, 1994. The "*negative*" direction of rotations corresponds to subtracting columns of matrix G from each other while adding the corresponding rows of the matrix F to each other. The opposite "*positive*" direction means subtracting F rows and adding G columns (calling the directions positive and negative is quite arbitrary). In some applications, good rotations correspond to situations where all possible positive or all possible negative rotations have been performed until the increase of Q prevents further rotations (see Paatero, 1997, for a spectroscopic example). The parameter FPEAK of PMF2 causes positive (if FPEAK>0) and negative (if FPEAK<0) rotations.

Using FPEAK for producing rotations must be based on trial and error. First try with small values, e.g. FPEAK=+0.1 and FPEAK=-0.1, and observe the increase of Q and changes of G and F factors. Then increase FPEAK and repeat the exercise until the results are not meaningful any more. It may be practical to start each new computation so that the results of the previous computation with a smaller FPEAK are used as the starting point of the next computation.

There is no theoretical reason for assuming that any value of FPEAK will produce a “good” rotation. In many applications, a good rotation is found. However, it is easy to set up examples where no value of FPEAK will produce a good rotation. Tha main virtue of FPEAK is that it enables one to inspect the range of possible. In some sense, the rotations achieved by largest possible positive and negative values of FPEAK are extreme, so that all other rotations are qualitatively “between” these extreme rotations.

Inducing rotations in PMF2 (2): parameter *rotcom*

The matrix *rotcom* may be used for detailed control of individual rotations. However, this option has not been used in practical applications and its use is not recommended. The other suggested techniques should be used instead.

Inducing rotations in PMF2 (3): pulling individual factor elements towards zero

Sometimes, *a priori* knowledge tells that a certain factor element should be zero or very small. In analyses of atmospheric aerosol, it may be known that certain key elements are not emitted by certain sources. For example, the concentration of sulfate is very low in seawater. Thus one might wish to have no sulfate in the marine factor. This may be achieved by *pulling down* the sulfate concentration in the marine factor in the following way.

First run PMF2 in the usual way, without enforcing rotations. Identify the element in factor matrix F that corresponds to sulfate in the marine factor. Call this element f_{ms}. Then set up an .ini file that defines a continuation run of PMF2. The data file and std-dev file are used in the same way as in the original run. The lims value of the first iteration level should be decreased so that it is (almost) the same as for the second level. The results (G and F) of the previous run shall be input as the initial values of the factor matrices. (If headers were written by the previous run, you must remember to also input those headers in the continuation run).

The *pulling down* operation is controlled by the matrix "Fkey". This matrix of integer values is of same size as F. Each element of Fkey controls the behavior of the corresponding element in the factor matrix F, see Part 2 of the User's guide. Note that reading of Fkey is influenced by the "*transpose*" parameter similarly as reading/writing F itself: factors run along rows if *transpose* is not specified. The influence of Fkey is exponential, thus an increase or decrease of two units is often quite significant. The numerical values of factors will influence the value of Fkey or Gkey that is needed for achieving the desired effect: for trace elements, different Fkey values may be needed than for the main constituents of the samples.

Construct the matrix Fkey so that all elements equal zero, except the element $Fkey_{ms}$. Set $Fkey_{ms}$ =9, this causes that the concentration f_{ms} is pulled towards zero with a "medium-strong" pull. Change the .ini file so that instead of specifying the zero code value (FIL=0) for Fkey, Fkey is input from a file. It may be convenient to include the matrix Fkey as part of the .ini file by setting its file code to be FIL=4. It is convenient to use the repeat notation when writing Fkey. If one row of Fkey consists of 25 zero values, then this row is simply written as 25*0 (do not use a period in the integer values of Fkey).

Run PMF2 with the modified .ini file and observe the results. If Q has decreased from the initially obtained reference value, it means that the original solution was in fact a local minimum. Then adopt the new solution as the reference solution for all further experiments. Usually Q increases from the reference value. If this increase was too large, or if the factors appear distorted, the pulling-down was too strong. In such a case, run again with $Fkey_{ms}$ =7 or $Fkey_{ms}$ =8, say. If Q increased by a large amount, while the factor values hardly changed at all, it means that no free rotation was possible for decreasing factor element f_{ms}. Such a result would indicate that the initial assumption, underlying the rotational attempt, was in fact not correct. In the marine aerosol, chlorine might be leached out in an exchange reaction where sulfate replaces chlorine. If such a reaction has happened, then sulfate would be an essential part of the modified composition of the marine aerosol, and it might be impossible to eliminate sulfate from the marine aerosol by performing rotations.

It is possible to use the matrix Fkey for pulling several factor elements towards zero. Simply insert positive entries for all $Fkey_{ij}$ that correspond to factor elements f_{ij} which should be pulled towards zero. It will probably be safest to introduce these puller entries one by one. In that way one sees at once if one of the puller terms is too strong, causing an excessive increase of Q. If you are going to publish factor analytic results that have been obtained by pulling-to-zero, it is essential that you also publish details of the pulling: what factor elements were pulled, why, and how strongly. Without such information, it would be impossible for a reader to replicate your analysis.

Inducing rotations in PMF2 (4): using target factor shapes

Target Transformation Factor Analysis has been used successfully for many applications, including environmental factor analysis (Hopke 19??). Target shapes for factors may be included in a PMF2 analysis. This approach is based on "pseudo measurements" that are included in the data matrix **X**. Each pseudo measurement represents the target shape for one F factor (one row of the F matrix). The std-dev values corresponding to each pseudo measurement should be set so that each std-dev value reflects the strength of the information regarding the corresponding element in the target vector. If some elements in the target factor profiles are in fact unknown, then large values should be inserted for the corresponding std-dev elements. It is essential that the analysis is configured so that each of the target rows of the enhanced data matrix is explained by only one factor. This is achieved so that most of the corresponding weight coefficients in the G matrix are pulled down, leaving only one non-zero element on each row of the G matrix.

Using target factor shapes is illustrated by an example. Assume that the chemical composition has been analyzed of 111 atmospheric aerosol samples; 22 elements/compounds have been determined. Then the dimension of the matrix X is 111x22. Assume that 5 factors are used in the analysis. Then the dimensions of the factor matrices G and F are 111x5 and 5x22, respectively. Non-negativity of all factor elements is required. First perform an ordinary PMF2 run in order to have good starting values for the rotational experiments. Call the factors computed in this initial run G^o and F^o.

In order to apply target factors, the dimensions of the following matrices are changed: X, T, G, G^o, and Gkey. To matrix X, 5 new rows are introduced, on top of the matrix. The rows 6 to 116 of the enhanced matrix X contain the original X. The rows 1 to 5 are reserved for the target shapes of factors 1 to 5. Similarly, 5 rows are also introduced on top of the std-dev/T matrix T. These rows are reserved for the tolerance information of the target shapes. A matrix of dimensions 5x5 is introduced on top of the matrices G, G^o, and Gkey. The non-diagonal entries of the extra 5x5 G matrix are pulled strongly to zero so that $g_{ik} \approx 0 \quad \text{if } (i \neq k \text{ and } k \leq 5)$. Then the least squares fit equations for the pseudo measurements are

$$x_{ij} = \sum_{k=1}^{5} g_{ik} f_{kj} + e_{ij} \approx g_{ii} f_{ij} + e_{ij}$$

showing that each pseudo measurement is indeed approximated by the corresponding row of the matrix F, multiplied by the corresponding element g_{ii}.

In transposed form, the enhanced matrix Gkey that effects pulling down of G is typically

$$\mathbf{G}key^T = \begin{array}{|c|c|c|c|c|c|}\hline 0 & 9 & 9 & 9 & 9 & 111*0 \\ \hline 9 & 0 & 9 & 9 & 9 & 111*0 \\ \hline 9 & 9 & 0 & 9 & 9 & 111*0 \\ \hline 9 & 9 & 9 & 0 & 9 & 111*0 \\ \hline 9 & 9 & 9 & 9 & 0 & 111*0 \\ \hline \end{array}$$

Rows 1 to 5 of the enhanced matrix Go should be set to zero. Rows 6 to 116 of Go should contain the G matrix, as computed in the preliminary PMF2 run.

The new rows of X and T should be set according to what *a priori* information is available about factor shapes. There may be some factors without any *a priori* information at all. The no-information X rows corresponding to such factors should be set to zero, and the corresponding no-information T rows to unity.

For other factors, there will be information about some factor elements, or perhaps about all of the factor elements. The new X elements should be set equal to the assumed *a priori* values of the factor elements, or equal to zero for factor elements lacking *a priori* information. The corresponding new T elements should be set so that they reflect the uncertainty of *a priori* information for each of the new X elements. For such X elements where no *a priori* information is available, the corresponding T elements should have a very large value, e.g. 10^8. Whenever very strong information is available, the T element might be between 0.01 and 0.001 times the corresponding X element. If robust mode is being used, then the T elements may need to be smaller than in a non-robust computation.

In the atmospheric example, there would probably be reliable information about most components of the marine factor. The composition of sea salt would be used as the assumed factor profile. However, for certain elements no information should be assumed: It is well known that Chlorine may be leached out of aerosol particles. Because of their organic precursors, Iodine and Bromine are enriched in marine aerosol.

Several elements of the soil factor could also have some *a priori* information. Unfortunately, several "standard soils" have been proposed, with different compositions. Thus the T elements for the soil factor should be larger than for the marine factor. The uncertainties of the composition of the soil factor may well be close to the factor of two.

For pollution factors, there may be *a priori* information in some special cases. One should be cautious and avoid using questionable information, except perhaps in the style of "what if" studies. Some information may come in the negative form, saying that factor xx (whose most important element is yy) should not contain element zz. This information may presented in the following way: set X elements yy and zz of factor xx to 1.0 and 0.0, respectively, and the corresponding T elements to 0.01 and 0.001, say.

The results of a PMF2 run with target shapes should be carefully evaluated in order to avoid using such results where *a priori* information distorts the true factor shapes. Part of the observed increase of Q comes from the target equations; the magnitude of this part can be estimated by observing the residuals.

A worked-through real example: aerosol measurements in Thailand

To be written....

Requesting support from the author

The author would like to help new users of PMFx in getting their jobs done. This help is, however, limited due to time constraints. If other work is pressing, this help will have to wait. Some support will soon be installed on the WWW pages of the Department.

Preferably, ask for help by e-mail, the internet address is Pentti.Paatero@Helsinki.Fi. From time to time, check for new information and/or for new program versions in the ftp site ftp://rock.helsinki.fi/pub/misc/pmf/ .

Except during holidays and visits to other laboratories, this e-mail is read daily. Another possibility is to send a diskette (1.44 MB), the address is: Dr. Pentti Paatero // Dept. of Physics // University of Helsinki // BOX 9 // 00014 HELSINKI/UNIVERSITY // FINLAND

Although fax is handy, it may be less suitable for asking for help, because data files would have to be retyped after being transmitted by fax, and there is hardly any time for such work. The fax number is: +358-9-191-8680.

When you ask for help, provide all the following information, otherwise there might be no answer! Send the information as plain ASCII text, as part of your ordinary email or as a text-type attachments. Don't use any kind of encoding, such as uuencode or binhex or base64. Do not send Microsoft Word .doc files or Excel .xls files, because of the very significant virus risk that comes with these files (.rtf is OK). You should not automatically assume that the author has the tools for unpacking anything that you are able to pack! — In some cases some entries of the following list are not needed, but *you should have clear reasons for not sending any items from the list*.

- Date message (written when the program starts), file size, file date etc. of your PMF2.EXE or PMF3.EXE copy.
- If there are error messages, send a **verbatim** copy of them!
- A copy of your .INI file and of the data files it should read (unless the files are very large)
- A copy of the file PMF2.LOG or PMF3.LOG (preferably run with Monitor =-1)
- Copies of the result files written by the program. If you send the files as email attachments, long lines will not cause problems. Otherwise, in order that the files be e-mailable, you might

need to output the matrices so that there are no long rows. Write F side matrices in transposed layout or in a format where 5 values are written on one line!

- Your comments about the "error model" EM and about the standard deviations chosen for the array X: why did you select those values and not something else
- Your report about what is wrong or what is missing etc., what kind of results you are expecting but not getting, ... , or perhaps your questions "...how should I proceed in order to ..."

All kinds of feedback is also solicited:

- There may be unclear passages in the documentation. Please tell us, we will try to improve them.

- When using these programs, you are in the front line of the science of data analysis. Several features of the programs have not yet been fully evaluated, not even published. Comparable features don’t exist in other programs. This is especially true for the robust mode, for the estimation of rotational uncertainty, and for the estimation of std-dev of factor values, also for handling multiple solutions, etc. If you have new insight into these questions, please let us know!

- If you would like to use PMFx on another type of computer, let us know! The availability of good Fortran90 compilers is still limited, but the situation is improving! The PMF programs may be compiled for Linux operating system and for Digital Alpha processors (OSF/1).

Bibliography of PMF-related publications

Please send information about your publications where PMF2 or PMF3 has been used! Such information is desired for inclusion in this bibliography.

P. Paatero and U. Tapper, Positive Matrix Factorization: a non-negative factor model with optimal utilization of error estimates of data values, Environmetrics **5** 111-126 (1994)

S. Juntto and P. Paatero, Analysis of Daily Precipitation Data by Positive Matrix Factorization, Environmetrics **5** 127-144 (1994)

P. Anttila, P. Paatero, U. Tapper, and O. Jarvinen, Source Identification of Bulk Wet Deposition in Finland by Positive Matrix Factorization. Atmospheric Environment **29** No 14 pp.1705-1718, 1995.

M. Juvela, K. Lehtinen, and P. Paatero, The use of positive matrix factorization in the analysis of molecular line spectra. Monthly Notices of the Royal Astronomical Society **280** (1996) 616-626.

Pentti Paatero, Jyrki Mäkelä, and Vilho Jokinen, Factor analysis of aerosol size distributions, J Aerosol Sci **26** Suppl **1** S269-S270 (1995).

P. Paatero and U. Tapper, Analysis of Different Modes of Factor Analysis as Least Squares Fit Problems, Chemometrics and Intelligent Laboratory Systems **18** (2) 183-194 (1993)

P. Paatero, Least squares formulation of robust non-negative factor analysis, *Chemometrics and Intelligent Laboratory Systems* **37** (1997) 23-35.

P. Paatero, A weighted non-negative least squares algorithm for three-way "PARAFAC" factor analysis. *Chemometrics and Intelligent Laboratory systems* **38** (1997) 223-242.

P. K. Hopke, P. Paatero, H. Jia, R. T. Ross, and R. A. Harshman, Three-way (PARAFAC) factor analysis: Examination and Comparison of alternative computational methods, as applied to Ill-conditioned data. *Chemometrics Intell. Lab. Syst*., in print (1998).

R. A. Harshman and M. E. Lundy, PARAFAC: Parallel factor analysis, *Computational Statistics & Data Analysis* **18** (1994) 39-72.

Yu-Long Xie, Philip K. Hopke, and Pentti Paatero, Positive matrix factorization applied to curve resolution. *Journal of Chemometrics* 12 (1998) 357-364.

Yu-Long Xie, Philip K. Hopke, and Pentti Paatero, Calibration Transfer as a Data Reconstruction problem. *Analytica Chimica Acta* 384 (1999) 193-205.

Lee, E., Chan, C. K., and Paatero, P. (1999) Application of Positive Matrix Factorization in source apportionment of particulate pollutants in Hong Kong, *Atmospheric Environment*, 33(1999), 3201-3212.

Huang, S., Rahn, K.A., and Arimoto, R. (1999) Testing and Optimizing Two Factor-Analysis Techniques on Aerosol at Narragansett, Rhode Island, *Atmospheric Environment*, 33(1999), 2169-2185.

Pentti Paatero, The Multilinear Engine - a Table-driven Least Squares Program for Solving Multilinear Problems, Including the n-way Parallel Factor Analysis Model. *Journal of Computational and Graphical Statistics* (1999), Vol 8, Number 4, pp 854-888.