Applications of Factor Analysis to Spectroscopic Methods

Timothy Graham Brockwell

August 1992

A thesis submitted to the Council for National Academic Awards in partial fulfilment of the requirements for the degree of Doctor of Philosophy

Sponsoring establishment:

the
UNIVERSITY
of
GREENWICH

Wellington Street,
Woolwich,
London,
SE18 6PF

Collaborating establishment:

DEFENCE
RESEARCH
AGENCY

Defence Research Agency,
Quality Assurance Technical
Support,
Royal Arsenal East,
Woolwich,
London,
SE18 6TD

This page intentionally left blank

Abstract

Applications of Factor Analysis to Spectroscopic Methods

Timothy Graham Brockwell

A computer program has been developed to perform factor analysis and targ testing on spectroscopic data. The program offers advantages over other softwa available on the PC in its ability to work with data matrices of a size limited only b memory and disk space on the computer, and the inclusion of target testing an iterative target testing routines. The program works with numbers of twice th precision of other software and evidence is presented of the improved accuracy of th calculations.

The application of factor analysis to the UV-Vis spectra of a series quaternary mixtures of transition metal ions is shown. The number of ions in th system is determined and the identities of the ions assigned using target testing. Th effect of inadequate sampling intervals of the data is discussed.

Factor analysis is also shown applied to temperature programmed pyrolysis mass spectrometry (TPPy-MS) data where multiple factors are found to arise from single component. The spectra of three substituted ferrocenes are give 1,4,5,6,7,7-hexachloro-5-norbornene-2,3-dicarboxylic anhydride ferrocene, 1-1'(2,4-dichlorobenzoyl) ferrocene, and 3,4-dichlorobenzoyl ferrocene. The sourc of the multiple components from each of the samples is discussed and used to propo pyrolysis and fragmentation mechanisms. The sensitivity of the factor analys technique is examplified by identification of factors due to contaminated source improper calibration, adsorption effects within the source and bias in the measureme of mass spectra.

The identification of components within mixtures is shown using target testin and the isolation of unknown components demonstrated using both accepted and nov methods of iterative target testing. The properties of the new methods of iterati target testing are investigated. The application of the techniques developed is show using unknown samples analysed by normal analytical techniques. The results of th analysis showed broad agreement with the normal analysis, though the factor analys indicated the presence of a component not observed in normal analysis and al showed incorrect assignment of another.

Acknowledgements

My thanks go to all the people who have made it possible for me to complete this work. To my supervisor, Ed Metcalfe, who has had to put up with my constant demands for his time, goes my greatest thanks. To Tony Walmsley, a fellow chemometrician, without who's ear to bend in times of crisis, I would undoubtedly have not achieved anywhere near as much. To Katherine Timmins, Simon Mundy an everyone at Royal Arsenal East for their assistance and sponsorship.

As usual there is a long list of people who have also helped in a myriad of ways and so my thanks go to John Mendham, Stewart McNaughton, Nick Rees, Geoff Brown, Vernon Rogers, John Gates, Sian Howells, Mick Brown and Gurvinde Wall (The Coffee Crew), Philippe Kahn (Borland), Bill Gates (Microsoft), Bernard Vandeginste, Faye Walker and Gordon Dixon.

This work has been carried out with the support of the Defence Research Agency, Quality Assurance Technical Support, Royal Arsenal East, Woolwich, London.

Contents

Key to abbreviations and notation

Matrix notation

Throughout this text, use is made of matrices, vectors and scalars. The following notation is used to represent the different forms.

Scalars are single numbers and are represented by a lowercase letter, e.g. r and c.

Vectors are a one dimensional array of numbers and are represented by a bold lowercase letter, e.g. $\mathbf{x}$. By convention, all vectors are regarded as column vectors unless otherwise indicated by a prime, e.g. $\mathbf{x}'$.

The components of a vector are scalars and are represented by a lowercase letter and a subscript indicating their position in the vector, e.g. x_i, which is the i^{th} component of vector $\mathbf{x}$.

A matrix is represented by an uppercase bold letter or by an enclosure within square brackets. the position of a vector or scalar within the matrix is identified using subscripts. The first subscript identifies the row of the matrix and the second identifies the column, e.g. $\mathbf{x}_{.j}$ is a vector containing the j^{th} column of matrix $\mathbf{X}$ and x_{ij} is the element from the i^{th} row and j^{th} column of matrix $\mathbf{X}$.

The transpose of a matrix, whereby rows and columns are interchanged, is indicated by the use of a prime, ', in the same manner as that used for vectors.

Additional embellishments are used in some cases, a circumflex (or hat), ^, is used to indicate an estimated, or calculated, quantity in line with normal statistical notation.

A bar placed above a quantity indicates an estimation based upon a reduced factor space.

Following are some examples based upon this notation,

$$\mathbf{X} = \begin{bmatrix} x_{11} & x_{12} \\ x_{21} & x_{22} \\ x_{31} & x_{32} \end{bmatrix} \text{ is a matrix}$$

$$\mathbf{X}' = \begin{bmatrix} x_{11} & x_{21} & x_{31} \\ x_{12} & x_{22} & x_{32} \end{bmatrix} \text{ is the transposed matrix}$$

$$\mathbf{x}_{.j} = \begin{bmatrix} x_{1j} \\ x_{2j} \\ x_{3j} \end{bmatrix} \text{ is the } j^{\text{th}} \text{ column vector of } \mathbf{X}$$

$\mathbf{x}_{i.} = \begin{bmatrix} x_{i1} & x_{i2} \end{bmatrix}$ is the i^{th} row vector

x_{ij} is the i^{th} row and j^{th} column element of matrix $\mathbf{X}$.

Abbreviation	Description
%SL	Percentage significance level
1,1'DCBF	1-1'(2,4-dichlorobenzoyl) ferrocene
3,4-DCBF	3,4-dichlorobenzoyl ferrocene
A	The absorbance of a component
A	A matrix of absorbance values
AET	The apparent error in the test vector
ANOVA	Analysis of variance
ASCII	American Standard Code for Information Interchange
ASI	Automated spectral isolation
b	Coefficient of fit (Bessel's inequality)
C	A matrix of component concentrations
C	The abstract column, or loadings, or eigenvector, matrix
c	The concentration of a component
c	Number of columns in the data matrix
CR	Curve resolution
D	The data matrix
DEC	Digital Equipment Corporation
E	A matrix of absorptivity coefficients
ε	Molar absorptivity per unit path length
EDAS	Exploratory data analysis of spectra
EFA	Evolving factor analysis
EI	Electron Impact (ionization)
EMS	Extended memory specification
EPA	Environmental Protection Agency
ϕ	The angle between two vectors
FCA	1,4,5,6,7,7-hexachloro-5-norbornene-2,3-dicarboxylic anhydride ferrocene
FTIR	Fourier transform infra-red (spectrometry)
GC	Gas chromatography
GC-MS	Gas chromatography-mass spectrometry
GLC	Gas-liquid chromatography
H	The height of a peak in a mass spectrum
h°	The height of a pure component peak
HPLC	High pressure liquid chromatography
IE	The imbedded error term
IND	The indicator function
ISMA	Interactive self-modelling multivariate analysis
ITT	Iterative target transformation
ITTFA	Iterative target transformation factor analysis
Λ	The eigenvector matrix
λ	Eigenvalue (and wavelength)
LC	Liquid chromatography
MCA	Multi-component analysis
MS	Mass spectrometry
n	Number of primary factors
NIPALS	Non-linear iterative partial least squares
NMR	Nuclear magnetic resonance
p	The partial pressure of a component
PC	Personal computer
PCA	Principal component analysis
R	The abstract row, or scores, matrix
r	Number of rows in the data matrix
RE	The real error term
REP	The real error in the test vector
RET	Real error in the test vector
RMS	Root mean square
S/N	Signal to noise
s	Standard deviation
sd	Standard deviation
SEE	Standard error in eigenvalue
SPOIL	The SPOIL function
SVD	Singular value decomposition
T	The transformation matrix
TFA	Target factor analysis
TIC	Total ion current
TPPy-MS	Temperature programmed pyrolysis-mass spectrometry
UV-Vis	Ultra-violet visible (spectrometry)
VARDIA	Variance diagram
$\bar{x}$	Arithmetic mean of *x*
XE	The extracted error
XRD	X-ray diffraction
Z	The covariance matrix

1. Introduction

The following text provides the reader with an overview and general explanation of the principles of factor analysis. For the mathematically inclined, derivations of the formulae involved in factor analysis and the target testing procedure are contained in appendices at the end of this thesis. The details specific to the application of the method in the program are contained in the experimental section and the source code giving the implementation of the algorithms in Turbo Pascal is given in the appendices.

Factor analysis was originally developed for use in psychology where it was used to provide mathematical models describing human ability and behaviour. These models were based around the work of Charles Spearman[1] who proposed a theory of psychology postulating that human intelligence was composed of a single general factor and a specific factor. This theory was to stand for nearly 30 years before it became apparent, from the results of factor analysis, that considerably more than two factors were involved.

In his text on 'Modern Factor Analysis', Harman[2] suggests that the foundations of factor analysis were laid at the beginning of the century in a paper by Pearson[3], this paper, entitled 'On lines and planes of closest fit to systems of points in space', contained the statistical aspects of factor analysis. This work was then taken up and extended by Charles Spearman, to whom the origin of factor analysis is usually ascribed. The early works on factor analysis, though all aimed at explaining psychological theories of human behaviour, provided the impetus for the development of many different methods of factor analysis.

The method of principal component analysis as used in this work is based upon the early work of Pearson[3] and adaptations suggested by Hotelling[4]

1.1. General theory of factor analysis

The definition of the term factor analysis has changed and expanded since the first works on factor analysis. Since those early days the techniques and scope of factor analysis have changed so vastly that a multiplicity of meanings has become attached to the term. The definition adopted for this work is the one offered by Malinowski[5] ,'*Factor analysis is a multivariate technique for reducing matrices of data to their lowest dimensionality by the use of orthogonal factor space and transformations that yield predictions and/or recognizable factors*'.

The procedure of factor analysis involves three main steps; data pre-treatment, reproduction and transformation. The first and last steps can be accomplished via many methods, the second, reproduction, step is always performed to produce a model of the data based around an orthogonal factor space, generally by principal component analysis.

The principal component analysis finds a subset of orthogonal axes within the space described by the data that can be used to describe each of the data points. Because of redundancy in the data the number of factors necessary to describe the data to within experimental error is generally less than the number of samples in the original data set resulting in factor compression. In order for the subset of orthogonal axes to be defined the data must be due to a linear combination of factors, if this criterion cannot be met then the method of principal component analysis is not applicable.

1.1.1. Data pre-treatment

The simplest forms of data pre-treatment are algebraic transformations such as exponentiation, logarithms, reciprocals, etc. These treatments are applied on the basis of theoretical considerations, for example the logarithms of data obtained from a kinetics experiment may be used to reduce the effects of rate constants to an additive property, which is then factor analysable.

Other pre-treatments are applied to assist in cases where different units of measurement or different orders of magnitude are involved. The four most commonly used are covariance and correlation about both the mean and the origin. These four pre-processing methods are all forms of linear transformations and are given in most standard texts on factor analysis.

Correlation about the mean is the traditional form of pre-processing applied before factor analysis, it maintains the spatial information contained in the data but looses both the origin and the magnitude of the original information. Correlation about the origin maintains the zero point of the data but still looses the relative size information. Covariance about the mean maintains the relative size information but looses the zero point of the data. Covariance about the origin does not alter the data in any way thus preserving the magnitude and origin information. The different techniques find uses depending upon the characteristics of the data; mass spectroscopy data has both an absolute zero point and a common scale for magnitude.

The use of the four pre-treatments above was studied by Rozett and Petersen[6] using the mass spectra of 22 alkyl benzenes and they concluded that with both R and Q analysis (R analysis has the data with rows composed of the samples and columns of spectra, Q analysis is the opposite) the use of covariance about the origin was the best method of pre-treatment as it preserved both the origin of the factor space at zero and also the relative sizes of the components. For this reason the analyses performed in this work on experimental data all use covariance about the origin as the data pre-treatment.

1.1.2. Reproduction

In the reproduction step the data is decomposed using principal component analysis into a series of eigenvectors. The eigenvectors form an optimised orthogonal co-ordinate system. The optimization is accomplished by extracting the eigenvectors from the data successively and finding each eigenvector such that it accounts for the maximum variance in the data. On removal of an eigenvector from the data, the contribution made to the data by that eigenvector is then removed thus allowing the next eigenvector to be identified by the maximum variance criterion once more. By removing the contribution of each eigenvector from the data after the analysis the orthogonality of the eigenvectors is assured. The term orthogonality is defined in mathematics as two vectors being orthogonal if their dot product is equal to zero, this can be described graphically as visualizing the eigenvectors as axes on a scatter chart and orthogonality meaning that the axes are perpendicular to each other and are therefore independent.

1.1.2.1. Principles of factor analysis

The previous explanation is likely to mean little to a reader not versed in the ideas and terminology of factor analysis so an example is given below to explain the rationale of factor analysis.

Consider a hypothetical data matrix **A** containing absorbance measurements of 3 mixtures of the same absorbing species at 3 different wavelengths, such data might be obtained from a kinetics experiment.

It is known from the Beer-Lambert law that at constant path length the absorbance, A of a component is described by

$$A = \varepsilon c \qquad (1)$$

Where ε is the molar absorptivity per unit path length and c is the concentration, and that for a mixture the absorbances are additive so

$$A = \varepsilon_1 c_1 + \varepsilon_2 c_2 + \cdots \varepsilon_n c_n = \sum_{j=1}^{n} \varepsilon_j c_j \qquad (2)$$

Where ε_j is the molar absorptivity coefficient of component j, c_l is the concentration of component 1 and the sum is taken over n components.

The data matrix **A** contains nine absorbance readings each of which may be reduced to a summation as in equation (2). An example is shown below for 2 components.

$$\mathbf{A} = \begin{array}{c|ccc|} & & \text{mixture} & \\ \lambda & 1 & 2 & 3 \\ 1 & 0.371 & 0.713 & 0.219 \\ 2 & 0.271 & 0.515 & 0.202 \\ 3 & 0.349 & 0.641 & 0.265 \end{array}$$

$$= \begin{bmatrix} \varepsilon_{11}c_{11} + \varepsilon_{12}c_{21} & \varepsilon_{11}c_{12} + \varepsilon_{12}c_{22} & \varepsilon_{11}c_{13} + \varepsilon_{12}c_{23} \\ \varepsilon_{21}c_{11} + \varepsilon_{22}c_{21} & \varepsilon_{21}c_{12} + \varepsilon_{22}c_{22} & \varepsilon_{21}c_{13} + \varepsilon_{22}c_{23} \\ \varepsilon_{31}c_{11} + \varepsilon_{32}c_{21} & \varepsilon_{31}c_{12} + \varepsilon_{32}c_{22} & \varepsilon_{31}c_{13} + \varepsilon_{32}c_{23} \end{bmatrix}$$

where ε_{12} denotes the absorptivity coefficient at λ_1 of component 2, and c_{21} denotes the concentration of component 2 in mixture 1. (3)

This large matrix may, by applying the standard rules of matrix multiplication, be broken down into two matrices comprising one of concentration values, **C**, and one of absorptivity coefficient values, **E**.

$$\mathbf{A} = \begin{array}{c|cc|} & \multicolumn{2}{c}{\text{factors}} \\ \text{coeff.} & 1 & 2 \\ 1 & \varepsilon_{11} & \varepsilon_{12} \\ 2 & \varepsilon_{21} & \varepsilon_{22} \\ 3 & \varepsilon_{31} & \varepsilon_{32} \end{array} \begin{array}{|ccc|c} & \text{concentrations} & & \\ 1 & 2 & 3 & \text{factors} \\ c_{11} & c_{12} & c_{13} & 1 \\ c_{21} & c_{22} & c_{23} & 2 \end{array} \qquad (4)$$

Thus in matrix notation equation (4) becomes

$$\mathbf{A} = \mathbf{EC} \qquad (5)$$

This solution of the equation is readily acceptable to the chemist as it involves parameters that are known and understood. To calculate this solution mathematically requires complete knowledge about either E or C. In most chemical problems this information is not available and the solution not calculable. It is however possible to find an infinite number of mathematically correct, but abstract, solutions to this equation giving the form

$$\mathrm{A} = \mathrm{RC} \qquad (6)$$

In this equation the matrix, **R**, is related to the rows of the data matrix and has dimensions $r \times n$ (number of rows by number of factors) and the matrix, C, is related to the columns of the data matrix and has dimensions $n \times c$ (number of columns by number of factors). It should be noted that the matrix, C, is not the same as the matrix, **C**. The two matrices are related, as are matrices R and **E**, but whilst **E** and **C** contain physically recognizable values of concentration and absorptivity coefficient the matrices **R** and **C** contain abstract values which when multiplied together produce the original data. The relationship between the pairs of matrices is explained later in the transformation section.

1.1.2.2. Graphical interpretation of principal component analysis

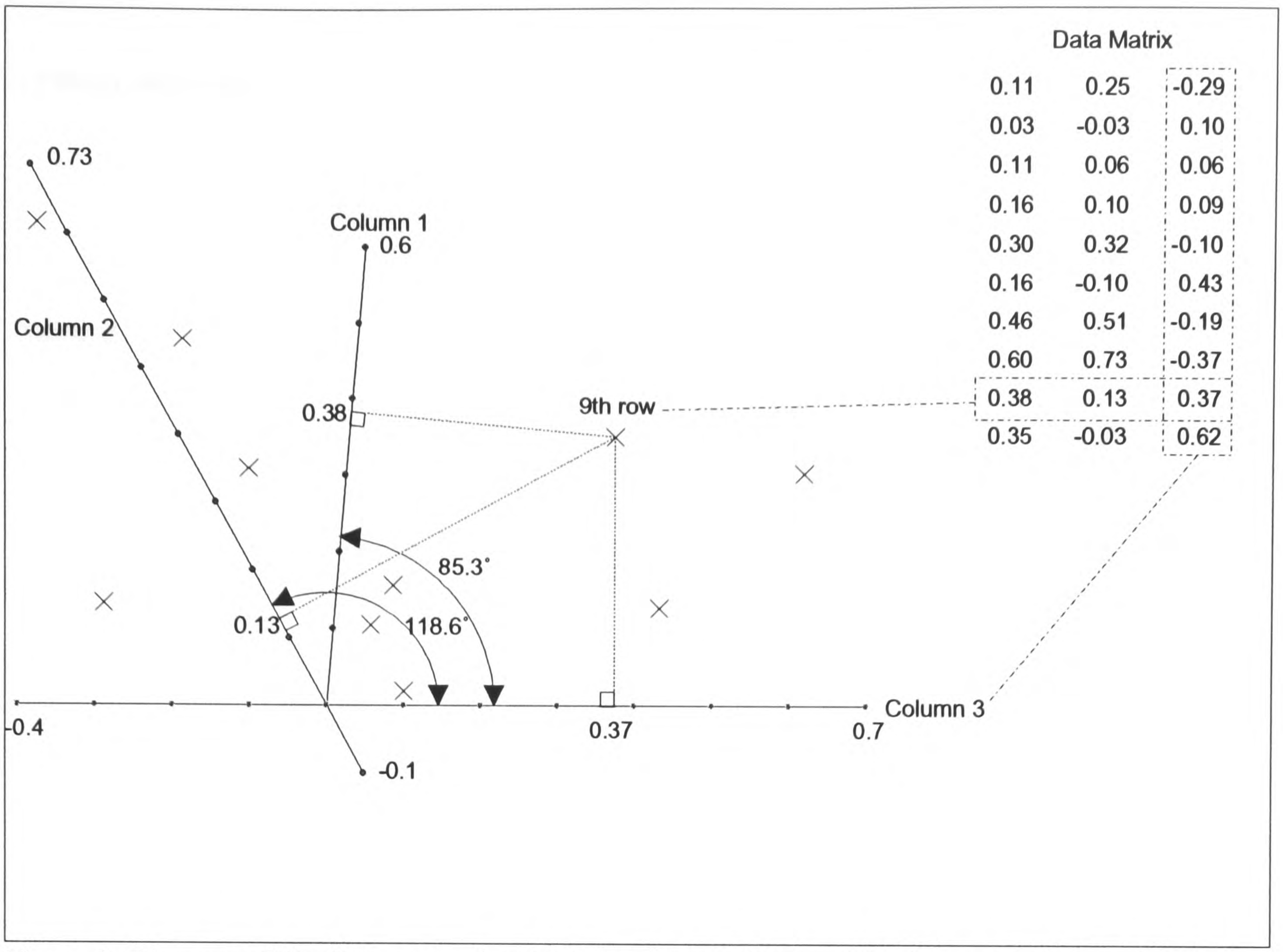
To understand how it is possible to have an infinite number of solutions it is necessary to explain the idea of the data defining a space. The measurements on the columns (samples) in this example are a series of absorbances at different wavelengths. Each of the columns can be thought of as forming a vector or axis, the space contained between the axes contains all the data points. The angles, ϕ, between the vectors formed by the axes can be found from the covariance matrix using a theorem of analytic geometry[7], which states that for vectors **x** and **y** the following relationship holds.

$$\phi = \arccos \frac{\mathbf{x}'\mathbf{y}}{\|\mathbf{x}\| \|\mathbf{y}\|} \qquad (7)$$

This allows the vectors to be placed in space as shown in figure 1. Each column of the data is treated as a vector and the angles between the vectors found from equation 7. Each row of the data matrix forms a point within the space bounded by the column vectors and the value of each of the data points on that row can be found by the projection of the point onto each column vector in a manner identical to the Cartesian co-ordinate projection of a point onto the axes of an XY graph.

In our example it is known that all of the measured values can be produced from a linear combination of two factors, namely, concentration and absorptivity coefficient. These two factors form a plane in which must lie each of the vectors for the samples. This can be seen in figure 1 where all three columns of the data matrix, representing three samples, lie in the plane of the paper.

The abstract matrices **R** & **C** are arrived at by a mathematical method called eigenanalysis and is normally carried out using a least squares technique called principal component analysis (PCA).



Data Matrix

0.11	0.25	-0.29
0.03	-0.03	0.10
0.11	0.06	0.06
0.16	0.10	0.09
0.30	0.32	-0.10
0.16	-0.10	0.43
0.46	0.51	-0.19
0.60	0.73	-0.37
0.38	0.13	0.37
0.35	-0.03	0.62

Figure 1: Diagram of the data space of a simple two factor data set. The data space is bounded by the columns of the data matrix treated as vectors, the individual data points are found from the perpendicular projection of a row designee (of which there are ten in this example) onto each column vector.

PCA works by defining a vector that passes through the data such that it accounts for the largest amount of variance of the data. A second axis is then found which is orthogonal to the first and which accounts for the largest amount of residual variance. This process of defining mutually orthogonal axes continues until all of the variance of the data has been accounted for.

The above explanation of PCA whilst accurate is not particularly easy to understand so explanation by example is advantageous.

Consider the series of data points used in figure 1, though the data originates from three column vectors each point may be identified spatially by a pair of co-ordinates defining its position in relation to two perpendicular axes in an identical manner to co-ordinates on an XY graph. The two axes necessary to obtain the co-ordinates of any point may be positioned at any angle about the origin (so long as they do not lie on top of each other) and still be able to define the position of the data points. This explains how it is possible to have an infinite number of possible solutions to the factor analysis problem.

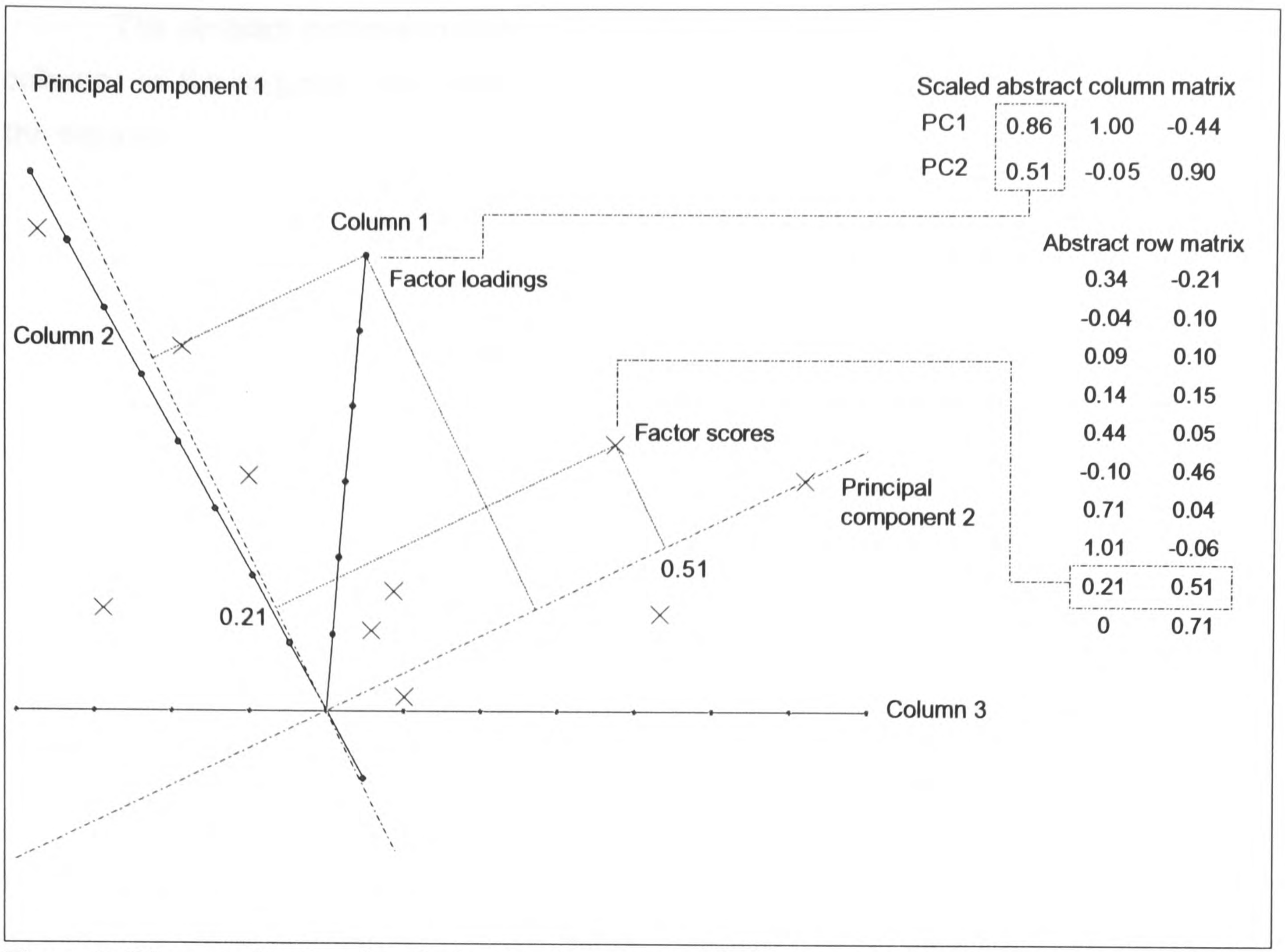


Figure 2: Diagram showing the relationship between the eigenvectors and the data vectors indicating how the values in the abstract row and column matrices can be arrived at geometrically.

To reduce the number of possible solutions some constraints are placed upon the analysis. The axes, or factors, when they are found must be orthogonal. For this simple planar example this means that the axes will be perpendicular to each other. Secondly, the axis is placed to account for the maximum variance in the data, this can be visualized as the axis passing through the greatest concentration of data points.

PCA finds the first of the required two axes in a manner similar to linear regression whereby the axis passes through the highest concentration of data points and therefore accounts for the largest variance in the data.

A second axis is then found which is perpendicular to the first and which in this case would account for all of the residual variance in the data set.

Figure 2 shows the position of the two factor axes (principal components) for the data in figure 1. It can be seen that the first axis has been positioned to pass through the greatest concentration of data points. The second axis lies perpendicular to the first and in this example allows the position of every point to be uniquely identified. The position of each data point is identified by reference to where lines drawn from the point, perpendicular to the axis (shown by the dashed line), intersect the axis. The distance along the axis from the origin (called the projection) produces the score of the data point on that factor. The values of all the scores for each axis comprise the abstract row (or score) matrix, **R**.

The abstract column matrix, which contains the eigenvectors, is related to the columns of the original data when treated as vectors. The relationship is defined by the equation

$$\mathbf{d}_{\cdot k} = \sum_{j=1}^{n} \sqrt{\lambda_j} c_{jk} \mathbf{c}_j$$

where

$\mathbf{d}_{\cdot k}$ = the k$^{\text{th}}$ data column,

λ_j = the eigenvalue of the j^{th} eigenvector,

c_{jk} = the j^{th} coefficient of the k^{th} eigenvector and

$\mathbf{c}_j$ = the j^{th} eigenvector. (8)

The eigenvalue is a constant that describes the importance of the associated eigenvector (factor) to the data. The PCA procedure gives eigenvectors of unit length with the magnitude removed in the eigenvalue.

The $\sqrt{\lambda_j} c_{jk}$ values resulting from this equation are the cosines of the angles between the column vectors and the factor axes and are called the loadings. The loadings measure the relative importance of each eigenvector on each column vector.

The values listed for the abstract column matrix in figure 2 have been multiplied by the root of the eigenvalue and are therefore the cosines of the angles between column vectors and factor axes.

The example used here is trivial in that only two factors and few samples and measurements are involved. Normally much larger numbers of samples and observations are made and the resulting data space can have many more than two dimensions. Depiction of systems of this form is obviously impossible using graphical methods and it must be stressed that here lies the difference between orthogonality and perpendicularity, which thus far have be treated identically. It is mathematically possible to have any number of factors that are not dependant upon each other whereas only three axes can possibly be perpendicular to each other. However, the simple two factor example remains valid for the example in question and will be referred to again in the explanation of transformations.

A complete derivation of the eigenanalysis procedure is given in appendix 1.

1.1.2.3. Effects of error in the data

After the principal component analysis, reproduction of the original data can be achieved by multiplication of the two abstract matrices according to equation 6.

The size of the matrices produced from PCA is determined by the number of rows and columns of the original data matrix, with the row matrix **R** having the same number of rows as the data matrix but only as many columns as there are factors and the column matrix **C** having columns equal to the data matrix and rows equal to the number of factors. There is a limitation placed on the number of factors in that there cannot be a larger number of factors than the smaller dimension of the data matrix. This means that if the evidence suggests that more factors are required to fully describe the data then further measurements must be taken to enlarge the smaller dimension of **D**, the data matrix. This requirement may be readily fulfilled in spectroscopic observations where large numbers of data points can readily be obtained.

In the example in figure 1 the dimensions of the matrices shown are less than those defined above. This is because the data is completely modelled using two factors, the third factor produced has an eigenvalue of zero and its associated vectors in the row and column matrices are also zero and thus may be neglected from the calculation.

The importance of a particular factor is indicated by the size of its eigenvalue. Factors are produced by PCA in decreasing order of importance, the value of each eigenvector's eigenvalue is related to the amount of variance accounted for by that eigenvector. As there is no experimental error in the values used for the example then the third factor has no variance to describe.

If random error were added to the data it would be seen on the figure as a perturbation of the data points in a random direction. It is important to realise that the direction of the perturbation is not restricted to the plane of the paper but might also move a data point out of the plane. This has the effect of increasing the dimensionality of the data as it is now necessary to find a further axis that will come directly out of the plane of the paper to define the extra variance introduced by the error. With experimental data the effects of random error generally ensure that as many factors are found as the smaller dimension of the data matrix. However, as the effect of random error on the data is generally small then the magnitude of the eigenvalue associated with the extra factors will be small. One of the useful by-products of the PCA process is that these eigenvectors, caused by experimental error, may be removed along with their associated columns in the row matrix and the data matrix may then be reproduced by multiplication of the row and column matrices according to equation 9 resulting in an immediate improvement of the accuracy of the data.

$$\overline{\mathbf{D}} = \overline{\mathbf{R}}\overline{\mathbf{C}} \qquad (9)$$

Where the bar denotes the matrix with the secondary factors deleted. It should be noted that only the portion of random error that produces unique vectors can be removed in this way. The portion of error that resulted in perturbation within the plane of the real factors remains in the data.

The separation of eigenvectors into those describing the data (primary eigenvectors) and those describing only the effects of error on the data (secondary eigenvectors) has been the subject of much work resulting in a plethora of different criteria. Many different methods are described in the standard texts on factor analysis but these are generally empirical in their approach to determining the size of the factor space (the number of primary eigenvectors). For use with chemical data matrices several workers have developed measures aimed at determining the dimensionality of the data matrix.

One of the simplest and earliest used for chemical data is the Scree test. This test developed by Cattell[8] was used in the work of Rozett and Petersen[6]. The Scree test operates on the assumption that the residual variance should level off before the factors account for random error when plotted against factor number. The Scree test is not used in this work but a variant on the method is used where the eigenvalue is plotted against factor number and the change in gradient used to indicate the change from primary to secondary eigenvectors.

Another early method for indicating dimensionality is the distribution of misfit. This term is calculated as the number of points in the reproduced data matrix, $\bar{\mathbf{A}}$ in this example, which deviate from the true value more than three times the standard deviation of the original data matrix. The data is reproduced using factors successively from the first factor until a tolerable number of misfits are found. The problem with this method is the arbitrary choice of the number of misfits that may be tolerated. The method was included in this work in an attempt to spot outliers in the data, unfortunately the large number of calculations involved in its determination precludes its use with very large data matrices.

The standard error in eigenvalue was developed by Hugus and El-Awady[9]. Based upon a statistical criterion for the vanishing of an eigenvalue, it is calculated using an estimate of the error in the data. When the eigenvalue falls below the value calculated for the standard error then the factors must describe only error.

A substantial amount of work has been performed by Malinowski on the subject of error in factor analysis. In his paper developing the theory of error in factor analysis, Malinowski[10] derives three related functions for use in determining the size of the factor space. The real error (RE), imbedded error (IE) and the extracted error (XE) are all related via a Pythagorean relationship. The terms RE and IE are used in this work and are calculated for successive factors to produce graphs of error term against factor number.

RE gives the difference between the pure data unaffected by error and the raw experimental data. This means that if an estimate of the error contained in the data is known then comparison of that with the value of RE will define the size of the factor space when the value of RE falls below the estimate.

Interpretation of IE is not as simple. The IE is a measure of the difference between the pure data and the factor analysis reproduced data. This function will start at a large value because too few factors are being used and a large difference exists between the pure and factor analysis reproduced data. Its value will fall as more factors are included until, after the dimensionality of the factor space is reached, pure error vectors are being added. This addition will increase the difference and once again the IE term will increase. When plotted, the correct number of factors should appear as the minimum on the chart.

Unfortunately the IE term is not always sensitive enough to the minimum and in some further work Malinowski[11] develops an empirical term based upon IE called the indicator function (IND). The IND appears to be more sensitive to the minimum than IE and both terms are used to complement each other in this work.

The final error term used in this work is the percentage significance level developed by Malinowski[12] as an extension of work done by Lorber[13]. This term has the advantage that it is based upon well-known and well-understood statistics and can give an estimate of the size of the factor space without any knowledge about error in the data. An F ratio is calculated for the eigenvalue of each factor and using the F distribution converted into a significance level. Work on data sets of previously determined dimensionality indicates that the eigenvalues pass from primary to secondary between 5% and 10% significance.

In almost all cases it is unwise to assign the size of the factor space based upon the evidence of one criterion without considering the evidence presented by the others. In many cases the values offered are ambiguous and can cover a range of one or two factors. In such cases credence should be given to those estimates that use an estimate of error in the data, if the estimate is trustworthy, or if no such estimate is available then the percentage significance level has proved the most reliable in published examples.

1.1.3. Transformation

The transformation step of factor analysis has been performed using many different techniques. One of the earliest and still popular are the rotations. These are more commonly known as cluster analyses and involve the use of one of a multitude of available mathematical transforms which operate on the data to rotate the axes so as to present the data in a form that can be interpreted graphically. They can be useful for identifying groups of points with similar behaviour and thus find an application in chemistry for identifying sources of materials, etc. but they provide a dead end for investigating the data further.

Predictions are the simplest form of transformation possible. The prediction ability of factor analysis is readily explained using figure 2. If the values are known for two of the samples in the data set then the projection of the point onto the two axes identifies its position in space. Once its position is known it is a simple matter to obtain its projection onto the unknown axis thus predicting its value in that sample. The example might be trivial in this case but if a large data matrix containing many samples and had a dimensionality of two then the values for all the samples can be predicted on the basis of two measurements. It is important to note that the prediction is made without any chemical knowledge of the system and it is this modelling ability that has made factor analysis so useful.

A more complicated, but more useful to the chemist, method of transformation is the target test. In the target test a property or function is tested to see if it is a factor in the data or not.

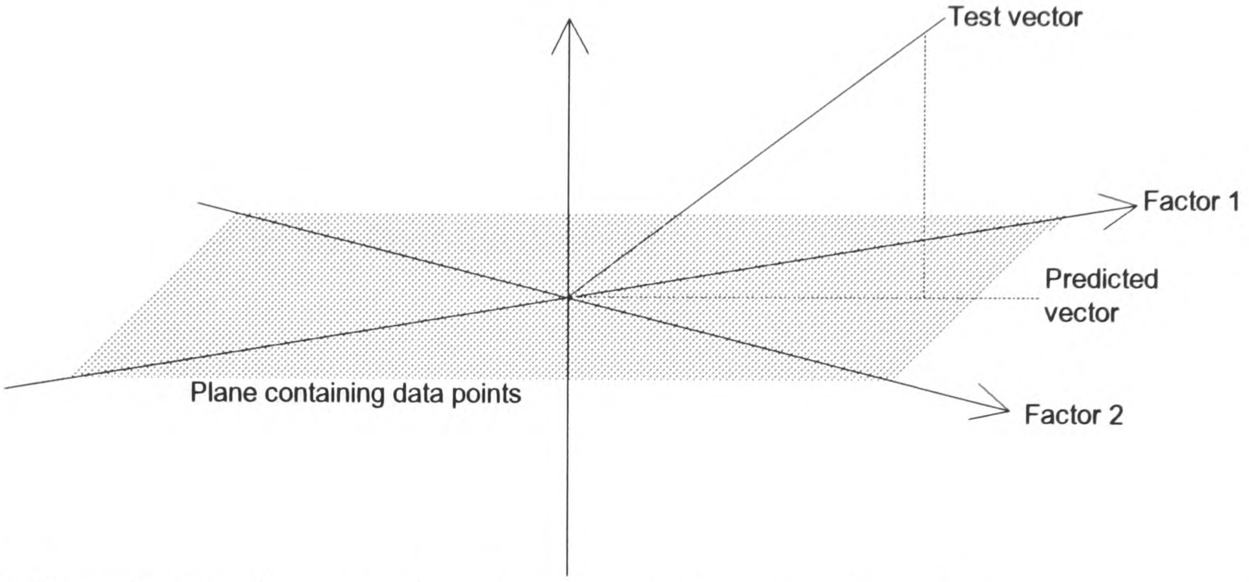


Figure 3: The principle of target testing. The test vector is projected into the data space to produce the closest fit to the test vector of the data.

In target testing, a vector is defined which is believed to be a basic factor of the data set, the vector is shown plotted in the factor space of the data in figure 3. The data, which was described by two factors, lies in a plane defined by factors 1 & 2. The test vector, if it is not a true factor of the data, will lie outside this plane. The target testing process projects the test vector onto the plane of the data producing the predicted vector. The predicted vector is the closest fit to the test vector it is possible to transform the data into. This allows the test and predicted vectors to be compared in order to determine whether the factor is important to the data or not. If the test vector is a true factor in the data then the values for test and predicted vector must be the same.

A complete derivation of target testing is given in appendix 2.

The power of the target test is in the number of properties that can be tested. Columns of the original data can be used which, by definition, must give good tests but these don't give rise to any new information about the data. Tests using chemically meaningful vectors are the most useful. The choice of vector depends upon the data being analysed, in the example, as the data is governed by the Beer-

Lambert law then test vectors of concentration and extinction coefficient would be appropriate. It is also possible to use structural test vectors for testing. These are based upon the structure of the samples and can indicate things like number of carbon atoms or chain branching. The vectors are composed by assigning an arbitrary scale to the features being tested for and creating a test vector.

The target test allows the chemist to probe the innermost workings of a system in the complete absence of any prior knowledge, it also allows single factors to be tested for without interference from other factors in the data.

1.1.4. UV-Vis spectroscopy

The applicability of factor analysis to UV-Vis spectroscopy has already been clearly demonstrated in the preceding example. To reiterate, the pre-requisite condition for factor analysis is that the data under investigation is formed from a linear combination of factors. In UV-Vis spectroscopy with solutions having concentrations low enough to not suffer from non-linear effects, the absorbance of a solution is governed by the Beer-Lambert law.

$$A = \varepsilon c \tag{1}$$

Which when expanded to describe several species takes the form.

$$A = \varepsilon_1 c_1 + \varepsilon_2 c_2 + \cdots \varepsilon_n c_n = \sum_{j=1}^{n} \varepsilon_j c_j \tag{2}$$

This linear sum of products satisfies the requirements for successful factor analysis.

1.1.5. Mass spectrometry

The spectrum produced from mass spectrometry is dependant upon many factors, the one producing the most variation is the type of ionization method employed. In the work performed here all ionization was performed using electron impact (EI) ionization. With EI the intensity of a signal is due to the concentration of a component within the source and the ionization efficiency for that component. The ionization efficiencies can vary widely from component to component. However, the efficiencies for a component will be the same for a given set of conditions thus allowing reproducible spectra to be produced. If two components produce a signal at the same m/e ratio the signal is additive. It can be seen that from the preceding discussion that a similar situation exists to UV-Vis spectroscopy with the ionization efficiencies similar to extinction coefficients. The situation in the mass spectrometer is further complicated by the concentration not being linked directly to the intensity of signal but being linked to the partial pressure of the component in the ion source.

Malinowski and McCue[14] have presented an equation to link the intensity (height) of a mass peak to the contributions of each of its components.

$$H_{(i,\alpha)} = \sum_{j=1}^{n} h^{\circ}_{(i,j)} p_{(j,\alpha)}$$

Where,

$H_{(i,\alpha)}$ = the height of the i^{th} peak in the mixture α,

$h^{\circ}_{(i,j)}$ = the height of the pure j^{th} component and

$p_{(j,\alpha)}$ = the partial pressure of component j in mixture α. (10)

Malinowski and McCue then go on to include modifications to allow for discrimination in a molecular leak inlet and to relate partial pressure in the source with partial pressure in the inlet and thus allow quantification of the mixtures investigated. In the work performed here the sample is vaporized directly into the ionization chamber thus precluding any mass discrimination and allowing the use of the formula as it stands.

1.1.6. Temperature programmed pyrolysis mass spectrometry

Westall and Pidduck[15] have described the application of temperature programmed pyrolysis-mass spectrometry (TPPy-MS) to polymer samples. The technique involves the placing of a small sample of the solid under investigation into a sample tube contained in an electrically heated probe. The probe is then inserted into mass spectrometer source so that the vapours exit the sample tube directly into the ionization chamber of the EI source.

The probe is then subjected to an accurately controlled temperature programme where the temperature of the probe tip is raised at a fixed rate. The mass spectrometer is set to scan over the required mass range during this temperature program. The data collected when presented as a total ion current (TIC) against time trace produces the pyrogram for the sample. Features of interest in the pyrogram can be selected and the mass spectrum of the corresponding scan studied to identify the components present.

The temperature program affords a coarse separation of the sample similar in intent to gas chromatography-mass spectrometry (GC-MS) but the nature of polymer samples is such that events occur so slowly that they generally overlap with the evolution of other components of the sample. This co-evolution of components complicates the interpretation of the individual mass spectra and it is here that factor analysis is to be applied.

TPPy-MS provides a perfect example of the ability of modern analytical instruments to acquire many measurements in a short period of time and how much of the information is left unused by the analyst.

1.1.7. Use of factor analysis in chemistry

The growth in the use of factor analysis in chemistry has paralleled the availability of cheap computing power in the laboratory. The large number of calculations involved in multivariate statistics made analysis of even modest data sets time consuming and labour intensive. Early attempts involving the use of statistics used some innovative techniques to simplify the calculations and generally only gave an indication of the rank of the data. Serious attempts to use factor analysis for the interpretation of data had to wait until access to mainframe computers was available.

Malinowski credits the first application of factor analysis to Higman[16] who attempted to use graphical methods to analyse linear free-energy relationships. The impact of the computer into the laboratory began to be seen in the 70's when investigators started to see some of the possibilities of factor analysis and applications in many different areas were investigated.

1.1.7.1. General theoretical development

Many papers have been published containing different approaches and methods of calculation for use with factor analysis.

Brown[17] has published a review of published Chemometrics materials for the period December 87 to November 89. This contains details of papers published in all areas of chemometrics including those related to factor analysis. Brown also gives details of software packages that have been produced to perform chemometric analyses.

This section contains some of the papers that have offered different approaches to solving problems in factor analysis. Following this section are papers where the theory has been applied to specific techniques.

Malinowski[10] proposed a theory for the way in which error mixes into factor analysis based around the fact that pure data results in factor compression and real data invariably gives a number of factors equal to the smaller dimension of the data matrix. He argues that all the factors beyond the true rank of the data are composed of only error and their rejection will result in the factor analysis reproduced data containing less error than the original experimental measurements. The separation of the factors into primary (due to the data) and secondary (due only to the error) allows this error reduction benefit to be achieved. Malinowski points out that identification of primary and secondary factors has been an obstacle to the acceptance of factor analysis into general use in chemistry and proposes three error terms to assist in the identification of the primary factors.

The error terms are developed around the postulate that the addition that random error makes to the data may be split into a portion that is irretrievably mixed in with the data point and a portion that produces the extra secondary axes. The result of his hypothesis is an equation to describe the error of the secondary axes (called the real error, RE) which he argues is a measure of the difference between the pure data and the experimental data thus allowing comparison of this value with a known estimate of error in the data enabling the number of primary factors to be identified. In addition to the RE term Malinowski also derives terms to describe the amount of error remaining in the data (imbedded error, IE) and the amount of error extracted from the experimental data (extracted error, XE). The three error terms are shown to be linked by a Pythagorean relationship enabling the third term to be calculated if two are known. To provide proof for the derived error terms Malinowski uses a variety of synthetic data sets to which a known error matrix has been added and finds that for small data sets the terms are in good agreement.

Malinowski[11] applied his theory of error in factor analysis to a variety of published data sets. The data are from NMR, MS, UV-Vis, gas chromatography and drug activity analysis. Use is made of the IE function to identify the number of primary factors. The theory of errors suggests that for data sets with uniformly distributed errors the IE function will decrease rapidly as the primary factors are extracted from the data, reach a minimum at the correct number of factors and then rise as more and more secondary eigenvectors are added. The results of Malinowksi's work show that if the error is not uniformly distributed then the IE function can fail to reach a minimum. In these situations the use of an empirical function called the indicator function (IND) is proposed. IND has the same behaviour as IE but appears to be more sensitive to the minimum in cases where the IE fails. The application of these terms to the published data allows Malinowski to determine the number of primary factors and then to determine the experimental error in the data set by calculation of the real error function. Comparison of the calculated RE terms and the number of factors produced good agreement in most cases and reasonable explanations were produced for any discrepancies. The analysis of the drug activity data set produced IE and IND values that increased from factor 1 onwards, Malinowski had observed this behaviour in data sets of random numbers and suggests that it is an indicator that the data set is not factor analysable.

Malinowski[18] extended his theory of error in target factor analysis to allow the calculation of an estimate of the error contained in the data produced from combination target factor analysis. The resultant equation is compared with the error estimates produced by the jack-knife method of Weiner et al.[19] for GLC data and produces similar values for the RMS error.

Roscoe and Hopke[20] give the derivation of a function that may be used to calculate an error estimate for each individual loading vector. The new term is calculated alongside the error estimates produced by the jack-knife method and the two values shown to compare favourably. The primary benefit in the use of the new calculation method is in the time saved in calculating the value.

Malinowski[21] considered a special case of the theory of error applied to target testing where a pure vector containing no error is used, such as unity tests or uniqueness tests. In modifying an assumption made in the original derivation the error introduced into the transformation matrix from the data is accounted for and is used to produce a new definition for the SPOIL function with narrower region of acceptability for when pure targets are used.

Eastment and Krzanowski[22] have applied the cross validation method of Wold[23] using a technique based upon singular value decomposition (an alternative method of eigenanalysing the data matrix). Their method only involves one complete decomposition of the data set and the characteristic factors of the sub-matrices are calculated by an updating process involving considerably less calculation. Adoption of this approach makes the use of the technique in determining the rank of the data feasible in terms of calculation time, unlike the computationally intensive method of Wold.

Hopke et al.[24] have described a computer program developed for the analysis of sources in environmental samples. The program runs on a CDC Cyber 175 computer and is written in CDC FORTRAN level 4 extended. The program allows the calculation of the abstract matrices via a Q mode diagonalization procedure. Error terms are calculated for the decomposed data, chi squared, Exner function and indicator function are all calculated and may be used to identify the rank of the data. Target testing is implemented using a system similar to that derived by Malinowski with the inclusion of a weighting matrix allowing the estimated error in the values of the test vector to weight the fit of the predicted vector. This weighting process allows the target test to be used iteratively to arrive at the true factors underlying the data after using a start point of a uniqueness vector (one containing only one value of unity the other members being zero). After selection of the true factors target combination may be performed using concentration values from known samples allowing the concentrations of the sources to be identified.

Hopke and Dharmavaram[25] describe changes made to their software for target transformation factor analysis. Improvements include a modified calculation for indicator function and the inclusion of R-mode analysis in addition to the original Q-mode. They mention that some confusion exists as to the difference between R- and Q- mode analysis, some people define the difference to be the order of multiplication of the data matrix and its transpose whilst the definition preferred by Hopke is that Q-mode analysis is the diagonalization of the matrix of correlations about the origin between the objects and the R-mode analysis is the diagonalization of the matrix of correlations about the origin between the variables.

Johansson et al.[26] investigate the effects of closure on data used for factor analysis. Closure occurs when the original data's dynamic range is compressed by treatments such as normalization. They identify that the risk involved in closed data is one of spurious correlations between variables. They illustrate their argument with examples of GC analysis of pine and spruce trees clearly demonstrating the distortion of the data introduced by the closure. The method commonly used in mass spectrometry of setting the largest peak to 100% and expressing the other peaks as percentages is shown to lose most of the variance in the data reducing the chance of extracting any information from it.

Braden et al.[27] have proposed an alternative to free floating data points in a target test vector. Their method uses iteration of a single data point by starting with a zero value, testing and replacing the zero with the value from the test. The vector is then re-tested to produce a better estimate, this procedure is repeated till a consistent value is obtained. A proof of the convergent nature of the iterative process is given and it is demonstrated that the true answer is the one converged upon.

Malinowski[12] presented a lucid description of development of Fisher variance ratio tests for both abstract factor analysis and target testing. The theory behind the formulae is developed and its application to a wide range of data from previous publications is given. Malinowski demonstrates that the F-tests based on the Fisher variance ratio show good agreement with the known number of factors in the data sets and also compare favourably with the values obtained from the empirical Indicator function. In target testing the F-test shows itself to able to quantify clearly the fit of a test vector and compares well with the spoil function.

Lorber and Kowalski[28] in this correspondence have given an overview of the generalised theory of abstract factor analysis and discuss the relationship between

Malinowski's target test, Brayden et al.'s iterative target testing of a single point and Vandeginste and Gemperline's iterative target testing with non-negative constraints. All the techniques are described as subsets of the EM algorithm[29].

Donahue and Brown[30] have presented a new and computationally efficient method for the orthogonalization of a data matrix. The successive average orthogonalization method uses the fact that the maximization of variance is in certain conditions identical to the mathematical mean and borrows a technique from Gram-Schmidt orthogonalization to adjust spectra to be orthogonal to a vector. The process was born out of the observation that the first eigenvector is always the average of the data set and that the first eigenvector can therefore be calculated in just that manner. The data matrix is then modified by making each spectrum orthogonal to the average by subtracting an amount equal to the dot product between the vectors multiplied by the average spectrum. The resulting data matrix is orthogonal to the first factor thus ensuring the orthogonality criteria, which is not maintained in the related Gram-Schmidt orthogonalization process. The averaging process is then repeated for the modified data set but to allow for the direction of each spectrum the first spectrum is taken as the reference and its dot product with the second spectrum calculated. if the result is positive the spectrum is added to the first or subtracted if not. This new running average is then used for the same calculation with the next spectrum. This results in the second eigenvector and the entire process repeats for successive eigenvectors. As only one pass through the data matrix is required for each extracted vector the technique is computationally very efficient. Donahue and Brown compare the results produced via this method with those produced from the more commonly used methods of eigenanalysis and find that for primary factors they compare very well. They identify a problem with determining the direction of a spectrum when its noise component becomes large but suggest that in these cases the factor is probably secondary anyway. Overall the method presents itself as very usable for data sets with good signal to noise and uniform error distributions.

1.1.7.2. Elution profile and spectral isolation

Since it was first realised that factor analysis could give the number of components present in a data set and that knowing one of the controlling factors allowed the other to be readily calculated (e.g. knowing the spectrum of a component allows its concentration to be determined), workers have searched for method that allow both controlling factors to be identified *a priori*. The following papers contain work performed with this end in mind and chart the development of ever more sophisticated techniques to achieve their aims.

Bulmer and Shurvel[31] used the techniques developed by Hugus and El-Awady to confirm results obtained from band resolution studies of acetic acid in carbon tetrachloride. They used the non-assumption of peak shape offered by factor analysis to complement the lorentzian based band resolution studies and find identical results as to the number of absorbing species in the solutions.

Lin and Liu[32] describe a numerical method they call automated spectral isolation (ASI) for isolating the spectra of component substances in mixtures. The data used must be normalized to a maximum value of 1 before factor analysis. The decomposed data is then tested with all possible vectors containing all values of 0.08 except for one position that contains the value 1. The predicted spectra produced are considered possible candidates for the pure component spectra. Their suitability is quantified by using the RISK function, which gives a measure of the difference between the predicted vector and the pure component factor space. The best fitting test vectors up to the number of components in the system are used as the pure component spectra. The method was tested by recording the IR spectra of 4 compounds and using the Beer-Lambert law to generate 8 mixture spectra. These mixture spectra were then subjected to the ASI procedure and the spectra of the pure components calculated with good agreement. The method is shown extended to electronic spectra where the test vector is modified to a gaussian form to produce suitable predicted vectors. The criteria for the gaussian function, peak location, baseline height and half width are selected using a simplex optimization routine. Again the results are tested on simulated spectra of a mixture of four aromatic compounds and the predicted spectra show good agreement with those of the pure components. A study of the effects of noise on data is shown by use of data simulated with a baseline shift between -0.1% and +3.5% of the full scale added. The predicted spectra show considerable inaccuracies but are still recognisable as the pure component spectra.

Gemperline[33] has shown the use of target testing in the determination of elution profiles of pure components in overlapped liquid chromatography peaks. His method used the data set presented as for Q analysis, with rows as spectra and columns as mixtures. This allows the target test to operate upon the row designees that are related to the concentration domain. The individual pure component peaks are searched for using uniqueness test vectors. The sum of squares of the difference between the test and predicted vectors is used as an indicator of approximation of fit of the test vector. The number of predicted vectors, equal to the rank of the data, which have the smallest difference are then used in further tests as better approximations to the true factors. In order to improve the approximation of the vector all negative values are removed and the vector tested. This iterative process is repeated until the RET is less than the RE in the data or the difference in RMS values of the truncated negative values between successive test vectors is less than RE. On completion of the iterative procedure a complete transformation matrix may be formed and the predicted spectra of the components calculated by target combination. Gemperline demonstrated the use of the technique on simulated data containing 2,3 and 4 overlapping components, he also investigated the effect of chromatographic resolution of the error in the predictions of the elution profiles and found that at a resolution of 0.64 the elution profiles faithfully follow those of the pure components.

Vandeginste et al.[34] used a system of iterative target transformation to resolve elution profiles in overlapping peaks from HPLC with diode array detection. Their approach is similar to that adopted by Gemperline but instead of testing every possible uniqueness vector they align the eigenvectors produced from principal component analysis with the axes of the data space by use of a varimax rotation. After the rotation the eigenvector maxima are used to give the positions of the unity value in the uniqueness vector. The conditions applied for adaptation of the vector were that no point could be below a threshold of 0.05 and that if more than one peak appeared in the vector separated by one or more zeros then the largest peak would be retained and the others set to zero. On solving the elution profiles of the components the original spectra were calculated by target combination. The technique was tested on a series of synthetic data sets containing three, four and six overlapping peaks and comparison was made with the results obtained from curve resolution studies for the three component system. The recorded results compared favourably with the original data used in data synthesis and also with the results from curve resolution.

Maeder and Zuberbüehler[35] have developed a method of factor analysis which uses the information contained in data sets whose samples are related by time, such as chromatographic methods. The evolving factor analysis (EFA) method uses the fact that the components within the sample will evolve at different times. The analysis is performed by factor analysing the first two samples. The next sample is added and the data re-analysed, this procedure is repeated to the last sample. The reverse procedure is possible, starting with the last two samples and adding their predecessors. The resulting values for eigenvalue, when plotted, show the evolution (for forward analysis) and decay (for backward analysis) for each component in the system. The resulting curves are arranged into the columns of the concentration matrix as first approximations to the concentration profiles. The absorbance matrix is then calculated using the pseudoinverse to give an approximation to the absorbance profiles. A further approximation to the concentration profiles is calculated via the pseudoinverse again to produce a better approximation to the concentration profiles. This iterative process is repeated to convergence after modifying the concentration profiles so that negative and non significant values are set to zero. The results of the analysis are absorbance profiles and concentration profiles without any prior knowledge of elution profile or response. The results of one analysis on model data containing three overlapping peaks are given with good agreement between the concentration profiles and absorption spectra being seen.

Gemperline[36] has extended his earlier work on curve resolution[33] using target transformation factor analysis by addressing problems of secondary maxima appearing during the iteration of components and also the difficulty of determining when the component has been completely resolved. His new approach involves the same start to the analysis, searching all possible uniqueness vectors to discover the positions of the components. He then departs from his previous treatment by using a constraints matrix instead of the eigenvector matrix in the target transformation. The constraints used are that only one maxima can exist and that negative values are not allowed. The method is tested on synthetic data sets containing added error and found to produce better approximations to the elution profiles of the species than the previous method.

Gamp et al.[37] have produced a modification to the rank annihilation factor analysis algorithm (rank annihilation is another mathematical technique for determining the number of components in a data set) which allows quantitation of a species with a known spectrum in the absence of any other knowledge of the response of individual components. The method is applicable to data sets that compose different continuous ranges of non-zero response in either the x or y directions, this includes data from LC with UV detection and spectrophotometric equilibrium studies. The algorithm works by using evolving factor analysis[35] to produce an initial solution to the species present. The knowledge of the spectral profile is then used to reduce the data matrix to those measurements which contain no influence from the known species. The reduced data set is then analysed by PCA and the previously determined eigenvector for the known species appended to the eigenvector matrix. Using linear regression the concentration of the known species is then calculated. The method is shown applied to both LC-UV data and equilibrium studies producing good approximations to the correct concentrations for a series of chromatographic resolutions and added errors.

Vandeginste et al.[38] have conducted a study of the effectiveness of two curve resolution algorithms (CR-2 and CR-3) and iterative target transformation factor analysis (ITTFA) in relation to quantitative liquid chromatographic analysis. Their study looked at the effects of changing four main factors in the analysis, namely the ratio of the areas of the elution profiles in the standard, the chromatographic resolution of the components, the similarity of the spectra of the components and the ratio of the areas of the elution profiles in the sample. The experiments were performed according to a factorial design to allow the influence of each factor on the results to be determined by ANOVA. The results for synthetic data containing two components showed that the ITTFA algorithm provided the most accurate results with 0.7% error and the CR-2 and CR-3 algorithms giving 1% and 4.5% errors respectively. The interaction terms of the factorial design showed that the CR algorithms were strongly affected by noise in the data, chromatographic overlap and spectral similarity. The results for the three component systems indicated similar trends. The ITTFA and CR-3 algorithm were then tested on two sets of three component experimental data and the ITTFA method continued to produce the best results. It was noted that the ITTFA method began to breakdown when the chromatographic resolution fell below 0.5 or the relative absorbances of the components differed greatly. The high sensitivity of the technique to small systematic error was noted and it was suggested that a calibration sample should always be used in preference to spectra from another source.

Strasters et al.[39] has provided a very readable study of the strengths and weaknesses of three methods of determining peak identity and retention time in liquid chromatographic analysis. The three methods used in comparisons are multi-component analysis (MCA), target factor analysis (TFA) and iterative target transformation (ITT). The ability of each technique to identify unambiguously the components in a data set and the accuracy of predicted elution profiles and absorbance spectra are considered. Particular emphasis is placed on the effects of chromatographic resolution and media effects on the accuracy of the results. The conclusions drawn are that in all circumstances the spectra of the component in the solvent system being used should be chosen for analyses using MCA or TFA to minimise inaccuracies. For unambiguous identification of components or for quantification of systems with low resolution and all species known, TFA should be chosen. For analysis of systems where the species are not known ITT is the only option and for systems with very low resolution MCA is the method of choice if the correct set of spectra in appropriate solvent can be obtained.

Strasters et al.[40] have continued their work on peak tracking in liquid chromatography by quantifying the observations made in their earlier paper. They derive three equations from experimental results to describe the dependence of the reproduced spectra on relative concentration, resolution and spectral similarity. The three equations are then combined to give a single figure describing the reliability of the spectra produced from the ITT analysis.

Schostack and Malinowski[41] give details of a new method for the separation of overlapping components. The technique uses a form of evolving factor analysis based on the premise that in a well-behaved system the concentration profiles of the components can only be positive and the spectral profile must also be positive. The analysis uses the following steps. Abstract factor analysis (PCA) to determine the number of components in the system. This is followed by iterative key set factor analysis to identify a key set of spectra that best describe the components. If no overlap of the components occurred then this would result in the pure spectra and component concentrations. In the case of components overlapping the concentration profiles of the two overlapping components will exhibit a common negative component. This negative component is added to the respective components profile and the new spectra calculated using the pseudoinverse. The technique is shown applied to both simulated and experimental data producing good agreement with the known results.

1.1.7.3. UV-Vis spectroscopy

The following papers are concerned with the application of factor analysis to UV-Vis spectroscopic data.

Kankare[42] demonstrated the use of factor analysis as a means of determining the number of components in a complex equilibrium mixture. The work involved the study of bismuth-chloride complexes by UV-Vis spectrometry. 17 solutions at 27 wavelengths were studied and the data factor analysed to find its rank. Before factor analysis the experimental data was smoothed by replacing any point whose factor analysis reproduced value differed by more than three times the standard deviation. Points outside this value were replaced by the factor analysis reproduced value. The smoothed data was then factor analysed resulting in what was believed to be more reliable results. The rank of the data was determined by comparison of the residual standard deviation of the absorbances with the estimated standard deviation of the absorbance measurements. The rank is then given by the number of factors whose residual standard deviation is greater than the estimate. Kankare was then able to convert the abstract concentration matrix to reflect real values by the use of existing values of formation constants of the suspected species as initial approximations in a Newton-Raphson iteration method to calculate a transformation matrix. The transformation matrix was then used with a least squares process to determine the best approximation to the true formation constants. Having calculated the formation constants the spectra of the individual species, which would otherwise have been unobtainable, were readily calculated.

Hugus and El-Awady[9] provide an explanation of the applicability of matrix rank analysis for the determination of the number of species present in data governed by the Beer-Lambert law. They point out that because of the addition of random error to experimental data the mathematical rank of an experimental data matrix is generally equal to the smaller dimension of the data matrix. To determine the true rank of the data they develop a statistical criterion for the vanishing of an eigenvalue based upon the idea that the addition of random error to a data point can be described as the addition of a vector to the mean of the observations of that point. In order to confirm their assumptions they also calculate values for misfit and chi-squared tests along with their standard error in eigenvalue function. Their function is tested on two data sets, one of data obtained from the hydrolytic depolymerisation of binuclear cobalt (III) complexes and the other from solutions of methyl red at varying pH's. The ranks of the data sets are found to be three in both cases with the error terms also in agreement.

McCue and Malinowski[43] have applied target factor analysis to unresolved chromatographic fractions. In their work they made two solutions containing o-,p-xylene and ethylbenzene. These mixtures were then passed through an HPLC system resulting in a single unresolved peak. Samples were taken across the peak by means of a 5ml siphon and the UV-Vis spectra of these aliquots determined using a spectrometer. The resulting sets of spectra were then analysed using principal component analysis and the rank of the data determined by calculating error functions real error, imbedded error and indicator function. The rank of the two data sets was expected to be 3 in accordance with the number of components in the system and this was shown to be the case for the first mixture. The values for the second mixture suggested that there were four components in the system. As a means of identifying the source of the fourth component the RMS absorbance differences were calculated between the true data and the factor analysis reproduced data for both 3 and 4 factors. Comparison of the figures showed that for three factors there was a large discrepancy between the true and calculated values for fraction 2. This identified the presence of the fourth factor only in fraction 2 and on removing fraction 2 from the data set and re-analysing the expected three factors was seen to be present. The fourth component in fraction 2 was explained as contamination in the receiving vessel. The identity of the three components was then searched for using target testing. The test vectors used were the three pure components of the mixtures and the pure spectra of m-xylene, toluene and benzene. The results of the tests clearly showed that the latter three components were not in the data and that the former three were present.

Haldna and Murshak[44] have used target testing to estimate the basicity constants of weak bases. Their method involved the measurement of the UV-Vis spectra of solutions of the bases in aqueous solutions of strong acids and the factor analysis procedure was used to overcome medium effects present with some bases. The UV-Vis data was decomposed using abstract factor analysis and the rank of the data determined by the size of the variance accounted for in an eigenvector. The target testing was performed using a series of test vectors calculated by the program and based around the calculated values for protonation fraction of the base. All of the test vectors were then tried by the program and the fit determined by the sum of squares of the differences between the test and the predicted vectors. Having determined the best fitting test vector the data was then target combined to produce the spectra of the mixtures devoid of any medium effects.

Gemperline et al.[45] have shown the application of principal component regression for background correction in multi-component spectroscopic analysis. Their technique relies upon the analyst being able to recreate the interfering species creating the background though no knowledge of the species or their spectra is necessary. Calibration is achieved by measuring the spectra of a series of standards of the species of interest and including in at least one of the standards, the interfering species. Principal component analysis is then performed on the data matrix formed row-wise from the spectra of the standards. The rank of the data is determined from the real error criterion. A transformation matrix is then calculated from the known compositions of the standards. The sample data is formed into a matrix in the same way as the standards and then the data projected into the concentration space of standards with a transformation similar to target transformation. The transformation matrix calculated from the standards is then used to calculate the concentrations of the samples by the normal target combination step. Gemperline and co-workers demonstrate their technique on two experimentally generated data sets, one using transition metal ions and the other using analysis of active ingredients in tablets. Their answers show good agreement to the known concentrations where the rank of the data has been correctly identified, however large errors are introduced if the effect of the background from the interfering species is not adequately modelled.

Malinowski[46] has produced a review of the work done in the application of factor analysis to absorption spectra. An overview of the theory of the analysis is presented and the information available from each step is discussed by reference to work performed by Malinowski and other workers in the field. Applications shown include visible spectra of dyes and Raman spectra of aqueous sulphuric acid.

1.1.7.4. Mass spectrometry

This section contains papers relating to the application of factor analysis to mass spectrometry in general and pyrolysis mass spectrometry specifically. It will be noted throughout that the researchers have used some form of reduced data set in the form of normalized data, averaged scans or averaged analyses or a limited mass range from within the data.

Davis et al.[47] applied factor analysis to the separation of unresolved peaks in GC-MS. To investigate the usefulness of the method they used artificial data to assess the effect of the addition of noise, overlap of the two peaks, peak width of perfectly overlapping peaks and the effect of tailing of one or both of the peaks. They found that so long as the difference between the two peaks (peak width or peak centre) was slightly larger than the added error then the two peaks would be identified even if they overlapped completely. Their results were also tested on experimental data using incompletely resolved mixtures of isotopic carbon monoxide and mixtures of n-hexane and n-heptane and both components were identified in each experiment.

Justice and Isenhour[48] applied factor analysis to a data set of 630 mass spectra containing compounds with seven functional groups. To identify the most important factors for each functional grouping they developed a weighting equation to rank the factors in order of importance and applied it to the factor analysis data after application of a varimax rotation. From the most important factors they identified the masses in the spectra characteristic of each functional group thus demonstrating the ability of factor analysis to divine relationships between functional groups and mass spectra.

Rozett and Petersen[6] have studied the use of factor analysis with mass spectral data. They point out the peculiarities of mass spectral data, namely that it contains a true origin of zero and that intensities at all masses for all compounds are reported in the same units. In their investigation they factor analyse the data after performing various transformations on the data. The transformations investigated are correlation about both the mean and the origin as well as covariance about the mean and origin. They conclude that mean centring of either form destroys the information about the zero point of the spectra and that correlation looses information about the comparative sizes of variables. A description is given of the terminology used to describe the forms of factor analysis, labelled O to Z, explaining that R analysis has rows of entities (compounds) and columns of characteristics (intensities) while Q analysis has rows of characteristics and columns of entities. They go on to test the different forms of transformation on both Q and R analysis for 22 alkyl benzenes using various subsets of the original data. They come to the conclusion that covariance about the origin is the best method to use with mass spectrometry data. The determination of the dimensionality of the data is discussed using methods based upon the Scree plot (residual variance versus factor number) and the mean square deviation between the original measurements and the data reproduced for successive factors and find in both cases that the rank is three. The theoretical attributes of factor transformations such as varimax and quartimax rotations and target testing are discussed.

Rozett and Petersen[49] study the classification of 22 isomers of $C_{10}H_{14}$ using typical factors, principal component analysis and varimax rotation. They determine that three factors are responsible for the data as they account for 99% of the variance in the data. The presentation of results is based around a variation of the Pearson correlation coefficient but lacking the mean centring of the vectors. This coefficient of congruence is used to produce polar plots of the isomers studied to identify clusters of related compounds. Polar plots are given for all three forms of analysis and comparisons noted, the same information is also given in the form of a tridimensional display where the values are plotted on axes that intersect at an angle of 120 degrees. The resulting triangular chart allows the clustering of the data to be observed. They observe three unique clusters and two smaller combination clusters and assign a fragment ion as the source of each cluster. The analysis is continued observing the dependence of the m/e values on the factors. They conclude that the orientation of the m/e values is complex and not as easily interpreted as the case of the isomers. Some tentative assignments are made as to the kinds of fragmentation occurring that give rise to the observed factors.

Ritter et al.[50] showed the application of factor analysis to chromatography-mass spectrometry. In their study, which attempted to simulate the effect of an unresolved peak in a chromatogram by using a series of mixtures of solutions with very similar mass spectra, they used Q-mode analysis. This is a form of PCA in which the data is column normalized before analysis by dividing each element by the mean of the data in the column. The eigenvectors of the data are then found by diagonalization of the covariance matrix to produce a series of eigenvalues and their associated eigenvectors. In order to determine the rank or number of components responsible for the data an error criterion was developed. The earlier work of Hugus and El-Awady was considered as unsuitable as its deduction was based upon the assumption than error in the data would be would be relative and constant. Ritter et al. obtained their data by manually digitising the mass spectra obtained on analysis of their mixtures. The error in their figures was considered to be larger than that present in the data from the instrument and was therefore constant and absolute. They calculated from their error estimate a value for the variance in each measurement and a figure for the standard deviation of each point in the matrix to form an error matrix. The error matrix was used by recalculating the covariance matrix using increasing numbers of eigenvectors until the difference between the covariance matrix and its reproduced form was less than the corresponding error terms in the error matrix. The analysis was applied to four series of mixtures each resulting in the correct number components being found whilst only using between 15 and 20 mass numbers carefully chosen to represent the spectra being investigated. Only a small number of mixtures were used (between 5 and 7). An example of the sensitivity of the technique was given when one of the sets of test mixtures produced a result of one more component than had been included in the mixtures. On investigation it was found that the mass spectrometer source was contaminated with nitrobenzene compounds from an earlier test. Upon re-selection of the m/e values tested to exclude any contribution from the contaminant the results then indicated the correct number of components.

Burgard et. al.[51] have used factor analysis to identify and quantify the nucleosides present in 32 oligodeoxyriboneucleotides. They identified unambiguously between presence and absence of nucleosides in selected subsets of samples but had less success when attempting to identify the four nucleosides present simultaneously. They note that a separation of the nucleosides along varimax rotated axes was clearly visible but that low intensities of some characteristic ions complicated interpretation especially for compounds containing all the nucleosides. They then used an R-type analysis to identify pairs of characteristic ions that gave a ratio between nucleosides. The compounds were then re-analysed using the ratios between the identified ion pairs as the data set and the relative loadings compared with the known ratios. Good agreement was found for those nucleosides with intense spectra and poorer values returned for the less intense spectra. An attempt was made to sequence the compounds but the differences between the spectra due to sequencing differences were submerged beneath experimental error.

Malinowski and McCue[14] use the data collected by Ritter et al.[50] to demonstrate the power of target transform factor analysis for the qualitative and quantitative determination of components in mixtures. An equation is developed which describes the components of a particular mixture in terms of their mole fractions expressed as a ratio between two components. This means that if the composition of one mixture is known then the others may be calculated. The investigation is carried out by the use of PCA on the data matrix followed by target testing using the pure component spectra. Examples of good and poor target tests are shown to demonstrate their differentiation. The data matrix is then expanded by the addition of a mixture of known composition and the decomposition performed again. The decomposed data is then subjected to a complete combination using the spectra of the pure components as constraints and then the compositions of the mixtures calculated using the developed equation. The calculated values were found to be in poor agreement to the values stated in the paper but as no figure was placed on the accuracy of the mixtures then it was concluded that the mixtures had be made with little attention to accuracy as this was not necessary to the earlier work.

In an extension of his earlier work on theory of error for abstract factor analysis[10], Malinowski[52] presents a theoretical derivation to describe the effects of error in target factor analysis. The effects of the error on target testing are quantified by the calculation of a series of root mean square error terms. The terms relate the error from the data incorporated into the predicted vector and the error present in the test vector to the error contained in the predicted vector. The relationship is shown to be Pythagorean allowing the simple calculation of the third error term if two are known. Equations are developed to allow the calculation of the apparent error in the test vector (the difference between the test vector and the predicted vector) and the real error in the predicted vector (the difference between the predicted vector and the pure test vector) thus allowing the third error term, the real error in the test vector (the difference between the test vector and the pure vector). The three terms allow an estimate to be made of the error contained within the test vector even when no information is available on the data used in the test vector. The interpretation of the values of the error terms is discussed and in an attempt to produce some simple values that allow the validity of a test vector as a primary factor in the data the functions RELI and SPOIL are proposed. The RELI function gives an indication of how well the test vector matches a true factor in the data, whilst the SPOIL function provides not only an indication of whether the test vector is a true factor but also gives an indication of how the inclusion of the vector in the target combination step will enlarge or decrease the error in the reproduced data. As a test of the validity of the theory a test matrix is calculated from known values and a defined amount of error added. The simulated data thus calculated was then analysed and target tested using test vectors to which had been added a known amount of error. The results of these simulations show a very good agreement between known and calculated values thus validating the theory. The theory was then applied to the mass spectroscopy data of

Ritter et al.[50] to demonstrate how the RELI and SPOIL functions may be used to determine good and poor test vectors. The data for the cyclohexane/cyclohexene mixtures was used showing good tests for cyclohexane and poor tests for hexane, which was not present in the mixtures. A further example related to NMR was presented in which a study of solvent effects was used. The gas-phase shift of the solvent molecules was used as a test vector giving good indication that it was a true factor in the data and the error terms indicating that the error in the vapour phase measurements was approximately four times that of the solution measurements as was expected from the extra difficulties involved in the vapour-phase measurements.

Rasmussen et al.[53] investigate the use of three separate methods for the identification of unknown components in mixtures analysed by mass spectroscopy. The first of the three methods used is Gram-Schmidt orthogonalization, which calculates a factor space with dimensions equal to the number of spectra used. The presence of a component is then searched for by calculating the orthogonal distance between the library test vector and the factor subspace. The second method uses target transformation as described by Malinowski and uses the Euclidean distance between the test and predicted vectors as the measure of fit. The final method investigated is the use of Bessel's inequality test. This has the disadvantage that the largest possible covariance matrix is created, increasing the calculations necessary. This problem is surmounted by a modification in the calculation of the eigenvectors allowing the smaller covariance matrix to be used. A coefficient of fit is calculated between the normalized test vector from the library data by calculation of the sum of squares of the dot products between the test vector and the primary eigenvectors. The methods were tested on calculated data based on pure spectra and also on real data. The results of the searches produced the correct components for each of the methods though some difficulty was apparent in determining one of the components with two of the methods namely the Gram-Schmidt and target testing on the artificial data. Consideration of the results and calculations necessary for each search allowed Rasmussen et al. to conclude that the factor space calculated by the Gram-Schmidt orthogonalization was very similar to that of principal component analysis but required more calculation. The calculations involved in Bessel's inequality test were the fastest and with the artificial data gave the best results however when tested with real data it was seen that in the case of components with similar spectra the test was unable to correctly identify the component and it was necessary to resort to using target testing and comparing the test and predicted vectors to identify the correct component. The combination of the two methods was chosen as the best method for the library search and further consideration was given to ways of reducing the number of searches necessary by the use of pre-filters. The two used were an overlap pre-filter, which rejected a test if sufficient of the mass values in the data were not in the test vector and also a maximum mass pre-filter, which rejected any test that contained masses above the maximum mass in the data.

Knorr and Futrell[54] used the factor analytical technique on mass spectroscopic data to find the true factors underlying the data. Their method relies upon their being a region in the data set for each component where only that component has a finite intensity and is not contributed to by any other component. The method works by first factor analysing the data and using the indicator function proposed by Malinowski[11] to determine the number of components in the system. Then the reduced loadings matrix is normalized row-wise so that the sum of squares is equal to one. The first mass unique to a component is found by the smallest loading in the first column. The justification for this is that the first eigenvector is an average of the data and the first row of the loadings matrix corresponds to the sum of intensities of all the components at each mass, for a mass number contributed to only by one component then the sum would be expected to be small. The second unique mass point is found to be the one whose value in the second column of the loadings matrix is the most different from the first unique mass. This is reasoned to be so as the data space is spanned by the vectors then if one boundary represents the pure case of one component then the furthest distant point representing the opposite boundary must also be a component. The third and later unique masses are found in the same way but using the value most distant from the average of the already known unique masses. Having found the unique masses the rows of the normalized loadings matrix associated with those mass numbers are used as a transformation matrix to transpose the scores matrix. This is achieved by column normalizing the loadings matrix (in its original form) and multiplying the rows of the scores matrix by the reciprocal of the normalization constant. The scores matrix is then transformed by multiplication by the transformation matrix and the new scores matrix calculated. The scores and loadings matrices are now the mass spectra and concentrations of the data. The technique is demonstrated using the cyclohexane/cyclohexene data of Ritter et al.[50] with the calculated spectra shown to be readily identifiable as the two components and the compositions to be in the correct regions. The inaccuracy in the concentration data is attributed to poor accuracy in the preparation of the mixtures and also to differences in the ionization cross section of the components, which was not taken into account in the calculations.

Aries et al.[55] have used canonical variates analysis to determine the functionality of the silane used in manufacture of reversed phase HPLC packing materials and also whether the packing has been end capped or not. Their method used Curie point pyrolysis of the packing and analysis of the resulting data using GENSTAT on a mainframe computer. The results of their analysis of a series of commercially available materials generally identified the capped materials correctly and allowed the functionality of the silane used to be determined. The results appeared to be affected by the particle size of the samples and unambiguous identifications were not possible in all cases. The results did indicate the applicability of the technique.

Windig et al.[56] in an extension to their earlier work on VARDIA (variance diagrams) produce a form of variance diagram of use in interpretation of time resolved mass spectral data. The VARDIA approach is an R-type analysis with entities (samples) arranged as rows in the data matrix. The data matrix is then column normalized and subjected to principal component analysis. The VARDIA approach then concentrates on the 2D plane formed by two of the factors. This plane is then divided into sectors of between 10 and 20 degrees. A sum of the variances accounted for by loadings contained within a window of 10 degrees either side of the sector orientation is calculated and the results plotted on a polar chart. The resulting plot indicates the direction of the sources of variance and by rotation of factors to align with these directions the spectra of the component responsible for the factor may be obtained. The VARDIA-S analysis is based upon a similar premise but instead of considering the 2D plane formed by two factors the rotations are considered about the origin formed by the spectrometer background and the angles related to the time the spectrum was gathered. The resulting diagram shows the positions of the components in the analysis and with suitable transformation the spectra of the individual components may be found. Examples of the analysis are shown using data obtained from a Curie point pyrolysis mass spectrometer and modified thermogravimetry system as a pyrolysis source for a mass spectrometer with analyses performed on samples of coal, rubber and wood. In all the samples tested the VARDIA-S technique was able to identify the major components and produce a good approximation to the component spectrum.

Aries et al.[57] have evaluated a purpose built pyrolysis mass spectrometry system by comparison of results of three sets of samples, orange juices, pectins and bacteria, between the instrument and a pyrolysis accessory on a conventional instrument. The evaluation of their results was performed using principal component analysis to identify the reproducibility between samples and also the discriminating power of the system. They found that the purpose built system was better at both discrimination and reproducibility for orange juice and pectins but that the conventional instrument performed better with the bacteria.

Aries et al.[58] have applied pyrolysis mass spectrometry to the characterisation of orange juice. In a study of 58 samples of orange juice from different sources it was possible to differentiate, though not unambiguously, between the country of origin and in some cases the processor. Analysis was achieved by Curie point pyrolysis of a centrifuged sample of the orange juice. The resulting scans were averaged and normalized to the total ion current. The data from all of the samples was analysed using principal component analysis and the principal component plots studied. The plot of the first two principal components allowed separation of the components roughly into their country of origin. To improve the separation canonical variates analysis was performed and autoscaling of the original data was tried with no improvement in the separation. The data set was then reduced to the 20 masses that produced the most discrimination in the data set and the data re-analysed. The results from this analysis exhibited a greater separation between the countries of origin and highlighted 6 samples that appeared to be improperly assigned. It was suggested that some impropriety may have occurred when declaring the country of origin with these samples. As a further analysis the samples from the individual countries were analysed and it was found to be possible to identify the individual producer in some cases.

Aries et al.[59] have used pyrolysis mass spectrometry to investigate pectin methylation. They used a Curie point pyrolysis of small samples of pectins of known degree of methylation, averaged and normalized the resulting spectra for each sample and performed principal component analysis on the resulting data set. They found that there was a linear relationship between degree of methylation and the first principal component. They further investigated to identify the mass numbers characteristic of low and high methylated pectins and using pyrolysis-gas chromatography-mass spectrometry made assignments for the pyrolysis products giving rise to the mass values. With this information they proposed pyrolysis mechanisms for the pectins and identified two masses whose ratio would indicate the degree of methylation with a precision close to that of the first principal component. They identified an impurity in one of the pectin samples and observe that the mass values chosen are not necessarily indicative of degree of methylation and may arise from another independent source but offer the work as confirmation of earlier methods of degree of methylation determination.

Magee et al.[60] used pyrolysis mass spectrometry in an attempt to classify 143 fusobacteria. Analysis was performed using Curie point pyrolysis and the data produced analysed using discriminant analysis and clustering techniques. In general they found that the bacteria could be classified correctly according to genus but with some notable discrepancies. They attribute the differences to the conventional classification being based upon chemical reactions and the small amount of enzyme necessary to perform such reaction being insufficient to change the composition of the microbe as a whole and thus the pyrolysis mass spectrometry is unable to determine a difference.

Price et al.[61] have studied the application of principal component analysis to polymer pyrolysis. In a series of experiments they used Curie point pyrolysis to study different types of commercial polymers. Their first experiment used polymethylmethacrylate, polypropylene, nylon 6, nylon 11, nylon 6,6, and polystyrene. The results of each individual analysis were reduced to the maximum 50 variables required by the software by taking the first 50 mass number with intensities greater than 2% excluding the air peaks of 28,32,40 and 44. Each spectrum was then normalized to the TIC and then autoscaled before analysis by PCA. The results were interpreted by means of principal component plots showing the clusterings of replicate analyses and the separation of different samples allowing the differentiation of the sample materials from each other. The second experiment involved the analysis of four polyacrylics differing only in slight changes in the polymer chain. Once again good reproducibility was observed between replicate analyses as clustering on the principal component plots and good separation was seen between the differing samples. The third experiment attempted to differentiate between several samples of commercial polypropylenes but failed to produce reasonable results. Price et. al. then went on to use thermal desorption gas chromatography mass spectrometry on the same samples to further attempt separation. The samples were heated in a furnace and the vapours produced trapped on an absorbent. The vapours were then desorbed into a narrow plug onto a capillary column where they were separated and passed into the mass spectrometer. The data from each run was converted into an average spectrum with the first and last spectra subtracted to reduce the background. The first 50 masses above 2% intensity were again chosen for analysis and this time a separation between the different samples was observed. The separation was swamped by two of the samples and an outlier, these were removed and the data re-analysed resulting in a better separation between the remaining samples. Price et. al. concluded that the pyrolysis-mass spectrometry was valuable in differentiating between different polymers but that for differentiating between different sources or blends of the same type of polymer the information contained in the volatile components of the blend was necessary and obtainable only via the thermal desorption route.

Lee et. al.[62] have applied principal component analysis to GC-MS data with the sole intention of reducing the amount of noise present in the system. Their application uses the non-linear iterative partial least squares (NIPALS) algorithm to extract successive eigenvectors. They use a simple, minimum change in eigenvalue criterion to identify when the eigenvectors being extracted are describing noise only. They chose this method above the real error and imbedded error terms because of the low mass skewed error distribution present in the mass spectroscopy data. They further point out that as the temperature increases then the error in high mass measurements may increase (due to column bleed). The residual matrix resulting from these calculations is then said to contain only noise and by subtracting it from the original data matrix an improvement in signal to noise (S/N) ratio is obtained. The benefit of this technique of noise reduction over the more usual smoothing approach is shown using a test sample of ethyl benzene in varying concentrations. The calculated S/N ratio for the 10 point Savitsky-Golay smoothed and PCA treated data are shown and for all concentrations the PCA treated data is shown to give a much improved S/N ratio. To check that no significant information is being deleted from the spectrum the smoothed and PCA treated spectra are matched with a spectral library producing a better match for the PCA treated data.

Snyder et al.[63] have taken the VARDIA approach of Windig and the KEY SET algorithm of Malinowski to produce a technique called Interactive Self-modelling Multivariate Analysis (ISMA) and applied it to linear programmed pyrolysis mass spectrometry using a triple quadrupole mass spectrometer and atmospheric pressure ionization source. Their study looked at biopolymers and bacteria in an attempt to identify the biopolymers in the bacterial samples. The ISMA approach uses Q analysis of normalized spectra for the principal component analysis. The data is then subjected to a KEY SET analysis where the mass values that are pure to a particular component are identified. The pure masses are then used in association with VARDIA analysis of the loadings allowing the masses associated with the pure masses to be seen and linked with the pure mass using colour coding. Mass spectra of the analysis are then produced maintaining the colour coding thus allowing the component spectra to be seen. To reduce the effect of the instrument background on the spectrum the standard deviation spectra are produced in which only the masses that vary over the course of the analysis are shown thus reducing the amount of background observed. This approach would obviously only work with a constant background. To test the analysis a series of biopolymers were analysed to find their characteristic pyrolysis component spectra. The bacterial samples were then analysed as well as a mixture of the biopolymers and the resulting data subjected to ISMA. The results allowed the identification of some of the biopolymers in the bacterial data and their presence was confirmed by daughter ion mass spectrometry.

Varmuza and Davies[64] give examples of the use of Exploratory Data Analysis of Spectra (EDAS) software. This software uses principal component analysis to produce projections of the data having maximum variance and thus showing clustering of samples with similar spectral features. Considerable emphasis is placed upon data pre-treatment systems in order to emphasise particular characteristics in the data. In an example using 143 hydrocarbon spectra the modulo 14 spectra were analysed (spectrum divided into intervals of 14 mass units and the contents of each interval summed) and shown to be separable into alkanes, alkenes and dienes. They also link one of the factors to conjugation in the compound. As an aid for identifying the class of a particular compound the package also performs discriminant analysis allowing interactive drawing of boundaries between the classes of compound.

1.1.7.5. Chromatography

The papers below are concerned with the work performed in the analysis of data from various chromatography techniques.

Howery[65] discusses the attributes of target transformation factor analysis that make it ideally suited to the chemist for investigating chemical systems. Emphasis is placed on the ability of target transformation to test an individual parameter in the data regardless of the complexity of the data space. Examples are shown of the use of physical and structural test vectors and the ability to free float unknown data points in a test vector and get an estimate of the value of the unknown data points. The use of the combination step is discussed in terms of its ability to model the data without constraining it to a particular model. The combination step is furthered by its use in the prediction of new rows of data by using the combination model.

Wold[66] proposed a method for the identification of the rank of the data space based upon cross validation. The method works by deleting a row from the data matrix and eigenanalysing the resulting matrix. The difference between the points in the deleted row and the target tested values for the deleted row using successive values for the rank of the data are calculated and tabulated. The process is repeated for every row of the data matrix. The prediction sum of squares for each number of primary factors is then calculated and compared with the value for one less factor. If the ratio is less than one then the extra factor yields a better prediction. this process is repeated until the ratio is greater than one giving the size of the factor space. Wold tested his method using the data of retention indices on different liquid phases of McReynolds[67] and finds five significant components after the rejection of 13 outliers in the data allowing the classification of the liquid phases into five groups.

Hirsch et al.[68] used a form of factor analysis known as Correspondence factor analysis (CFA), this form of analysis is popular in Europe and was developed by Frenchman J.P.Benzécri[69]. They analysed the Kovats retention indices of a series of hydrocarbons on a selection of cation exchange resins containing different cations. The CFA technique works by looking for deviation from proportionality between rows and columns using a common factor space thus allowing the deviations to be plotted on the same axis. The two sets of plotted data points, where they occur in close proximity to each other indicate a similarity in the deviation of their values and are said to correspond with each other. This allowed Hirsch et al. to identify specific selectivity effects between individual cations and types of hydrocarbons and to relate these effects to steric and pi-bonding capacity factors in the groups of hydrocarbons.

Howery et al.[70] assessed the predictive ability of target factor analysis by using it to predict retention data for 42 solutes and 24 stationary phases. They compared the data produced from two methods of prediction using TFA with the data obtained using multiple linear regression. Target combination was used to predict new data using a key set of typical vectors. The best set of typical vectors was determined by testing all combinations and keeping the set that produced the smallest RMS error for the reproduced data set. Target testing was used to predict data by use of the free floating feature of the test and the effect of using different numbers of known points was investigated. The results from both techniques were compared with results obtained from multiple linear regression using both specific models (modelling a single row) and global models (modelling the entire data set). Howery et al. found that the answers obtained from both of the TFA techniques were identical when using the same key vectors but that good answers could also be obtained using test vectors selected only on chemical insight. The results from linear regression were of similar accuracy for the specific model but the global model showed a significant drop in accuracy for data prediction. It was noted that the vectors used as independent variables by the linear regression were the same as those in the best key combination set or used a chemically similar species.

Howery and Soroka[71] used target factor analysis in an attempt to produce a model for solute-solvent interactions between 7 straight chain liquid phase solvents and 49 straight and branched chain solutes. Their data set was taken from the work of Zielinski and Martire[72] who had as a result of their own investigation presented an equation involving three terms to describe the analytical data. Howery and Soroka analysed the data set and using the evidence of RMS error calculated between original and factor analysis reproduced data and the theoretical error terms, real error and indicator function, deduced that the rank of the data was also three. Howery and Soroka then looked for any unique behaviour in the data. This was accomplished by reproducing the data after deletion of a single row or column from the data matrix and observing the effect on the calculated RMS error. A fall in the RMS error indicated that the removed vector was responsible for unique behaviour in the data. Two vectors were found to have unique behaviour and their existence was verified by target testing with a uniqueness vector. They then attempted to model the data using a key set of typical vectors and associated the components of the resultant key sets with different chemical properties namely alkane, bromide and iodide solvents. Target testing was then employed to determine the basic factors of the data. A series of different physical properties were tried as test vectors and the square, reciprocal and logarithmic transforms of the values tried. The parameters suggested by Zielinski and Martire were tested and two, alkene uniqueness and solute group delta, were found to be basic factors. The third term, ether uniqueness, produced a poor fit indicating it not to be a basic factor. After testing all the physical properties the three best basic factors were chosen to model the data, namely the unity vector, halide delta, and the logarithm of the refractive index, and an equation presented which allowed the calculation of the retention index with an error fairly near the experimental error.

Howery and Soroka[73] have investigated the basic factors that influence solute/stationary phase interactions. They worked with 18 monomeric stationary phases and 33 solutes forming 8 groups of chemically similar nature. In their analysis they identify 6 underlying factors of the data set by using the evidence obtained from RMS reproduction errors, maximum error in reproduced data, percentage of data points with error greater than the estimated experimental error and two theoretical error estimates RE and IND. A search was then performed for the vectors in the data that contributed most to the data. This was done by two methods, Vector deletion and Vector addition. In the deletion method a single vector is deleted from the data and the data reproduced, those vectors that contribute a lot of unique behaviour to the data will result in a fall in the RMS error in the data. The vector addition method uses the reversed criterion by using a subset of the data containing the most chemically simple type and needing only one factor to describe the data. To this subset is added one vector and the data reproduced, the sets of vectors resulting in the largest increase in RMS error and also in increase of the number of factors necessary to describe the factor space are the ones that exhibit unique behaviour. From these techniques Howery and Soroka noted three solvents with unique behaviour but no solute groups.

It was then attempted to identify key sets of typical vectors that accurately model the data. A percentage representation table was drawn up to identify the typical vectors of most importance to the data. This was achieved by tabulating the percentages of times each solvent vector occurred in a target combination model having an RMS error less than a specified cut off value. The cut off value was arbitrarily set at twice the RMS error of the corresponding data reproduced from the abstract factors. In the resultant key sets it was noted that the three solvents found to have unique behaviour were represented most frequently. A search was then made for factors basic to the data. First the complete set of uniqueness tests were run to confirm the results from the earlier tests, then a series of physical factors were used such as McReynolds constants, molecular weight, etc. The results were very good for McReynolds constants and diglycerol uniqueness and these factors along with others were used to form a key set for target combination that reproduced the data with an error only slightly greater than that in the original measurements.

Following on from earlier work by Hirsch et al.[68], Howery and Soroka[74] applied factor analysis to the retention indices measured at 180°C of 21 alkane and aromatic hydrocarbons on 10 ionic forms of Amberlyst 15 and Chromosorb P. In their analysis they discover evidence of four factors in the data and find unique behaviour for the sorbents Chromosorb P and the magnesium ion form of Amberlyst 15. In the solvents, unique behaviour was noted for 2,2,4-trimethylpentane and 1-heptane. Key combination sets of the abstract vectors were found to reproduce the data to values near the experimental error and a series of tests performed to identify basic factors in the data. The best sets of basic factors found were, for the sorbents, electron affinity, Craig and Nyholm R value, free energy of solvation and the unity vector. The solute basic factors were carbon number, critical volume, temperature coefficient of the enthalpy of vaporization and alkene uniqueness. Using the key sets of basic factors the values for solutes not used in the data set were calculated and found to produce answers in good agreement with the experimentally found results.

Lochmüller et al.[75] have produced, using TFA, a predictive model for retention behaviour in reversed phase chromatography. From the use of abstract factor analysis on 35 different benzene derivatives using a series of two types of ternary mobile phase they found a subset of three solvent systems and four solutes, which was able to characterise the system with an error close to the experimental value. This model was then used to predict the retention behaviour of both solutes and solvents and the predicted behaviour compared to that found from experiment. In tests of a four component mixture of solutes the prediction was found to be very good but an extension to a six component system showed a marked increase in error. It was noted that the model produced low error predictions even for values not spanned by the original data.

1.1.7.6. Other techniques

The application of factor analysis to chemistry is now extensive and the following papers cover a variety of areas that have been studied. The list is by no means exhaustive.

McCue and Malinowski[76] have used the technique of target factor analysis to investigate the infrared spectra of multi-component mixtures. A series of 10 mixtures of four strongly overlapping components, o-,m-,p-xylene and ethyl benzene, two further mixtures were prepared with chloroform as an impurity to test the robustness of the method. The infrared spectra of the mixtures and also of the pure components were obtained using an FTIR instrument and the resulting data factor analysed. The ability of target transformation to identify a component in the absence of information about the rest of the constituents was demonstrated by testing for the compounds using the pure spectra. Target combination was then performed using the absorptivities calculated from the pure component spectra. This procedure resulted in a transformed scores matrix containing the concentrations of each component in the mixtures which agreed well with the true values. The data from the contaminated mixtures was then added to the data and the presence of the contaminant noted by the increase in dimensionality of the data as indicated by the error terms calculated after abstract factor analysis. The contaminant was then tested for using the pure spectrum of chloroform and its presence confirmed by the similarity between the test vector and the predicted vector. Target combination was then performed using the absorptivities of the components and the impurity and the concentrations in the mixtures found. Again, they were in good agreement with the actual values. As a comparison the same concentration values were calculated for the four components using two other methods, solution by simultaneous equations using the four absorbance maxima of the components and regression analysis. The results for both techniques compared favourably for the uncontaminated mixtures but considerable inaccuracies were evident in the concentration values for the contaminated mixtures. These results highlight the power and utility of the target testing technique on systems where there is a lack of information about the identity and number of components in the system.

Roscoe and Hopke[77] used target factor analysis to identify the sources of mineral samples using an iterative target transformation technique. The iterative technique used unique vectors composed of only zero values except for one row designee that is given the value of unity. The transformation calculation is performed and the predicted vector used as a better approximation to the true factor. In the transformation calculation was introduced a weighting factor based on an error estimate for the designee or the data as a whole. The unweighted transformation was also tested. It was found that the weighted transformations converged to a steady value faster than the unweighted ones. It was also found that the sources thus found compared favourably with the results obtained by the normally used methods of construction of linear functions of elemental concentration.

Roscoe et al.[78] have investigated the sources of mineral matter inclusion in coal samples. Their analysis proceeds by abstract factor analysis of the data set and consideration of several error terms, RMS error, chi-square, Exner function and average percentage error, to identify the number of independent sources of mineral matter in the data. It was found that decomposition of the data gave ambiguous estimates for the rank of the data. On deleting the elemental data for carbon, nitrogen and hydrogen the number of factors appeared to be between 5 and 6. This was explained as being due to a natural variation in the ratio of these three elements during the formation of the coal and that also the indistinct rank was caused by the carbon data modelling the organic component in the coal. On the basis of this hypothesis the carbon elemental data was reintroduced and the rank set to 6. The next step in the analysis was to identify the sources of the mineral matter. Roscoe et al. proceeded by using a uniqueness vector for each measured element and then target testing each vector. The predicted vector from each test was then reintroduced as the new test vector. This iterative approach resulted in the predicted vector converging on an estimate of the true source. Information about the true elemental profile of known minerals was used to refine the iterated vectors. The end product of these tests was a set of vectors believed to represent the sources of the minerals in the coal. Target combination of the data was then performed to produce the relative concentrations of the mineral sources in the samples and this data was compared with the data obtained from x-ray diffraction studies of the same samples. The comparison was achieved using linear regression and determining the correlation coefficient of the data. Fit was found to be reasonable for most of the constituents and discrepancies were attributed to differences in the x-ray scattering properties of the different minerals.

Starks et al.[79] have applied target transform factor analysis to data produced from X-ray diffractometry. The samples analysed were of multiphase material such as rock. For each sample the XRD data was acquired and then data on the oxide composition of the bulk sample was obtained by chemical means. The data produced from the elemental analysis and a knowledge of the composition of various minerals allowed a series of test vectors to be constructed. The total number of mineral phases is obtained from the XRD data and the constructed vectors tested. The fit of the test vectors to the true factors is determined by the jack-knife procedure[80]. Once the set of true vectors was achieved the data was target combined to produce the weight proportions of the minerals in the samples.

Howery and Rubenstein[81] investigated bond dissociation data for 14 different radicals. They studied both the full 14 × 14 matrix and a series of selected sub-matrices chosen for their chemical properties. The dimensionality of the data space was determined by consideration of theoretical error terms RE and IND and also the RMS error from combination of the abstract matrices. Target testing was used to investigate the basic factors of the data. A large number (nearly 50) of possible candidates composed of physical properties of the radicals or the molecules formed by the radicals were tested and very good fits obtained. Tests for ionization potential and group electronegativity were used to predict data and fairly accurate values were obtained for the missing values.

2. Aims

I. To investigate the application of factor analysis to spectroscopic data through the development of a standalone PC based computer program for performing factor analysis and target testing with particular reference to temperature programmed pyrolysis - mass spectrometry (TPPy-MS) and UV-Vis spectrometry. The program should be capable of working with very large data sets and be readily modified for experimentation and development of new techniques for investigating spectroscopic data.

II. To study methods for the determination of the number of components in spectroscopic data and to identify and interpret the physical and chemical origins of the components.

III. To analyse the qualitative identification of components present in the data via target testing and to study various criteria of fit for the components tested.

IV. To develop methods for the investigation of the data using incomplete information about the components present and particularly using information gleaned from the data itself (i.e. iterative target testing).

2.1. General comments

The applicability of factor analysis to absorption spectra has been amply demonstrated in many papers, Malinowski[46] provides a review of some of the work performed. There has been considerably less work in the field of mass spectrometry, Malinowski and McCue[14] have produced a theoretical expression for the intensity of signal in mass spectroscopy that conforms to the linear sum constraint necessary for successful factor analysis. Rozett and Petersen[6] have studied the used of data pre-treatments and their effects on the analysis of mass spectroscopy data and concluded that covariance about the origin was the best method of pre-treatment. This pre-treatment method will be adopted for use in this work also.

The factor analysis of spectroscopic data has been achieved using several techniques. Methods such as rank annihilation[37], successive average orthogonalization[30], evolving factor analysis[35], Gram-Schmidt orthogonalization[53] and singular value decomposition[22] have all been used. The method adopted for use in this work is that described by Malinowski & Howery[87]. This calculates the characteristic eigenvectors of the data using a power method with a Wielandt's deflation. The requirement of large data matrices and accuracy will require the development of specific data structures and mathematical routines prohibiting the use of library routines available commercially. The type of analysis performed will be R analysis where the spectra are arranged in columns and the samples or consecutive measurements as rows.

The determination of the rank of the data poses a perennial problem in all fields of factor analysis. The presence of random error in the data invariably means that the result of the analysis will be a number of factors equal to number of rows or columns of the data whichever is the smaller. Many workers have proposed different methods for determining the number of significant factors, the earliest was the variance criterion where the number of factors used to describe the data was increased until some arbitrary value for the percentage variance accounted for was reached, usually around the 99% region. This empirical method is unable to determine the difference between primary and secondary eigenvectors and so other methods were developed. Another empirical measure used was the number of misfits. In this test the data was recreated using consecutive factors and the number of reproduced data points outside a chosen multiple of the standard deviation of the original data are regarded as misfits. The empiricism enters in determining the number of tolerable misfits and the multiple of the standard deviation. Cattel[8] proposed the use of the Scree test in which the residual percentage variance left unaccounted for in the data is plotted against factors used. The graph was interpreted by looking for a change in gradient marking the change from primary to secondary eigenvectors. A modified form of this method will be used in this work, where eigenvalue will be plotted against factor number and the gradient change observed. The Scree test is a considerable improvement over the percentage variance criterion but still lacks a quantitative measure. Malinowski[10] developed a theory for how the error mixes in with the data in a factor analysis and developed a triad of quantitative terms to assist in determining the rank of the data. The real error, imbedded error and extracted error were the terms proposed and the real error and imbedded error will be used for this work. Later work by Malinowski[11] saw the improvement of the information available from the imbedded error by the development of the empirical, but more sensitive indicator function and this will also be adopted for use in the program. Hugus and El Awady[9] proposed a function to calculate the standard error in eigenvalue which when used in conjunction with a good estimate of error in the data has given reliable estimates of rank and this will be included in the program. The final function to be used in the program for rank determination is the F-test described by Malinowski[12] which allows the application of well known and understood statistics to the determination of the rank of the data. In the application of the F-test it is intended to modify the algorithm given by Malinowski to allow the calculation of the F-test after extraction of each factor instead of at the end of the decomposition. This modification will allow the progress of the decomposition to be monitored and halted before completion resulting in a time saving for the operator, it also overcomes the problem of being unable to calculate the F-tests without a full decomposition. One method of rank determination that will not be implemented will be cross validation[66] this method was rejected because of the substantial increase in the number of calculations necessary to perform the method.

In the analysis of the data the problems of closure raised by Johansson et. al.[26] will be considered and in all cases original intensity data will be used as close as can be calculated.

The use of factor analysis in the interpretation of the physical and chemical origins of the data has been shown in work by Justice and Isenhour[48] and Rozett and Petersen[49] who both used factor analysis to identify m/e values characteristic of functional groupings.

The use of target testing in the identification of species present within a system is well demonstrated[65] but a quantitative measure for the quality of the fit is necessary. Work has been performed by Malinowski[52,21] to produce criteria which can be used. The terms apparent error in the test vector, real error in the test vector, real error in the predicted vector, reliability and spoil function will all be calculated in the program. In addition the F-test for target testing[12] will also be implemented. The product-moment correlation coefficient and the Bessel inequality function[53] will also be used to quantify the fit of test and predicted vectors.

The investigation of factor analysis data using iterative target testing has been shown by several workers[27,44,77,78] and a study of the merits of iterative target testing over target testing presented by Strasters et. al.[39,40]. The primary problem facing the use of any successful iterative technique is to find adequate convergence criteria. The use of chemical selection rules ensures convergence towards a chemically meaningful solution. Current users of iterative target testing[33,36,34] work with a Q analysis to enable them to test the information contained within the elution profiles of the species in the data. In these cases it is possible to use two selection rules such as;
'there can be no negative concentrations', and
'only one maxima is possible in an elution profile'.

In the case of the program where an R analysis is performed then the target test investigates the features of the spectra. In this case the only chemical selection rule applicable is that of 'no negative absorbances or intensities are possible' and the use of this rule and other non chemical criteria to aid convergence will be investigated.

3. Experimental

3.1. Data sets

3.1.1. UV-Vis spectrometry data

3.1.1.1. Transition metal ion data

Equipment

Kontron Uvikon 860[82]

Conditions:

Scan range	350-850nm
Scan speed	100 nm/min
Spectral bandwidth	2.0 nm
Wavelength accuracy	0.5 nm
Wavelength precision	0.02 nm
Absorbance accuracy	0.004
Absorbance precision	0.002
Sampling interval	1.00 nm
Path length	1cm
Cell	Polystyrene disposable

Materials

Unless otherwise noted all chemicals were obtained from BDH (Merck) Chemicals Ltd.[83]

Chemical:	Formula:	Quoted Purity:
Copper(II)chloride	$CuCl_2.2H_2O$	98.5%
Cobalt(II)chloride	$CoCl_2.6H_2O$	99.0%
Chromium(III)chloride	$CrCl_3.6H_2O$	97.0%
Nickel(II)chloride	$NiCl_2.6H_2O$	98%
Potassium permanganate	$KMn0_4$	99.5%

Method

Four transition metal ions whose spectra overlapped in the visible region of the spectrum, were used to make a series of solutions of differing concentrations and the solution spectra recorded.

The ions chosen were Cu^{2+} (λ_{max}~ 810nm), Co^{2+} (λ_{max}~ 510nm), Ni^{2+} (λ_{max}~ 395nm) and Cr^{3+} (λ_{max}~ 440nm). Solutions in de-ionised water were prepared from the hydrated chlorides of each of the ions at a rough concentration of 3mg/cm^3. The spectra were recorded on the Kontron spectrometer and approximate values for ε_{max} found. These values were then used to calculate the concentration necessary to produce solutions with unit absorbance and bulk solutions of each ion prepared.

Using a table of random numbers the compositions of sixteen mixtures of the four ions were calculated. Each set of four numbers was scaled to the volume of the flask and the appropriate quantities of bulk solutions combined to make the mixtures. The concentrations of the mixtures are given in table 1.

	Concentrations /mol l^{-1}			
Solution	Cu(II)	Co(II)	Ni(II)	Cr(III)
1	0.007	0.108	0.043	0.004
2	0.008	0.070	0.019	0.019
3	0.015	0.050	0.014	0.021
4	0.028	0.024	0.047	0.013
5	0.014	0.012	0.043	0.026
6	0.021	0.058	0.034	0.011
7	0.028	0.013	0.075	0.009
8	0.010	0.017	0.092	0.016
9	0.030	0.045	0.015	0.013
10	0.016	0.020	0.049	0.021
11	0.023	0.008	0.124	0.002
12	0.009	0.063	0.038	0.016
13	0.004	0.063	0.045	0.018
14	0.011	0.050	0.060	0.014
15	0.006	0.049	0.106	0.007
16	0.040	0.018	0.024	0.012

Table 1: Concentrations of metal ion in the sample solutions

The spectra of the bulk solutions and each of the mixtures were obtained as an analog printout. The size and scaling of the printed output was kept constant so that digitization error was the same for all mixtures. The spectrum of a 0.005M solution of potassium permanganate was also obtained for use in later testing.

The digitization was performed according to the method given later, at an interval of 20nm from 350nm to 850nm producing 26 measurements for each spectrum. The measured heights were then converted to absorbance values and entered into a spreadsheet for further analysis.

3.1.2. Pyrolysis - mass spectrometry data

3.1.2.1. Equipment

VG Analytical[84] 70-70HS double focusing, forward geometry (EB) mass spectrometer,

Variable rate temperature controller and direct insertion probe,

DEC PDP 11/84 data system,

RSX -11M -Plus Version 3.0 operating system,

VG Analytical System 11-250 R3.0-84-A data analysis software.

Unless otherwise stated the following operating parameters were used.

Scan range	10-650m/e
Ionization	EI
Accelerating potential	4kV
Source temperature	180°C
Threshold	500
Probe ramp	60°C/min
Scan type	Exponential down scan
Time per scan	6.61 sec/scan
Signal threshold	10 counts
Peak centre detection	Peak centroid
Spectra Library	National Bureau of Standards & In-house

3.1.2.2. Method

The mass spectrometer is calibrated in accordance with normal practices using perfluorokerosene as a standard.

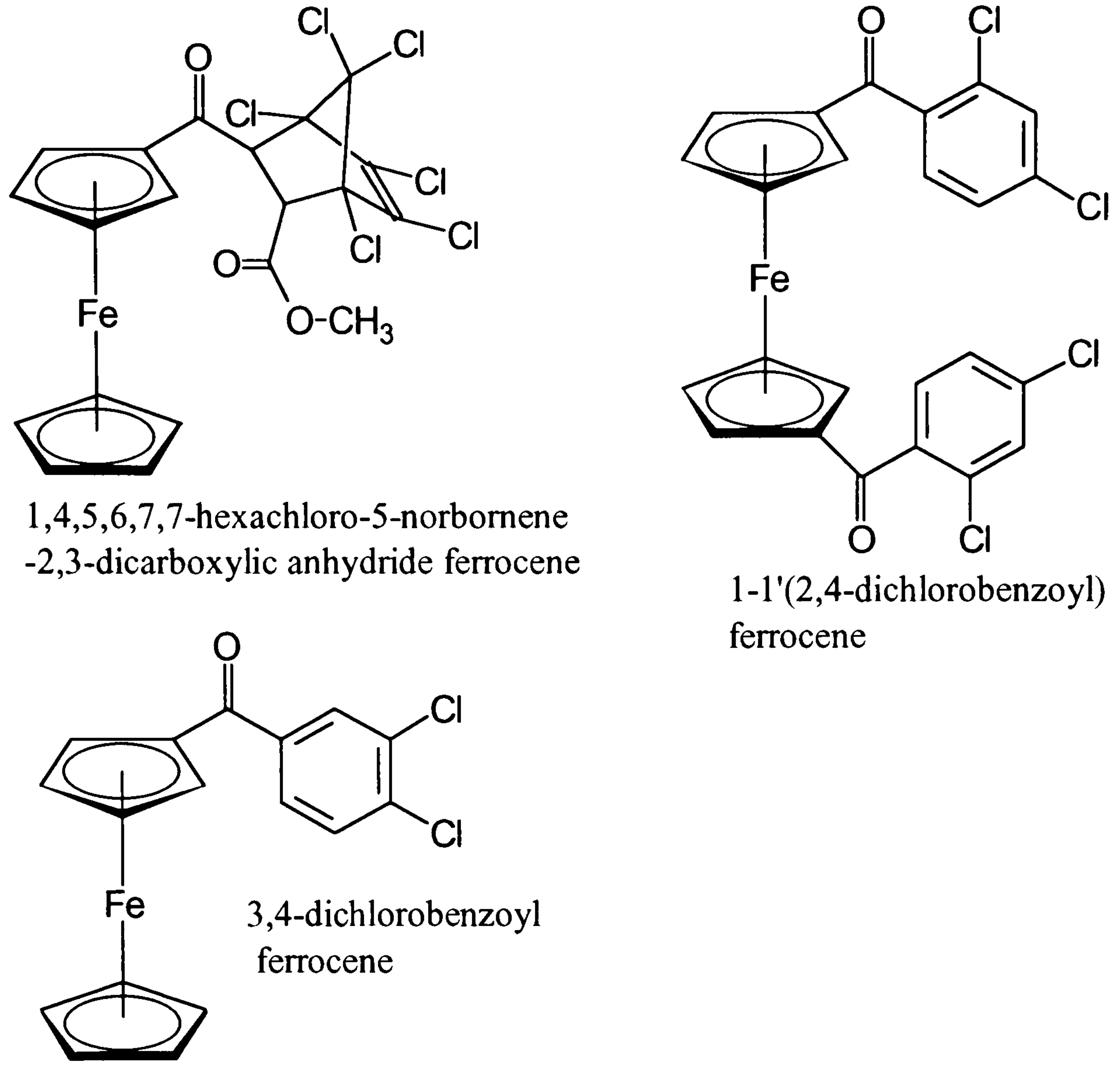
A sub milligram portion of the sample is placed into a 20mm × 1mm i.d. quartz tube, which is placed into the end of the direct insertion probe. Data acquisition is commenced and after collecting a few scans the probe inserted into the source. The probe heating ramp is then started and data acquired until evolution of sample is complete or the probe maximum temperature is reached. On completion of the run the probe is cooled by the integral water cooling system and withdrawn from the spectrometer.

For the initial work it was desired to analyse single component systems that could then be mixed. The TPPy-MS technique is normally used for the analysis of polymers but a pure sample of polymer is difficult to obtain. For this reason three substituted ferrocenes were used to collect data on single component systems. The ferrocenes have high melting points and therefore appear late in the temperature ramp, similar to polymer samples. Their purity has also been verified by elemental analysis, X-ray diffraction, and NMR spectroscopy[85] so the factors they produce will be due to the known sample and the spectrometer only.

The ferrocenes selected were, 1,4,5,6,7,7-hexachloro-5-norbornene-2,3-dicarboxylic anhydride ferrocene (FCA), 1-1'(2,4-dichlorobenzoyl) ferrocene (1,1'DCBF) and 3,4-dichlorobenzoylferrocene (3,4-DCBF). Their structures are given in figure 4.

A series of the above samples were tested and the data collected[86]. The samples and conditions are detailed in table 2.

In order to study how the factors underlying the data changed in mixtures and to study methods of investigating mixtures, a sample was prepared which contained a mixture of FCA and 1,1'DCBF.



1,4,5,6,7,7-hexachloro-5-norbornene -2,3-dicarboxylic anhydride ferrocene

1-1'(2,4-dichlorobenzoyl) ferrocene

3,4-dichlorobenzoyl ferrocene

Figure 4: The structures of the three ferrocenes used in the investigation.

Data set	TB11	TB12	TB13	TB15	TB18	TB19	TB20
Sample[1]	A	A	A	B	C	A+B	Blank
Sensitivity /AFS[2]	10^{-5}	10^{-5}	10^{-5}	10^{-5}	10^{-5}	10^{-5}	10^{-5}
Initial temp /°C	25	25	26	25	26	22	35
Final temp /°C	490	484	500	830	340	556	830
Probe insertion @ scan	8	(i)	8	8	8	8	(ii)
Ramp start @ scan	17	20	17	17	17	17	4
Run end @ scan	91	93	92	147	68	100	132

[1] Sample codes: A=FCA, B=1,1'DCBF, C=3,4-DCBF

[2] AFS = amps full scale

i) An error occurred starting the ramp so acquisition was restarted with the probe in situ.

ii) This run was performed with the pyrolysis tube from TB19 without removal from the spectrometer.

Table 2: Ferrocene data collected and run conditions.

Complex polymer samples

To test the techniques developed it was necessary to use real samples. Two data sets that had been previously analysed using standard techniques were acquired.

Also collected were the spectra of the components believed to be in the samples. These data sets, MS1930 and MS1930A[86], were samples of an easily identifiable polymer but which contained an unknown number of additives.

3.1.3. Literature data

In the course of program development it was necessary to determine that the software produced the correct values. In order to test these, several data sets published in the literature were used and the values produced by the program checked against the published values. The literature data sets are listed in table 3 below.

Authors	Calculations compared	Data source
Malinowski & Howery[87]	PCA, RE, IE, AET, REP, RET, SPOIL, RELI	Synthetic
Cartwright[88]	Factor extraction	UV-Vis spectra
Malinowski[12]	Eigenvalues, F-tests for eigenvalues and target tests, RE, IE, IND	^{1}H NMR - Weiner et al.[89] MS - Ritter et al.[50]
Lorber[13]	F-tests for target tests	MS - Stenhagen et al.[90]

Table 3: Data sets taken from the literature

The synthetic data sets of Malinowski & Howery[87] were based upon a 10×3 matrix of numbers calculated from the multiplication of 10×2 and 2×3 matrices. The data therefore contains only 2 factors. In order to produce error in the pure data a matrix of error values between +1 and -1, selected randomly, was added to the pure data. The data sets are reproduced below.

$$\underset{\mathbf{X}}{\begin{bmatrix} 0 & 4 \\ 1 & -1 \\ 2 & 0 \\ 3 & 0 \\ 4 & 3 \\ 5 & -4 \\ 6 & 5 \\ 7 & 8 \\ 8 & -2 \\ 9 & -5 \end{bmatrix}} \times \underset{\mathbf{Y}}{\begin{bmatrix} 2 & 5 & 2 \\ 1 & 10 & -5 \end{bmatrix}} = \underset{\mathbf{D}}{\begin{bmatrix} 4 & 40 & -20 \\ 1 & -5 & 7 \\ 4 & 10 & 4 \\ 6 & 15 & 6 \\ 11 & 50 & -7 \\ 6 & -15 & 30 \\ 17 & 80 & -13 \\ 22 & 115 & -26 \\ 14 & 20 & 26 \\ 13 & -5 & 43 \end{bmatrix}}$$

Figure 5: Calculation of the pure data matrix

Where **X** and **Y** are the components of the pure data matrix **D**. The raw data matrix ($\mathbf{D}_{raw}$) was created by adding an error matrix (**E**) to the data matrix as shown below. The RMS of the added errors was ±0.477.

$$\begin{bmatrix} 4 & 40 & -20 \\ 1 & -5 & 7 \\ 4 & 10 & 4 \\ 6 & 15 & 6 \\ 11 & 50 & -7 \\ 6 & -15 & 30 \\ 17 & 80 & -13 \\ 22 & 115 & -26 \\ 14 & 20 & 26 \\ 13 & -5 & 43 \end{bmatrix} + \begin{bmatrix} -0.1 & 0.6 & 0 \\ 0.3 & 0.5 & 0.3 \\ -0.7 & 0.2 & -1.0 \\ 0.9 & 0.2 & -0.5 \\ 0 & -0.5 & 0.5 \\ 0.1 & -0.2 & 0.4 \\ -0.6 & 0.4 & -0.4 \\ 0.6 & -0.9 & -0.3 \\ -0.1 & -0.3 & 0.8 \\ 0 & 0.1 & 0 \end{bmatrix} = \begin{bmatrix} 3.9 & 40.6 & -20.0 \\ 1.3 & -4.5 & 7.3 \\ 3.3 & 10.2 & 3.0 \\ 6.9 & 15.2 & 5.5 \\ 11.0 & 49.5 & -6.5 \\ 6.1 & -15.2 & 30.4 \\ 16.4 & 80.4 & -13.4 \\ 22.6 & 114.1 & -26.3 \\ 13.9 & 19.7 & 26.8 \\ 13.0 & -4.9 & 43.0 \end{bmatrix}$$

$$\mathbf{D} + \mathbf{E} = \mathbf{D}_{\mathrm{raw}}$$

Figure 6: The creation of the raw data matrix from its components.

The target testing sections of the program were also tested using synthetic data. The pure test vectors used were columns of matrix **X** and also a unity vector composed wholly of 1's. The raw test vectors were created similarly to the raw data matrix, by the addition of an error matrix to the vector as follows.

$$\begin{bmatrix} 0 & 4 \\ 1 & -1 \\ 2 & 0 \\ 3 & 0 \\ 4 & 3 \\ 5 & -4 \\ 6 & 5 \\ 7 & 8 \\ 8 & -2 \\ 9 & -5 \end{bmatrix} + \begin{bmatrix} 0.1 & -0.1 \\ -0.2 & 0 \\ 0 & 0.2 \\ -0.1 & -0.2 \\ 0.2 & 0.1 \\ 0.1 & -0.1 \\ -0.1 & 0 \\ 0 & 0.1 \\ -0.2 & 0.1 \\ 0.1 & 0.1 \end{bmatrix} = \begin{bmatrix} 0.1 & 3.9 \\ 0.8 & -1.0 \\ 2.0 & 0.2 \\ 2.9 & -0.2 \\ 4.2 & 3.1 \\ 5.1 & -4.1 \\ 5.9 & 5.0 \\ 7.0 & 8.1 \\ 7.8 & -1.9 \\ 9.1 & -4.9 \end{bmatrix}$$

$$\mathbf{X} + \mathbf{E} = \mathbf{X}_{\mathrm{err}}$$

Figure 7: Calculation of the raw test vectors

The columns of the resulting matrix, $\mathbf{X}_{\mathrm{err}}$, were used as the test vectors. The error in each vector is the RMS error in the corresponding column of the error matrix, **E**, for the first column this is ±0.130 and for the second ±0.118.

The UV-Vis data set of Cartwright[88] is composed of a series of spectra of aqueous solutions containing Cu(II) and other transition metal ions. The spectra of the six solutions published is shown reproduced in the following table.

	Solution					
Wavelength	1	2	3	4	5	6
850.0	0.017	0.064	0.013	0.251	0.124	0.126
825.2	0.025	0.065	0.019	0.262	0.129	0.133
799.6	0.039	0.065	0.029	0.266	0.13	0.137
774.8	0.057	0.061	0.042	0.258	0.126	0.138
750.0	0.077	0.055	0.057	0.236	0.116	0.132
725.2	0.09	0.047	0.067	0.201	0.101	0.119
699.6	0.085	0.038	0.064	0.156	0.079	0.097
674.8	0.077	0.03	0.058	0.111	0.06	0.074
650.0	0.077	0.024	0.059	0.075	0.044	0.058
625.2	0.052	0.02	0.041	0.045	0.03	0.038
599.6	0.027	0.017	0.023	0.024	0.02	0.023
574.8	0.013	0.025	0.014	0.015	0.021	0.017
550.0	0.007	0.071	0.019	0.017	0.05	0.033
525.2	0.006	0.148	0.034	0.028	0.102	0.064
499.6	0.004	0.157	0.034	0.028	0.107	0.066
474.8	0.009	0.122	0.031	0.022	0.084	0.053
450.0	0.019	0.075	0.029	0.015	0.052	0.036
425.2	0.069	0.027	0.056	0.013	0.024	0.032
399.6	0.204	0.011	0.153	0.028	0.028	0.068
374.8	0.128	0.004	0.095	0.017	0.015	0.041
350.8	0.023	0.003	0.017	0.003	0.004	0.007

Table 4: Digitized spectra between 850nm and 350nm of aqueous solutions containing Cu(II) ions in the presence of other metal ions.

Solutes	Solvents								
	CH_3CN	CH_2Cl_2	$CHCl_3$	CCl_4	CS_2	CH_2Br_2	$CHBr_3$	CH_3I	CH_2I_2
CH_4	12.1	12.1	12.7	13.8	13.3	13.8	15.2	12.9	15.1
CH_3CN	117.6	118.0	120.0	117.4	114.8	122.7	127.3	122.9	128.8
CH_3Cl	181.6	181.1	180.2	178.8	176.6	182.2	184.1	180.6	185.3
CH_2Cl_2	326.9	319.8	317.4	317.1	313.9	321.2	321.8	322.5	323.5
$CHCl_3$	455.7	438.9	436.1	435.0	432.5	440.8	439.6	446.1	441.4
CH_3Br	160.4	158.8	158.7	157.2	155.8	161.3	162.9	159.6	163.7
CH_2Br_2	305.2	297.6	295.5	295.4	292.9	300.3	299.0	301.0	301.0
$CHBr_3$	425.4	412.8	410.0	409.2	406.9	412.6	411.0	416.6	410.9
CH_3I	130.4	129.3	129.6	128.9	128.5	132.3	133.7	131.0	135.7
CH_2I_2	238.1	233.6	232.1	232.2	232.1	234.9	234.6	235.5	235.0
CHI_3	303.3	295.8	294.5	294.7	293.1	293.9	292.5	296.0	288.2
CH_2ClBr	317.9	311.2	309.4	308.3	306.4	313.9	312.4	314.8	315.6
CH_2ClCN	256.8	248.2	246.1	244.2	242.8	252.3	253.0	255.3	257.8
$CHBrCl_2$	449.7	432.1	430.4	430.0	429.0	435.4	433.0	439.6	434.9

Table 5: Chemical shifts of substituted methane solutes in polar and non polar solvents, in Hz at 60MHz, relative to internal TMS.

	% cyclohexane			
m/e	80%	60%	40%	20%
27	2.3	3.2	3.4	2.1
28	1.2	1.3	1.3	0.7
29	1.1	1.1	1.1	0.5
39	3.9	5.8	6.8	4.9
40	0.7	1.0	1.0	0.6
41	8.6	10.5	10.5	6.0
42	3.5	3.7	3.1	1.1
43	1.6	1.7	1.4	0.4
51	0.5	1.1	1.5	1.0
53	0.9	1.7	1.7	1.7
54	3.6	8.0	11.3	10.1
55	5.1	5.4	4.5	1.9
56	14.2	14.4	11.4	3.6
67	4.4	11.1	16	15.1
68	0.5	0.8	1.1	0.8
69	4.1	4.3	3.3	1.0
79	0.4	0.8	1.2	1.0
81	0.5	1.0	1.6	1.4
82	1.5	4.1	6.1	5.4
84	10.5	11.8	8.5	2.4

Table 6: Digitized intensity values for the cyclohexane/cyclohexene mixtures of Ritter et al.[60]

	% Cyclohexane						
m/e	100%	90%	80%	50%	20%	10%	0%
27	1.8	1.6	2.4	2.6	2.8	2.7	2.8
28	2.1	1.8	4.1	1.8	6.8	2.8	2.0
29	1.3	1.5	2.0	3.3	4.4	4.2	5.1
39	2.5	2.0	2.2	2.5	2.0	2.0	1.6
40	0.7	0.5	0.6	0.5	0.3	0.4	0.4
41	7.1	6.4	8.0	9.3	9.1	8.8	8.8
42	3.5	3.4	3.7	4.6	4.7	4.6	4.7
43	2.2	2.5	3.3	6.2	7.4	7.9	8.6
44	0.2	0.2	0.2	0.2	0.3	0.2	0.3
54	0.8	0.6	0.6	0.5	0.3	0.2	0.1
55	4.6	3.7	4.3	3.5	2.4	1.1	0.9
56	13.5	11.6	12.6	12.5	9.9	8.4	6.8
57	1.2	2.3	3.2	7.3	9.6	11.0	12.2
69	3.8	3.4	3.4	2.7	1.6	1.0	0.2
83	0.8	0.6	0.6	0.5	0.4	0.4	0.1
84	10.7	8.2	8.6	7.3	3.6	2.0	0.1
85	0.9	0.8	0.8	0.6	0.5	0.3	0.1
86	0.1	0.4	0.6	1.6	2.4	2.5	2.8

Table 7: Digitized intensity values for the cyclohexane/hexane mixtures of Ritter et al.[60]

The ^{1}H NMR data of Weiner et al.[89] was used in the validation of the F-test calculations and is listed in table 5. The data was collected in a study of the proton shifts of simple substituted methanes in a variety of different solvents.

Two data sets mass spectrometer data sets of Ritter et al.[50] were used. The first is composed of selected m/e values of a series of mixtures of cyclohexane in cyclohexene and is given in table 6.

The second data set is composed of the intensities of selected m/e values of a series of mixtures of cyclohexane in hexane and is given in table 7.

The test vectors used in conjunction with table 6 were given by Lorber[13] and are reproduced below.

m/e	Test vector					Key	
	a	b	c	d	e	a	cyclohexane
27	18.1	19.4	13.6	11.3	24.9	b	cyclohexene
28	8.9	6.3	12.5	6.5	10.7	c	bicyclo[3.1.0]hexane
29	8.9	2.6	1.0	4.7	4.9	d	fluorocyclohexane
39	20.5	36.6	30.5	13.5	52.0	e	bicyclopropyl
40	5.0	5.0	3.0	2.7	5.1		
41	56.9	36.6	42.5	39.1	64.6		
42	25.8	2.7	2.0	11.4	4.5		
43	12.2	0.2	0.5	7.2	0.3		
51	2.8	8.7	2.2	3.5	5.6		
53	4.1	12.9	8.1	6.1	15.5		
54	6.2	76.9	64.4	27.5	100.0		
55	34.2	5.6	3.2	17.7	7.3		
56	100.0	0.4	0.4	25.1	0.4		
67	2.9	100.0	100.0	100.0	92.6		
68	1.7	5.5	4.6	5.6	3.4		
69	23.4	0.1	0.3	4.1	0.3		
79	0.6	7.4	3.5	1.4	4.5		
81	0.5	11.5	9.6	8.8	8.5		
82	0.2	40.9	21.8	31.1	4.5		
84	72.9	0.1	0.3	0.7	0.3		

Table 8: Composition of test vectors used with data in table 6.

3.2. Program development

3.2.1. Data processing equipment

Dell[91] System 310 PC compatible computer, 20MHz 80386 with 80387 maths coprocessor and 8Mb of RAM.

Borland[92] Turbo Pascal Professional, Version 5.5.

TurboPower[93], Object Professional 1.0 Library.

Borland[92] Quattro.

3.2.2. Data structures

Data can be stored in many representations in the computer and the selection of the most appropriate form can have a major bearing on the ease of design, implementation and modification of the program. The data analysed by PCA is composed of real numbers (not integers) arranged in matrices or vectors. Vectors can be considered as matrices with only one column (or row) so a data structure based around the matrix was chosen as the most appropriate. From the number of possible variables of real type available in Turbo Pascal, extended type was chosen to maintain the maximum numerical accuracy in all calculations.

3.2.2.1. Characteristics of mass spectrometer data

The mass spectrometric data to be investigated in this work was obtained by scanning over a mass range for a number of scans. The mass spectrum produced from each scan is composed of a series of mass numbers and corresponding relative intensities. The sample in the instrument produces signals at discrete m/e values so even for complex mixtures there are many m/e values scanned for which no intensity figure is recorded (electronic noise is removed by the mass spectrometer). For the data set TB19 (see later) nearly 80% of the data points are zeros. In the case of temperature programmed pyrolysis there can be periods in the pyrogram where nothing is evolved resulting in only background peaks being recorded. The combination of these factors results in a large amount of redundant information in the recorded data. The second important characteristic of the resultant data is the number of recorded values, for a typical TPPy-Ms run 150 scans each covering a mass range of 10-650 are recorded resulting in 96,000 data points, or 288,000 values including the corresponding m/e and scan number values.

3.2.2.2. Sparse array development

The PCA procedure requires the use of the data matrix for calculation of the covariance and row matrices. The data could be stored on disk or in memory. Retrieving values from disk is several hundred times slower than accessing a value in memory so as many accesses are involved then the data need to be stored in memory for the shortest calculation times. The simplest method of storing the data would be as an array with array indices of m/e value and scan number. The Pascal extended type variables each occupy 10 bytes of memory. Thus for the typical data discussed 960,000 bytes of memory would be necessary. This value far exceeds the space available to the operating system (655,360 bytes) and is thus impossible to use. As a further problem the predefined Pascal data types can only occupy a maximum size of 65,536 bytes inhibiting the use of the Pascal array type.

Turbo Pascal offers a method of circumventing this size restriction by using the heap. The heap is all the memory available to the operating system and not used by the program or the stack. Unfortunately in order to find any variable placed in the heap it is necessary to keep a record of its memory location. Numerous techniques have been developed to allow use of the heap.

The simplest and most closely related to the array structure is to use storage mapping functions. With storage mapping functions a block of the heap memory is allocated large enough to store all the required data points. The data are then placed into this area in the form of an array. As the size of each data point is known it is simple to calculate the appropriate position for each point. This approach circumvents the Pascal data type size restriction but unfortunately still results in a matrix too large to fit into the operating system memory.

To reduce the number of data points needed to be stored a type of data structure called the sparse array may be used. In a sparse array only the non-zero data points are stored. This would appear to result in a size reduction directly proportional to the number of zeroes in the data. Unfortunately the position of each data point and the overall dimensions of the matrix must now be stored as well. Furthermore, as the data is stored on the heap a record of its memory address must also be kept.

These problems were addressed by using a data structure composed of records and pointers. Records are a feature of Turbo Pascal and are a series of variables of differing data types that are held together and referenced as if they were a single variable. Pointers are simply a variable that holds (points to) a memory address. The data structure was constructed using a pointer, held on the stack, which points to the start of the array. The array is composed of a circularly linked list of circularly linked lists. A linked list is a series of records with each record containing the address of the next and the previous record in the list; the list is circular because the last entry in the list points to the first entry and vice versa. The first linked list can be thought of as the rows of the matrix. From each record (row) of the first linked list, starts a second linked list that contains each value across the row. This structure allows an individual record to be placed anywhere in the array simply by inserting it into the relevant linked list. The records also hold the co-ordinates of the data point thus allowing only the non-zero points to be stored. A full graphical explanation is given in Figure 8. The figure also includes the index, a singly linked list, used to allow creation of more than one matrix as is required for PCA. A complete listing of unit DATSTRU2.PAS is included in appendix 3.

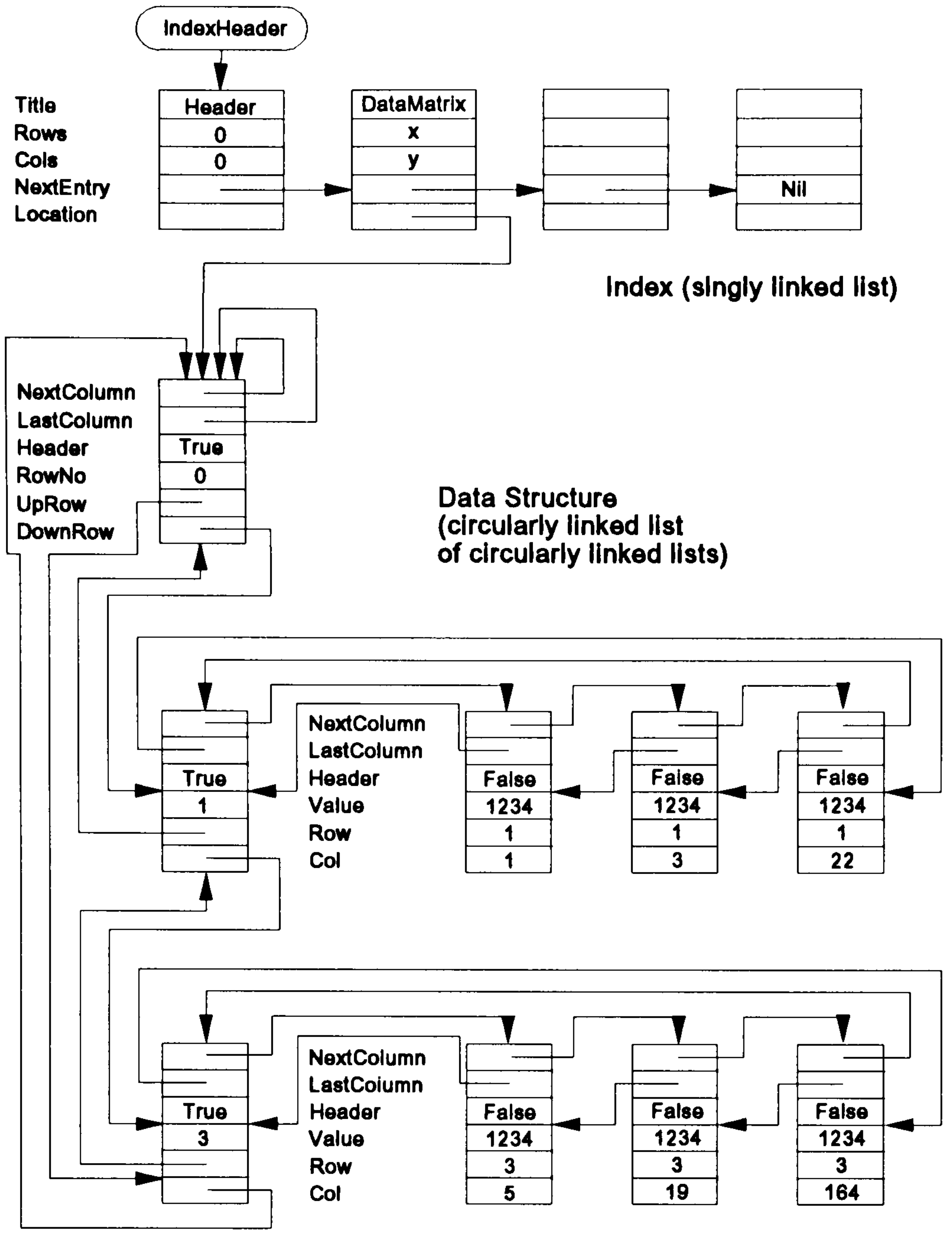


Figure 8: Diagrammatic representation of the sparse array data structure.

The sparse array data structure had the following advantages:

- Small memory requirements for sparse data,
- Dynamic structure (The size of array could be altered easily);

and the following disadvantages:

- Large memory overhead for dense arrays (15 bytes/value overhead),
- Slow access times as linked list has to be traversed each time.

The result of PCA is a number of dense arrays that have to be stored in memory as well as the original data matrix. So the use of the sparse array structure for these arrays was wasteful of memory and restricted the maximum size of data that could be analysed.

3.2.2.3. *Extended Memory Specification (EMS) data structures*

The lack of memory available to the MS-DOS operating system was placing severe constraints upon the data sets that could be analysed. To overcome this limitation a different data structure was developed. The new structure was based around the storage mapping function design but the data was stored in extended memory (EMS).

The EMS data structure was written using the OPLARRAY object in the Object Professional library. This allowed access to the EMS functions contained in the Version 3.0 EMS specification. The EMS data structure divides the data up into pages and stores each page in EMS or on disk in a virtual array. Using the objects supplied a new data structure was created. The new code was written using object oriented programming techniques to create the EMS arrays and a linked list to index them. A series of short functions were written to act as an interface between the calls from the program to the old data structure and the objects of the new data structure.

A complete listing of the EMS data structure unit, EMSDAT.PAS, is given in appendix 3.

3.2.3. Principal component analysis

3.2.3.1. Data pre-treatment

Digitization

The UV-Vis spectra of the transition metal mixtures were digitized from the original spectra recorded on chart paper. The digitization of the absorbance data was performed using a ruler. The height of the absorbance line above the baseline was measured. This was performed at regular intervals along the whole chart length and repeated in the same manner for each recorded spectrum.

The measurements were entered into a spreadsheet and converted to absorbance using a scale factor from the graph using equation 11.

$$\frac{(Abs_{\max} - Abs_{\min})}{Length_{axis}} \times Length_{abs} = Absorbance$$

where:

$Abs_{\max}$ = Maximum value on absorbance scale

$Abs_{\min}$ = Minimum value on absorbance scale

$Length_{axis}$ = Length of axis (mm)

$Length_{abs}$ = Distance between base and absorbance line (mm) (11)

File conversion

EPA files

Files collected by the mass spectrometer were stored in a format proprietary to VG analytical on the attached data system. In order to transfer them to the PC for analysis the files were converted to ASCII using the EPAF program in the HOUSEKEEPING utility on the data system. The converted files were then transferred to the PC using the Kermit[94] file transfer program.

The ASCII files are coded in VG analytical's EPA file format (originally from the Environmental Protection Agency) and software was written into the program to enable the file to be read and converted to a data matrix of intensities. The description of the file format is included in Appendix 4.

Lotus 123 files

Spreadsheets were used throughout the investigation for editing of data and to produce graphical output. A conversion program was necessary to convert between the spreadsheet file format and the programs on data format. The most popular file format for interchange of information between spreadsheets is the format used by Lotus 123 Version 2. The file format is published by Lotus[95] and the conversions were programmed according to those specifications. Only a limited conversion program was developed as only numbers are used by the program. Any records referring to formulae or text are ignored so the incoming data must be in the form of a matrix or vector starting in cell A1. The data must contain no gaps or other non-number entries. The conversion program will convert integers to reals automatically. A further feature is the ability to write matrices in Lotus format with more than 256 columns though these can only be read by spreadsheets that accommodate more columns such as Wingz[96]. It was necessary to make modifications to circumvent bugs in the Quattro file import routines. A complete listing of the conversion unit, LOTUSFIL.PAS is given in appendix 3.

Background subtraction

A common technique used to improve the spectra in mass spectrometry data is background subtraction. This is achieved by selecting a scan from the beginning of the analysis when no sample is being evolved (generally after insertion of the probe but before the temperature ramp is started) and subtracting it from all the scans in the analysis.

Scan range reduction

With many of the samples analysed most of the data is due to background signals with peaks of interest over only a few (10-30) scans. It may also only be necessary to investigate a small region as the other components are well separated. In these cases the number of scans subjected to analysis was reduced, with a consequent benefit in analysis time. The reduction was achieved using a spreadsheet to delete the unwanted columns from the data the remainder being saved.

Air peak subtraction

The background in the mass spectrometer is due mostly to residual air. The m/e peaks produced in the spectrum are at low mass and unlikely to be due to any other significant fragment from the sample. The air peaks can be deleted from the data matrix without loss of sample information. This is achieved by setting to zero the rows in the data matrix corresponding to m/e 17,18,28,32,40 and 44.

Range scaling

In some cases it was desired to have spectra of identical maximum magnitude. To achieve this the spectra were scaled according to equation 12.

$$d_{ik}(\text{scaled}) = \frac{d_{ik} - d_{\cdot k}(\text{min})}{d_{\cdot k}(\text{max}) - d_{\cdot k}(\text{min})}$$

where

$d_{\cdot k}(\text{min})$ = minimum value of d

$d_{\cdot k}(\text{max})$ = maximum value of d

d_{ik} = i^{th}data point of k^{th}row or column (12)

3.2.3.2. *Calculation of the covariance matrix*

The covariance matrix **Z** is calculated by pre-multiplying the data matrix **D** by it's transpose **D′** as follows:

$$\mathbf{Z} = \mathbf{D'D} \tag{13}$$

Only half of the covariance matrix is calculated by the program, as all off-diagonal values are mirrored across the leading diagonal to reduce calculation times. The covariance matrix is calculated in procedure `Covariance_matrix` in MATHUNIT.PAS, which is contained in appendix 3.

3.2.3.3. *Factor extraction procedure*

The covariance matrix is decomposed into its characteristic eigenvalues via the following procedure.

1. All elements of the first eigenvector, $\mathbf{c}$, are set to an initial approximation according to the equation

$$\mathbf{c}_{i1} = \frac{1}{\sqrt{c}}$$

where c = number of columns (14)

2. A better approximation to the eigenvector is calculated from

$$\mathbf{Z}\mathbf{c}_1 = \lambda_1 \mathbf{c}_1 \tag{15}$$

the resulting eigenvector is normalized by dividing each element of the eigenvector by the normalization constant given by

$$\lambda_1 = \left(\sum_i c_{i1}^2 \right)^{\frac{1}{2}} \tag{16}$$

Note that the normalization constant becomes the estimate for the eigenvalue λ_1 as shown on the right-hand side of equation 15.

3. Step 2 produces an approximation to the eigenvector and eigenvalue. Once calculated the figures are used to repeat the calculation producing better and

better approximations, until the required accuracy is obtained as determined by equation 20.

4. The contribution of the first eigenvector to the covariance matrix is then removed according to the following equation

$$\mathfrak{R}_1 = \mathbf{Z} - \lambda_1 \mathbf{c}_1 \mathbf{c}_1' \tag{17}$$

Where $\mathfrak{R}_1$ is the first residual matrix.

5. Steps 1 to 4 above are then repeated to calculate $\mathbf{c}_2$. With the appropriate substitutions the equations become

$$\mathfrak{R}_1 \mathbf{c}_2 = \lambda_2 \mathbf{c}_2 \tag{18}$$

for the approximation to the eigenvector, and

$$\mathfrak{R}_2 = \mathfrak{R}_1 - \lambda_2 \mathbf{c}_2 \mathbf{c}_2' \tag{19}$$

for the second residual matrix. This process is repeated for every eigenvector in the covariance matrix. The eigenvectors are stored in the form of a matrix, $\mathbf{C}$, and the eigenvectors are stored as one row in a matrix.

3.2.3.4. Testing for completion of extraction

The approximations to the eigenvector will get closer and closer together with each iteration. With each calculation a small amount of calculation error will be introduced into the eigenvector so that the values reached will never be identical. To determine when the iterations are no longer converging each of the elements of the eigenvector are compared with the previous values as follows

$$|\mathbf{c}_i(\text{old})| - |\mathbf{c}_i(\text{new})| < \text{threshold} \tag{20}$$

The threshold is a constant that may be altered to change the accuracy of the factor extraction (default 1×10^{-19}).

3.2.3.5. Calculation of the row matrix

Once the eigenvector matrix has been calculated the row matrix, $\mathbf{R}$, is found using the following equation

$$\mathbf{R} = \mathbf{DC}' \tag{21}$$

This is the last step in the PCA procedure. The listings of the procedures necessary to accomplish the previous steps are contained in the unit MATHUNIT.PAS in appendix 3

3.2.4. Determining the dimensions of the data space

3.2.4.1. Variance

The PCA procedure calculates the eigenvectors in the data, in order of decreasing importance to the data. The eigenvalue associated with the eigenvector gives a measure of the importance of the eigenvector to the data. The sum of all the eigenvectors is the total variance in the data set. Knowing the total variance and the individual variances the percentage variances accounted for by each factor is readily calculated from

$$\%\text{variance} = \frac{\lambda_j}{\sum_{j=1}^{c} \lambda_j} \times 100 \tag{22}$$

and the cumulative percent variance found for the n^{th} factor using

$$\text{cumulative \% variance} = \frac{\sum_{j=1}^{n} \lambda_j}{\sum_{j=1}^{c} \lambda_j} \times 100 \tag{23}$$

3.2.4.2. Error estimates

In order to assist in the determination of the number of factors necessary to describe the factor space the error terms developed by Malinowski[10] are calculated (see the introduction for an explanation of the terms).

The Real Error, RE, is calculated from

$$\text{RE} = \left[\frac{\sum_{j=n+1}^{c} \lambda_j^{\circ}}{r(c-n)} \right]^{\frac{1}{2}} \tag{24}$$

and the related Imbedded Error, IE found using

$$\text{IE} = \text{RE}\sqrt{\frac{n}{c}} \tag{25}$$

also calculated is the empirical Factor Indicator Function, IND using the equation

$$\text{IND} = \frac{\text{RE}}{(c-n)^2} \tag{26}$$

3.2.4.3. Misfits

This term gives a measure of how well the factor analysis regenerated data fits the observed data. The fit is determined by counting the number of misfits as a function of the number of factors used to regenerate the data. A regenerated point is regarded as a misfit if its value differs from the observed by more than three times the standard deviation of the experimental data.

The standard deviation of the data is calculated using the standard formula

$$s=\sqrt{\frac{\sum_{i=1}^{r}\sum_{j=1}^{c}\left(d_{ij}-\bar{d}\right)^{2}}{(rc-1)}} \qquad (27)$$

and the data regenerated according to the equation below

$$\mathbf{D}_{n}=\begin{bmatrix} r_{1} & r_{2} & \cdots & r_{n}\end{bmatrix}\begin{bmatrix} c_{1}' \\ c_{2}' \\ \vdots \\ c_{n}'\end{bmatrix} \qquad (28)$$

where $\mathbf{D}_n$ is the reproduced data matrix using n factors.

3.2.4.4. Standard error in eigenvalue

This term, developed by Hugus and El-Awady[9], is based upon a statistical criterion for the "vanishing" of an eigenvalue. The calculation is performed using the following equation

$$\sigma_{m}=\left[\sum_{j=1}^{c}\sum_{k=1}^{c}c_{mj}^{2}c_{mk}^{2}\sigma(Z)_{jk}^{2}\right]^{\frac{1}{2}}$$

where

σ_m = standard error in m^{th}eigenvalue

c_{mj} = j^{th}component of the m^{th}eigenvector

c_{mk} = k^{th}component of the m^{th}eigenvector

and

$$\sigma(Z)_{jk}^{2}=\begin{cases}\sum_{i=1}^{r}(d_{ij}^{2}\sigma_{ik}^{2}+d_{ik}^{2}\sigma_{ij}^{2}) & \text{for}\, j\neq k\\ \sum_{i=1}^{r}4d_{ij}^{2}\sigma_{ij}^{2} & \text{for}\, j=k\end{cases}$$

where

σ_{ij} = error in d_{ij} (29)

This equation allows an individual error term to be entered for every data point. In this particular implementation a single error term was used for every data point in the experimental data.

3.2.4.5. *Significance level*

The percentage significance levels for the eigenvalues are calculated from the Fisher variance ratios as described by Malinowski[12]. The original paper contains some typographical errors in the equations given so the expressions have been corrected. The variance ratio is calculated from

$$F(1,s-n)=\frac{\sum_{j=n+1}^{s}(r-j+1)(c-j+1)}{(r-n+1)(c-n+1)}\frac{\lambda_n}{\sum_{j=n+1}^{s}\lambda_j^0}$$

where

$s = r$ or c whichever is smaller (30)

The percentage significance level is calculated from an algorithm presented by Cooke, Craven and Clarke[97] which uses the following integral

$$I_x(a,b)=\frac{\Gamma(a)\Gamma(b)}{\Gamma(a+b)}\int_0^x t^{a-1}(1-t)^{b-1}dt$$

The integral $I_x(a,b)$ equals the value of the F distribution function, with k_1 and k_2 degrees of freedom and with argument F when $a=k_2/2$, $b=k_1/2$ and $x=k_2/(k_2+k_1F)$. (31)

3.2.5. Optimisation of PCA

3.2.5.1. *Accuracy of factor extraction*

In the eigenanalysis process each factor is found by convergent series. A starting approximation to the eigenvector is calculated from equation 14. A better approximation to the eigenvector is then calculated using equation 15

The left hand side of the equation is calculated and the eigenvalue, λ, calculated from equation 16. The sum of the values is used to normalize the eigenvector to unit length.

The newly calculated eigenvector must be compared with the old one to determine if it is sufficiently similar to be regarded as identical and therefore fully extracted. Comparison is achieved by subtracting each of the corresponding values of the new and previous approximations and comparing these values with a threshold figure, according to equation 20.

In order to investigate the effect of the magnitude of the threshold on the eigenanalysis a series of decompositions were performed varying the value the factor extraction test threshold over the range 1×10^{-5} to 1×10^{-22}. For each decomposition the number of iterations necessary for each factor to be extracted was noted and at the end of decomposition (determined by the number of columns in the data, not by the residual test procedure discussed later) the residual matrix was saved for later processing. The residual matrix contains the variance left after the factors have been removed and theoretically is zero upon completion of decomposition.

The data was interpreted by adding the number of iterations and calculating the RMS average of the residual matrix. The RMS average was used, as the residual matrix contains both positive and negative numbers and it was desired to have an indication of the magnitude of the numbers in the matrix independent of their sign.

Three data sets were tested in this way; the transition metal ion data, UV-Vis absorbance data from Cartwright[88] and a mass spectroscopy data set for FCA (TB11). The mass spectrometric data set was modified to produce a smaller number of columns thereby reducing the calculation times. The modifications were deletion of columns 1-37 (no sample evolved in this region), and deletion of columns 39,41,43,45,47,49,51,53 and 55 (every other column to reduce the number of scans to 10).

3.2.5.2. Determining completion of extraction

The diagonalization process (PCA) fully describes the data set after the extraction of a number of factors from the covariance matrix equal to the number of rows or columns of the original data, whichever is the smaller. However, because of redundancy in the data the number of factors necessary to describe the data completely is usually less than the smaller dimension of the data. In order to determine when the data has been adequately modelled the residual matrix is tested by comparison of the values in the matrix with a test value. If all of the values in the residual matrix are smaller than the test value, then the data has been adequately described and the decomposition process halted.

In order to study the size of the residuals after each factor has been extracted the program was modified to allow the residual matrix to be saved to disk after each factor extraction. Data was obtained from this procedure for the transition metal ion absorbance data and the modified mass spectroscopic data set, TB11.

If the magnitude of the test value is to be modified for each type of data then a method of determining the test value must be found. The theoretical zero for any set of data must be the limit of its accuracy. For the absorbance data the accuracy is simply determined by the manufacturer's specification for the instrument. For the mass spectrometer no such value is available and one must be estimated.

For the transition metal ion data the spectrometer used to collect the spectra has a quoted accuracy of 0.004 absorbance units. This will translate to a value of 1.6×10^{-5} in the covariance matrix (equation 13).

3.2.5.3. *Estimation of error in the mass spectrometer data*

For the mass spectrometric data an estimate of the error in the procedure was determined using a blank pyrolysis run (TB20). In the blank run the only peaks expected are those due to the residual air present in the instrument at all times and possibly a small amount of hydrocarbon from the pump oil.

The determination of the error in the spectrometer is complicated by the use of the EPA file format as an intermediary between the mass spectrometer and the PC. The EPA file format is constructed such that for every scan the peak of maximum intensity is used as a normalization constant and the magnitude of the rest of the spectrum coded as an integer between 0 and 999 where 999 is the intensity of the most intense peak. This introduces an absolute error of 1/1000 of the highest intensity of each scan for each peak in the spectrum except for the peak of highest intensity. Consideration of the data showed that the largest signal in the background was from nitrogen in the air at m/e 28, as the intensity of this peak is measured at the full accuracy of the data system then its error will be a true measure of the data systems accuracy unaffected by the EPA file format encoding process. The RMS error for the N_2 peak was calculated and converted to a relative error that was then applied to the TIC of the sample data and the number so produced used as the absolute error in the data.

3.2.6. Validation of results from PCA

3.2.6.1. *Synthetic data sets*

The results from the program were compared with those published by Malinowski & Howery[87] for both the pure and raw data sets. The error terms calculated for the raw data set were compared with the figures given.

3.2.6.2. *Published data and results*

Most published work does not include the abstract matrices so comparisons were limited to eigenvalues and error terms. The works of Weiner et al.[89] and Ritter et al.[50] were used to compare values produced from the program and their own calculations.

3.2.7. Target testing

The abstract factors produced from PCA are transformed into the 'closest fitting' vector to the input test vector by a transformation vector $\mathbf{t}_l$. The vector is calculated by a least squares method which is described by the equation

$$\mathbf{t}_l = \overline{\boldsymbol{\Lambda}}^{-1}\overline{\mathbf{R}}'\mathbf{x}_l \tag{32}$$

where the bar denotes a matrix composed of primary eigenvectors only, $\boldsymbol{\Lambda}$ is a diagonal matrix of eigenvalues and $\mathbf{x}_l$ is the test vector.

3.2.7.1. Transformation constant calculation

It can be seen that part of equation 32 will remain constant for any test vector if the number of primary eigenvectors is unchanged. To avoid recalculation of this term for each test it is calculated once at the beginning of any target testing from the following

$$\mathbf{T}_k = \overline{\boldsymbol{\Lambda}}^{-1}\overline{\mathbf{R}}' \tag{33}$$

thus simplifying the calculation of the transformation vector to

$$\mathbf{t}_l = \mathbf{T}_k\mathbf{x}_l \tag{34}$$

The calculation of the constant is performed in procedure `Calculate_TKON` in unit TARGTEST.PAS, which is contained in appendix 3.

3.2.7.2. Predicted vector determination

The vector predicted by the target testing process is calculated from

$$\hat{\mathbf{x}}_l = \overline{\mathbf{R}}\mathbf{t}_l \tag{35}$$

where $\hat{\mathbf{x}}_l$ is the predicted vector. This calculation is performed by procedure `Calculate_pred_vector` in unit TARGTEST.PAS, contained in appendix 3.

3.2.8. Criteria for the fit of a test vector

3.2.8.1. Error estimates

The program calculates several error terms developed by Malinowski[52] to aid quantifying the validity of a test vector.

Apparent error in the test vector (AET)

This term which expresses the difference between the predicted vector and the test vector is calculated as follows

$$\mathrm{AET} = \left(\frac{\sum_{i=1}^{r} (\hat{x}_i - x_i)^2}{r} \right)^{\frac{1}{2}} \tag{36}$$

where the summation is the difference between the elements of the test and predicted vectors.

Real error in the predicted vector (REP)

This term is calculated from the following equation

$$\mathrm{REP} = \mathrm{RE}\left\|\mathbf{t}_l\right\| \tag{37}$$

where the vector norm $\left\|\mathbf{t}_l\right\|$ is calculated from

$$\left\|\mathbf{t}_l\right\| = \sqrt{\mathbf{t}_l \times \mathbf{t}_l} \tag{38}$$

Real error in the test vector (RET)

The RET is calculated using the Pythagorean relationship between the terms AET, REP and RET as follows

$$\mathrm{RET} = \sqrt{(\mathrm{AET})^2 \times (\mathrm{REP})^2} \tag{39}$$

Reliability function (RELI)

This comparison of the real and calculated errors in the test vector is found from

$$\mathrm{RELI} = \left(\frac{1 - (\mathrm{RET})^2 - (\mathrm{RET})^2_{\mathrm{est.}}}{(\mathrm{AET})^2} \right)^{\frac{1}{2}} \tag{40}$$

where $(\mathrm{RET})_{\mathrm{est.}}$ is the value of the real error in the test vector estimated from a knowledge of the experimental error in the data used to construct the test vector.

Spoil function (SPOIL)

The SPOIL function is calculated from the ratio

$$\mathrm{SPOIL} = \frac{\mathrm{RET}}{\mathrm{REP}} \tag{41}$$

All of the error terms above are calculated in procedure `Errors_in_test_vector`, contained in unit TARGTEST.PAS and listed in appendix 3.

Significance testing

Percentage significance levels are calculated from the variance ratios described by Malinowski[12] after removal of the typographical errors. Variance ratios are found according to the following expression

$$F(r-n-b, s-n) = \frac{\sum_{j=n+1}^{s} (r-j+1)(c-j+1)}{(r-n+1)(c-n+1)} \frac{r(s-n)\left(\overline{\mathbf{R}} - \overline{\overline{\mathbf{R}}}\right)^2}{(r-n-b)\sum_{j=n+1}^{s} \lambda_j^0 \sum_{j=1}^{n} \mathbf{t}_j^2} \tag{42}$$

and the percentage significance level calculated from the incomplete beta function given in equation 31.

3.2.8.2. Visual inspection

Test and predicted vectors were also compared graphically by plotting the vectors. Two major types of plot were used.

Scatter plots with both test and predicted vectors on the same chart. These were used when a small number of points needed to be displayed. The test and predicted vectors were overlaid so that deviations could clearly be seen.

Column charts were used to compare vectors with large numbers of points such as the mass spectra. The vectors were plotted on separate charts with identical scales and arranged vertically above each other thus allowing a visual comparison. If a more careful study of a particular region was needed then an appropriate scale expansion would be used.

3.2.9. Validation of results from target testing

3.2.9.1. Synthetic data sets

The data sets of Malinowski & Howery[87] were used to compare the results from the target tests and also for the calculation and comparison of error terms listed above.

3.2.9.2. Published data and results

Test vectors were used from Lorber[13] in conjunction with data from Ritter et al.[50] to compare target test results and error terms including F-tests.

3.2.10. Iterative target testing

3.2.10.1. Vector modification

The result of a target test is a vector that has been oriented in the co-ordinate system described by the factors (i.e. the data space) to most closely approximate the test vector. If a poor test vector is tested then the result will be oriented to give the closest fit possible to the test. If the test vector is modified and then resubmitted for testing, then the orientation within the data space will change to best fit the new vector. By modifying the test vector according to chemical knowledge it is possible to iterate the test vector towards a chemically meaningful solution.

If chemically meaningful rules for perturbing the vector cannot be found or are insufficient to induce convergence to a true factor then an abstract perturbation may be used. The abstract perturbation is added in an attempt to unsettle the vector sufficiently to allow the chemical selection rule to move the factor in the direction of the true factor.

Note: In the following methods all rejected peaks are set to zero.

Chemical selection rules

The program performs target testing on the columns of the data matrix which for the mass spectrometer data are the mass spectra. As the column is a mass spectrum then it may not contain any negative values as these would be impossible to obtain. Any spectra produced as a result of a target test that contains negative values must therefore be regarded as improperly aligned in the data space for representing a true component spectrum.

Rejection of the negative values in the spectrum and re-submission of the test vector will allow a closer approximation to the true factor being produced as a result of the target testing procedure.

Abstract selection rules

Tests showed that the chemical selection rule did not produce a rapid enough convergence so a selection of abstract methods were developed and tested. Details of the methods and their development are contained in the results section.

3.2.10.2. Control of iterative process

The control of the iterations in the program was designed to pause the calculation after each iteration and to update the display enabling the vector to be observed. Pressing Escape stops the iterative process and any other key continues to the next iteration. The iterative process was designed to be re-entrant so it may be left and restarted at any point, allowing intermediate results to be saved. The iterations will automatically cease if the positive difference between test and predicted vectors is less than 1×10^{16}.

4. Results & discussion

4.1. Program description

4.1.1. Overview

The user interface for the program was designed to appear similar to a spreadsheet. The screen is divided into a series of regions with different functions. The top line of the display contains the address of the cell currently pointed to by the cursor and its value expressed in scientific notation to the full number of stored digits. At the right hand end of the top line is the status flag; which gives the current status of the program.

Possible status conditions are listed in table 9 below.

Status flag	Indication
Ready	Program awaiting instructions
ERROR	Awaiting clearance of an error condition
Wait	Program busy
Menu	Awaiting selection from the menu
Edit	Editing a value in a matrix
Point	Awaiting selection from displayed choices

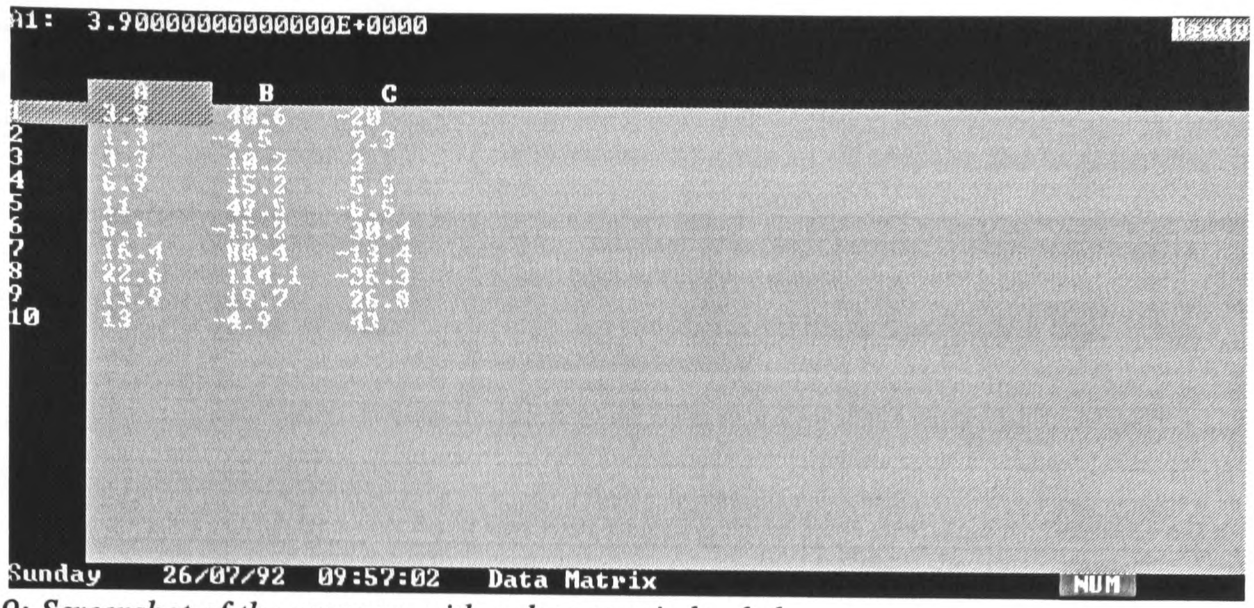
Table 9: Explanation of status flag indications.

The second line of the screen is the menu/command line. When the menu is active the possible selections appear on this line. All typed input to the program is entered on this line, e.g. editing values, filenames, etc. The third screen line, the message line, contains a description of the item on the menu/command line or a prompt for input.

The fourth line of the screen contains the column headings for the matrix; these may be in standard spreadsheet letter format or in number format. The column that contains the cursor is indicated by the column heading being highlighted.

Lines 5 to 24 contain the data for the matrix currently being displayed arranged in columns. The first column contains the row numbers of the data on display with the row containing the cursor being highlighted as for the column headings. The data values are displayed left aligned and as plain numbers unless there are more digits than may be displayed in the current column width in which case the number is rounded and displayed in scientific notation.

Line 25, at the bottom of the screen, contains the current date and time on the left. The centre of the line contains the name of the matrix currently being displayed and the right hand side contains flags to show the current positions of the CAPS LOCK, NUM LOCK and SCROLL LOCK keys.



```
A1:  3.900000000000000E+0000                                     Ready

       A        B        C
1     3.9      40.6     -20
2     1.3      -4.5      7.3
3     3.3      10.2      3
4     6.9      15.2      5.5
5     11       49.5     -6.5
6     6.1     -15.2      30.4
7     16.4     80.4     -13.4
8     22.6     114.1    -26.3
9     13.9     19.7      26.0
10    13       -4.9      43

Sunday     26/07/92   09:57:02    Data Matrix                   NUM
```

Figure 9: Screenshot of the program with a data matrix loaded.

4.1.2. Keyboard reference

All control input to the program is via the keyboard and control is almost exclusively achieved via the menu system. There are a limited number of shortcut keys and these are listed later.

Cursor movement mostly conforms to the keystrokes found in spreadsheet packages but with some extensions to make moving around in large matrices easier. Cursor movement commands are listed in table 10.

Key(s)	Cursor movement
→	Moves one column right
↓	Moves one row down
←	Moves one column left
↑	Moves one row up
PgUp	Moves one screen page up
PgDn	Moves one screen page down
Tab *or* Ctrl+→	Moves one screen page right
Shift+Tab *or* Ctrl+←	Moves one screen page left
Home	Moves to cell A1 (Row 1, Column 1)
End	Moves to last cell in matrix
Ctrl+Home	Moves to the top row in the same column
Ctrl+End	Moves to the bottom row in the same column
Ctrl+PgUp	Moves to the leftmost column in the same row
Ctrl+PgDn	Moves to the rightmost column in the same row

Table 10: Cursor control keys.

The shortcut keys perform operations available on the menu system but in a single key press. The following table lists the function keys that have been set up to perform shortcuts.

Function key	Name	Shortcut
F2	Edit	Edits the number pointed to by the cursor
F5	Go To	Allows user to enter a cell address and then moves the cursor to that address
F6	Col Headings	Swaps column headings between numbers and the spreadsheet letter system
F7	Col Width	Asks for a column width and reformats data to the new column width
F8	Change Matrix	Allows the matrix being displayed to be changed

Table 11: Function key shortcuts.

At various points in the use of the program input from the user is necessary. Two possible conditions arise, where there is existing data, and where there is none. If data already exists the current value is offered to the user for modification; the existing value is displayed on the menu/command line, after a prompt, with no cursor visible. If a new value is to be entered then it may be typed in. Immediately a character key is pressed the existing value is deleted from the line and replaced with the new character. If it is desired to edit the existing value then an editing key must first be pressed. This can be the left or right cursor key, Home, End, Delete or Insert. Pressing any of these keys' results in the cursor becoming visible at the end of the current value (unless the key pressed moves the cursor, e.g. Home) ready for editing. The default editing mode is overtype, which will replace the character at the cursor. Pressing Insert at any time or to enter editing mode will result in the cursor changing from an underline (indicating overtype mode) to a flashing block (indicating insert mode) which will allow characters to be inserted at the cursor.

When the modification or new entry is complete it may be entered by pressing Return; pressing Return without entering anything leaves the original value unchanged. Pressing Escape at any time will revert to the step before data entry and leave the value unchanged.

Key	Function
←	Moves the cursor one character to the left
→	Moves the cursor one character to the right
Home	Moves the cursor to the leftmost character
End	Moves the cursor to the rightmost character
Delete	Deletes the character at the cursor
Backspace	Deletes the character to the left of the cursor
Insert	Switches between insert and overtype mode

Table 12: Summary of keys used for editing.

The program stores its data in a series of matrices with predefined names. The display shows only one matrix at a time with the name of the matrix displayed in the centre of the bottom line. The possible matrices and the data they contain are listed in table 13. The displayed matrix may be changed using the menu system or the F8 shortcut key. Changing the displayed matrix results in the screen showing a list of all the matrix names with the current matrix being highlighted. The new matrix to be displayed can be selected by moving the highlight with the cursor keys to the desired name and then pressing Return.

Matrix name	Description
No Matrix to display	Null entry for blank screen
Data Matrix	Original data values
Covariance Matrix	Values of the covariance matrix
Residual Matrix	Covariance matrix with factors removed
Abstract Column Matrix	Eigenvector matrix
Abstract Row Matrix	Contains the scores on the eigenvectors
Composite Matrix	Contains eigenvalues and error terms
Intermediate Matrix	Used for intermediate calculations
Transform constant	Contains the constant for target testing
Target Test Matrix	Contains the test & predicted vector & errors
Vector Differential Matrix	Used for intermediate results in ITT

Table 13: Table of matrix names.

4.1.3. Menu system

The menu is activated by pressing the / key on the keyboard. The menu is displayed on the menu/command line as a horizontal list of choices separated by spaces, with one of the options highlighted to indicate which is selected. The options in the menu may be selected by use of the left and right cursor keys to move the highlight and pressing Return to proceed. The message line carries a description of the currently highlighted command or a listing of the options in the submenu selected. Menu options may also be chosen by pressing the first capitalized letter of the option. This is equivalent to selection with the cursor keys and pressing Return, and provides the fastest method of navigating the menu system.

Some of the menu options perform operations; others lead to a submenu. The submenu is operated in the same way as the main menu and may contain entries leading to further submenus. To return to the previous menu or to leave the menu system the Escape key may be pressed. In general the Escape key may be used to go back a step or to stop and abandon what ever operation is currently in progress.

Attempting to move the cursor beyond the beginning or end of the line or pressing a letter key for an option not present in the current menu will result in the program responding with a beep and the cursor remaining unmoved.

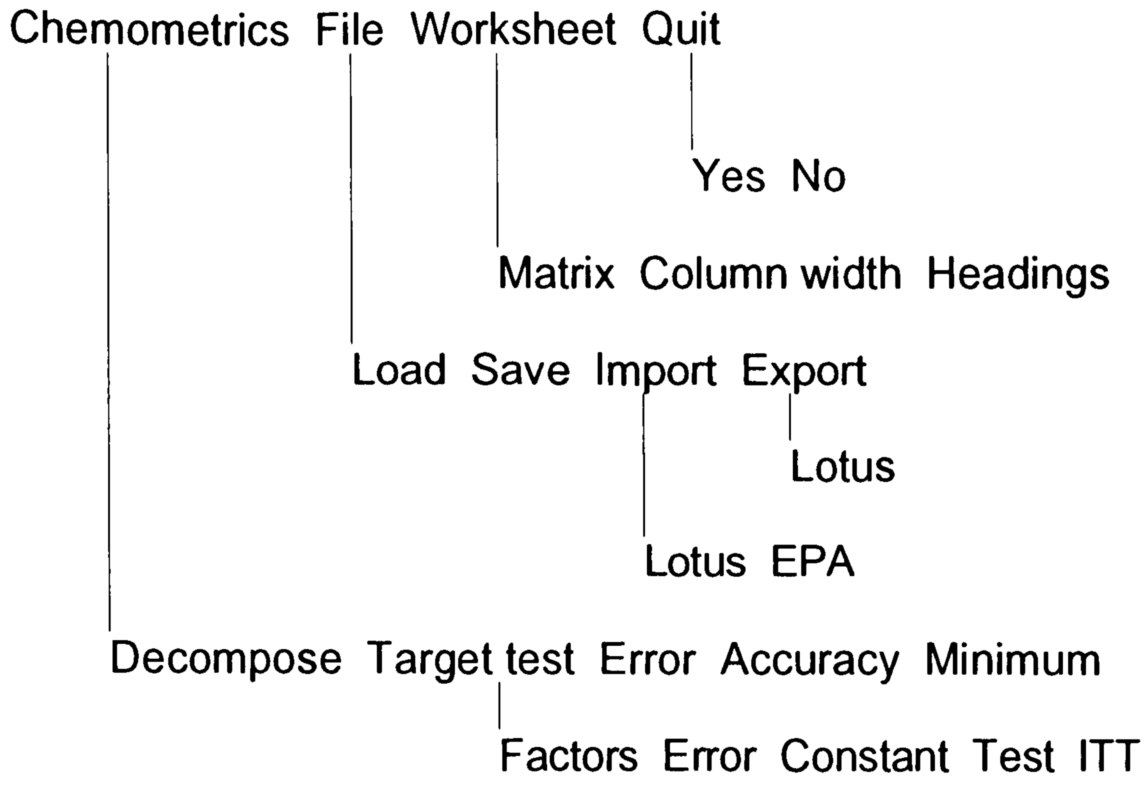

Chemometrics File Worksheet Quit
Yes No
Matrix Column width Headings
Load Save Import Export
Lotus
Lotus EPA
Decompose Target test Error Accuracy Minimum
Factors Error Constant Test ITT

Figure 10: Diagram of the menu hierarchy.

The menu hierarchy is shown in figure 10. The main menu, which appears when / is pressed, contains four options and is shown as the top line in the figure. Selection of any of the options leads to a submenu where further choices must be made. The menu entries are described below.

Main menu:

Quit: This option leads to a simple Yes/No submenu. Selection of Yes leaves the program (If the data has been edited and not saved a further prompt indicating the fact will be seen). Selection of No returns to the Ready state.

Worksheet: This option leads to the Worksheet submenu.

File: This option leads to the File submenu.

Chemometrics: This option leads to the Chemometrics submenu.

Worksheet submenu:

Matrix: Choosing this option makes the matrix selection screen appear; this contains an entry for each of the matrices it is possible to store in the program. The matrix to be displayed is selected using the cursor keys to place the highlight over the appropriate name and pressing Return. The shortcut key for this option is F8.

Column width: This option allows the selection of a new column width for the display. The current column width is offered and a new one prompted for. Typing the new width and pressing Return results in the display being reformatted to the new column width. The shortcut key for this option is F7.

Headings: Switches the display between spreadsheet style letters for the column headings and column numbers. The shortcut key for this option is F6.

File submenu:

Load: This option allows files stored in the program's internal (native) data storage format to be loaded. The filename for the file is asked for and then the name of the matrix it is to be loaded into is selected. Any existing data is overwritten and the newly loaded matrix is displayed.

Save: Allows a matrix to be stored to disk in the program's native format. The filename for the matrix and the matrix to be saved are asked for before the file is saved; if an existing file is specified then confirmation of the overwrite is requested.

Import: Selection of this option leads to the Import submenu.

Export: Selection of this option leads to the Export submenu.

Import submenu:

Lotus: This option allows files written in Lotus WK1 format to be translated and loaded. The filename and matrix to be loaded into are asked for and the file converted and loaded.

EPA: This option enables files to be translated from the mass spectrometer EPA format. The filename is prompted for and the matrix to be loaded selected before converting and loading the file.

Export submenu:

Lotus: This option allows files to be translated and written in Lotus WK1 format. The filename and matrix to be saved are asked for and the file converted and saved.

Chemometrics submenu:

Decompose: This option initiates the PCA calculation sequence. The data matrix is subjected to PCA, error terms calculated for the eigenvalues and the abstract row matrix calculated. During the course of the factor extraction the screen shows the current factor, the number of iterations, the percentage and cumulative percentage variance and the calculated significance level. If the Escape key is pressed during factor extraction then on completion of the current extraction the PCA is aborted and the error terms and abstract row matrix calculated using only the factors already found.

Target test: This option leads to the Target testing submenu.

Error: Allows an estimate for the error in the data matrix to be entered. This has the effect of enabling the calculation of the standard error in eigenvalue. If the value is set to zero then the calculation is not performed.

Accuracy: This option allows the accuracy of factor extraction to be set. The default value is 1×10^{-19} and it is normally unnecessary to change it.

Minimum: Setting this value allows the program to determine when the residual matrix no longer contains any useful information. Selection of an appropriate value is performed by entering the smallest possible meaningful value, e.g. for the mass spectroscopic data, which is all integer, any number less than 1 is meaningless and this represents the minimum value. The default value is 1×10^{-19}.

Target test submenu:

Factors: This option must be set to the number of factors that are believed to model the data accurately.

Error: This option enables an estimate for the error in the test vector to be entered. This estimate is used in the calculation of the reliability error term for the predicted vector.

Constant: Part of the calculation involved in target testing is a constant for a particular combination of abstract row matrix, associated eigenvalues, and number of factors. Selection of this option calculates the constant part of the calculation and must be performed before testing and again if any of the previous list is changed.

Test: Target test the current test vector and calculate associated error terms.

ITT: Starts iterative target testing; after each iteration the error terms are calculated, the test vector modified in preparation for the next iteration and the display updated to show the new values. The program then pauses and waits for a key press. Pressing Escape leaves the testing loop or any other key moves on to the next iteration.

4.1.4. PCA

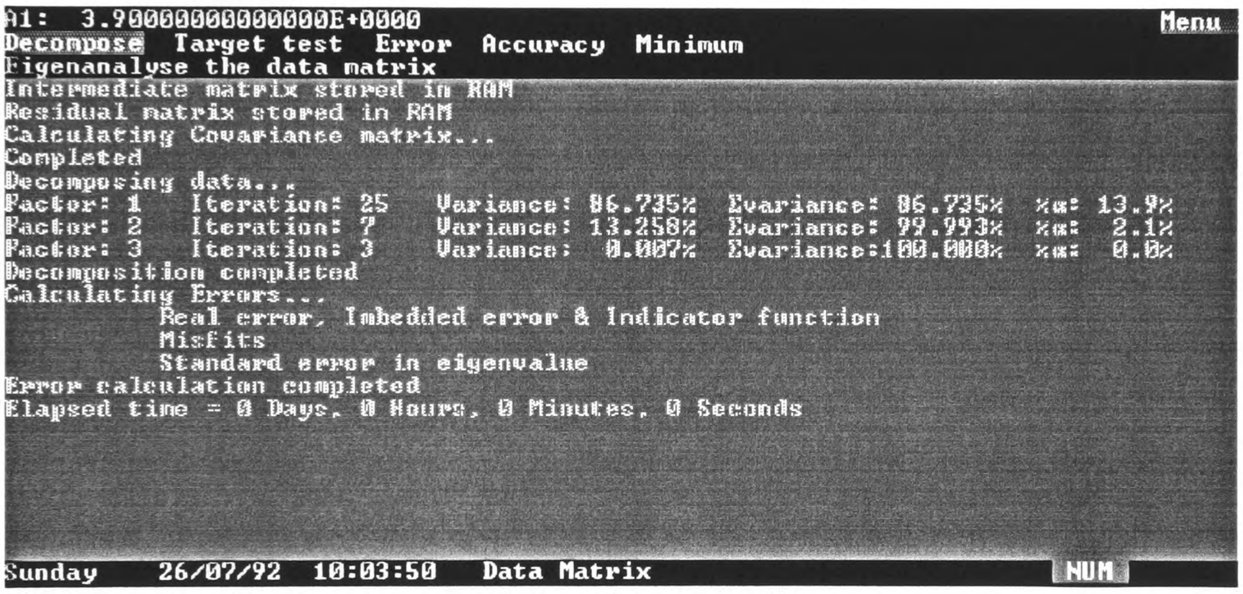
The PCA analysis is initiated through the Chemometrics submenu (Decompose). The only prerequisite for PCA is a data matrix loaded into the matrix of the same name. Optional entries may be made for the error in the data matrix (Error), the accuracy of factor extraction (Accuracy) and the smallest meaningful value for the data (Minimum). Initiating PCA results in the matrices Covariance Matrix, Residual Matrix, Abstract Column Matrix, Abstract Row Matrix, Intermediate Matrix and Composite Matrix being deleted and dimensioned to appropriate sizes resulting in the loss of any data not saved.

Once initiated the decomposition routine covers the display of the matrix to show a series of messages. Firstly the program indicates where the Intermediate and Residual matrices are being stored. This gives an indication of the speed of the decomposition, if the matrices are stored in RAM then the fastest calculation speeds will result, storage in EMS will slow the program and storage on disk will result in the slowest calculation speed. The speed is affected because of the overhead in retrieving the data values.

The program then indicates that the covariance matrix is being calculated and also the completion of the calculations. The factors are then extracted. For each factor its number is displayed and the number of iterations performed updated until extraction is complete. The variance and cumulative variance for the factor are then displayed and the significance level for the factor shown. After all the factors have been extracted a message is given to show the completion of decomposition and that the calculation of errors has begun.

Messages are produced for the Real error, Imbedded error and Indicator functions, Misfits and the standard error in eigenvalue. The error calculations are then shown to be completed and the elapsed time for the calculations displayed.

The decomposition display remains on the screen until the menu system is exited by pressing Escape twice to go up through the two levels of menu when the matrix is redisplayed.



```
A1:  3.90000000000000E+0000                                           Menu
Decompose  Target test  Error  Accuracy  Minimum
Eigenanalyse the data matrix
Intermediate matrix stored in RAM
Residual matrix stored in RAM
Calculating Covariance matrix...
Completed
Decomposing data...
Factor: 1   Iteration: 25   Variance: 86.735%  Σvariance: 86.735%  %α: 13.9%
Factor: 2   Iteration: 7    Variance: 13.258%  Σvariance: 99.993%  %α:  2.1%
Factor: 3   Iteration: 3    Variance:  0.007%  Σvariance:100.000%  %α:  0.0%
Decomposition completed
Calculating Errors...
          Real error, Imbedded error & Indicator function
          Misfits
          Standard error in eigenvalue
Error calculation completed
Elapsed time = 0 Days, 0 Hours, 0 Minutes, 0 Seconds

Sunday    26/07/92  10:03:50   Data Matrix                          NUM
```

Figure 11: Screenshot of program on completion of decomposition.

The results of PCA may be seen by viewing the appropriate matrix. The Abstract Column Matrix contains the eigenvectors for the factors arranged in rows. The abstract Row Matrix contains the associated scores on the row variables. The Composite Matrix contains the eigenvalues and their associated error terms arranged in columns with the column number corresponding to the factor number. The order of terms is given in table 14.

Row number	Values
1	Eigenvalues
2	% variance
3	Cumulative % variance
4	Real error
5	Imbedded error
6	Indicator function
7	Misfit
8	Standard error in eigenvalue
9	Significance level

Table 14: Key to rows in the composite matrix.

4.1.5. Target testing

Target testing is controlled through the Target test menu. The prerequisites for target testing are an Abstract Row Matrix with its associated Composite Matrix (produced by the PCA procedure or may be loaded from disk as required) and a Target Test Matrix. The test vector is entered as the first column of the Target Test Matrix and the length of the vector must equal the length of the Abstract Row Matrix. There must also be included two further columns for the predicted vector and the error terms. Supplementary to these requirements the number of factors to be used for modelling the data must also be entered and the Transform Constant calculated.

The number of factors to be used in modelling the data is entered using the first (Factors) option in the menu. The second option allows the entry of an estimate of the error in the test vector which if entered is used in the calculation of the reliability estimate for the test. Constant, the third entry in the menu, calculates the constant part of the calculations involved in a target test.

The fourth entry on the menu initiates the target testing procedure. During the test the predicted vector is calculated and the error terms determined, the predicted vector is placed in the second column of the Target Test Matrix and the errors in the third column; the key to the error terms is given in table 15, some of the terms listed are only calculated during iterative target testing (ITT).

Row number	Value
1	Estimated error in test vector
2	Apparent error in test vector (AET)
3	Real error in predicted vector (REP)
4	Real error in the test vector (RET)
5	Spoil function (SPOIL)
6	Reliability estimate (RELI)
7	Significance level for test vector (%SL)
8	Iteration number (ITT only)
9	Difference term (ITT only)
10	Correlation coefficient (ITT only)

Table 15: Key to error terms in the third column of the Target Test Matrix.

Iterative target testing is performed using the ITT command on the menu. The ITT process calculates the predicted vector and associated errors as for a normal target test. A series of extra procedures is also performed as follows. The first iteration, as indicated by the absence of an iteration number in cell C8 causes the Vector Differential Matrix to be dimensioned and the original test vector to be copied to its first column. For each iteration the iteration number is incremented by one, the difference term and correlation coefficient determined and the error terms calculated using the original test vector. Depending upon the type of ITT being performed, the first and second differentials of the predicted vector are also calculated. Finally, the test vector is modified according to the criteria in use and the screen updated. The program then pauses until a key is pressed to go on to the next iteration or Escape is pressed to break out of the loop.

4.2. Optimisation of PCA

4.2.1. Accuracy of factor extraction

The results for the transition metal ion data set are shown in figure 12. It can clearly be seen that the total number of iterations necessary to complete decomposition increases linearly with decrease in factor extraction test threshold.

The magnitude of the residuals can be seen to fall sharply after test value 1×10^{-8} to a value of 1×10^{-18} at test value 1×10^{-12}. The behaviour of the RMS average after this point then varies slightly and unpredictably until it becomes constant at a test value of 1×10^{-19}.

The quoted accuracy of calculations for the extended type variables used in the program is 19 or 20 decimal places. This makes the theoretical minimum value obtainable around 1×10^{-19}, i.e. the point at which the RMS average becomes constant. The constant values for the residual matrix are taken to indicate that the limit of accuracy of the calculation method has been achieved. The fact that this value is virtually achieved with a much lower extraction test suggests that this lower value might be used to achieve a saving in total number of iterations and therefore a time saving in decomposition. Reducing the extraction test below 1×10^{-12} results in erratic behaviour of the RMS average which at one point passes close to 1×10^{-20}, lower than the value at which the residuals become constant; this would at first sight appear to be impossible but a consideration of the range of values covered in the residual matrices and the position of their arithmetic means as shown in figure 13 indicates that the residuals for test value 1×10^{-15} have an arithmetic mean closer to zero than any of the other residuals and the range of values is reasonably evenly distributed about the mean resulting in a very small RMS average.

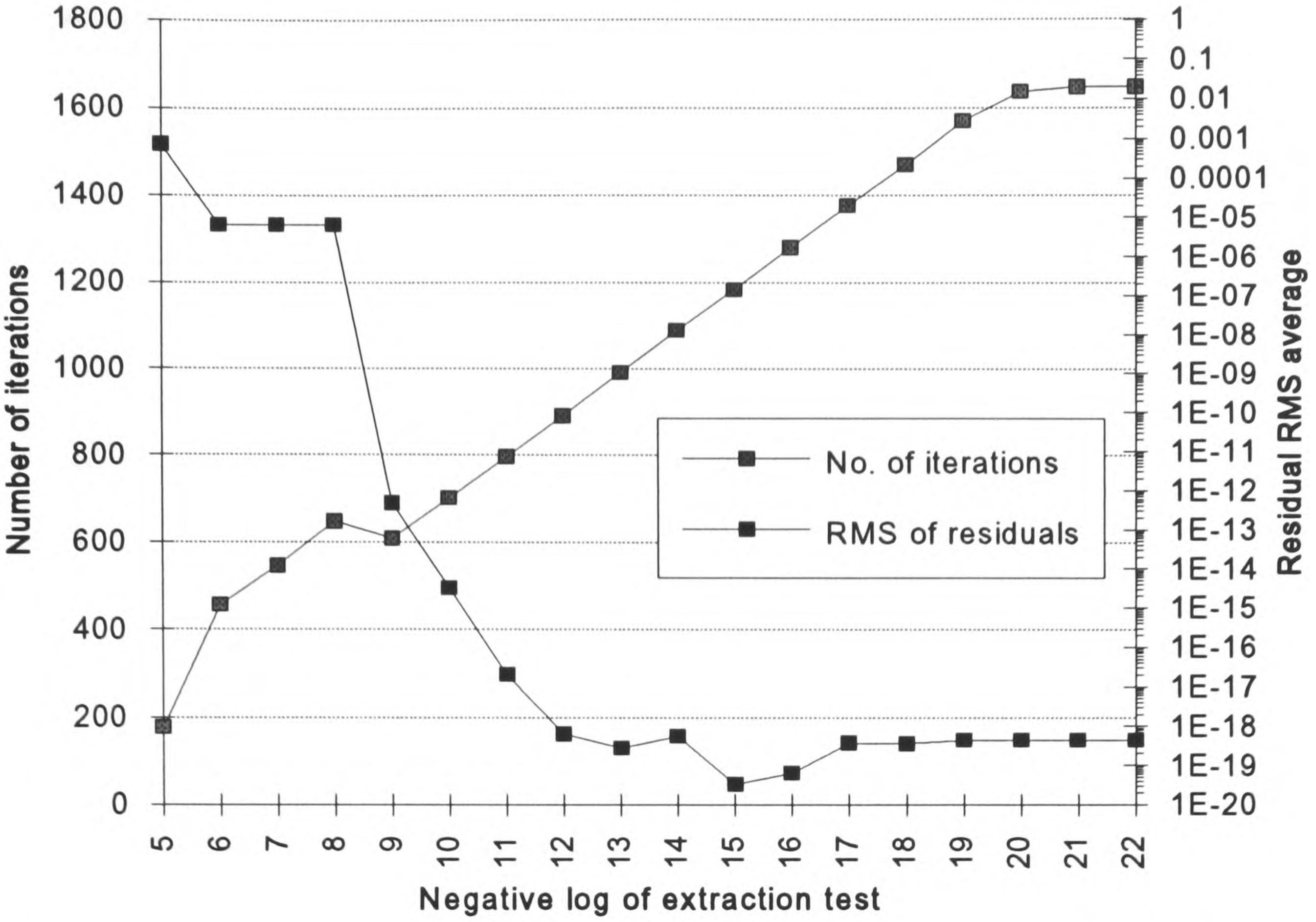


Figure 12: Plot of total iterations against RMS average of residuals for the transition metal ion data set.

More curious and less easy to explain is why the RMS averages and arithmetic means for the last four test values are constant but the range changes. The difference in absolute values in the residual matrices is very small and is probably lost in calculation error on numbers this small. It also not clear why a wholly negative residual matrix is produced from an entirely positive data set but study of the other data sets tested shows that this is not always the case and the final result is thought to be influenced by the effects of cumulative errors on the calculation producing a non-random distribution of errors in the final residual matrix.

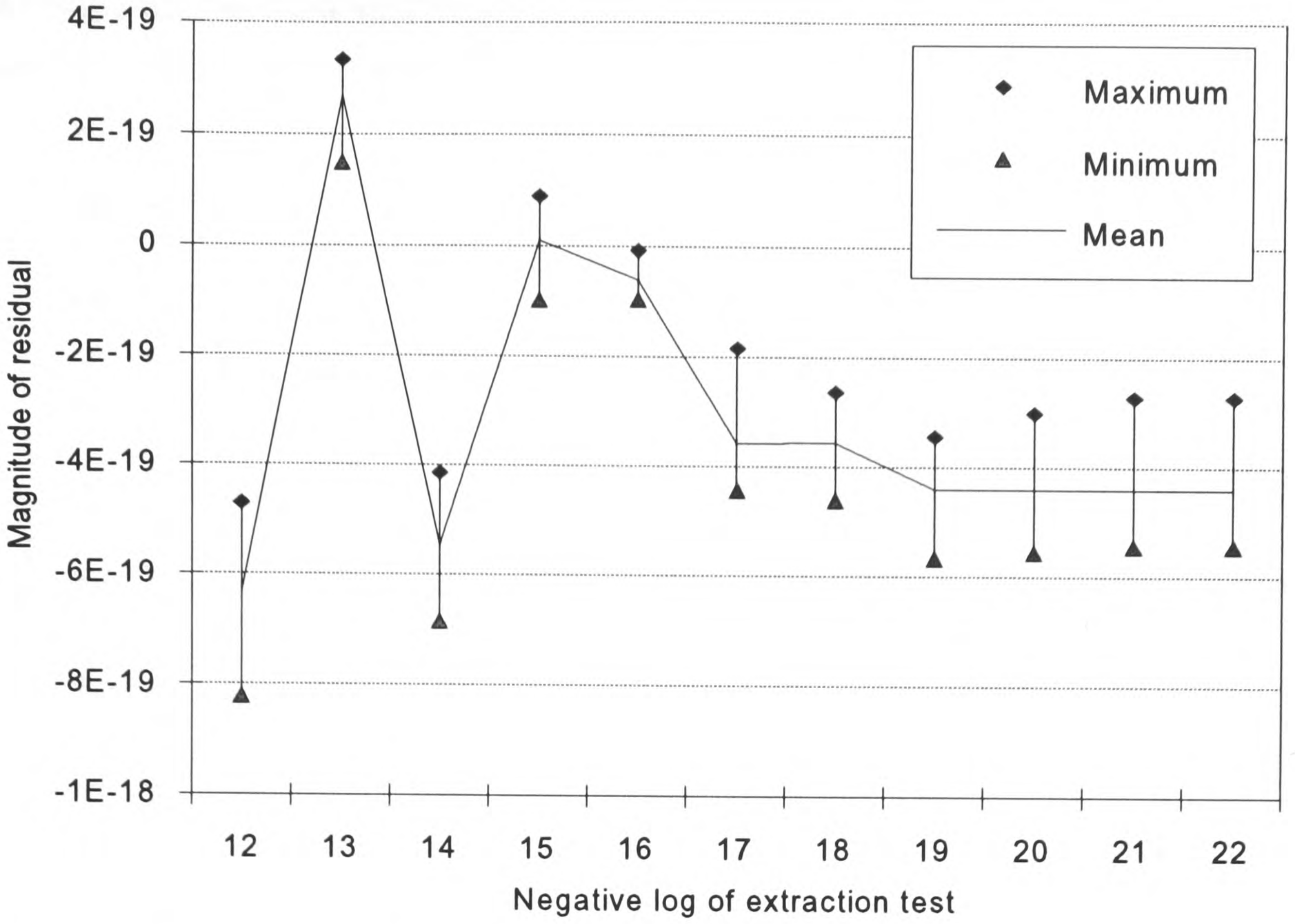


Figure 13: Maximum, minimum and mean values for the transition metal ion data set.

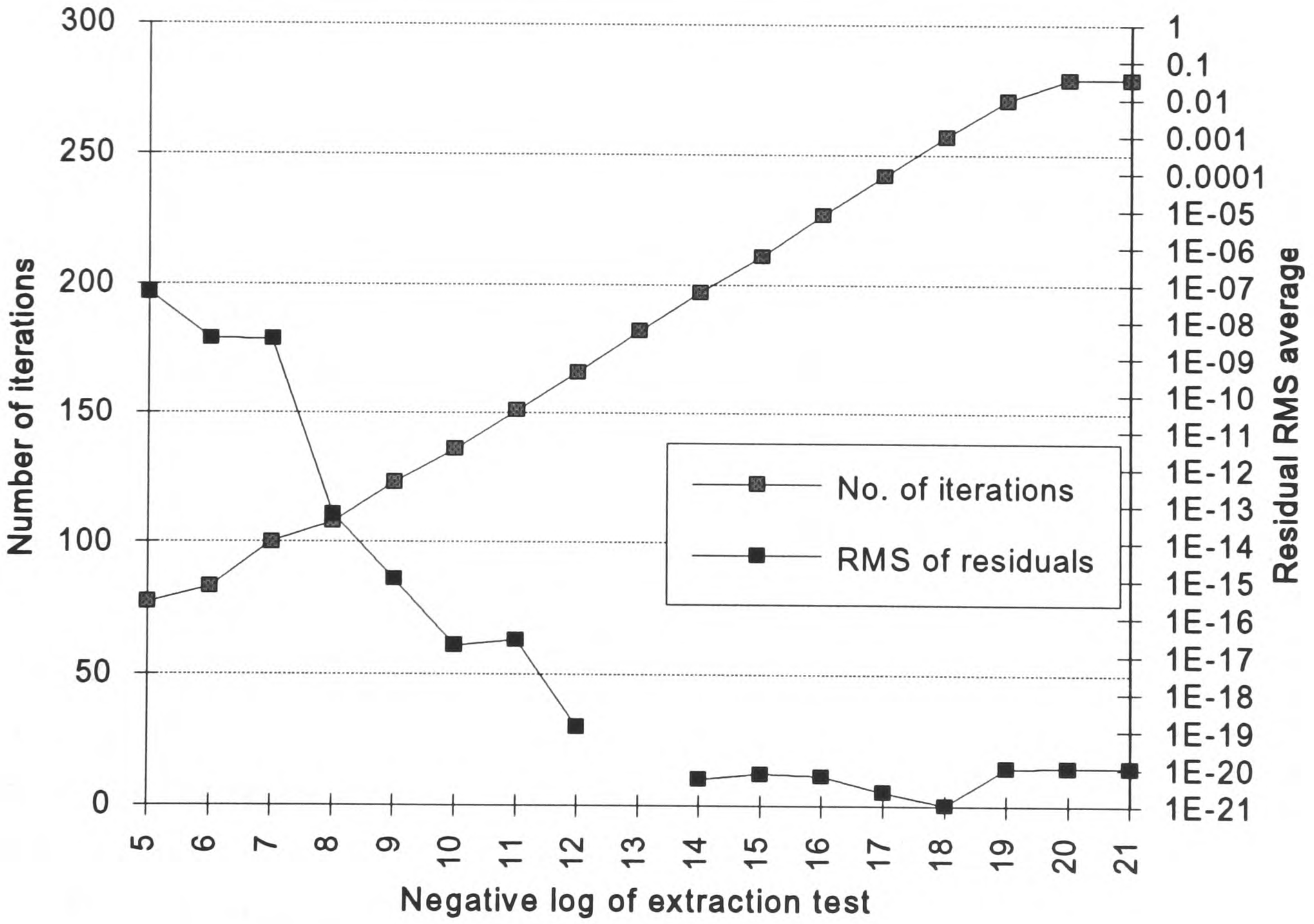


Figure 14: Plot of total iterations against RMS average of residuals for the Cartwright[88] data set.

The results of the analysis of the Cartwright[88] data set are given in figure 14, as can be seen this data set gave very similar results to the above, but it should be noted that the lowest residual RMS average now occurs at a test value of 1×10^{-18}. The results for the mass spectrometer data set (figure 15) also exhibit a similar small minimum suggesting that this effect is characteristic of the decomposition process.

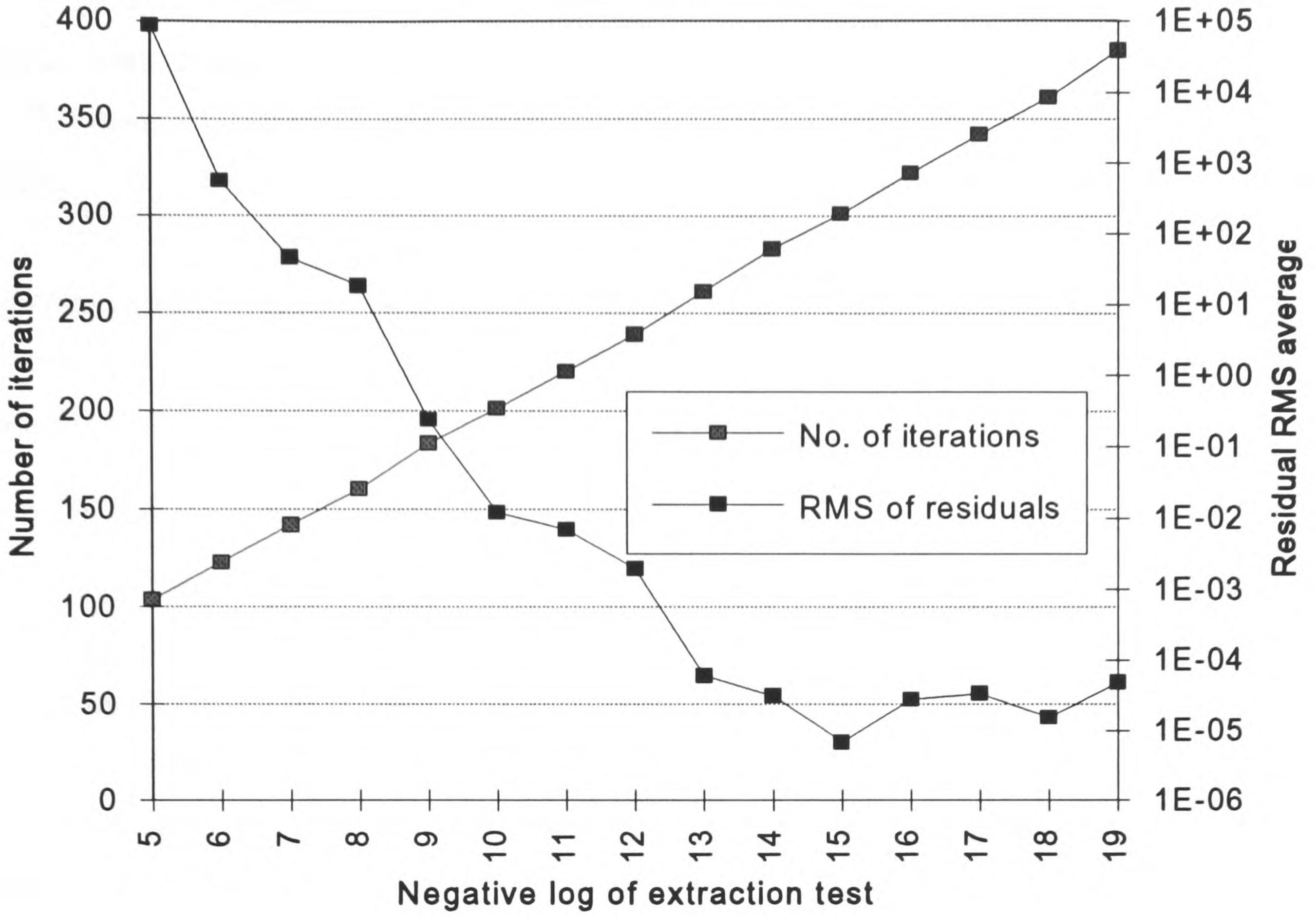


Figure 15: Plot of total iterations against RMS average for the TB11 mass spectroscopy data set.

As the residual matrix is expected to be zero at the completion of decomposition the selection of the test value that produces the smallest range of values centred about zero might be expected to produce the most accurate results. However, the results considered here indicate that the smallest RMS average will not occur at a fixed test value for every data set, precluding the selection of test values by that criterion.

It is also interesting that the magnitude of the final residual matrix for the mass spectroscopy data is 15 orders larger than that of the absorbance data. The absorbance data covers a range 0 to 1 whilst the mass spectrometer data is in the range 0 to 10^7. Evidently the difference is responsible for the larger residuals but whether this is due to the range of values or the magnitude is uncertain. Figure 15 shows the same behaviour of the residuals as the absorbance data sets including the very small, zero centred residual, in this case at a test value of 1×10^{-15}.

From the results considered here it is apparent that the factor extraction test may be set to a larger value than the theoretical minimum to obtain a saving in time of decomposition of around 50%. The data indicates a maximum test value of 1×10^{-14} to obtain a reasonable decomposition, setting a value above this may result in the relaxing of the constraint of orthogonality in the diagonalization process and therefore invalidating the linear combination model necessary for successful interpretation.

The results also seem to indicate that the effects of cumulative errors in the calculations can be minimised by selection of the appropriate test value to produce the smallest zero centred residual matrix indicative of the optimum modelling of the data set. The appropriate test value has been shown to vary for each data set thereby making its selection impracticable except by the method shown here.

The final recommendation is that the factor extraction test is best set for the theoretical calculation limit of 1×10^{-19} where an accurate diagonalization is performed for any data set albeit with a penalty in time taken and in the introduction of slightly more cumulative calculation error than is possible.

4.2.2. Determining completion of extraction

The diagonalization process fully describes the data set after the extraction of a number of factors from the covariance matrix equal to the number of rows or columns of the original data, whichever is the smaller. However, because of redundancy in the data, the number of factors necessary to describe the data completely is often less than the smaller dimension of the data. In order to determine when the data has been adequately modelled the residual matrix is tested by comparison of the values in the matrix with a test value. If all of the values in the residual matrix are smaller than the test value then the data has been adequately described and the decomposition process can be halted.

In order to study the size of the residuals after each factor has been extracted the program was modified to allow the residual matrix to be saved to disk after each factor extraction. The data obtained from this procedure for the transition metal ion data set and the mass spectroscopic data set TB11, modified as listed previously, are discussed below.

If the magnitude of the test value is to be modified for each type of data then a method of determining the test value must be found. The theoretical zero for any set of data must be the limit of its accuracy. For the absorbance data the accuracy is simply determined by the manufacturer's specification for the instrument or by well defined experimental means. For the mass spectrometer no such value is available and one must be estimated (see later).

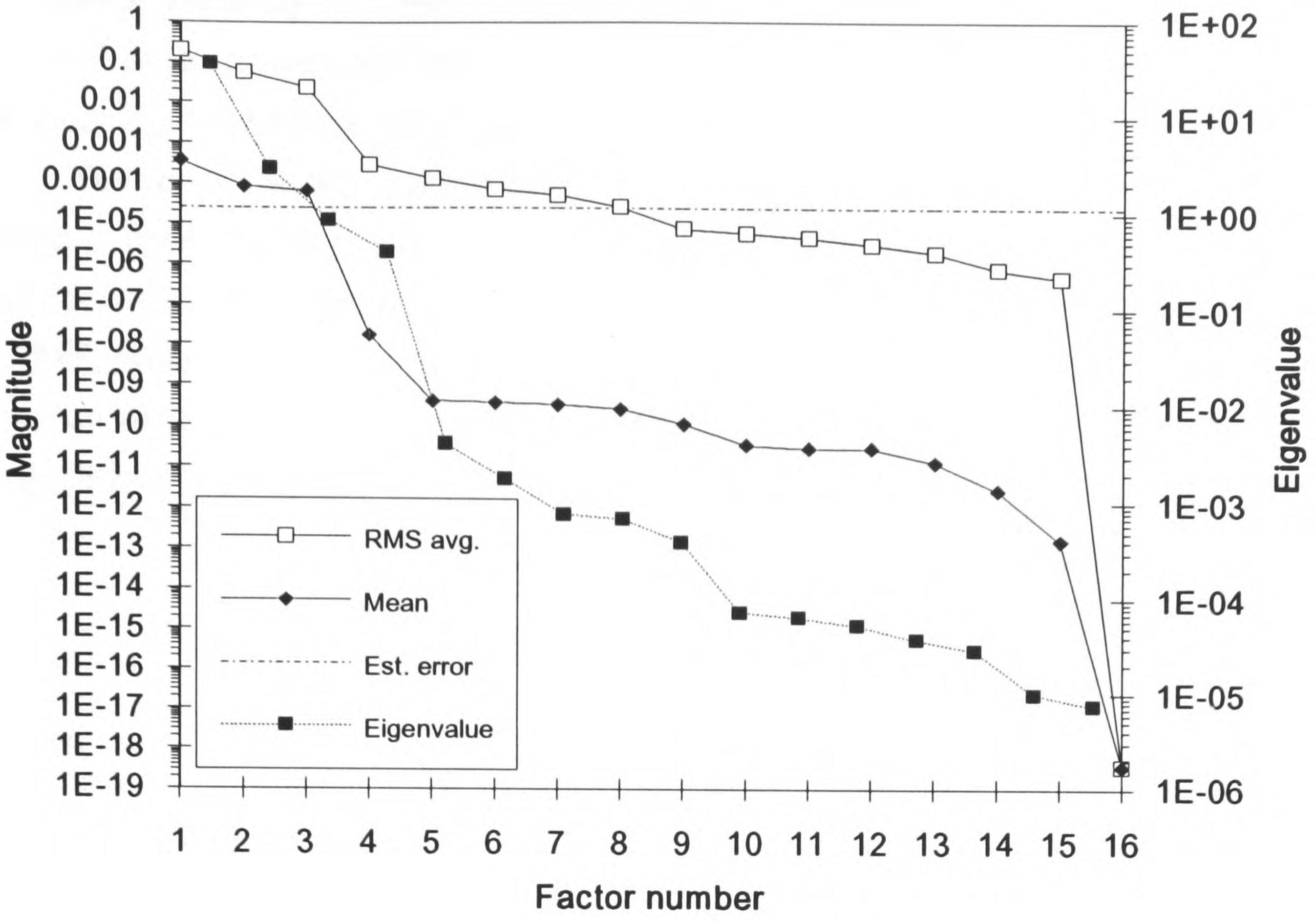


Figure 16: Plot of the magnitude of the residual matrix and eigenvalue against factor number for the transition metal ion data set.

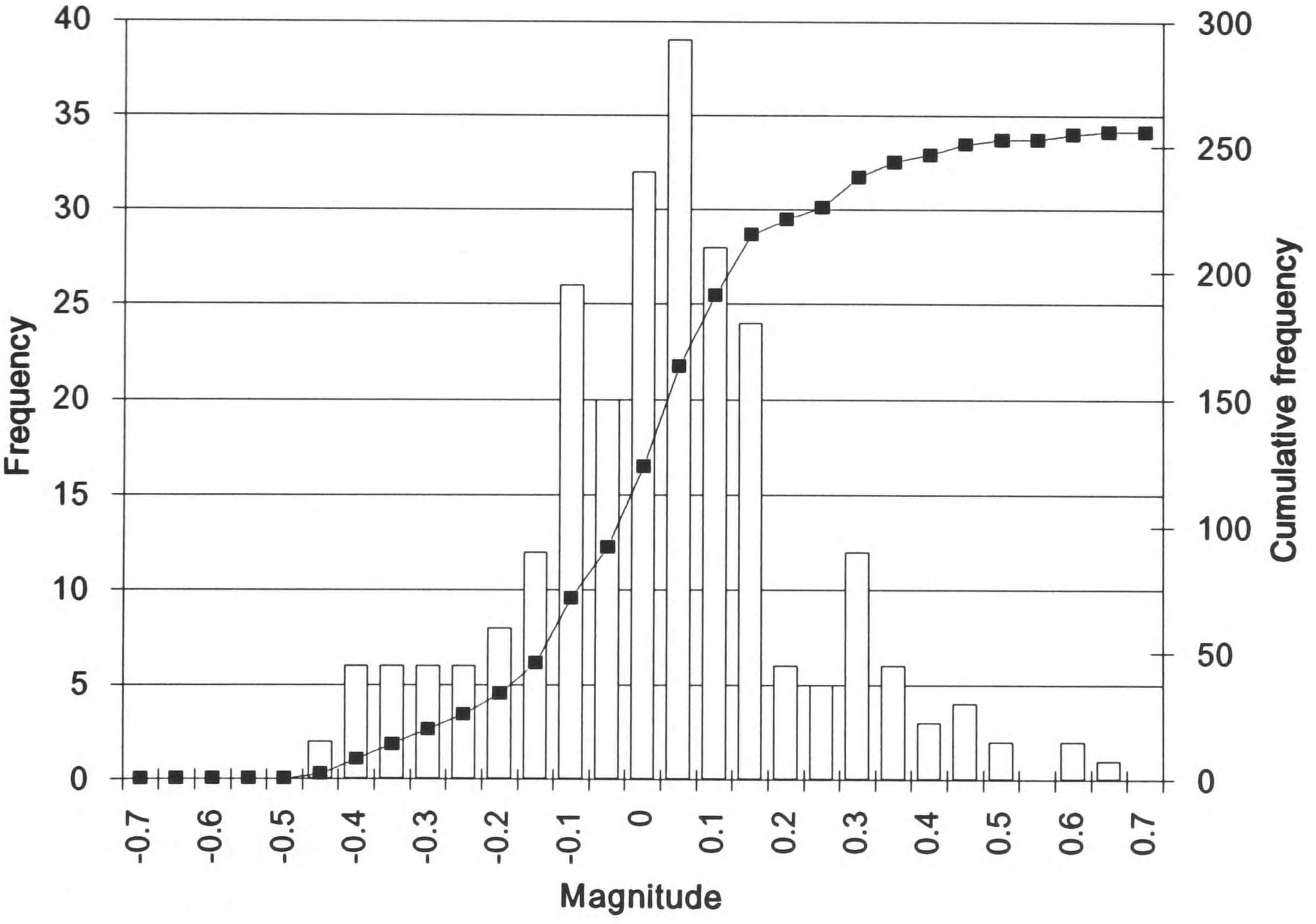


Figure 17: Frequency distribution for the first residual matrix of the transition metal ion data set.

For the transition metal ion data the spectrometer used to collect the spectra has a quoted accuracy of 0.004 absorbance units. This translates to a value of 1.6×10^{-5} in the covariance matrix as the covariance matrix is found from equation 13 thus squaring the error in each value.

The results of the analysis of the residuals are given in figure 16. The RMS average is used to give the most representative value for the magnitude of the residual matrix but the values for the standard deviation were found to be almost identical to the RMS values for all except the original covariance matrix. A consideration of the frequency distributions of the residual matrices showed that they have a roughly even distribution either side of zero (see figure 17).

$$\mathrm{RMS} = \sqrt{\frac{\sum_i x_i^2}{n}} \tag{43}$$

$$s = \sqrt{\frac{\sum_i (x_i - \bar{x})^2}{n-1}} \equiv \sqrt{\frac{\sum_i x_i^2}{n-1} - \frac{\left(\sum_i x_i\right)^2}{n(n-1)}} \tag{44}$$

The RMS average is calculated from equation 43 and the standard deviation from equation 44. It can be seen that the first term in the second equation for standard deviation is essentially the RMS average; and that for a zero centred distribution, the result of the second term will be very small thus explaining the similarity. The fact that the data is now spread about zero when all the measured values were positive serves to highlight the way the eigenvectors are found. The first eigenvector passes through the greatest concentration of data points producing, in effect, an average of the data. The residual matrix will contain the distance between the eigenvector and the data points and will therefore contain both positive and negative values spread around zero. The spread is due to unextracted factors and error. After all the real factors have been extracted, the error, if it is purely random, would be expected to be arranged in a normal distribution with a mean of zero. This behaviour should allow identification of real factors from error factors by determining how good the fit of the residuals is to the normal distribution. Unfortunately the data sets studied here did not show such behaviour and this is thought to be due to inaccuracies in the diagonalization process used.

The behaviour of the mean is seen to be markedly different to that of the RMS average due to the presence of negative values in the residual matrix after the extraction of the first factor. The mean gives a measure of how well centred on zero the distribution of values in the residual matrix is and should move towards zero as factors are extracted. The RMS average gives an indication of the spread of the data in a manner similar to the standard deviation. The spread of the data is an indication of the amount of information remaining in the residual matrix, so as the factors are removed it should fall to some minimum value.

The transition metal ion data set contains four components. The eigenvalues show a sharp fall after four factors in agreement with this. The RMS average and the mean also show a drop after the extraction of the fourth factor. The mean shows a further drop after the extraction of the fifth factor that is not reflected in the RMS average or the eigenvalue and the interpretation of this is not clear. A further fall is visible after the extraction of factor 9 and again the mean shows a fall for the next factor as well. This secondary fall is interpreted as the boundary between error in the data and error introduced by the calculation process itself, which should be considerably smaller than experimental errors.

This indicates that the mean and RMS average can be used to determine the number of components in the system but that generally the eigenvalue gives a more pronounced indication.

From consideration of the accuracy of the instrument, the threshold between factors describing information and those describing error should be at 1.6×10^{-5}. As the data was digitised by hand it might be expected that the errors introduced from this source will be greater than that from the instrument but a consideration of the measurement error produces an estimate of 0.003 absorbance units, which is less than the error from the spectrometer. The theory of propagation of errors results in the most probable error estimate as follows

$$\begin{aligned}\text{Most probable error} &= \sqrt{0.004^2+0.003^2}\\ &= 0.005 \text{ absorbance units}\end{aligned} \tag{45}$$

Using the RMS average this shows that the correct number of factors has been extracted at factor 9. If the mean were used instead then only four factors would be extracted, in agreement with the number of factors known to be in the data. The test performed on the residual matrix looks for any absolute number with a magnitude greater than a threshold and in this case the use of the error estimate as a threshold would have halted the decomposition at factor 12. The shallow gradient of both the RMS average and the mean effectively preclude their use in the determination of the size of the factor space by this method as only a small uncertainty in the error estimate may change dimensionality by several factors.

The error estimate compares very favourably with the value for real error calculated by the program for four factors, 0.00509 absorbance units.

It is clear from the results that neither the RMS average nor the maximum value produce an estimate of the dimensionality of the factor space but that either will reduce the number of factors extracted without loosing any of the primary factors, the mean appears in this case to produce the correct dimensionality for the factor space.

Figure 18 shows the results of the residual analysis for the mass spectrometer data set TB11. Once again the RMS error and the standard deviation values were found to be very similar.

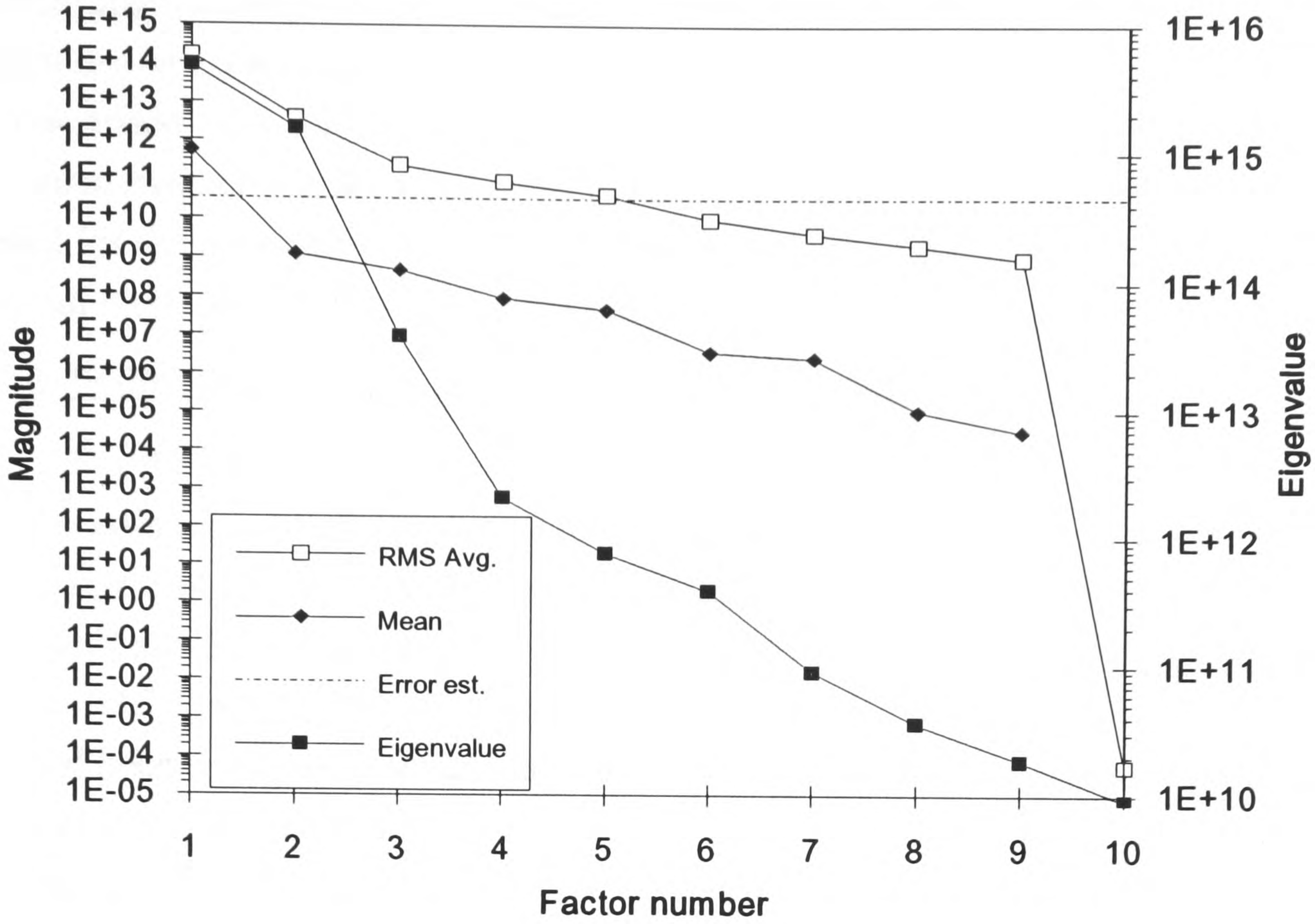


Figure 18: Plot of the magnitude of the residual matrix and eigenvalue against factor number for the mass spectroscopic data set TB11.

The level of the error estimate, calculated using the largest peak in the data set (scan 40, m/e 28) and the estimate of error in the data (see later), is shown on the graph and indicates that in this case according to RMS average 5 factors are necessary to adequately span the data space. The mean value suggests that 1 factor is responsible for the data, in accordance with known facts. The information from the eigenvalues is not as clear with this example as the previous one but, as one compound is involved in the analysis only one factor was expected. A further factor is attributable to the air background present in the spectrometer and it can be seen that after two factors a considerable fall in the eigenvalue occurs. Another fall occurs after factor 6 and this is reflected in the behaviour of the mean. The theoretical limit of accuracy for the mass spectrometer data system is 1, as it is capable of unit resolution. Selection of this value as the criteria for halting the decomposition results in the whole ten possible factors being extracted as the magnitude of the residual matrix remains above 1×10^9 until extraction of the last factor when it drops to around 1×10^{-5}.

The results from both data sets show some promise for the reduction of the number of factors necessary to be extracted from the data. In both the cases here the combination of the error estimate and mean of residual matrix give the appropriate size for the data space but it would be imprudent to recommend this as a method without further testing.

It would appear that if a reasonable estimate of the error in the data can be made then the factor extraction process can be shortened by using the comparison with the RMS average and a time saving made in the calculation process. If the current test of searching for the first value above a threshold is abandoned in favour of comparison with the RMS average then the extra calculation time necessary to determine such an average must be considered when evaluating the overall time needed for decomposition. For these reasons the original system was retained but the program modified to allow the user to halt the decomposition process at will.

4.2.3. Estimation of error in the mass spectrometric data

For the mass spectrometric data an estimate of the error in the procedure was determined using a blank pyrolysis run (data set TB20).

In the blank run the only peaks expected are those due to the air present in the instrument at all times and possibly a small amount of hydrocarbon from the pump oil. The TIC spectrum of TB20 is shown in figure 19, it can be seen that though the tube had been pyrolysed once a small signal from the sample was evident causing the increase in the later part of the pyrogram. This may have been due to some sample remaining in the tube or, more likely, some of the sample may have condensed on the source and is being vapourized by the heat from the close proximity of the probe tip; the source is held at a high temperature and the distance from the probe tip to the ion chamber kept to a minimum to reduce this problem but this evidence indicated that this source of error in the data will also have to be considered when interpreting factors. To avoid any problems that might be caused by this residue, only scans 4-70 were used to calculate the error estimate.

The determination of the error in the spectrometer is complicated by the use of the EPA file format as an intermediary between the mass spectrometer and the PC. The EPA file format is constructed such that for every scan the peak of maximum intensity is used to scale the rest of the spectrum as an integer among 0 and 999, where 999 is the intensity of the most intense peak. This introduces an absolute error of 1/1000 of the base peak of each scan for each peak in the spectrum except for the peak of highest intensity. Consideration of the data showed that the largest signal in the background was from nitrogen in the air at m/e 28; as the intensity of this peak is measured at the full accuracy of the data system then its error will be a true measure of the accuracy of the data system unaffected by the EPA file format encoding process.

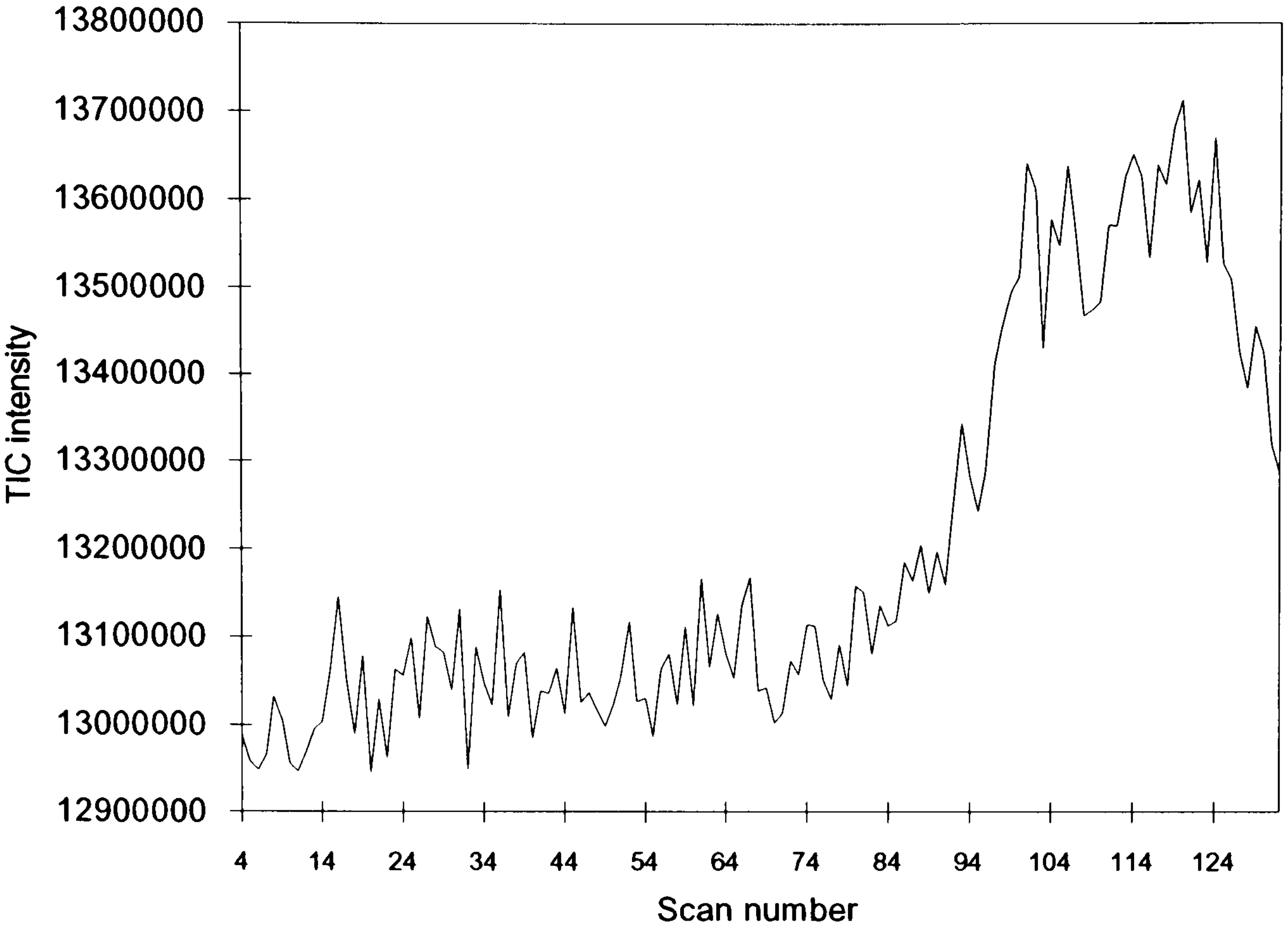


Figure 19: TIC pyrogram for mass spectroscopy data set TB20 (Blank run).

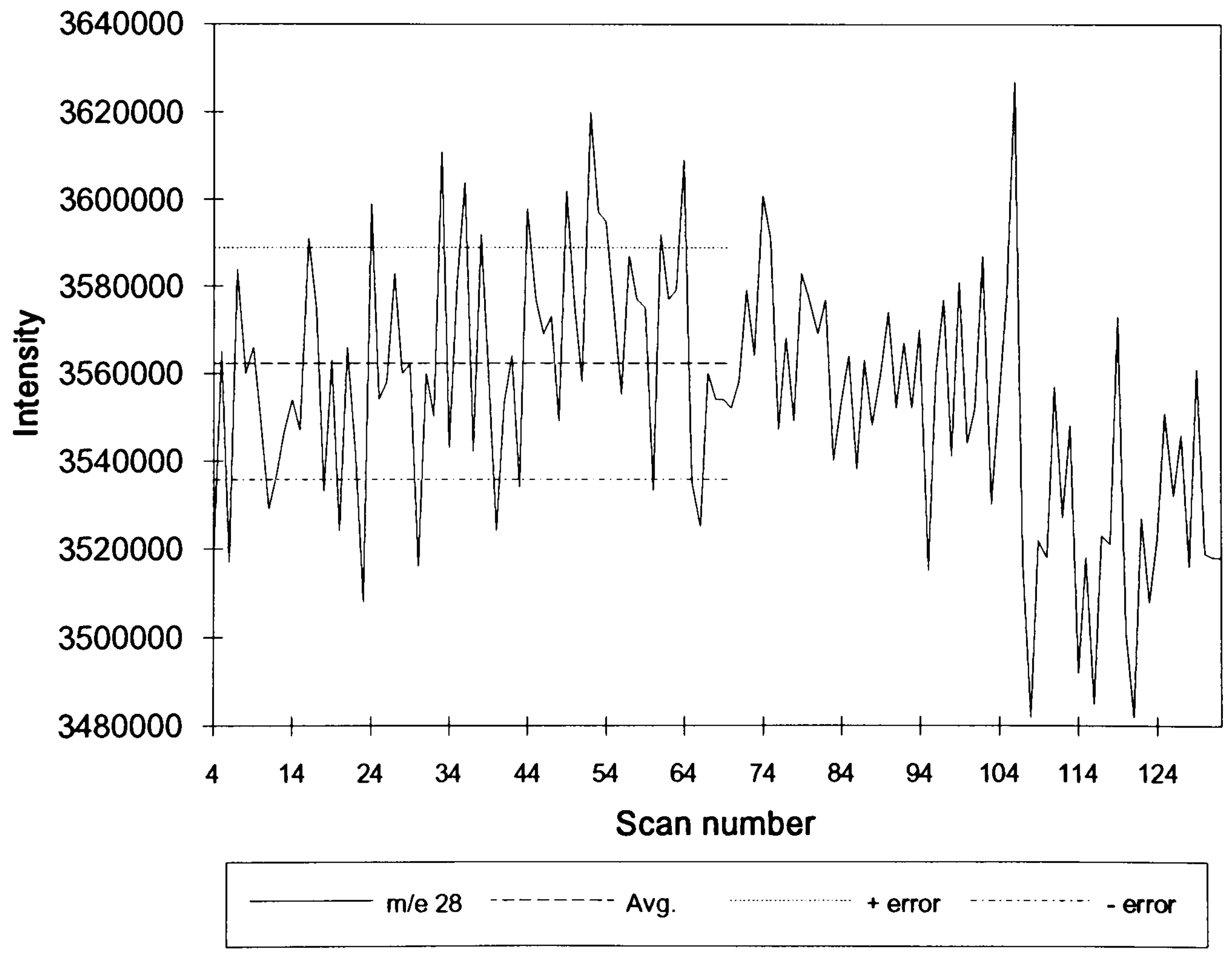


Figure 20: Single ion pyrogram for m/e 28 in mass spectroscopy data set TB20 showing RMS error estimate for scans 4-70.

The intensity of the m/e 28 peak is shown plotted in figure 20, also shown are the values calculated for the average and the spread of the RMS error. The RMS error for the N_2 peak was calculated from the displacement from the mean and converted to a relative error. The error for scans 4-70 was found to be 0.75% and for the whole run, 0.81%. In order to provide the absolute value for error required for interpretation of the error terms, the largest peak in the entire data set is found and 0.75% of its intensity used as the estimate.

Several assumptions are made in this approach to determining the error in the data, firstly that the error in the instrument is proportional to the measured intensity and is not absolute. This is assumed as the signal from a component is due to its partial pressure within the source and its ionization efficiency the latter being a effectively unchanging. The partial pressure is due to the amount of sample being volatilized from the probe and will vary according to the total amount of gas being produced. As the volatilization will be subject to bulk effects, especially in the case of polymer samples, then the partial pressure is expected to vary considerably producing variation in the signal proportional to the size of the signal. It is important to note that in this case there will be no bulk effects as only the background is being measured and that for samples the error may be considerably higher than estimated here.

The second assumption is that the error does not change with m/e ratio, the major factor affecting this is the detector sensitivity. The detectors response is assumed to be linear to different m/e ions and also linear over the dynamic range of the instrument as the higher mass ions tend to be of lower abundance.

The last factor not considered is that of interference from neighbouring peaks or peaks that move in the m/e scale. This is a problem of sampling interval, the spectra are determined at instrument resolution but are stored in the EPA file format as unit resolution. If there are peaks present in the data which have a centriod close to 1/2 m/e units then variance in the m/e scale will allow the peak to move from one unit m/e value to another producing a large error in the data and complicating the factor analysis. This effect is made even worse if there is a neighbouring peak which will then vary widely in intensity depending upon into which unit m/e value the varying peak is placed.

A further problem can be seen in that when the most intense peak is small the quantization limit imposed by the EPA file format results in small intervals allowing the modelling of small changes of intensity. If however the most intense peak is large then the quanta are also large and small changes of intensity are lost in the large intervals. The net result of this will be to vary the sensitivity of the analysis by altering its dynamic range throughout the analysis. This will necessarily complicate the interpretation of the smaller factors that would describe small changes within the data set thus making the identification of components of low concentration that co-evolve with another more major component difficult if not impossible.

During a run the maximum intensity varies with the amount of material being evolved from the probe thereby altering the magnitude of the error for each scan. An upper limit to the error is set by the limit of the analogue to digital conversion of the mass spectrometer and a lower limit set by the magnitude of the largest air peak when nothing is being evolved from the probe.

The magnitude of the largest error caused by quantization (0.1%) is still small compared to the error estimate calculated from the blank run (~ 1%) and the effect of this source of error on the data is expected to be small.

4.3. Validation of PCA

4.3.1. Synthetic data sets

4.3.1.1. Pure data matrix

The pure data matrix used by Malinowski & Howery[87] was analysed using the program and the results compared with those published.

Covariance matrix

$$\begin{bmatrix} 1364 & 4850 & 212 \\ 4850 & 24725 & -5230 \\ 212 & -5230 & 4820 \end{bmatrix} \equiv \begin{bmatrix} 1364 & 4850 & 212 \\ 4850 & 24725 & -5230 \\ 212 & -5230 & 4820 \end{bmatrix}$$

Published data   Calculated results

Figure 21: Comparison of published and calculated results for the covariance matrix of the pure data set.

Abstract column matrix

$$\begin{bmatrix} 0.180200 & 0.957463 & -0.225372 \\ 0.348777 & 0.151999 & 0.924790 \end{bmatrix} = \begin{bmatrix} 0.180197 & 0.957462 & -0.225378 \\ 0.348804 & 0.152039 & 0.924781 \end{bmatrix}$$

Published data   Calculated results

Figure 22: Comparison of published and calculated results for the abstract column matrix of the pure data set.

The small discrepancies in values between the two matrices is ascribed to the greater accuracy of the program. The published data is calculated by hand and will contain considerably more rounding error than the values from the program which are calculated with a precision of 19 significant figures.

Abstract row matrix

$$\begin{bmatrix} 43.5267 & -11.0207 \\ -6.1847 & 6.0624 \\ 9.3939 & 6.6143 \\ 14.0909 & 9.9214 \\ 51.4330 & 4.9630 \\ -20.0419 & 27.5564 \\ 82.5903 & 6.0669 \\ 119.9323 & 1.1084 \\ 15.8124 & 31.9674 \\ -12.4357 & 43.5401 \end{bmatrix} = \begin{bmatrix} 43.52683 & -11.0189 \\ -6.18476 & 6.060277 \\ 9.3939 & 6.614729 \\ 14.09085 & 9.922093 \\ 51.43292 & 4.96532 \\ -20.0421 & 27.55567 \\ 82.59024 & 6.070623 \\ 119.9323 & 1.11385 \\ 15.81219 & 31.96834 \\ -12.136 & 43.53984 \end{bmatrix}$$

Published data   Calculated results

Figure 23: Comparison of published and calculated results for the abstract row matrix of the pure data set.

Eigenvalues

Published data	Calculated results
26868.9	26868.88
4040.2	4040.122

Table 16: Comparison of published and calculated results for the eigenvalues of the pure data set.

4.3.1.2. *Raw data matrix*

The raw data matrix used by Malinowski & Howery[87] was analysed by the program and the results compared with those published.

Abstract column matrix

$$\begin{bmatrix} 0.180847 & 0.956468 & -0.229048 \\ 0.349120 & 0.155291 & 0.924121 \\ 0.919462 & -0.247089 & -0.305838 \end{bmatrix} \equiv \begin{bmatrix} 0.180847 & 0.956468 & -0.229048 \\ 0.349120 & 0.155291 & 0.924121 \\ 0.919462 & -0.247089 & -0.305838 \end{bmatrix}$$

Published data Calculated results

Figure 24: Comparison of published and calculated results for the abstract column matrix of the raw data set.

The values for the raw data matrix are identical to the published figures, it is thought that the values for the raw data set were calculated using a computer and not by hand as was the case for the pure data set.

Abstract row matrix

$$\begin{bmatrix} 44.1189 & -10.8161 & -0.3292 \\ -5.7411 & 6.5011 & 0.0746 \\ 9.6656 & 5.5084 & -0.4036 \\ 14.5264 & 9.8520 & 0.9064 \\ 50.8233 & 5.5204 & -0.1289 \\ -20.3982 & 27.8625 & 0.0670 \\ 82.9352 & 5.8277 & -0.6886 \\ 119.2441 & 1.3044 & 0.6305 \\ 15.2177 & 32.6784 & -0.2836 \\ -12.1847 & 43.5148 & 0.0127 \end{bmatrix} \equiv \begin{bmatrix} 44.1189 & -10.8161 & -0.3292 \\ -5.7411 & 6.5011 & 0.0746 \\ 9.6656 & 5.5084 & -0.4036 \\ 14.5264 & 9.8520 & 0.9064 \\ 50.8233 & 5.5204 & -0.1289 \\ -20.3982 & 27.8625 & 0.0670 \\ 82.9352 & 5.8277 & -0.6886 \\ 119.2441 & 1.3044 & 0.6305 \\ 15.2177 & 32.6784 & -0.2836 \\ -12.1847 & 43.5148 & 0.0127 \end{bmatrix}$$

Published data Calculated results

Figure 25: Comparison of published and calculated results for the abstract row matrix of the raw data set.

Eigenvalues

Published data	Calculated results
26760.421	2670.4214
4090.5369	4090.5369
2.0717	2.0717

Table 17: Comparison of published and calculated results for the eigenvalues of the raw data set.

Error terms

	Published data		Calculated results	
Factor	1	2	1	2
Real error (RE)	14.3049	0.4552	14.3049	0.4552
Imbedded error (IE)	8.2589	0.3716	8.2589	0.3716

Table 18: Comparison of published and calculated results for the error terms of the raw data set.

The values calculated for standard error in eigenvalue were validated using this data set by parallel calculation of their values using a spreadsheet and comparison of the results, which were found to be identical.

4.3.2. Published data and results

4.3.2.1. Mass spectrometer cyclohexane/cyclohexene data

The analysis used by Ritter et al.[50] was a variant of Q-mode factor analysis. The normal Q-mode factor analysis uses a columnwise normalization involving the division of each data point by the norm of its associated column, however, the method used by Ritter et al.[50] involved subtracting the column mean from each member of the column, usually referred to as covariance about the mean. This treatment was applied to the data given in table 6 before analysis using the program and comparison with the published results.

Abstract column matrix

Factor	Published data				Calculated results			
1	0.411	0.566	0.594	0.397	0.4112	0.5656	0.5945	0.3970
2	-0.633	-0.275	0.245	0.681	-0.6330	-0.2753	0.2452	0.6808
3	0.579	-0.768	0.235	0.143	0.5788	-0.7678	0.2345	0.1432
4	-0.309	-0.122	0.729	-0.599	0.3087	0.1217	-0.7290	0.5987

Table 19: Comparison of published and calculated results for the abstract column matrix of the cyclohexane/cyclohexene data set of Ritter et al.[50]

The first three factors show good agreement with the published data. The fourth factor has the correct magnitudes but appears to have its sign reversed and it is unclear whether this is a typographic error or a calculation error in the original paper; as the factor is describing only error the reversal is not critical.

Eigenvalues and error terms

	Published data				Calculated results			
Factor	1	2	3	4	1	2	3	4
Eigenvalue	1035.8	222.7	0.7	0.2	1035.82	222.74	0.70	0.23
Real error (RE)	1.932	0.154	0.118		1.9308	0.1524	0.1067	
Imbedded error (IE)	0.966	0.1092	0.0969		0.9654	0.1078	0.0924	
Indicator function (IND)	0.2147	0.0386	0.1118		0.2145	0.0381	0.1067	

Table 20: Comparison of published and calculated results for the eigenvalues and error terms of the cyclohexane/cyclohexene data set of Ritter et al.[60]

The calculated values agree well for the eigenvalues and the error terms for the first two factors. A small difference is apparent in the error terms for the third factor and it is thought to be caused by the different fourth eigenvector. The differences do not alter the interpretation of the error terms which still indicate two primary factors.

4.3.2.2. Mass spectrometer cyclohexane/hexane data

The mass spectrometer data for the mixtures of cyclohexane in hexane investigated by Ritter et al.[50] were converted to mean centered values as for the cyclohexane/cyclohexene data. The data was then analysed and the results compared with those published.

Eigenvalues and error terms

	Published data				Calculated results				
Factor	RE[a]	IE[a]	IND[a]×10^3	%SL[b]	Eigenvalue	RE	IE	IND×10^3	%SL
1	1.810	0.684	50.27	0.5	1059.052	1.807	0.683	50.19	3.245
2	0.465	0.249	18.26	0.2	333.860	0.456	0.244	18.25	0.159
3	0.128	0.084	8.03	0.7	17.561	0.127	0.083	7.95	0.681
4	0.111	0.084	12.30	37.9	0.524	0.109	0.082	12.10	31.676
5	0.098	0.073	24.56	46.1	0.314	0.095	0.080	23.81	44.922
6	0.074	0.068	73.51	47.2	0.243	0.068	0.063	67.97	43.065
7					0.083				

Key: RE = Real error, IE = Imbedded error, IND = Indicator function, %SL = percentage significance level.

[a]Reproduced from Malinowski[11]. [b]Reproduced from Malinowski[12].

Table 21: Comparison of published and calculated results for the eigenvalues and error terms of the cyclohexane/hexane data set of Ritter et al.[60]

The values calculated for the error terms agree well except for the percentage significance levels, these show considerable deviation from the published values. The trend is the same and the dimensionality of the data space is found to be the same from both sets of results. The discrepancies between %SL values will be looked at in more detail with the ^{1}H NMR data set below.

4.3.2.3. ^{1}H NMR data

The ^{1}H NMR data of Weiner et al.[89] was analysed using the program and the results compared with the published values.

Eigenvalues and error terms

	Published data				Calculated results				
Factor	RE[a]	IE[a]	IND[a] $\times 10^2$	%SL[b]	Eigenvalue	RE	IE	IND	%SL
1	2.32	0.77	3.62	0.0	10243870.24	2.32	0.77	3.62	0.0
2	1.12	0.53	2.29	1.4	477.09	1.12	0.53	2.29	1.5
3	0.58	0.33	1.60	3.0	95.46	0.58	0.33	1.60	3.0
4	0.48	0.32	1.93	29.0	11.63	0.48	0.32	1.93	25.1
5	0.37	0.27	2.30	24.4	8.79	0.37	0.27	2.30	22.3
6	0.29	0.24	3.26	33.2	3.94	0.29	0.24	3.26	29.3
7	0.27	0.24	6.79	51.0	1.56	0.27	0.24	6.79	51.1
8	0.22	0.21	22.43	53.0	1.36	0.22	0.21	22.43	55.7
9					0.70				

Key: RE = Real error, IE = Imbedded error, IND = Indicator function,
%SL = percentage significance level.
[a]Reproduced from Malinowski[11]. [b]Reproduced from Malinowski[12].

Table 22: Comparison of published and calculated results for the eigenvalues and error terms for the ^{1}H NMR data set of Weiner et al.[89]

	Published data				Calculated results				
Factor	λ	$\bar{\lambda}$	F	%SL	λ	$\bar{\lambda}$	F	%SL	%SL*
1	10243900	81300.8	51986.6	0.0	10243870	81300.6	51986.3	0.0	0.0
2	477.09	4.5874	10.406	1.4	477.09	4.5874	10.405	1.5	1.5
3	95.46	1.1364	7.96	3.0	95.46	1.1364	7.958	3.0	3.0
4	11.63	0.1762	1.401	29.0	11.63	0.1763	1.401	25.1	25.1
5	8.79	0.1758	1.86	24.4	8.79	0.1759	1.860	22.3	22.3
6	3.94	0.1094	1.33	33.2	3.94	0.1094	1.328	29.3	29.3
7	1.56	0.065	0.631	51.0	1.56	0.0650	0.629	51.1	51.0
8	1.36	0.0971	0.83	53.0	1.36	0.0972	0.828	55.7	55.7
9	0.70	0.1167			0.70				

Key: λ = eigenvalue, $\bar{\lambda}$ = reduced eigenvalue, F = F-ratio, %SL = percentage significance level, %SL* = %SL calculated using F given in published data.

Table 23: Comparison of published and calculated values for eigenvalue, reduced eigenvalue, f-ratio, and percentage significance levels for the ^{1}H NMR data set of Weiner et al.[89]

The values for error terms RE, IE and IND can be seen to have identical values to those published. The significance level again shows a discrepancy between the two figures. To further study the differences values calculated at intermediate points in the calculation of the %SL values were compared with the same values published in Malinowski[12]. The intermediate values checked were the reduced eigenvalue ($\bar{\lambda}$) and the f-ratio (F) and their results are listed in table 23.

From table 23 it can be seen that the eigenvalues from the program are very similar though not identical and thus produce different F values. The differences are thought to be due to the published data being calculated in double precision numbers where the program uses extended precision numbers with twice the number of significant figures. In order to check that the conversion from F-ratio to percentage significance level produced the correct results the values for F given in the paper were used to calculate %SL. The values can be seen to be little changed from those produced by the program suggesting a problem with the calculation of the F distribution function. The calculation of the areas in the tail of the F distribution was then checked using test data provided supplied by Cooke et al.[97] and the results obtained are tabulated below.

Input			Output	
F	K1	K2	Result	True value
18.51	1	2	0.9500	0.95
4.41	20	10	0.9900	0.99
2.54	10	15	0.9497	0.95
4.56	15	10	0.9900	0.99

Key: F = F-ratio, K1,K2 = degrees of freedom of numerator and denominator.

Table 24: Validation of the calculation of the area in the tail of the F distribution using the data supplied by Cooke et al.[97]

The results shown in table 24 clearly indicate that the areas are being correctly calculated, it must therefore be assumed that the values given by Malinowski[12] are based upon an approximation to the F distribution, which, under certain circumstances, contains some error. As the significance level is only important around the 5-10% level then the approximation used, if correct in these regions, is adequate as appears to be the case. It should be noted that most approximations to the F distrbution tend to work only over a limited range for the degrees of freedom and for that reason the incomplete beta function calculation is to be preferred.

4.4. Target testing validation

4.4.1. Synthetic data sets

4.4.1.1. Pure data matrix

The test vectors for the pure data matrix described earlier (figure 5) were used to compare the results obtained for the predicted vector in target testing.

True factors						Unity test		
Test	Published	Predicted	Test	Published	Predicted	Test	Published	Predicted
0	-0.0001	-2.8E-16	4	4.0003	4	1	0.1376	0.137615
1	0.9999	1	-1	-1.0001	-1	1	0.1215	0.121560
2	1.9997	2	0	-0.0001	-4.40E-17	1	0.3119	0.311927
3	2.9996	3	0	-0.0001	-4.80E-17	1	0.4678	0.467890
4	3.9994	4	3	3.0001	3	1	0.7270	0.727064
5	4.9995	5	-4	-4.0005	-4	1	0.6421	0.642202
6	5.9991	6	5	5.0001	5	1	1.1077	1.107798
7	6.9988	7	8	8.0004	8	1	1.3668	1.366972
8	7.9990	8	-2	-2.0005	-2	1	1.1787	1.178899
9	8.9990	9	-5	-5.0008	-5	1	1.2315	1.231651

Published figures and test vectors reproduced from Malinowski & Howery[87].

Table 25: Comparison of published and calculated results of target tests on the pure data set.

The results in table 25 show the greater calculation accuracy of the program as the predicted results are much closer to the values of the test vector than published values.

4.4.1.2. Raw data matrix

The raw data matrix was target tested with test vectors described in figure 7 and the following results obtained.

	Impure factors						Unity test		
Vector	Test	Published	Predicted	Test	Published	Predicted	Test	Published	Predicted
	0.1	0.0880	0.0885	3.9	4.0270	4.0268	1	0.154	0.154
	0.8	1.1130	1.1134	-1.0	-1.0000	-1.0000	1	0.139	0.139
	2.0	1.7590	1.7592	0.2	0.1460	0.1460	1	0.280	0.280
	2.9	2.9910	2.9914	-0.2	0.0720	0.0723	1	0.470	0.470
	4.2	4.0730	4.0728	3.1	2.9580	2.9584	1	0.739	0.739
	5.1	5.0080	5.0080	-4.1	-3.9980	-3.9983	1	0.643	0.643
	5.9	5.9440	5.9437	5.0	5.1250	5.1249	1	1.106	1.106
	7.0	6.9830	6.9834	8.1	8.0300	8.0297	1	1.370	1.370
	7.8	8.0710	8.0714	-1.9	-2.0140	-2.0137	1	1.190	1.190
	9.1	8.9260	8.9260	-4.9	-4.9000	-4.8997	1	1.223	1.223
Known error		0.130	0.130		0.118	0.118		Unknown	0.000
AET		0.172	0.172		0.125	0.125		0.518	0.518
REP		0.104	0.104		0.053	0.053		0.015	0.015
RET		0.137	0.137		0.113	0.114		0.518	0.518
SPOIL		1.32	1.32		2.16	2.16		34.30	34.33
RELI%		97	97		100	103		Unknown	3
%SL		55.9	55.8		41.2	41.2		3.0	3.0

Published figures and test vectors reproduced from Malinowski & Howery[87], except %SL which are reproduced from Malinowski[98].

Table 26: Comparison of published and calculated results of target tests on the raw data set.

It can be seen from the table that the program produces results sufficiently similar to the published values that it may be judged as functioning correctly. The small differences are ascribed to the greater calculation accuracy of the program as exemplified by table 25.

4.4.2. Published data and results

4.4.2.1. Mass spectrometer cyclohexane/cyclohexene data

Target tests were performed on the cyclohexane/cyclohexene data set of Ritter et al.[50] and their results compared with published values.

	Published data[a]		Calculated results	
Test vector[b]	SPOIL[c]	%SL[d]	SPOIL	%SL
Cyclohexane	2.25	29.4	2.25	29.5
Cyclohexene	3.25	17.2	3.20	17.2
Bicyclo[3.1.0]hexane	5.79	6.0	5.79	6.0
Fluorocyclohexane	14.73	1.0	14.73	1.0
Bicyclopropyl	14.74	1.0	14.74	1.0

[a]Published data reproduced from Malinowski[12]
[b]Test vectors reproduced from Lorber[13]
[c]SPOIL function calculated from 41 but using the unbiased estimate given by Malinowski[99] and shown in equation 46.
[d]Percentage significance level calculated from the F-distribution and equation 42.

Table 27: Comparison of published and calculated results of target tests on the cyclohexene/cyclohexane data set of Ritter et al.[50]

The table of results shows good agreement between the calculated values and the values published in the literature. The results were obtained by PCA of the untreated data matrix, ie. covariance about the origin, and not mean centred as in the original work of Ritter et al.[50]. The test vector was also untreated before testing. The SPOIL values for this data set were calculated using a modified AET term. The modification is added to account for loss in degrees of freedom resulting from using fewer than the maximum number of factors to describe the data. The modification is given in equation 46.

$$\mathrm{AET} = \left(\frac{\sum_{i=1}^{r} (\hat{x}_i - x_i)^2}{r - n} \right)^{\frac{1}{2}} \qquad (46)$$

4.4.2.2. Mass spectrometer cyclohexane/hexane data

The cyclohexane/hexane data set was modified by removing the columns containing the pure spectra and the row containing m/e 28 values deleted. The resulting data matrix was then factor analysed and the columns of pure spectra used as test vectors to produce the following results.

Cyclohexane			Hexane		
Test	Published[a]	Predicted	Test	Published[a]	Predicted
1.8	1.9	1.9	2.8	2.7	2.7
1.3	1.3	1.3	5.1	4.8	4.8
2.5	2.3	2.3	1.6	1.8	1.8
0.7	0.6	0.6	0.4	0.2	0.2
7.1	7.3	7.3	8.8	8.7	8.7
3.5	3.5	3.5	4.7	4.6	4.5
2.2	2.1	2.1	8.6	8.8	8.8
0.2	0.2	0.2	0.3	0.2	0.2
0.8	0.7	0.7	0.1	0.1	0.1
4.6	4.9	4.8	0.9	0.6	0.6
13.5	13.6	13.6	6.8	6.9	6.9
1.2	1.2	1.2	12.2	12.3	12.3
3.8	4.1	4.1	0.2	0.3	0.3
0.8	0.7	0.7	0.1	0.3	0.3
10.7	10.4	10.4	0.1	0.2	0.2
0.9	0.9	0.9	0.1	0.2	0.2
0.1	0.1	0.1	2.8	3.0	3.0
Known error	0.13[b]			0.13[b]	
AET	0.18[b]	0.15		0.17[b]	0.16
REP	0.13[b]	0.13		0.14[b]	0.14
RET	0.13[b]	0.08		0.09[b]	0.07
SPOIL	0.70[b]	0.62		0.56[b]	0.52
RELI	102[b]	121		112[b]	122

[a] Reproduced from Malinowski and McCue[14].

[b] Reproduced from Malinowski[52].

Table 28: Comparison of published and calculated results of target tests on the cyclohexane/hexane data set of Ritter et al.[50]

The results show good agreement for the predicted vectors of both components, the values for REP also agree closely. The discrepancies in the error term AET are responsible for the other differences observed. Consideration of the papers containing the original work suggests that the vectors were tested with missing values. The missing data points are allowed for in the calculation of AET by the following modification

$$\mathrm{AET} = \left(\frac{r-n}{p-n} \frac{\sum_{i=1}^{p} (\hat{x}_i - x_i)^2}{r} \right)^{\frac{1}{2}} \qquad (47)$$

where p is the number of missing points in the test vector. Is is clear that if points were missing from the test vector then the estimation of the AET figure will be impaired and will give a different answer to the value given by the program for the complete vector.

4.5. UV-Vis spectrometry

4.5.1. Transition metal ion data

4.5.1.1. Data analysis

The data matrix, collected and digitized as given in the experimental section, was analysed using the program. The variables in the program were set as follows; Error in the data set, 0.005 absorbance units (see equation 45) and the minimum value set to 0.001 as results from the spectrometer are only given to three decimal places. When decomposed the analysis ran to five factors before stopping automatically.

The data was re-analysed using the default minimum setting of 1×10^{-19} and the program then ran to the maximum 16 possible factors. This clearly indicates the time saving benefit of setting a minimum value especially for large data sets.

4.5.1.2. Determination of the rank of the data

The following table gives the results of the error term calculations and the calculated eigenvalues.

Factor no.	1	2	3	4	5
Eigenvalue	36.720	2.996	0.857	0.393	0.004
% variance	89.62	7.31	2.09	0.96	0.01
Cum. var.	89.62	96.93	99.02	99.98	99.99
RE	0.1044	0.0588	0.0344	0.0051	0.0037
IE	0.0261	0.0208	0.0149	0.0025	0.0021
IND	4.64E-04	3.00E-04	2.04E-04	3.51E-05	3.06E-05
Misfits	1	0	0	0	0
SEE	0.0118	0.0123	0.0121	0.0120	0.0128
%SL	0.00	0.28	0.56	0.00	5.61

Key: Cum. var. = % cumulative variance; RE = real error; IE = imbedded error; IND = indicator function; SEE = standard error in eigenvalue; %SL = % significance level.

Table 29: Table of eigenvalues and error terms resulting from PCA of the transition metal ion data set.

The values in table 29 are used to identify how many factors are responsible for the data. The eigenvalue gives a measure of how important the factor is to the data, generally it is not possible to use it alone to determine rank but in conjunction with the standard error in eigenvalue (SEE) term the error factors may be separated from the primary factors. Figure 26 shows the eigenvalues plotted with the SEE. The graph shows the eigenvalue falls sharply after the fourth factor, this behaviour is common especially when the factors being extracted change from accounting for real data and describing error in the data matrix. Caution should be exercised when using such a fall to indicate rank as large errors in the data or even a change in the magnitude of real factor being accounted for can produce such a drop. In conjunction with the SEE values it can be seen that the estimate of error in the data identifies the first four eigenvalues as having magnitudes larger than their standard errors and must therefore describe real information, and the fifth factor, being smaller than the SEE, must describe error in the data. The evidence of these two terms indicates that the data has a rank of four, in accordance with the known facts.

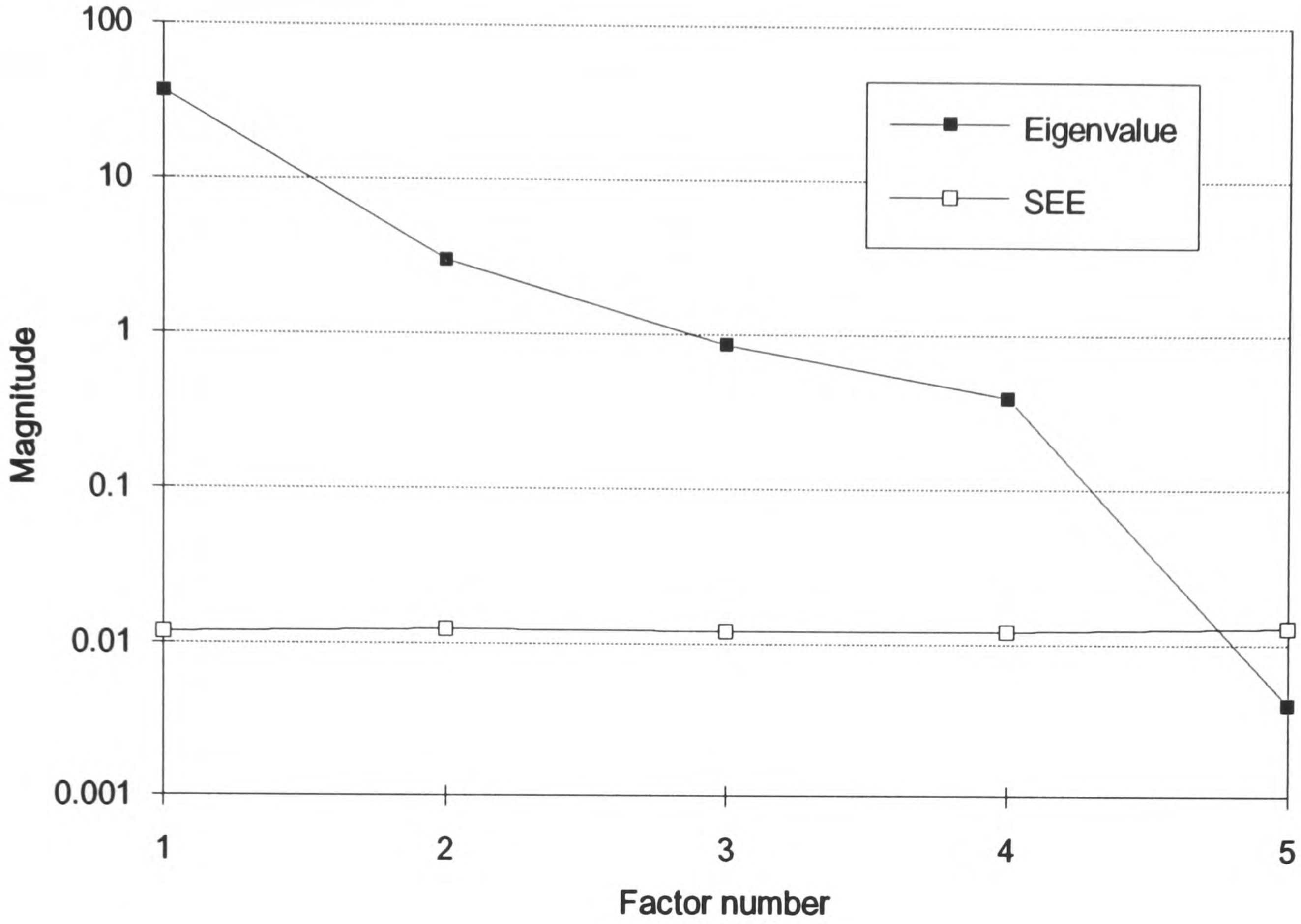


Figure 26: Plot of the eigenvalues and standard error in eigenvalue (SEE) against factor number for the transition metal ion data set.

A further estimate of the dimensionality of the data can be made by consideration of the real error (RE). As there is an estimate of the error present in the data then comparison of that with the values produced by the program may indicate the number of primary factors. Comparison of the error estimate of 0.005 absorbance units with the RE values in table 29 finds that factor four has an almost identical value (0.0051 absorbance units). The fifth value for RE would normally be undefined as the decomposition was stopped at factor five but it has been included in table 29 from a decomposition performed with no minimum value set, and at 0.0037 absorbance units is below the error estimate and must therefore be regarded as error.

The imbedded error (IE) and indicator function (IND) may also give an indication of the rank of the data. Their use does not involve a knowledge of the error in the data and thus they have an advantage over the previous two techniques, they also serve as a check on the rank in the case of an incorrect estimate of error in the data set. The theory from which the imbedded error term is derived suggests that its value should fall to a minimum when the correct number of factors are included in its calculation, and then rise again when error factors are included in the calculation. Unfortunately the theory relies upon the errors being in a uniform random distribution and in cases where non-random errors exist may fail to exhibit a minimum. The indicator function also suffers from the same problems as it is closely linked to the imbedded error.

Figure 27 shows the functions plotted for the first five factors (values for factor five are added from the full 16 factor decomposition). No minimum is seen for either function, even when the full 16 factors are considered, but the sharp fall from factor 3 to factor 4 is indicative of the change from primary to secondary factors at factor 4.

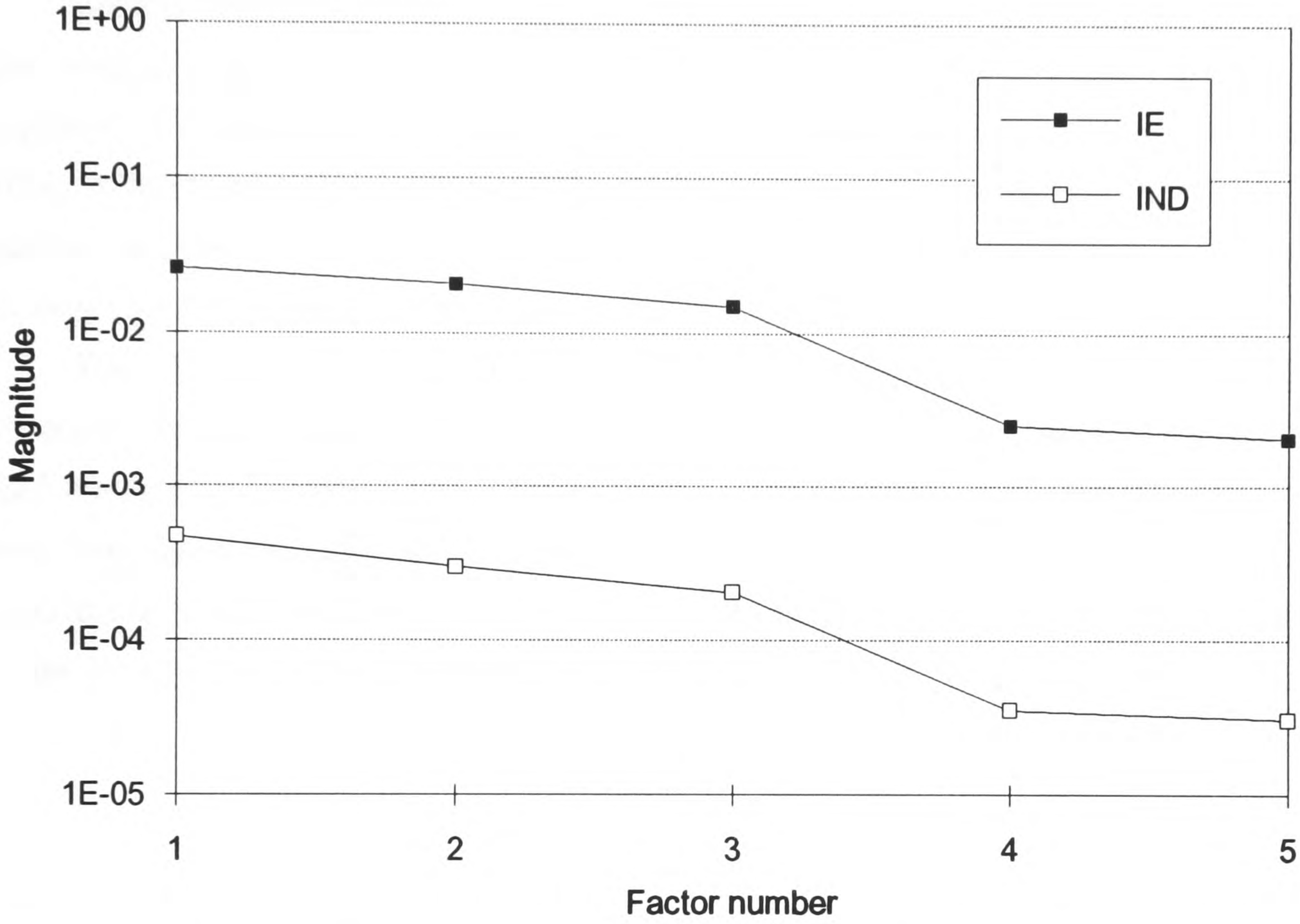


Figure 27: Plot of imbedded error (IE) and indicator function (IND) against factor number for the transition metal ion data set.

For this data set the misfit term is of no use in determining the rank as only one misfit is obtained for data regeneration using one factor. This serves to highlight the arbitrary nature of this criterion and the difficulties involved in its interpretation.

The percentage significance level (%SL) shows that factor five is significant at the 5% level and the factor is therefore indicated to be due to error.

4.5.1.3. Abstract matrices

The abstract matrices contain all the information necessary to recreate the data but arranged in order of importance to the data. This means that the factor that causes the most change in the data will be found first. The second factor will be from the second most important source of variation and will show how it adds or subtracts from the first factor and so on until all the variation in the data has been described. Studying the vectors of the abstract matrices can provide valuable information about the sources of variation in a data set. The structure of the abstract matrices is best observed graphically and there are two common methods of representation. A plot of the vector against sample (for eigenvectors) or variable axis (for scores) allows the importance of any sample (or variable) to the data to be gauged. This can be particularly useful for identifying outliers or unique behaviour in the data set. The second method is to plot vectors against each other; this technique allows structure in the data set to be seen, the relationships between the points being seen as clusters or as linear relationships between successive points.

Because of the way the PCA procedure finds the factors the first vector always passes through the centre of the data points in a fashion similar to simple linear regression, this results in the first vectors of the abstract matrices always being the average of each sample or variable. The second vectors show how the next most important source of variance in the data varies from the average and thus contains both positive and negative components.

The vector plots of the abstract column matrix (eigenvectors) show the average absorbance for each sample for the first vector. The second and following vectors would normally be expected to show structure in the data set related to the concentrations of the components in the different samples, unfortunately as the sample compositions were calculated from a random number table the structure is hidden and so little information can be gleaned from the plots.

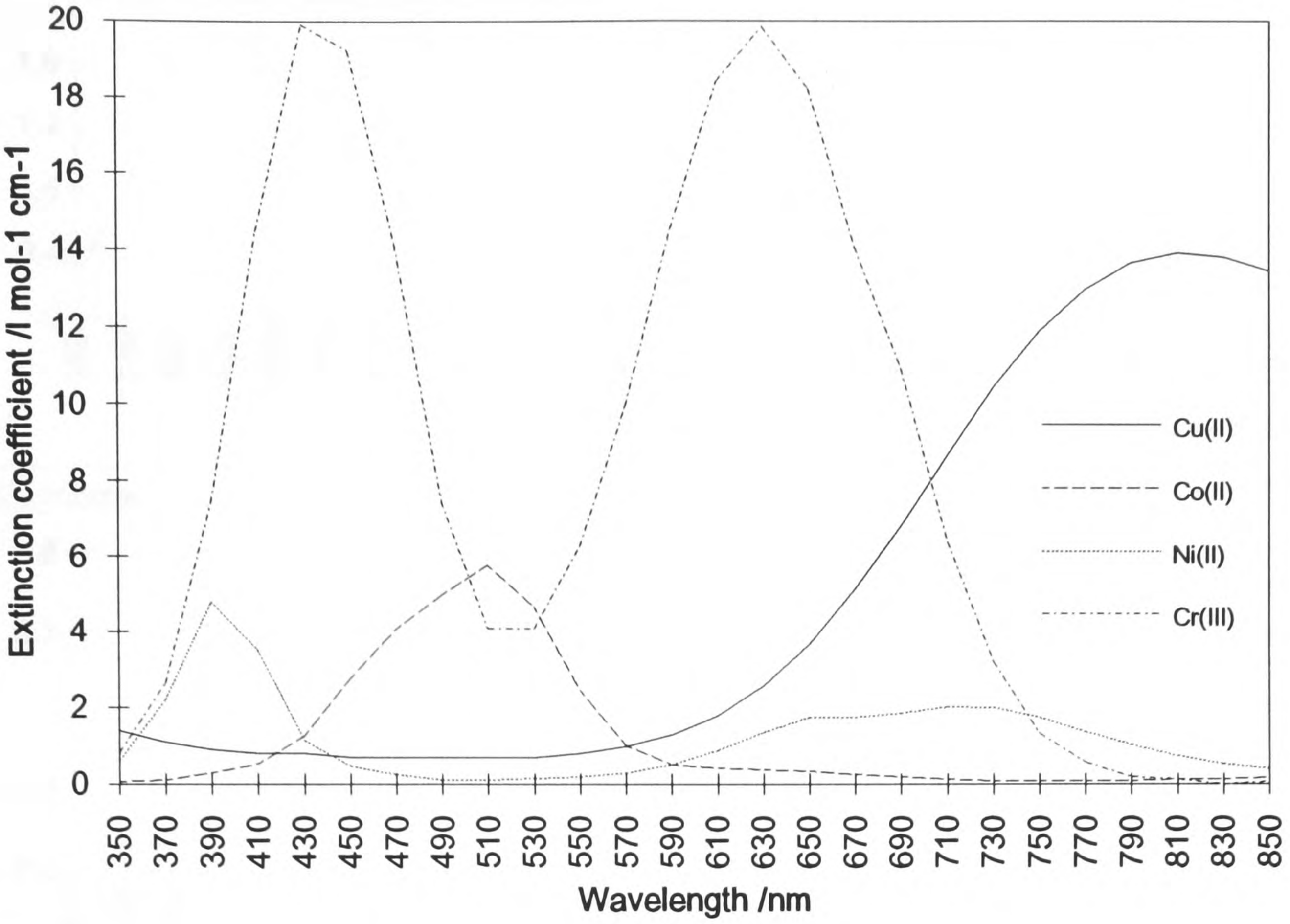


Figure 28: Plot of the spectra of the pure components used in the transition metal ion data set.

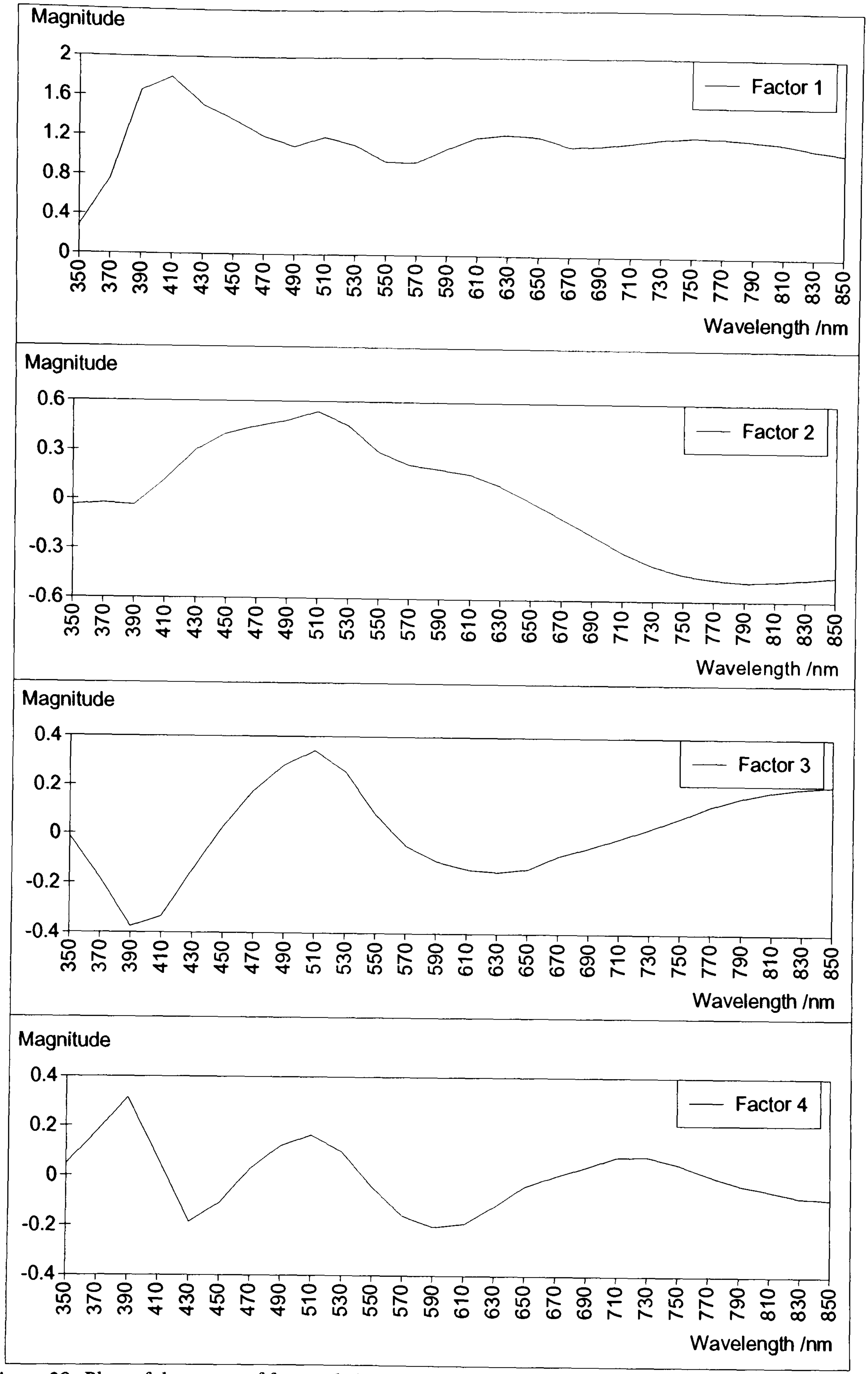


Figure 29: Plots of the scores of factors 1-4 against wavelength for the transition metal ion data set .

The vector plots of the abstract row matrix (scores) are a little easier to interpret as the wavelength scale is ordered. Figure 28 shows the spectra for the pure components of the solutions analysed; figure 29 shows the first four scores. The first factor is the spectrum found if the sample's contributions at each wavelength are averaged. The second factor shows a maximum at 510 nm corresponding to the maximum in the spectrum of Co(II), if factor 2 arises from the Co(II) component then its complementary eigenvector plot should follow the concentrations of Co(II) in the samples. Figure 30 shows the values for the second eigenvector and the concentrations after subtraction of the mean concentration. It can be seen that the fit is not perfect so factor 2 cannot describe the variance of Co(II) alone, indeed the later part of the factor shows a negative value corresponding to a negative weighting for Cu(II). This serves to highlight the fact that the PCA process finds vectors to describe the sources of variance in the data and not the underlying factors. In the case of factor 2 the direction of the factor must lie close to that of Co(II) but not along it. This point is further exemplified by the presence of peaks at 510nm in the third and fourth factors as well.

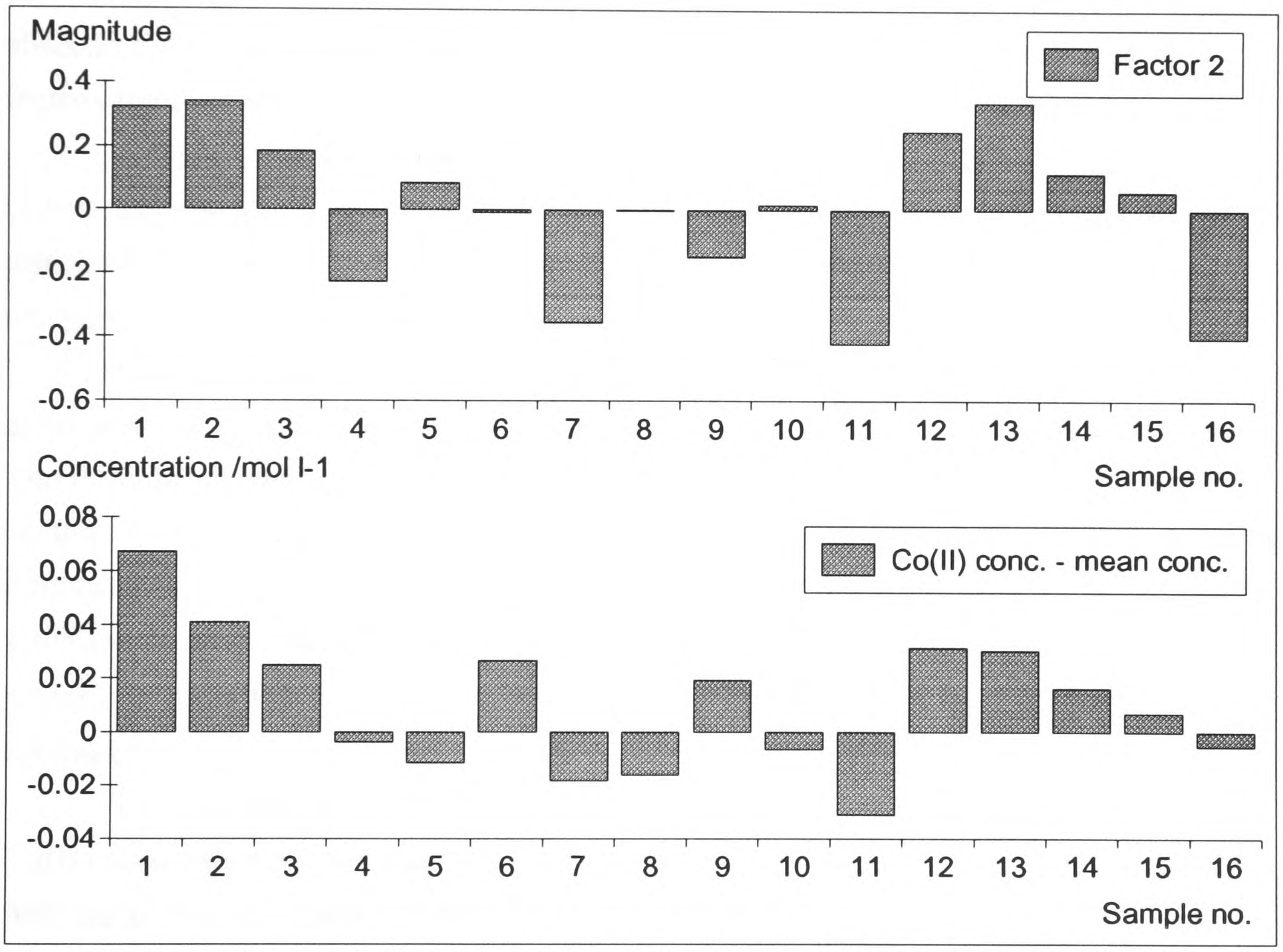


Figure 30: Plot of the second eigenvector of the transition metal ion data and concentration of Co(II) in the samples after subtraction of the mean concentration.

4.5.1.4. Target testing

Effect of rank on target testing

When target testing, the selection of the number of factors used in modelling the data is important; too few factors will always result in poor predicted vectors while too many will unnecessarily reproduce error in the data.

To investigate the effects of varying the rank of the data on target testing the Cu(II) extinction coefficients were used as a test vector and the vector target tested using from one to five factors to describe the data. The results are shown in figure 31. The test using one factor to model the data returns the average spectrum, or more accurately the first factor. This result is inevitable as using only one factor means that the data is one dimensional and can only have the values of the first factor.

The introduction of a second factor and retesting now allows the vector to align as best it can within the space defined between the two factors. The predicted vector now shows the peak at 810nm but also shows a peak at 390nm. The extra peak is at the position of the maximum for Ni(II) and consideration of the Ni(II) absorption profile (see figure 28) shows a broad peak around 710nm, i.e. underneath the Cu(II) absorbance. This similarity between the two components means that using only two factors the two components are linked and indistinguishable from each other.

Adding the third factor produces a much closer approximation to the Cu(II) test vector though a small amount of the Ni(II) can still be seen, also visible is a small peak at 510nm from the Co(II) indicating the data space still has not been spanned adequately.

Using four factors, the number of components in the system, produces a very good predicted vector with only a small deviation at the lower end of the spectrum. The deviation is due to error introduced by the sampling interval of 20nm. This occurs when a spectrum contains a peak with a sharp maximum between two sampling intervals; the digitized spectrum effectively moves the peaks towards the nearer of the sampling intervals thus distorting the spectrum and adding error to the data. There are no interferences from the other components and the data space has now been spanned adequately.

The addition of factor five does not significantly improve the shape of the Cu(II) absorption but the baseline has been slightly improved. The addition of the fifth factor has not added anything to the modelling of the Cu(II) peak, the improvement is due to the error introduced from the sampling interval digitization now being included in the model. These results show the importance of selecting the correct number of factors to describe the data and also that if the correct number is uncertain then it is better to have one too many than one too few.

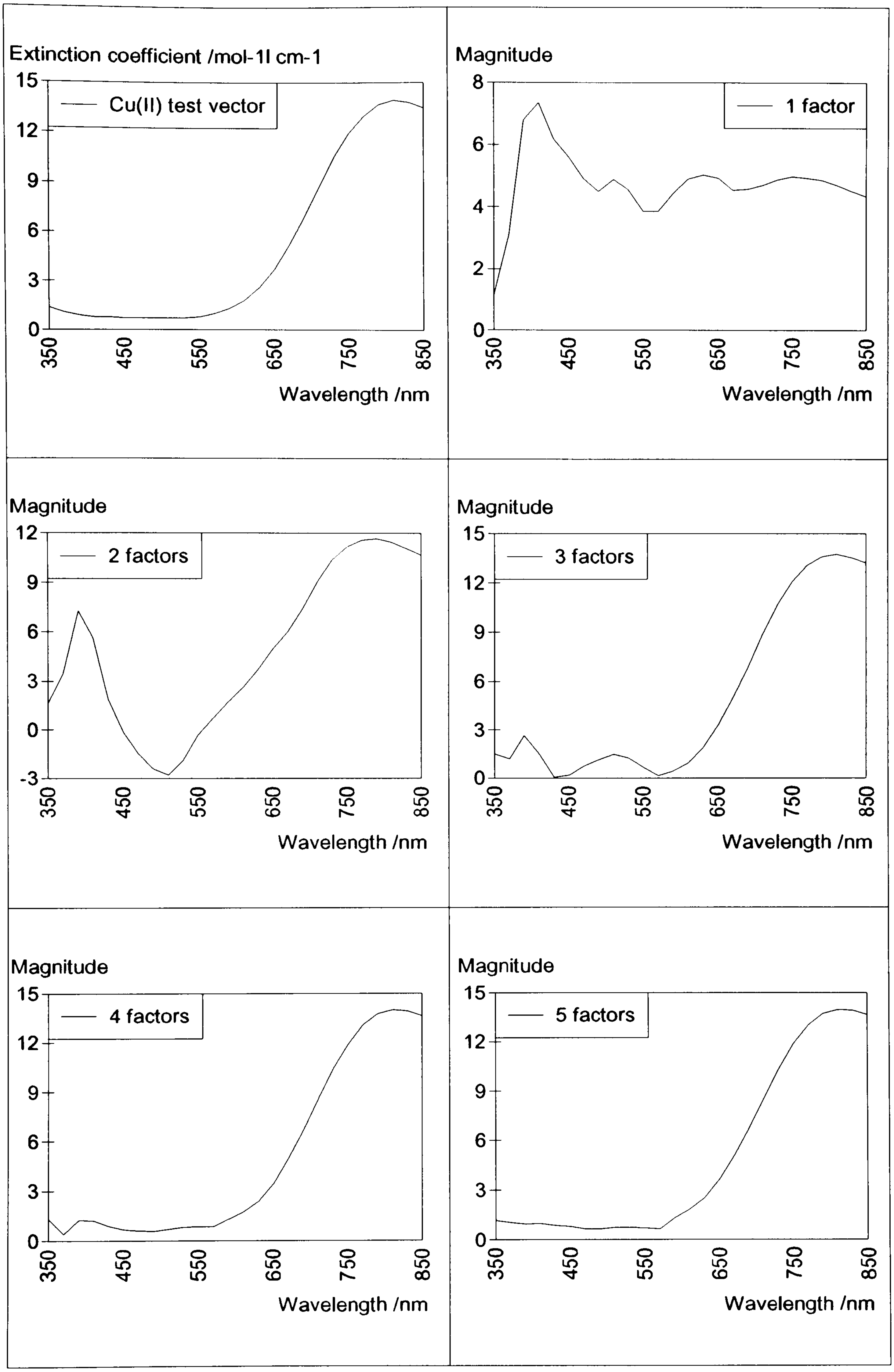


Figure 31: Plots of the extinction coefficients of Cu(II) and predicted vectors from target testing of the Cu(II) vector using from 1 to 5 factors to model the transition metal ion data.

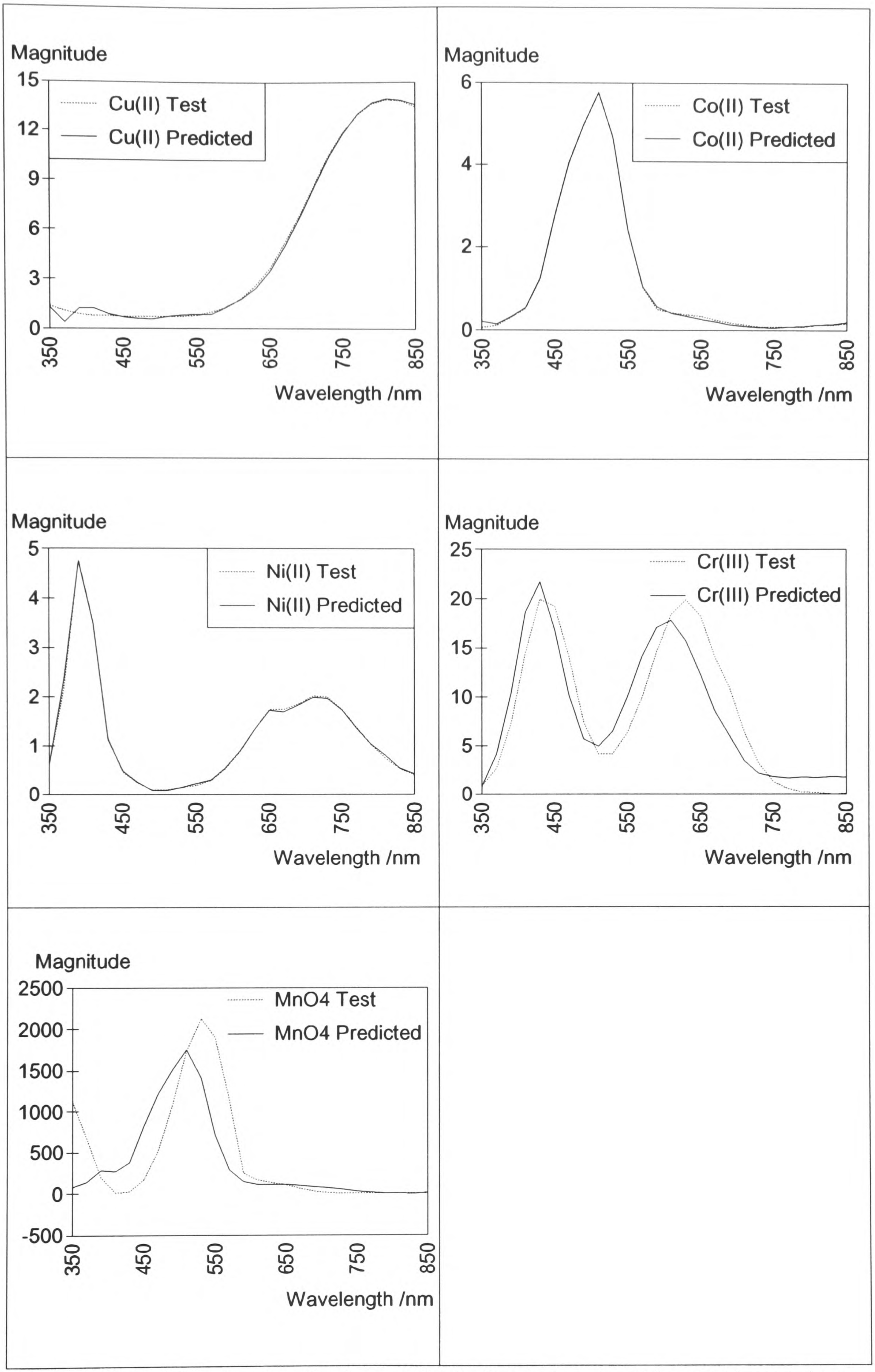


Figure 32: Plots of the test and predicted vectors for the four components of the transition metal ion data set (Cu(II), Co(II), Ni(II) and Cr(III)) and a non-component (MnO4).

True factors

All four of the known components in the transition metal ion data set and one that was known to be absent were target tested using four factors to model the data. The extinction coefficients of the pure components were used as test vectors and the results shown in figure 32 and table 30 obtained. The results of the target tests of the known components can be seen to mostly give very good predicted vectors; this is also shown by the error terms calculated for each predicted vector.

	Cu(II)	Co(II)	Ni(II)	Cr(III)	MnO4
Error est.	0.04	0.02	0.01	0.06	6.5
AET	0.20	0.04	0.04	2.99	454.25
REP	0.10	0.04	0.03	0.17	11.81
RET	0.17	0.02	0.03	2.98	454.09
SPOIL	1.74	0.40	0.88	17.81	38.46
RELI	54	104	79	6	3
%SL	12.5	89.4	65.2	1.73E-09	0

Table 30: Table of error terms for the predicted vectors shown in figure 32

The error terms for the predicted Co(II) vector are examples of a perfect result, the real error in the test vector (RET) matches the error known to be present in the test vector indicating that the vector is present in the data. The SPOIL term is within the acceptable region described by Malinowski[52], the significance level (%SL) indicates that the null hypothesis must be rejected thus the vector is a true factor and the reliability function (RELI) suggests a better than perfect vector (the greater than 100% result arises from approximations made in the derivation).

The values for Cu(II) and Ni(II) do not display the same level of perfection but still indicate that the vectors are true factors. The differences between the known error in the vectors and their respective RET terms is due to the sampling interval digitization problem discussed earlier. This inaccurate representation of the data means that the same levels of error cannot be achieved in the predicted vector. A further example of the distortion introduced by the sampling interval is that of the Cr(III) test vector, the calculated error terms suggest that the vector is not a true factor but consideration of the plotted vectors shows a good fit but with a shift towards the higher wavelengths. This is due to the peak shape being distorted by the sampling interval being too wide and having the effect of shifting the spectrum in wavelength.

Bad targets

The test vector for permanganate (MnO4) shows clearly that the data cannot be made to fit the test vector, the best that can be achieved appears to include mostly the Co(II) contribution to the spectrum. The error terms also reflect the poor fit of the predicted vector.

The fact that a reasonable approximation to the Co(II) spectrum is produced from testing for MnO4 shows how iterative target testing may be used to find true factors in the data. The predicted vector produced from the test can be re-presented as a test vector that is hopefully a better approximation to the true factor and an even better approximation produced. This process can then be repeated until converged on the true factor.

4.6. Pyrolysis - mass spectrometry

4.6.1. Single component data

The mass spectra of the three substituted ferrocenes used in this investigation are given on the following page. Where possible the background subtracted spectra have been used and in all cases the spectrum shown is from the scan corresponding to the maximum in the TIC pyrogram. All spectra have been normalized to 100% base peak.

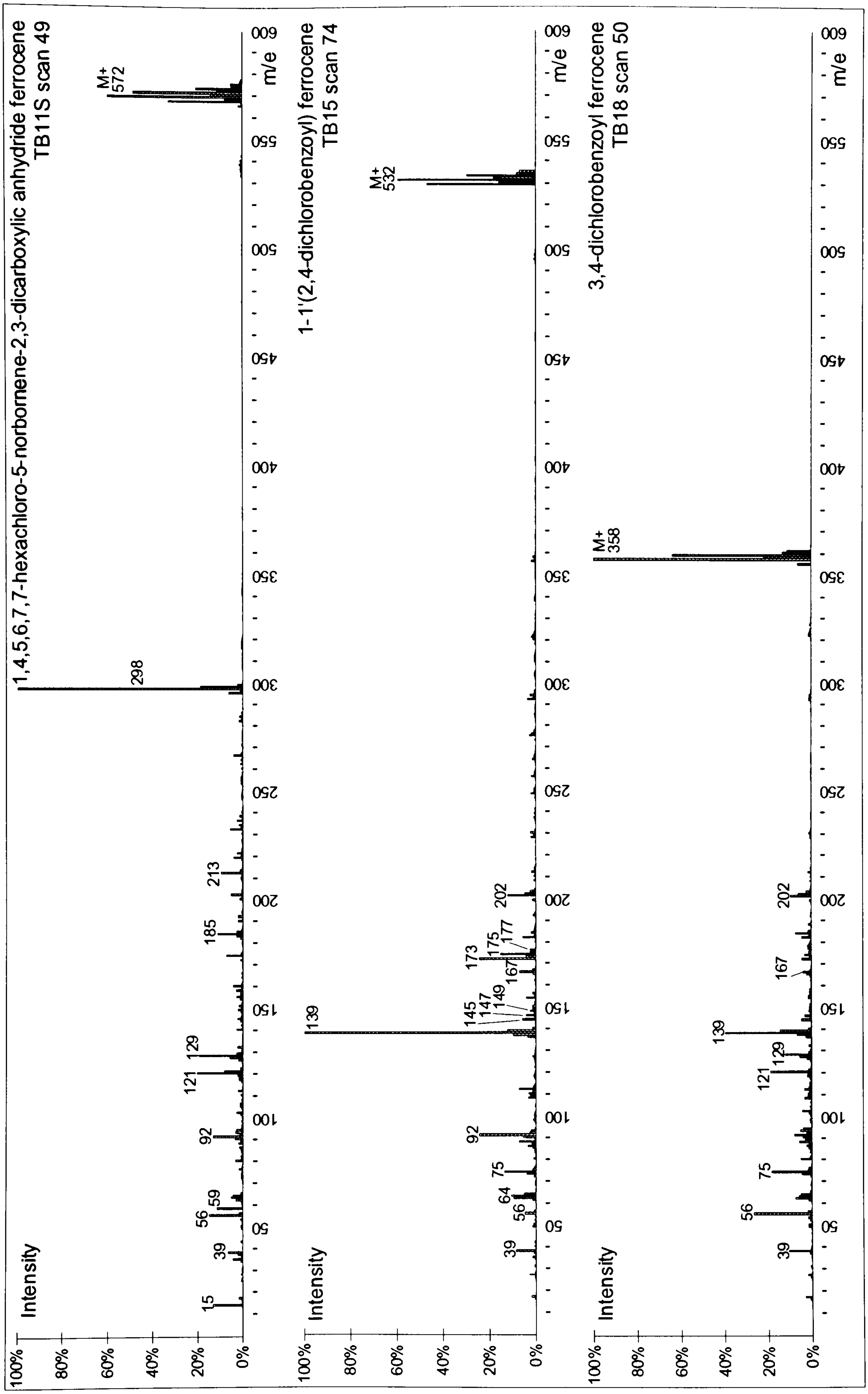


Figure 33: Normalized mass spectra of FCA, 1,1'DCBF and 3,4DCBF.

4.6.1.1. 1,4,5,6,7,7-hexachloro-5-norbornene-2,3-dicarboxylic anhydride ferrocene (FCA)

Three samples of the FCA were subjected to temperature programmed pyrolysis mass spectrometric investigation as detailed in the experimental section.

The TIC pyrograms for each of the analyses are shown in figure 34, it can be seen that runs TB11 and TB12 both produced similarly shaped pyrograms and run TB13 shows a much sharper and smaller area peak, this is due to a smaller sample size than in the other two runs.

The TIC_{max} were at scans 49,51 & 48 for TB11,12 and 13 respectively, a spread of nearly 20 seconds. Given that the probe ramp is started manually, a spread of 2 or 3 seconds is possible, as the spread is larger than this then another factor must be responsible. This may be due to thermal gradients in the probe brought about by sample shape affecting the contact area between the tube and the sample, or by differences in the sizes of the tubes affecting their contact between the probe tip and the tube and thereby influencing transfer of heat from the probe to the tube.

This spread of peak maxima imposes a severe limit on the usefulness of the pyrograms in identifying a component and highlights the difficulties that exist when attempting to perform reproducible analyses on the same sample.

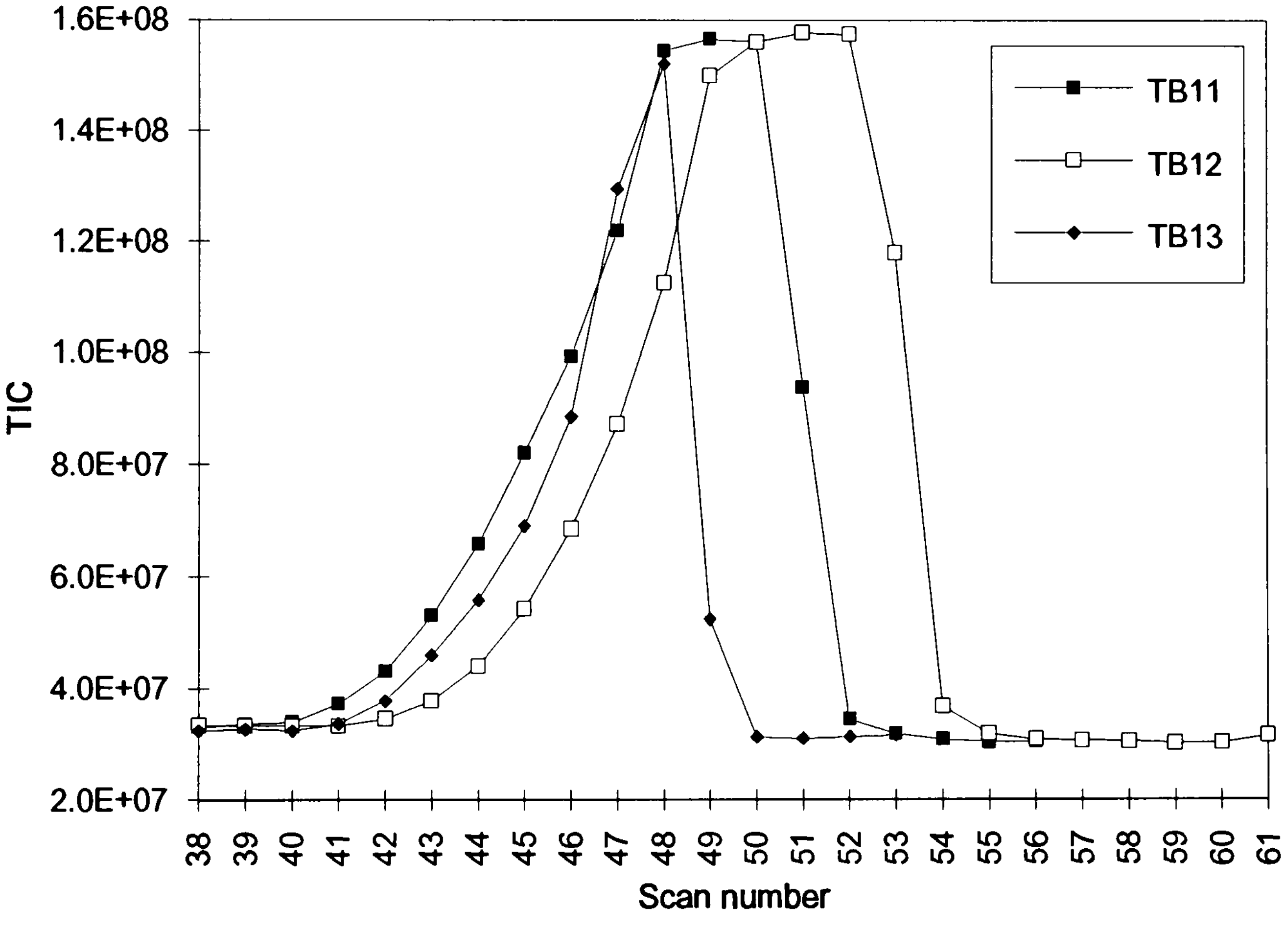


Figure 34: TIC pyrograms for data sets TB11, TB12 and TB13.

Data pre-treatment

The data sets were converted to EPA format using the mass spectrometer data system and in the process reduced to contain only scans 17 to 91. A further three data sets were created by subtracting the third scan from every scan in each data set. For the data set TB12 it was found impossible to convert to EPA format after subtraction, believed to be due to negative peaks being created in the file as the probe was already inserted into the spectrometer and a small amount of the sample was detectable. To overcome this problem the third scan of data set TB11 was subtracted instead and all three subtracted files converted to EPA format. The background subtracted files are referred to by appending an S to the name of the data set, i.e. TB11S, TB12S and TB13S.

It was attempted to analyse the EPA files as produced from the mass spectrometer data system but it was found that the size of the data sets was such that the program was unable to keep all the necessary matrices in memory. In an attempt to overcome this problem the program was modified to allow matrices to be written to disk for intermediate storage. It was found however that even with these modifications insufficient memory remained to allow the decomposition to complete.

The data sets from the mass spectrometer were therefore truncated in the number of scans that they contained as shown in table 31. Two of the data sets were truncated using the mass spectrometer data system and reconverted to EPA format, the third was shortened using a spreadsheet.

Original file	Scan range	Reduced range	Truncation method
TB11	17-91	38-56	by MS data system
TB12	17-91	38-61	by spreadsheet
TB13	17-91	38-53	by MS data system

Table 31: Scans included in data files TB11, TB12 and TB13 for analysis.

The smaller data sets reduced the memory overhead sufficiently to allow the program to complete the analysis calculations without error, this memory shortage prompted the creation of the second data structure to enable the data to be stored in EMS and allow the program to deal with much larger data sets.

Data analysis

Each data file and its subtracted version were analysed, producing a total of six sets of results. Program parameters were set as follows,

	TB11	TB11S	TB12	TB12S	TB13	TB13S
Error est.[a]	246510	207200	242070	208890	236300	178410
Accuracy[b]	1E-19	1E-19	1E-18[1]	1E-19	1E-19	1E-19
Minimum[c]	13296	730	16625	1104	13578	968
Max intensity[d]	24651000	20720000	24207000	20889000	23630000	17841000
@[e]	m/e 28, scan 41	m/e 298, scan 50	m/e 28, scan 44	m/e 298, scan 52	m/e 28, scan 39	m/e 298, scan 48
Min. base peak[f]	13283000	730000	16608000	1103000	15364000	967000

[1] Accuracy reduced because factor 3 wouldn't converge at the normal limit of accuracy.
[a] Estimate of error in the data set, based on 1% of largest peak in data set.
[b] Accuracy of factor extraction test - see Optimization of PCA for details.
[c] Minimum possible data value, set to smallest base peak / 999 - see Appendix 4.
[d] Largest peak in data set.
[e] m/e ratio and scan number of largest peak.
[f] Smallest base peak of all of the scans in the data set.

Table 32: Table of operating parameters for the analysis of data sets TB11, TB11S, TB12, TB12S, TB13 and TB13S.

Table 33 lists the total number of factors found for each data set. It can be seen that the subtraction of the background from each scan has not reduced the number of factors necessary to describe the data. The number of factors found for each data set is equal to the number of scans in each data set; this is the maximum number of factors possible for each data set. As decomposition did not stop earlier then either all of the factors are important to the data or the estimate for the residual test was too low.

Data file	Untreated data	Subtracted data	No. of scans in data set
1TB11	19	19	19
1TB12	24	24	24
1TB13	16	16	16

Table 33: Number of factors found in the data sets TB11, TB11S, TB12, TB12S, TB13 and TB13S.

As the value for the residual test was set at the smallest possible value measurable by the data system then it would be expected that the program would attempt to model all the data including noise from the data system and the instrument. Increasing the value for the residual test would reduce the number of factors found but a valid method for producing an accurate estimate of the noise in the system must be found first to avoid loosing real factors from the data.

Determination of the rank of the data

The amount of variance accounted for by each factor gives an indication of the number of significant factors in the data. Table 34 shows clearly that the first two factors account for nearly all of the variance in the data and that in the background subtracted data the importance of the second factor is reduced considerably. This evidence leads to the conclusion that the background is an important factor in the data and that its importance to the data may be reduced considerably by background subtraction.

	Data set					
Factor	TB11	TB11S	TB12	TB12S	TB13	TB13S
1	77.66	98.54	77.08	98.13	88.23	98.53
2	22.03	1.01	22.58	1.09	11.70	1.07
3	0.28	0.35	0.30	0.69	0.04	0.25
4	0.02	0.06	0.02	0.05	0.01	0.08
5	0.01	0.03	0.01	0.03	0.01	0.03
6	3.05E-03	0.01	2.35E-03	0.01	3.97E-03	0.02
7	1.16E-03	3.05E-03	1.05E-03	3.01E-03	1.44E-03	0.01
8	9.24E-04	2.46E-03	8.09E-04	1.83E-03	8.43E-04	4.38E-03
9	6.60E-04	1.44E-03	5.07E-04	1.64E-03	6.07E-04	3.13E-03
10	4.41E-04	1.11E-03	4.21E-04	1.39E-03	3.13E-04	1.59E-03
11	3.25E-04	7.11E-04	4.02E-04	1.06E-03	2.57E-04	8.97E-04
12	2.36E-04	6.69E-04	3.06E-04	7.72E-04	1.79E-04	5.19E-04
13	1.91E-04	3.14E-04	1.92E-04	5.25E-04	8.03E-05	2.37E-04
14	1.43E-04	2.17E-04	1.53E-04	2.58E-04	3.83E-05	8.07E-05
15	7.73E-05	1.82E-04	1.40E-04	2.25E-04	3.05E-05	6.65E-05
16	6.92E-05	1.50E-04	6.53E-05	1.74E-04	7.04E-06	2.61E-05
17	2.73E-05	6.55E-05	3.10E-05	9.58E-05		
18	2.52E-05	5.90E-05	2.21E-05	5.39E-05		
19	1.29E-05	7.08E-06	1.39E-05	2.82E-05		
20			1.13E-05	1.80E-05		
21			6.43E-06	1.04E-05		
22			4.66E-06	7.25E-06		
23			1.28E-06	2.55E-06		
24			5.12E-07	3.95E-07		

Table 34: Table of percentage variances accounted for by each factor in data sets TB11, TB11S, TB12, TB12S, TB13 and TB13S.

To consider further the number of factors that comprise the data it is necessary to look at the error terms calculated during the decomposition stage.

Figure 35 shows the eigenvalues for the first six factors of the TB11 data set along with the calculated value for standard error in eigenvalue (SEE). It can be seen that the first three eigenvalues are identified as true factors and that there is a distinct change in gradient in the eigenvalue plot from factor four onwards which adds support to this conclusion.

Figure 36 shows a plot for the first six factors of the real error (RE) and the estimated error in the untreated data. The real error shows a large fall from the first to the second factor, as suggested by the percentage variance data. The real error crosses the estimated error after only one factor indicating only one factor in the data, this appears to be in conflict with the results from the eigenvalues but as the estimate was based upon the observed variation in the background, it should mark the change from describing factors larger than the background and smaller factors.

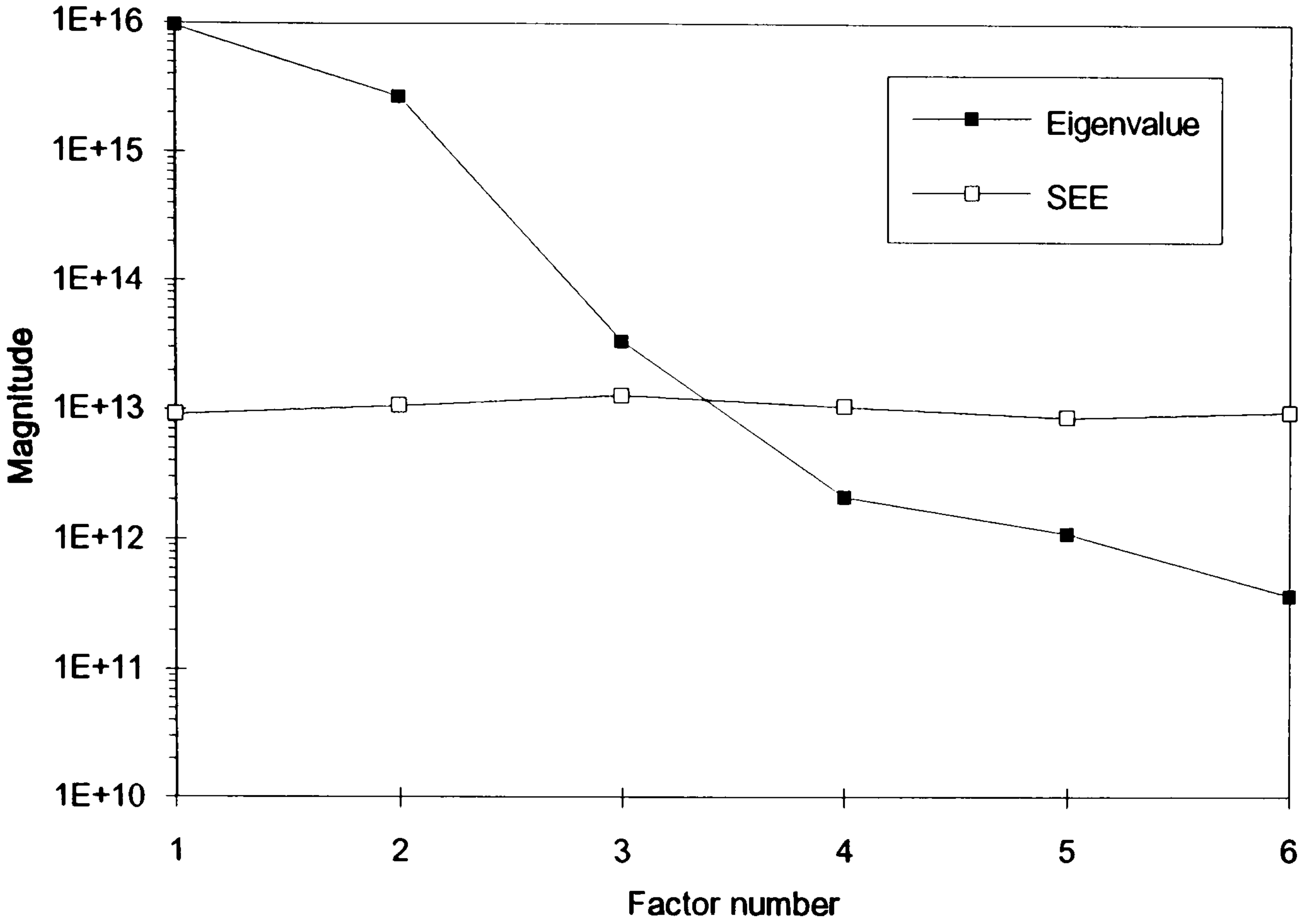


Figure 35: Plot of eigenvalue and standard error in eigenvalue (SEE) for data set TB11.

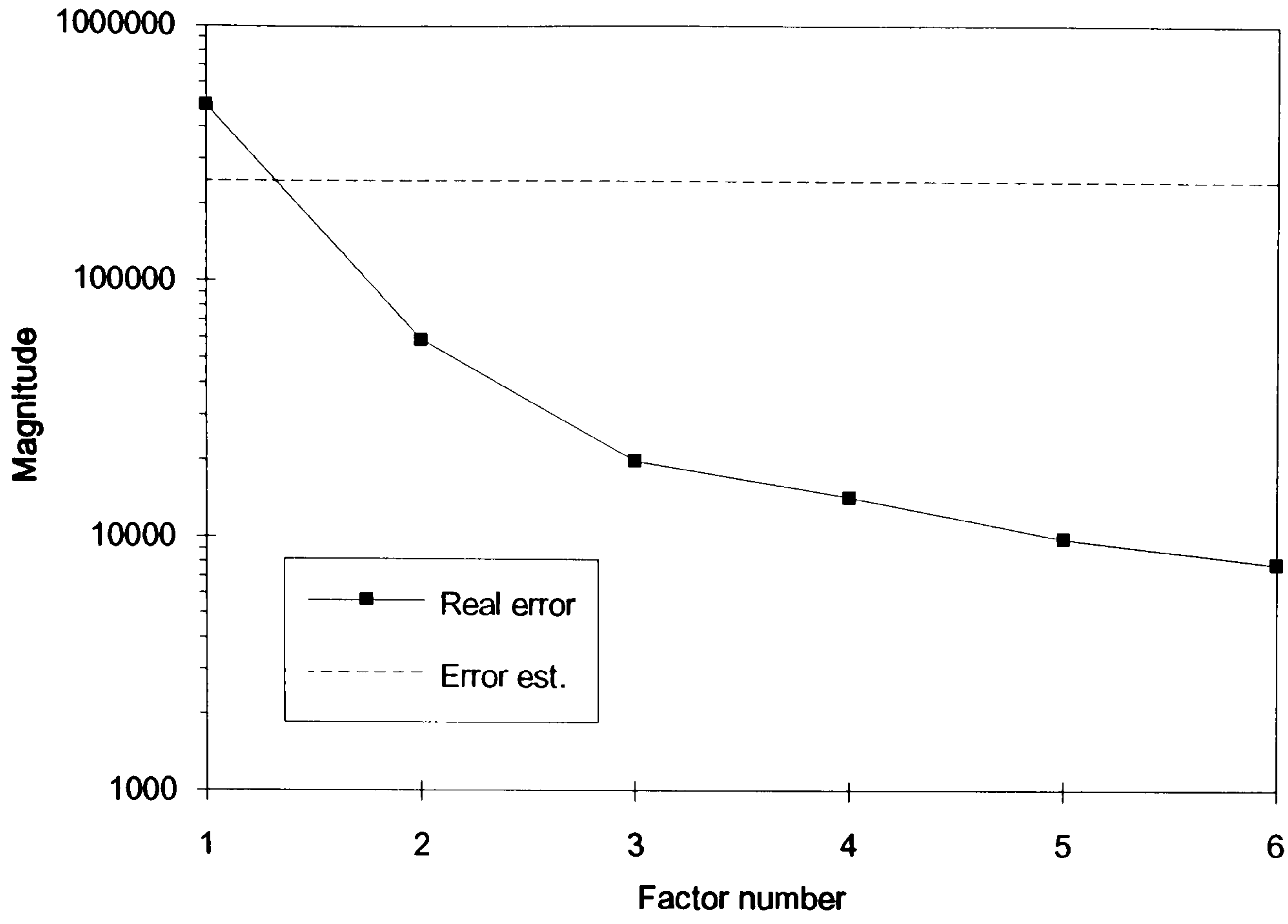


Figure 36: Plot of real error and estimated error for data set TB11.

Unfortunately as the estimate of error in the data is apparently too large then the values calculated for SEE, which use the error estimate, must be regarded as suspect. When the values for SEE are recalculated using the RE estimate of error in the data set for six factors (see indicator function estimate of rank) the new SEE values also predict six primary factors.

The error terms that rely upon no prior knowledge of the error in the data may aid in determining the dimensionality of the factor space. Figure 37 shows that the imbedded error value falls sharply to factor two but continues to fall with further factors. Two changes of gradient are visible, one at three factors and another at six factors. The theory of imbedded error suggests that it should reach a minimum at the number of factors responsible for the data, this event does not occur with this data implying that the correct number of factors responsible for the data have not been extracted or that the function does not work for this type of data. In cases like this, the change of gradient may mark the change from primary to secondary factors, but such interpretation must be performed with caution and is best used as a confirmation of estimates from other sources. Further information imparted by the imbedded error plot is whether the data is factor analysable or not, non-factor analysable data produces an increasing imbedded error function from the first factor onwards[80], so this suggests that the data is produced from the predicted linear combination of factors and is therefore amenable to factor analysis.

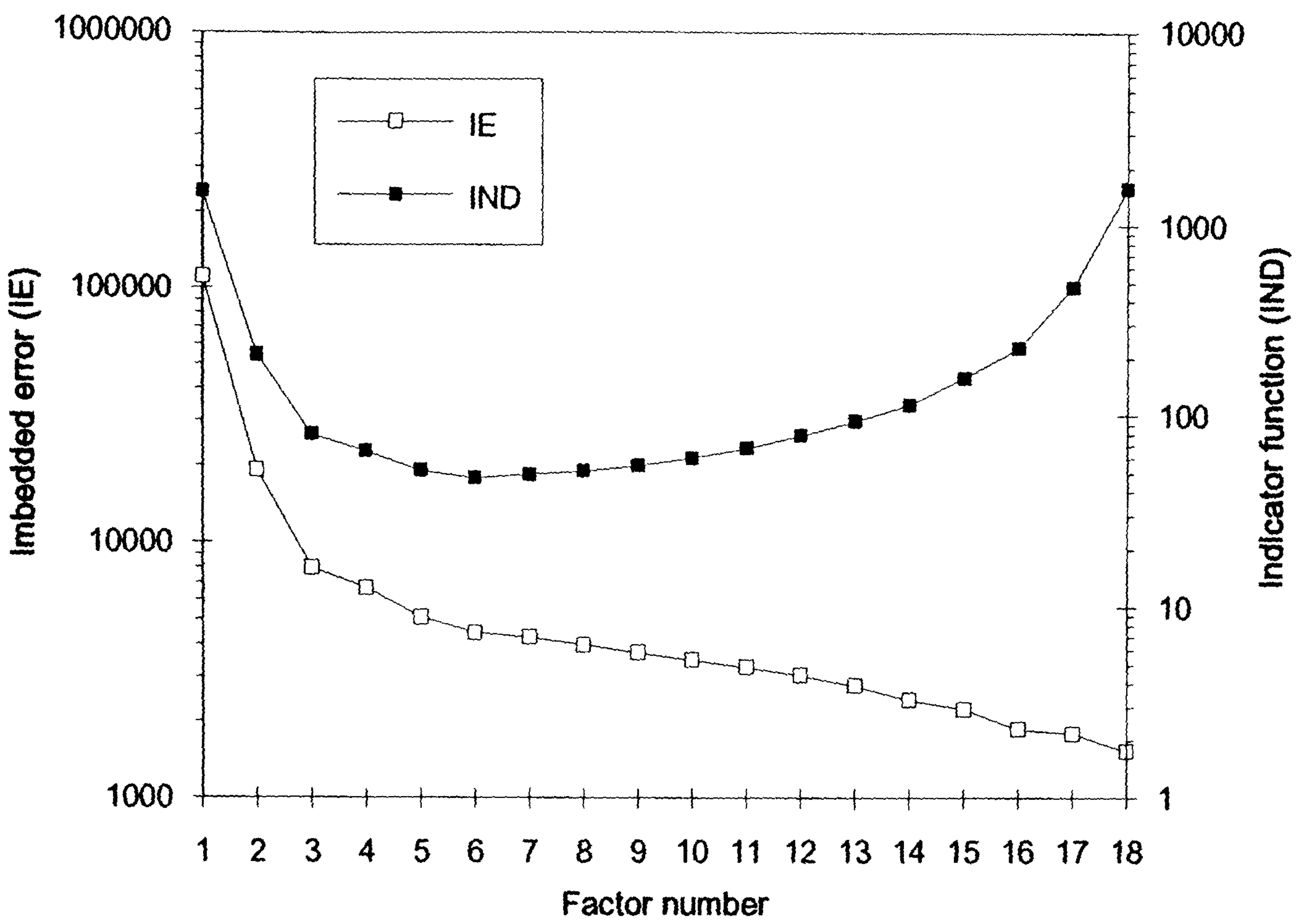


Figure 37: Plot of imbedded error (IE) and indicator function (IND) for the data set TB11.

The indicator function does show a minimum at factor six and a sharp gradient change at factor three. The indicator function is an empirical function derived from the imbedded error and is generally more sensitive to the minimum value.

The final piece of information of use in determining the dimensionality of the data is the F-test. The work performed previously by Malinowski[12] led him to the conclusion that the significance level for the change between primary and secondary eigenvalues is between 10% and 5%. For TB11 the seventh factor has a significance level of 15% and the sixth factor a significance level of 5% thus indicating 6 primary factors in agreement with the indicator function.

Abstract column matrix

The nature of operation of factor analysis is such that the first factor found by the decomposition is an average of the data set. Because of the way the data was presented to the program, the eigenvectors are related to pyrograms of the data and the scores are related to the mass spectra; the plot of the first eigenvector is therefore the average pyrogram. As only one component was present in the system the average pyrogram is expected to resemble the TIC pyrogram.

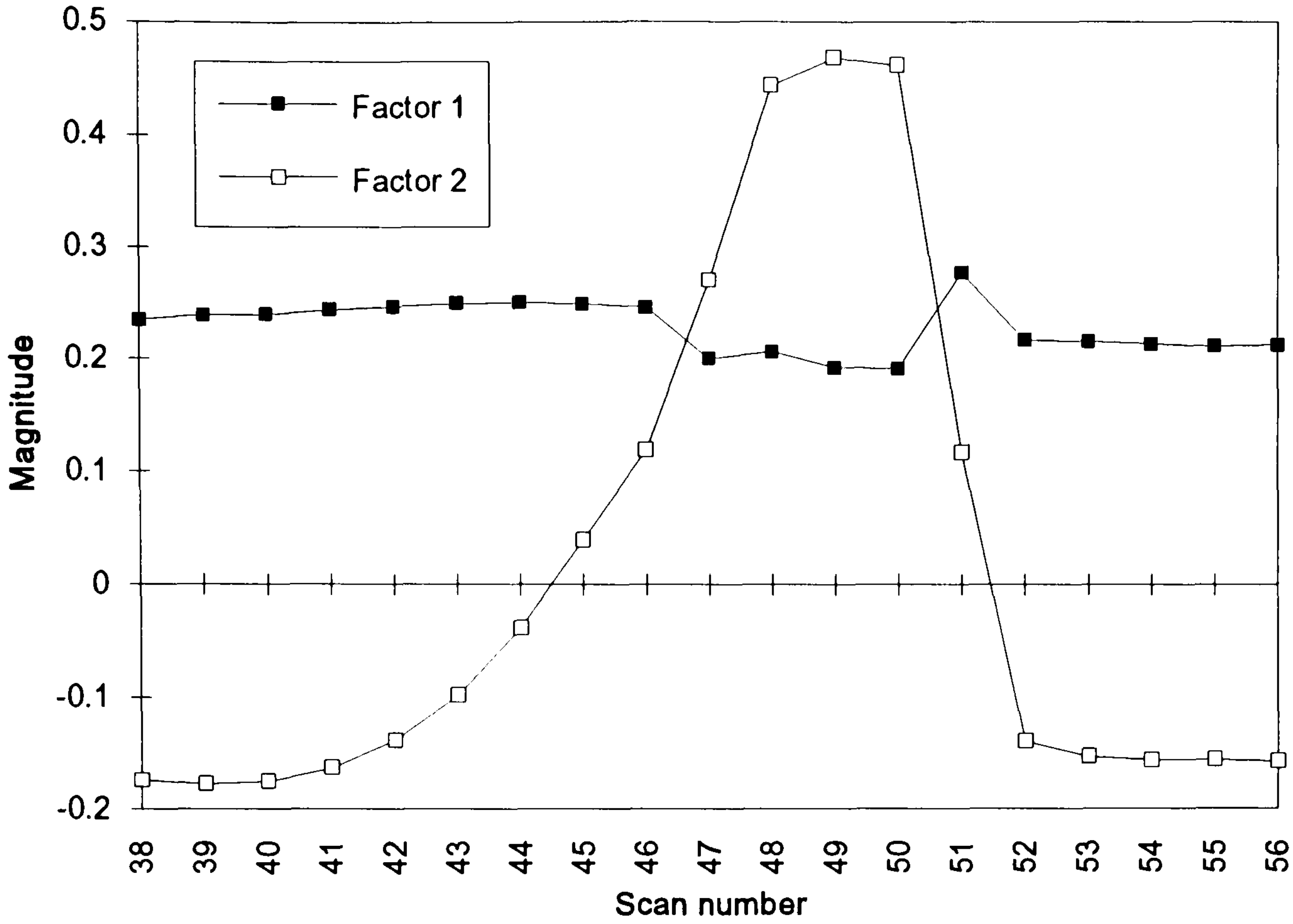


Figure 38: Plot of factors 1 and 2 for the data set TB11.

Figure 38 shows the first two factors plotted against scan number, it is immediately obvious that the expected pyrogram is in fact factor two and factor one is nearly flat. This means that the most dominant factor in the mass spectrum is the background and not, as expected, the sample peak. The dominance of the background is assured by its presence in every scan and not by its magnitude. The shape of factor two is quite clearly the same shape as the TIC pyrogram and follows it almost exactly; this factor is related to the sample.

This leaves us with an interpretation of the two most important factors in the data accounting for 99.7% of the variance in the data set. The error produced in measurements from the mass spectrometer has been estimated to be as large as 1%, therefore further factors may not contain significant information, the error term rank estimate is six factors suggesting that the smaller factors do describe real factors in the data.

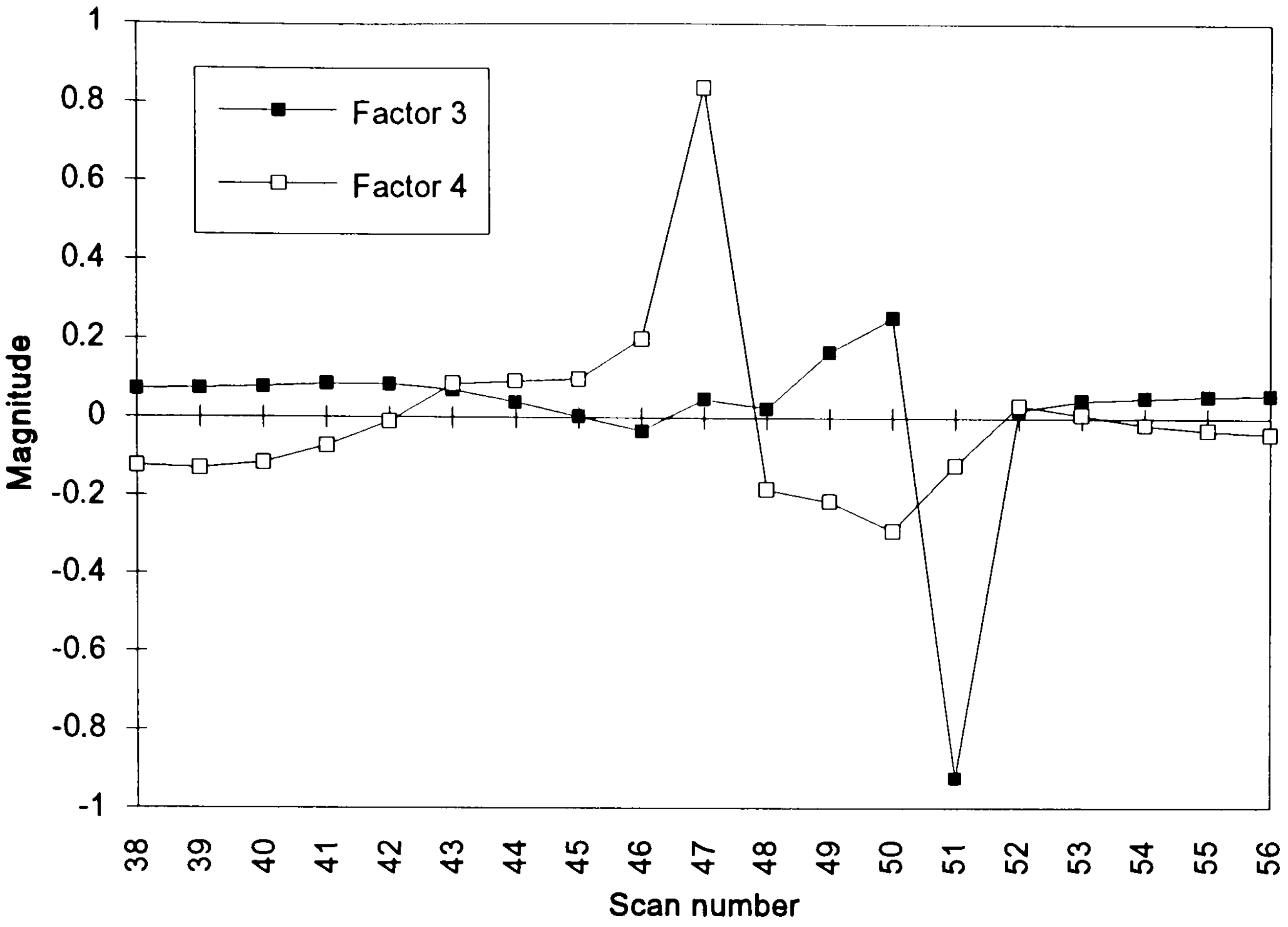


Figure 39: Plot of factors 3 and 4 for the data set TB11.

Figure 39 shows the eigenvectors for factors 3 and 4, which bring the total variance accounted for to 99.99%. These plots are more difficult to interpret on their own and little can be said about them without further information. What is clear is that both factors are effectively constant and near-zero either side of the sample peak, which indicates that they are related to the sample and not to the background and that factor 3 is associated with the end of the sample peak and factor 4 the beginning. This means that the sample has given rise to more than the one factor expected. Factors 5 and 6 (not shown) exhibit a more complex shape than factors 3 and 4 but still appear to be related to the sample. This shows that the decomposition of even a single pure compound can be complex producing more than one factor.

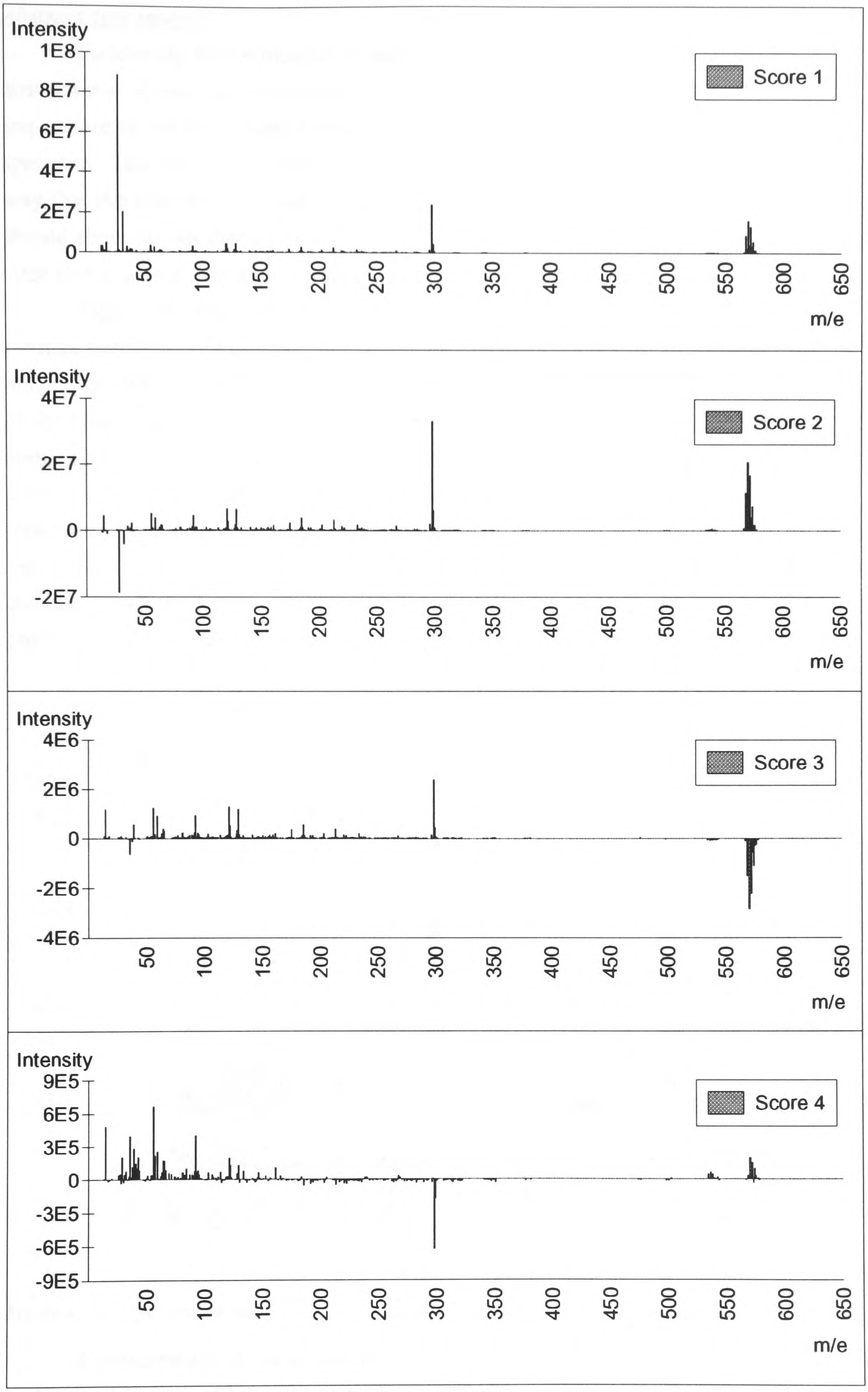


Figure 40: Plots of factor scores 1-4 for the data set TB11.

Abstract row matrix

To identify the sources of variation giving rise to the various factors the abstract row matrix can be studied. The scores contained in this matrix show the importance of each m/e ratio to the factors and when plotted appear as a mass spectrum. The first factor scores will display the average mass spectrum in the same way that the first factor showed the average pyrogram. The second factor scores should show the sample spectrum as its eigenvector shows the TIC pyrogram. The later factor scores may give an insight into the cause of their factors.

Figure 40 shows the factor scores for the first four factors, score 1 shows the average spectrum as expected. The dominance of the background is shown by the magnitude of the peaks at m/e 28 and 32 (N_2 and O_2) and also visible is the average of the sample spectra at a much reduced magnitude compared to the background. Score 2 shows the spectrum of the ferrocene derivative on the positive side of the axis and the peaks m/e 28 and 32 on the negative side. This factor can be thought of a separating the background from the sample signal. The completeness of this separation can be seen in figure 41 where the positive portion of the score spectrum is compared with the spectrum at TIC maximum from the background subtracted data, providing an extremely good match.

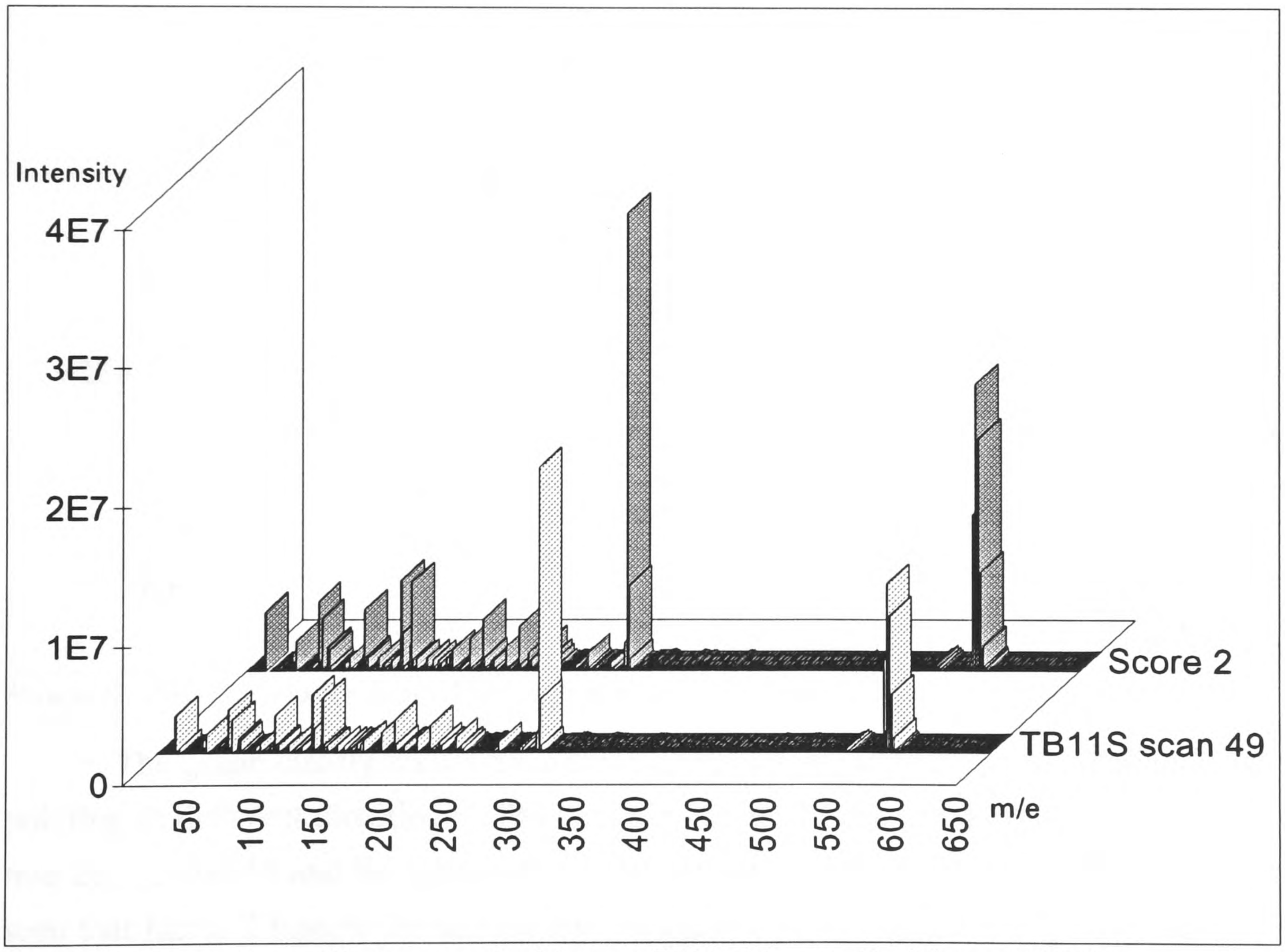


Figure 41: Comparison of factor score 2 of data set TB11 with scan 49 of data set TB11S.

Consideration of the spectrum for score 3 shows negative values for the molecular ion peak at m/e 570 and also for m/e 36 and 38 whilst the base peak at m/e

298 and the rest of the fragmentation peaks have positive values. The nature of factor analysis means that the first factor produces an average of the data, the second will divide the two major sources of variance (sample and background) and the third will divide any variance left unaccounted into two sources and so on until all the data is described uniquely. The background is relatively unchanging and as such will probably be adequately described by the first factor. The sample is separated from the background very effectively by the second factor, so the third factor must describe the two major sources of variance in the sample signal. This implies that the molecular ion peak and the m/e 36 and 38 peaks vary differently to the base peak.

As an aid to the visualization of the different sources of variance in the data, scores may be plotted against each other to give a 2D or 3D spatial representation of the factors. Figure 42 shows the 2D graph obtained from plotting the scores for factors 2 and 3.

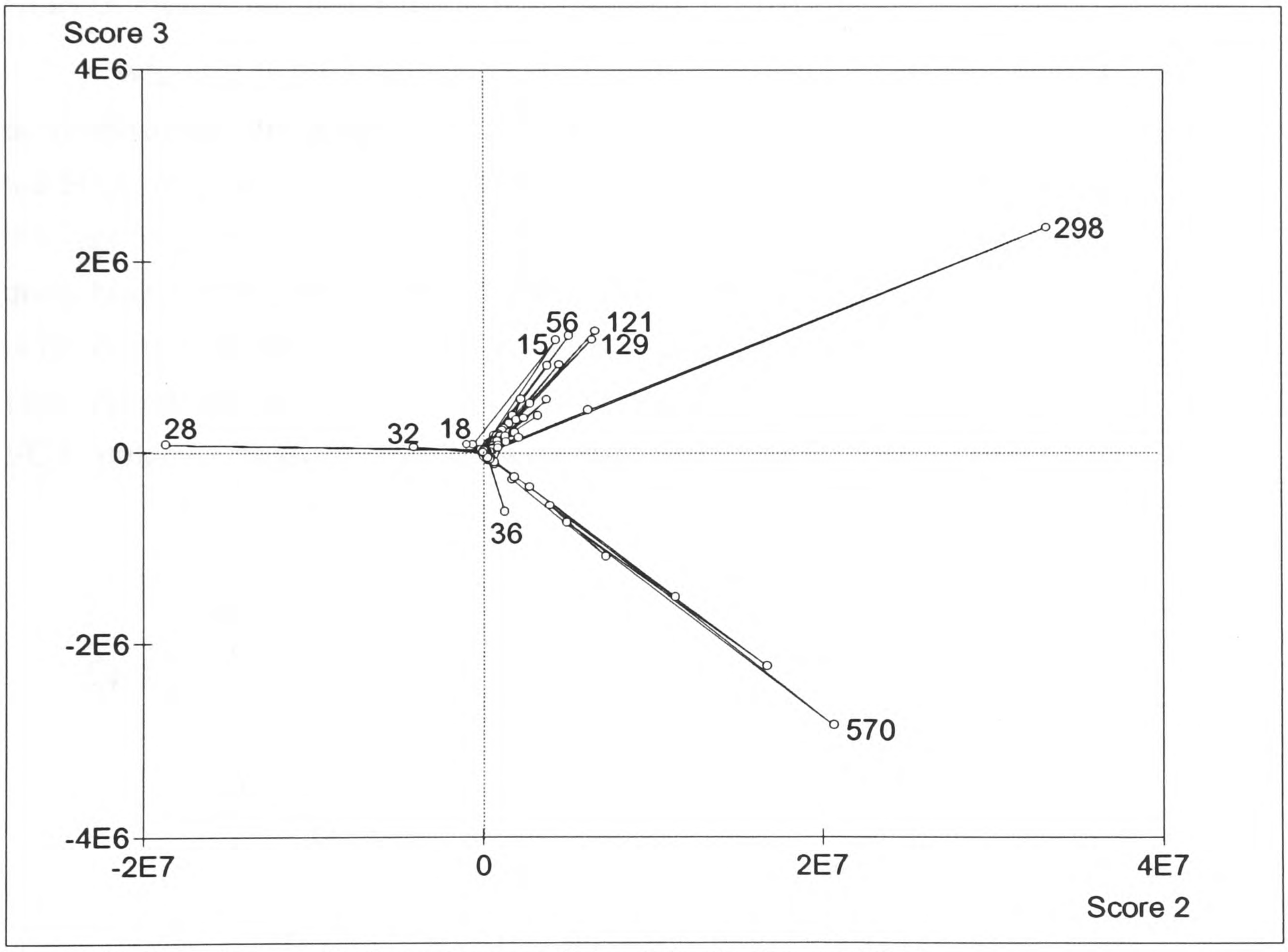


Figure 42: Plot of scores for factor 2 versus factor 3 for the data set TB11.

The graph clearly shows the different sources of variation in the data as lines pointing in different directions. The axis due to the background is visible containing m/e 28, 32 and 18 and lies perpendicular to the axis defined by factor 2. It can be seen that factor 2 bisects the sample and background information thus giving the negative background and positive sample spectrum seen in the scores for factor 2. The differing variances of the molecular ion (m/e 570) and the base peak (m/e 298) is also obvious. Rather more interesting is the m/e 36 value, which is close to but not completely aligned with the m/e 570 source of variance. Most of the other fragment peaks align closely but not exactly with the m/e 298 peak suggesting that their production is closely linked with the variance of the base peak but is not exactly the same.

Ion source chemistry

To explain the sources of the variance it is necessary to look at the chemistry occurring in the mass spectrometer. The following reactions are likely to occur in the probe to provide gaseous species that will enter the source and be ionized.

$$R_{(s)} \longrightarrow R_{(g)}$$
$$R_{(s)} \longrightarrow P_{(s)} \longrightarrow P_{(g)}$$
$$R_{(s)} \longrightarrow R_{(g)} \longrightarrow P_{(g)}$$

Figure43: Possible reactions occurring at the probe tip.

Where R is the reactant (FCA) and P is the product of some reaction (i.e. decomposition). In order for the m/e 570 molecular ion to appear then vaporization of the FCA must occur. The gas phase reactions will be limited by the residence time of the vapour in the probe tip, once leaving the tip the temperature will fall, effectively quenching any reaction. The residence time will be affected by the pumping rate and is believed to be short as evidenced by the sharp fall of the peak in the TIC pyrogram. Decomposition may occur from the solid or the gas phase and a consideration of the FCA molecule produces the following likely decomposition mechanism.

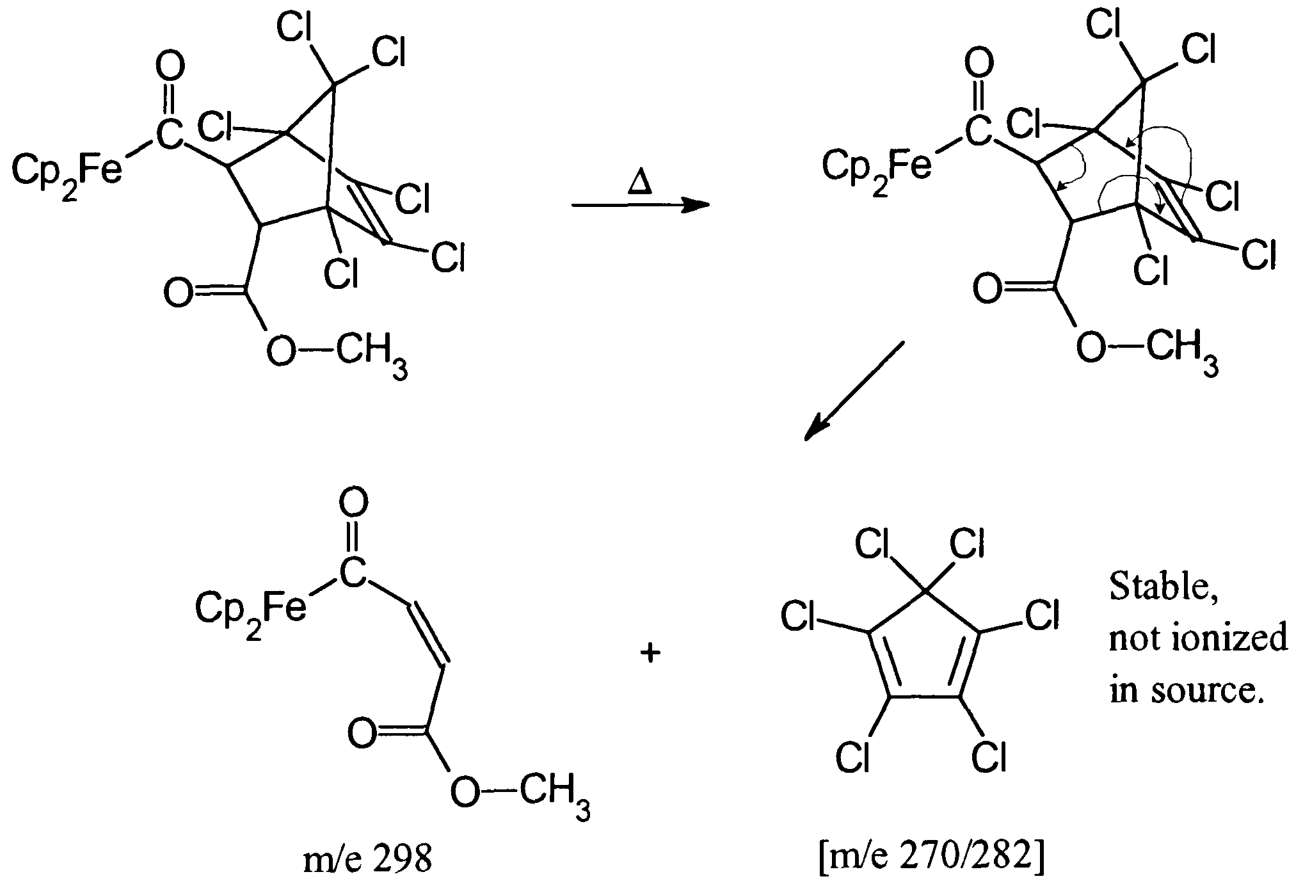


Figure 44: Proposed decomposition pathway for FCA (Cp = cyclopentadiene).

This decomposition would give rise to the base peak observed in the spectrum and fragmentation of the m/e 298 ion results in the lower mass peaks observed. These two possible sources can explain why the m/e 570 and m/e 298 signals are of different variance and possibly their opposite sign, but it does not provide a rational explanation for why all the lower mass peaks have positive weightings apart from m/e 36 and m/e 38 (HCl).

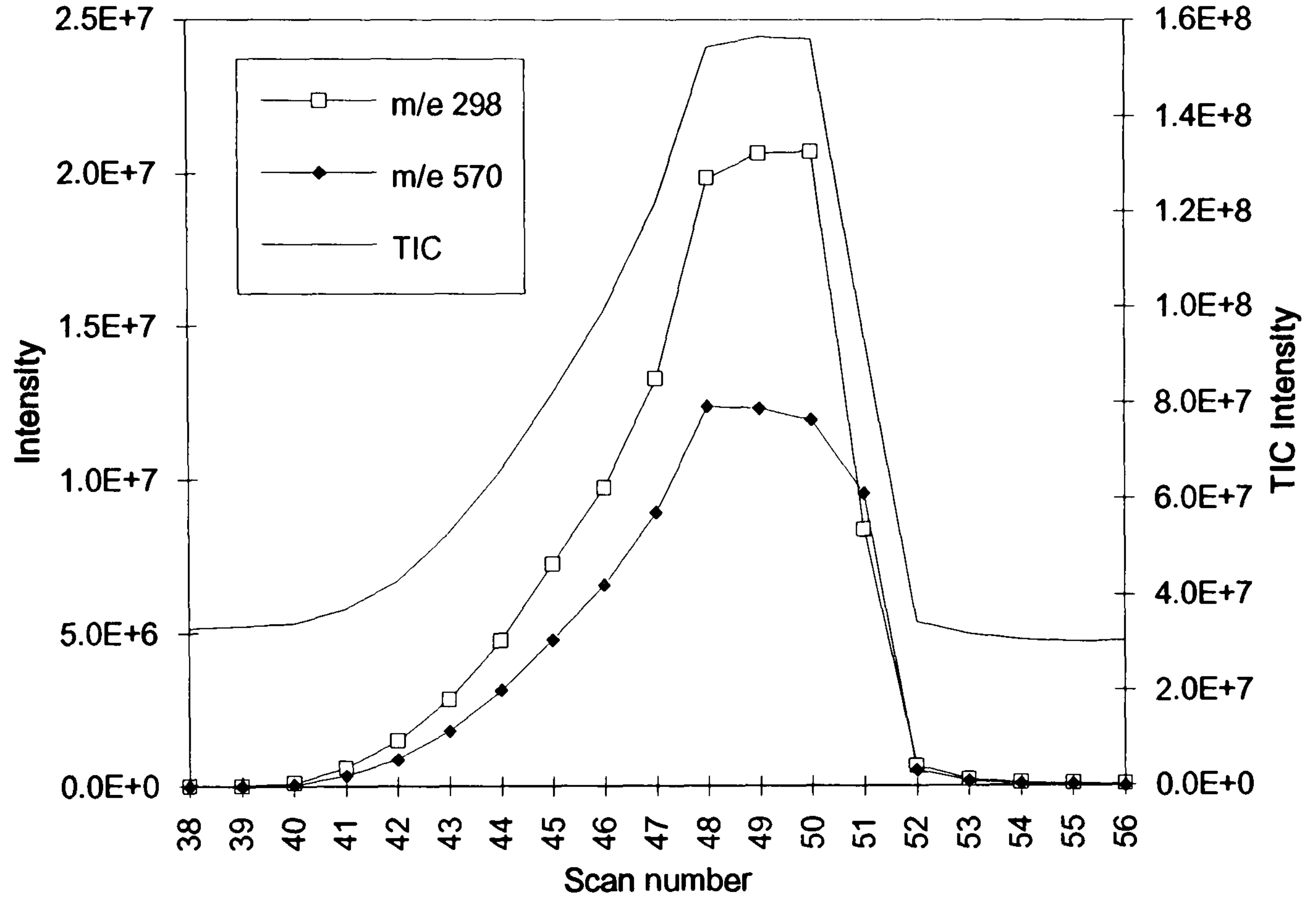


Figure 45: Single ion pyrograms for m/e 298 and m/e 570 for data set TB11.

To understand the causes of these other variances it is necessary to look at the single ion pyrograms of some of the m/e values involved. Figure 45 shows the single ion pyrograms for m/e 570 and m/e 298. It can be seen that the signal for m/e 298 rises faster than that for m/e 570 but otherwise mirrors the shape of the TIC until scan 51 when the fall for m/e 298 is much greater than for m/e 570. If we assume that the sharp fall in TIC corresponds to the solid sample being used up then there is no chemical reason why m/e 570 should fall less quickly. The difference is explained as being due to bias in the measurement of the mass spectrum. The spectrum is recorded as an exponential down scan and takes around 6.7 seconds per scan. When scan 51 was started the concentrations of all components were falling rapidly, m/e 570 was measured first, as the scan passed down to m/e 298 the concentrations fell further until the m/e 298 value is measured, the concentrations were still falling as the scan passed on to lower m/e numbers recording still lower concentrations. By this method the bias arises in the scan giving an artificial intensity distortion biased towards the higher masses.

In terms of variances this can be seen as the separate directions for m/e 570, m/e 298 and the lower mass fragments in figure 42, the bias has the effect of a rotation anticlockwise about the origin when viewed in this format. The factor analysis process maximizes the variance on the factor and so factor three bisects the variance caused by bias thus producing the negative weighting for m/e 570 and the positive weighting for the other m/e values. Unfortunately bias in scan 51 does not explain the negative weights for m/e 36 and m/e 38 (HCl). Consideration of the single ion pyrogram for m/e 36 given in figure 46 shows that the evolution of HCl rises with the TIC and reaches a plateau at scan 46 where it stays until scan 51 when it shows a sharp peak and then an almost exponential fall. This behaviour is unlike the other components and at scan 51 apparently increases in concentration, for this reason the m/e 36 and m/e 38 signals are negative and close to m/e 570.

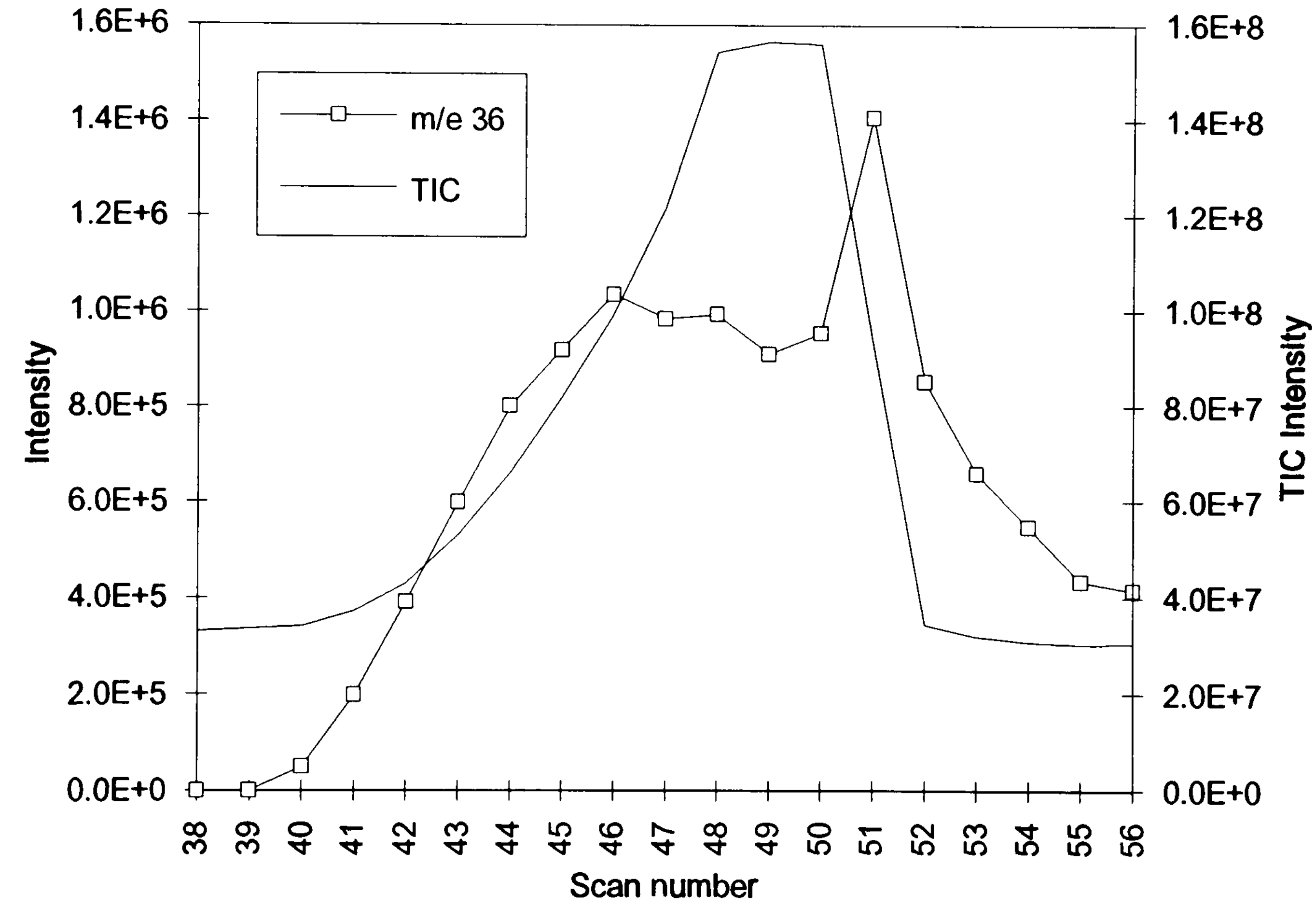


Figure 46: Single ion pyrogram for m/e 36 for data set TB11.

Explanations have now been found to explain all the sources of variance visible in figure 42 except the cluster of low mass peak close to the m/e 298 signal. To explain this the fragmentation in the source must be considered. The thermal decomposition product at m/e 298 fragments further to produce lower masses, the m/e 570 molecular ion also fragments to give lower m/e fragments via the route given in figure 47. This shows clearly how the same fragment molecules can arise from different precursors. The m/e 298 signal is the most dominant source as shown by figure 45 but is perturbed by the contribution from m/e 570 and thus does not align perfectly with the m/e 298 signal in figure 42.

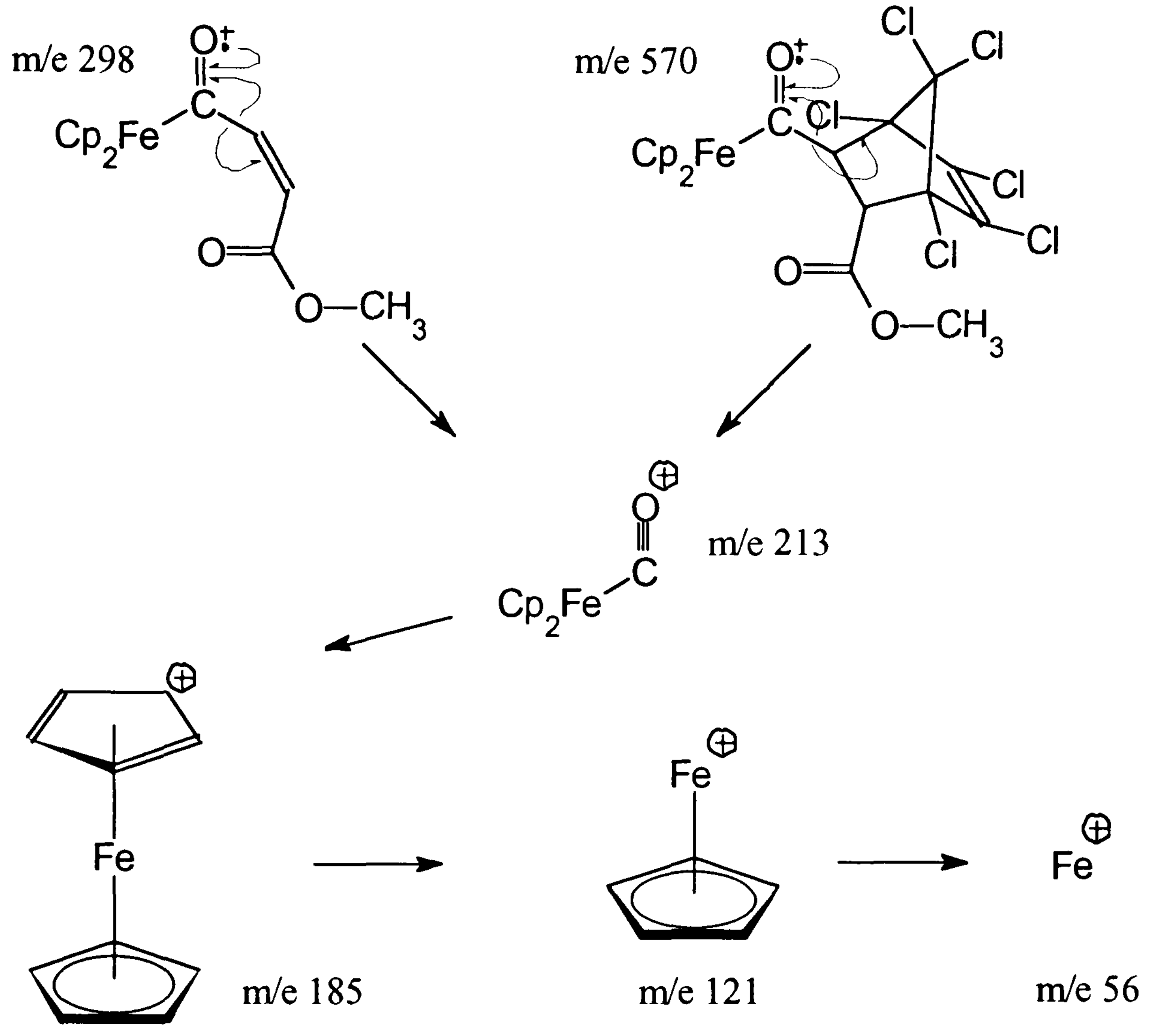


Figure 47: Partial fragmentation pathway for FCA and its decomposition product.

The eigenvector for factor four (figure 39) shows it to be significant only at the front edge of the TIC peak. The magnitude of the factor starts to rise from scan no 40 and shows a sharp peak at scan number 47. Looking at the TIC pyrogram it can be seen that factor 4 increases with the gradient of the TIC signal, this suggests that bias may be responsible for this factor too. In this case, for bias to be the cause, the size of high mass peaks would be artificially reduced and the intensities of the low masses artificially increased; the factor scores for factor 4 (figure 40) show this to be the case. The negative value for the m/e 298 peak is due to the negative weighting imposed by factor 3.

Factor 6 (not shown) has dominant values in its scores for HCl and the eigenvector is similar to the shape of the single ion pyrogram (figure 46) and is thus assigned to describing the unique behaviour of the m/e 36 and m/e 38 peaks. The shape of the first part of the single ion pyrogram roughly follows the TIC curve until the plateau at scan 46. Consideration of the sources of HCl from the sample shows that the major decomposition route (figure 44) produces hexachlorocyclopentadiene, which contains no hydrogen and therefore cannot produce HCl. Close study of the mass spectrum of FCA shows two very weak series of peaks at m/e 534 and m/e 498 corresponding to two losses of HCl, from this it is concluded that the HCl is produced from another less favourable decomposition pathway that competes with the major one. The decomposition probably occurs by loss of one of the two hydrogen atoms attached to the norbornane structure along with one of the attached chlorine atoms, as there are two hydrogen and two observed HCl losses. The explanation for the plateau is less certain but may be due to the formation of a char, trapping HCl, or simply a depression in the HCl signal due the rapid evolution of the m/e 298 product increasing total pressure in the source. There is some supporting evidence for the last hypothesis in the form of a sharp increase in the magnitude of factor 5 at scan 52 where the total pressure in the source will have fallen after the sample had been used up. This supposition begs an explanation for the presence of HCl after the sample signal has finished, the HCl ion profile indicates that either two separate formation processes are occurring, or that the HCl generated is being "stored" for later release. This can be explained if HCl is adsorbed on "cold spots" close to the probe, to be slowly desorbed as the temperature of the probe, and hence the nearby "cold spots", is raised further. The long tail evident on the m/e 36 trace at higher temperatures is consistent with this interpretation and it is well known that HCl is difficult to analyse quantitatively due to adsorption losses[100].

Factor 6, the last factor describing a true factor according to the error terms, seems to be related to the m/e 18 signal, which does not quite align with the background signal, and may be due to atmospheric moisture introduced into the source chamber when inserting the probe and being reduced by pumping throughout the analysis, this can be seen by comparison of the single ion pyrograms for m/e 18 and m/e 28 (not shown) where the signal for H_2O can be seen to be reduced over the period of the analysis.

In summary it can be said that the data is composed of six factors (IND and %SL evidence) and that the sources of variance can be assigned as follows:

- Factor 1 - the average of the data set.
- Factor 2 - separates the background from the sample signal.
- Factor 3 - describes bias from tail of peak.
- Factor 4 - describes bias from leading edge of peak.
- Factor 5 - models the unique behaviour of HCl.
- Factor 6 - models the decay of water vapour over the analysis.

Replicate analyses

Study of the other two data sets (TB12 and TB13) shows that in each case the error terms, IND and %SL, gave a rank of six and that the first three factors for each data set were almost identical, factors 4 to 6 had differing interpretations. Factor 3 can still be clearly identified with bias in the measurement of the scan at the end of the sample peak but has a greater magnitude for TB12 and a smaller magnitude for TB13. This can be seen reflected in the percentage variances listed in table 34, the change in magnitude can be explained by consideration of when the scan was taken. Figure 34 shows the TIC pyrograms of the three samples and the markers identify the start of each scan. TB11 shows scan 51 starting roughly halfway down the decay of the sample signal, TB12 at scan 53 starts slightly higher in the decay and TB13 starts the scan near the bottom of the decay. Ignoring the time taken for the system to reset for the next scan, the marker for the next scan marks the end of the scan. For TB13 the signal has reached background by this point so bias will only affect the high masses. Both TB11 and TB12 show that the signal still has further to fall at the end of their scans and will be more affected by bias. The decay of the sample signal appears to be exponential, so as TB12 started scanning further up the peak it will be affected to a greater extent than TB11 thus producing the observed effect in factor 3. These observations act as a confirmation of the interpretation of factor 3 and also explain why the later factors (4 to 6) are altered, the later factors still describe the other sources of perturbation in the data but because of the change in factor 3 have become distorted and some mixing of factors has occurred making interpretation less clear cut.

From the interpretation of the factors contributing to the observed data it is now clear that for events that occur as rapidly as these, the sampling interval in the time domain is not small enough and introduces sources of error that complicate interpretation. It is also clear that for major components the extra variance complicates the determination of the dimensionality of the data space but does not affect identification of the components, for minor components, whose contribution of variance to the data approaches that of the error introduced from inadequate sampling interval, unambiguous identification may be difficult due to the mixing of the sources of variance. Reduction of this problem may be achieved by increasing the scan rate (preferred but not possible with the equipment used) or by reducing the heating rate. The reduction of the heating rate may not entirely avoid the problem as pure components may still evolve fast enough to cause sampling rate distortion.

Background subtraction

The results for the background subtracted data error terms gave the same dimensionality of six factors as indicated by the percentage significance level, indicator function and imbedded error term. The real error estimate was of no use in the estimation as its value was found to be much larger than the real error term for even the first factor. This clearly shows the inadequacy of the current method of determining the absolute error in the data. The values of the error terms for data set TB11S are given in table 35 and are typical for all three of the background subtracted data sets.

Factor no.	Eigenvalue	RE	IE	IND	SEE	%SL
1	4E+15	67945	15588	210	9E+12	0.0
2	4E+13	38811	12592	134	7E+12	0.0
3	1E+13	18967	7537	74	2E+12	0.0
4	2E+12	13177	6046	59	8E+12	0.9
5	1E+12	8259	4237	42	2E+12	0.4
6	2E+11	6763	3801	40	5E+12	7.1
7	1E+11	5921	3594	41	9E+12	13.1
8	9E+10	5051	3277	42	4E+12	11.8
9	5E+10	4457	3067	45	9E+12	16.5
10	4E+10	3876	2812	48	3E+12	16.4

Key: RE = real error, IE = imbedded error, IND = indicator function,
SEE = standard error in eigenvalue, %SL = percentage significance level.

Table 35: Table of error terms for the first ten factors for data set TB11S.

The estimate of error in the data is found to be far too large as the largest peak in the data after background subtraction is the base peak from the sample and not, as previously, the N_2 peak from the air background. This indicates that the background contributes substantially to the error measured in the data and that the background subtraction technique reduces this source substantially and also that the error in the measurement of the sample data is less than 1% of the maximum base peak.

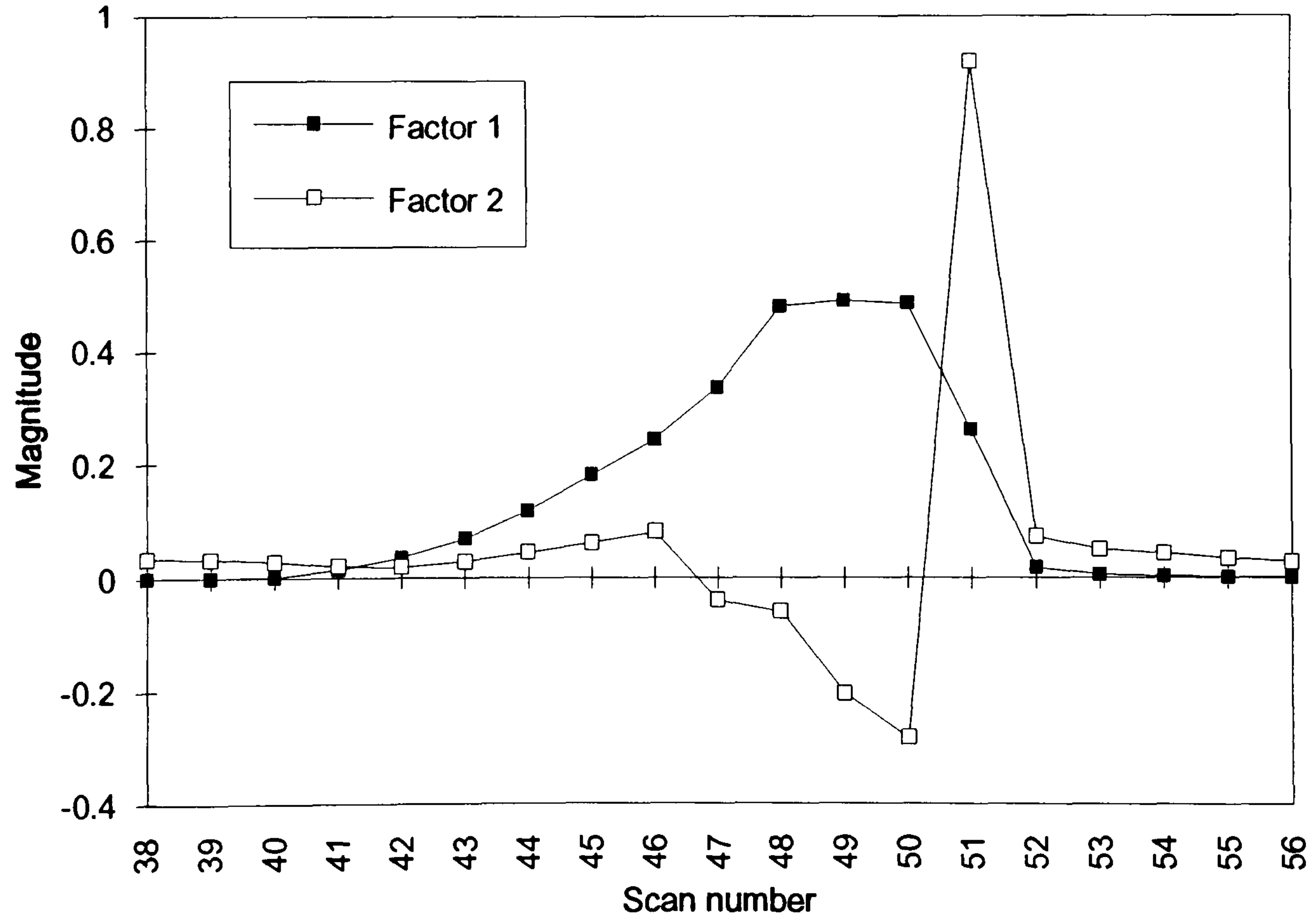


Figure 48: Plot of factors 1 and 2 for the data set TB11S.

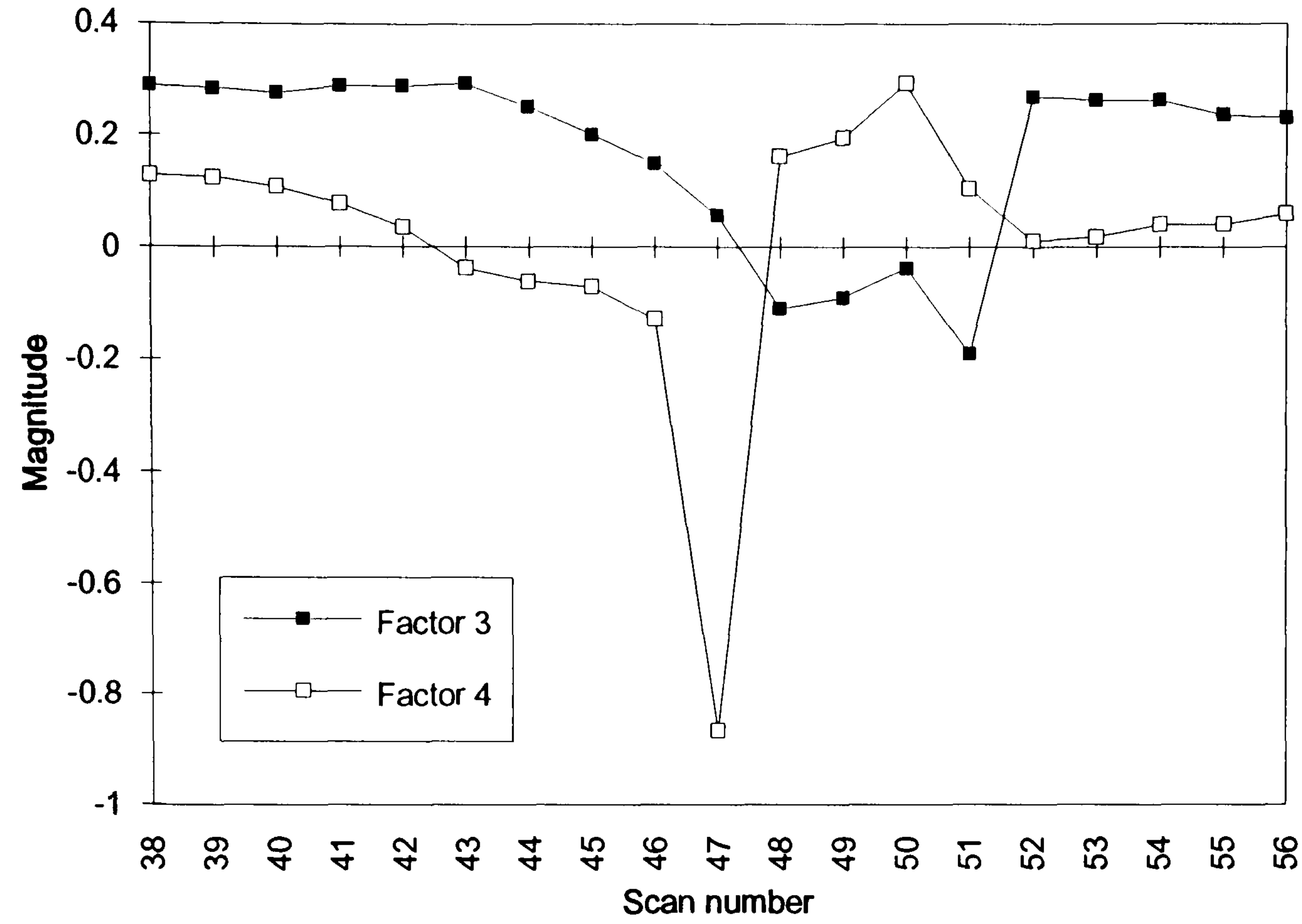


Figure 49: Plot of factors 3 and 4 for the data set TB11S.

The reduction of the importance of the background to the data can be seen in the eigenvectors of TB11S; figure 48 shows the eigenvectors for the first two factors. Factor 1 can now be seen to be the same shape as the TIC pyrogram and thus describes the sample signal, factor 2 follows approximately the same shape as factor 3 in the unsubtracted data and is therefore probably related to the bias in measuring scan 51.

Figure 49 shows the eigenvectors for factor 3 and 4 and it can be seen that the third factor describes the background, thus it can be deduced that the background subtraction has reduced the importance of the background to the data considerably. The fourth factor is similar to the fourth factor in the unsubtracted data and describes the bias at the front of the sample peak.

These results, which were typical for all the background subtracted data, indicate that the technique of background subtraction is useful for reducing the magnitude of the background in the data set. Reduction of the importance of the background in the abstract factors can be useful in cases where a small number of components are present as their presence in the abstract factors will become more obvious, unfortunately, the interpretation of smaller factors is made more difficult as the small residue of the background tends to mix with the smaller factors, this can be seen in the shape of factor 3 in figure 49, which shows some deviation from the ideal background behaviour due to the mixing in of the factors describing bias.

4.6.1.2. 1-1'(2,4-dichlorobenzoyl) ferrocene (1-1'DCBF)

A sample of 1-1'DCBF was analysed by TPPy-MS as detailed in the experimental section to produce the data set TB15. The TIC pyrogram for the analysis is shown in figure 50 along with the TIC trace for the subtracted data set detailed below.

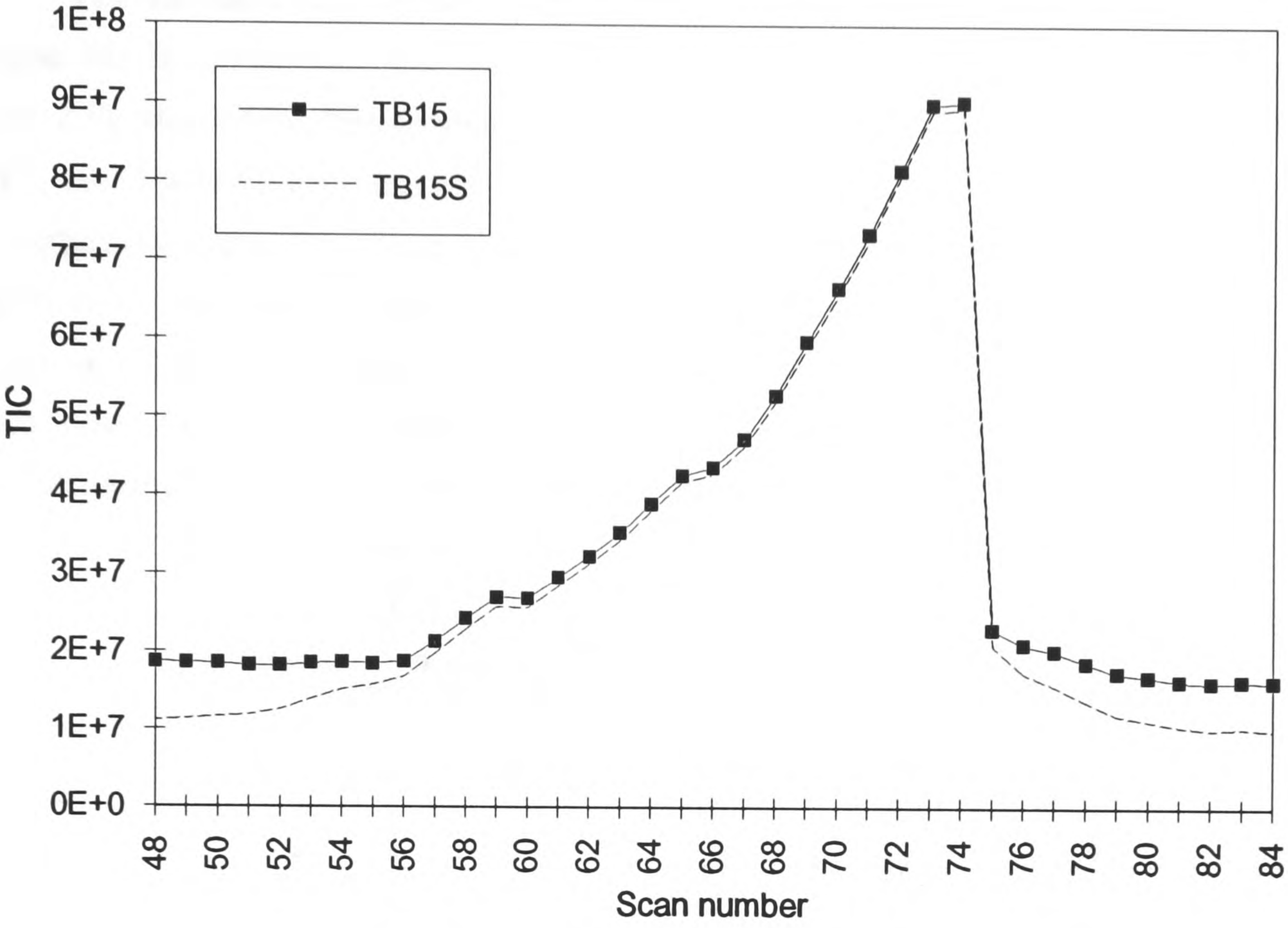


Figure 50: TIC pyrogram for data set TB15.

<u>Data pre-treatment</u>

The data was converted to EPA format using the mass spectrometer data system and the EPA format file was then converted to spreadsheet format using the program. The size of the data file was then reduced by removing scans from the beginning and end of the file, leaving only scans 48 to 84 that contained the pyrogram peak (all other scans were background only). The truncation of data was performed to reduce the analysis times.

A further data set was produced from TB15 using the spreadsheet. The m/e values relating to air in the spectra (m/e 17,18,28,32,40 and 44) were set to zero to study this method of background removal. The data set produced from this procedure was named TB15S.

<u>Data analysis</u>

The two data sets were analysed using the following operating parameters; Error estimate = 901531 (1% of largest peak in data set, m/e 139 @ scan 74).

Minimum = 16137 (smallest base peak / 999).
Accuracy = 1×10^{-19} (default value).

Both data sets ran to the maximum number of possible factors before completion of the decomposition suggesting that, as in the previous experiment, the minimum value is set too low.

Determination of the rank of the data

The variance accounted for by the first fifteen factors in the data sets is given in table 36. It can be seen that the air peak reduction has reduced the importance of factor 2 by about 6%, this is much less than the effect of background subtraction on TB11, etc. Factor 2 for this data does, however, appear to be less important than with the earlier data sets, this is supported by the fact that for all the earlier data sets the largest single peak in the data was due to N_2 and for this data the largest peak is the base peak of the sample spectrum indicating that the background has a smaller overall contribution to the data. Additionally the smaller reduction may be due to the air peak removal being less efficient compared with background subtraction.

	Data set			
	TB15		TB15S	
Factor	%Variance	%SL	%Variance	%SL
1	91.33	0.0	97.81	0.0
2	7.86	0.0	1.36	0.0
3	0.47	0.0	0.51	0.0
4	0.11	0.8	0.11	0.6
5	0.10	0.1	0.09	0.1
6	0.03	4.2	0.03	4.0
7	0.03	2.1	0.03	2.0
8	0.02	3.2	0.02	3.5
9	0.01	3.9	0.01	3.6
10	8.77E-03	4.8	9.29E-03	4.2
11	6.98E-03	4.5	7.50E-03	3.5
12	5.82E-03	3.4	6.21E-03	2.4
13	2.69E-03	10.2	2.27E-03	11.9
14	2.13E-03	11.3	1.48E-03	17.1
15	1.39E-03	16.2	1.30E-03	17.5

Table 36: Table of percentage variances and percentage significance levels for the first fifteen factors in data sets TB15 and TB15S.

The eigenvalues for data set TB15 are shown in figure 51. The eigenvalue can be seen to fall sharply to the fourth factor where a plateau occurs for one factor before falling sharply again to another one factor plateau; the eigenvalues then fall away at a much reduced gradient. Interpretation of the eigenvalues alone would be difficult given their behaviour in this case. The standard error in eigenvalue crosses the eigenvalue just after factor 3, so, if the error estimate for the data set is correct then two or three factors are responsible for the data.

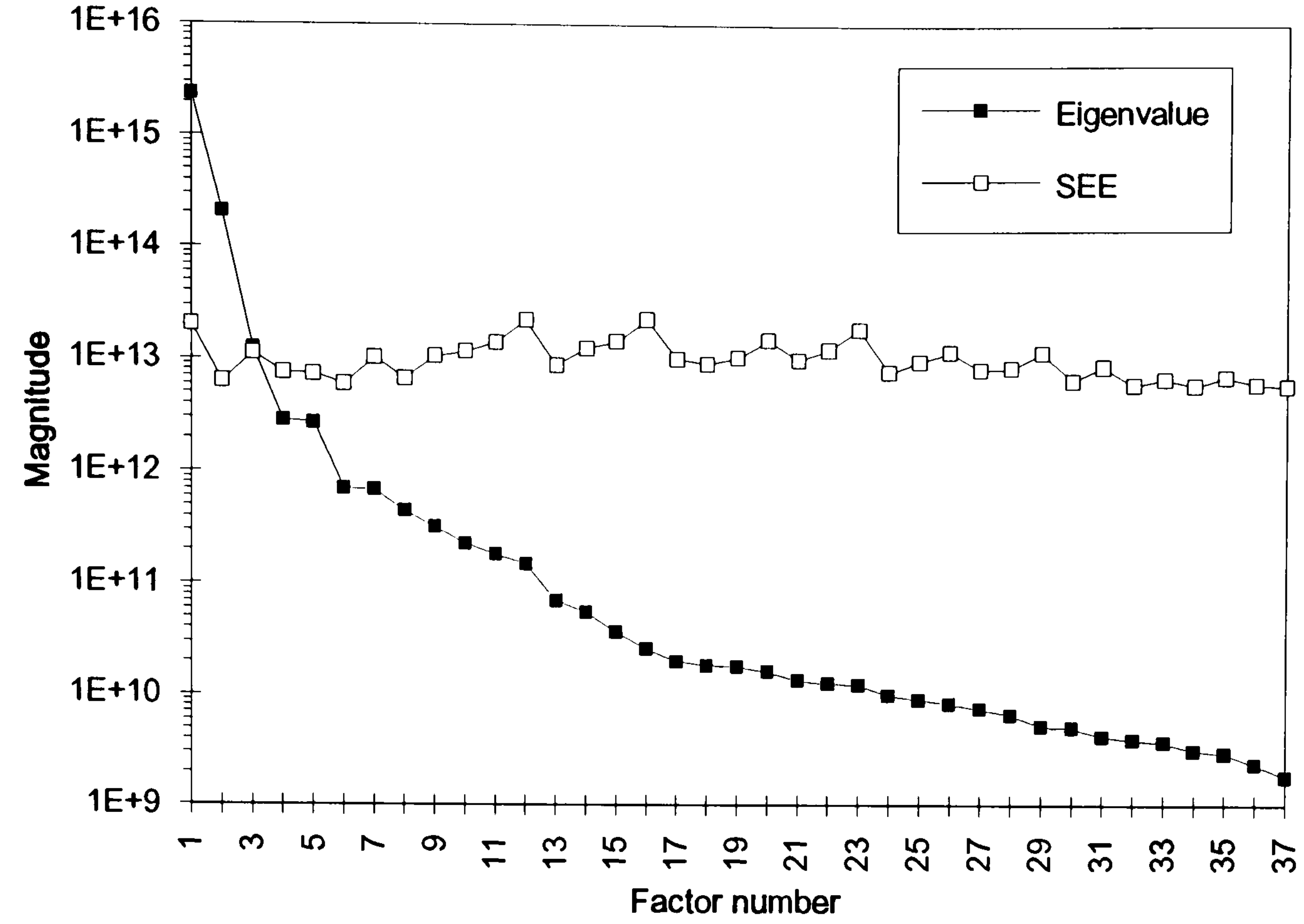


Figure 51: Plot of eigenvalue and standard error in eigenvalue (SEE) for data set TB15.

To look more closely at how well the error estimate fits the data set we can study the real error function; this is shown in figure 52. The estimate of error based upon 1% of the most intense peak in the data produces an estimate far too large for any of the real error values and must be regarded as suspect along with the standard error in eigenvalue calculated from it.

As an attempt to find a better error estimate, 1% of the average of the m/e 28 value at scans 48 and 84 was tried. This can be also be seen plotted in figure 52 passing just above the real error value for the second factor. This indicates one factor in the data as is known to be the case and thus may be a better method of calculating the error estimate.

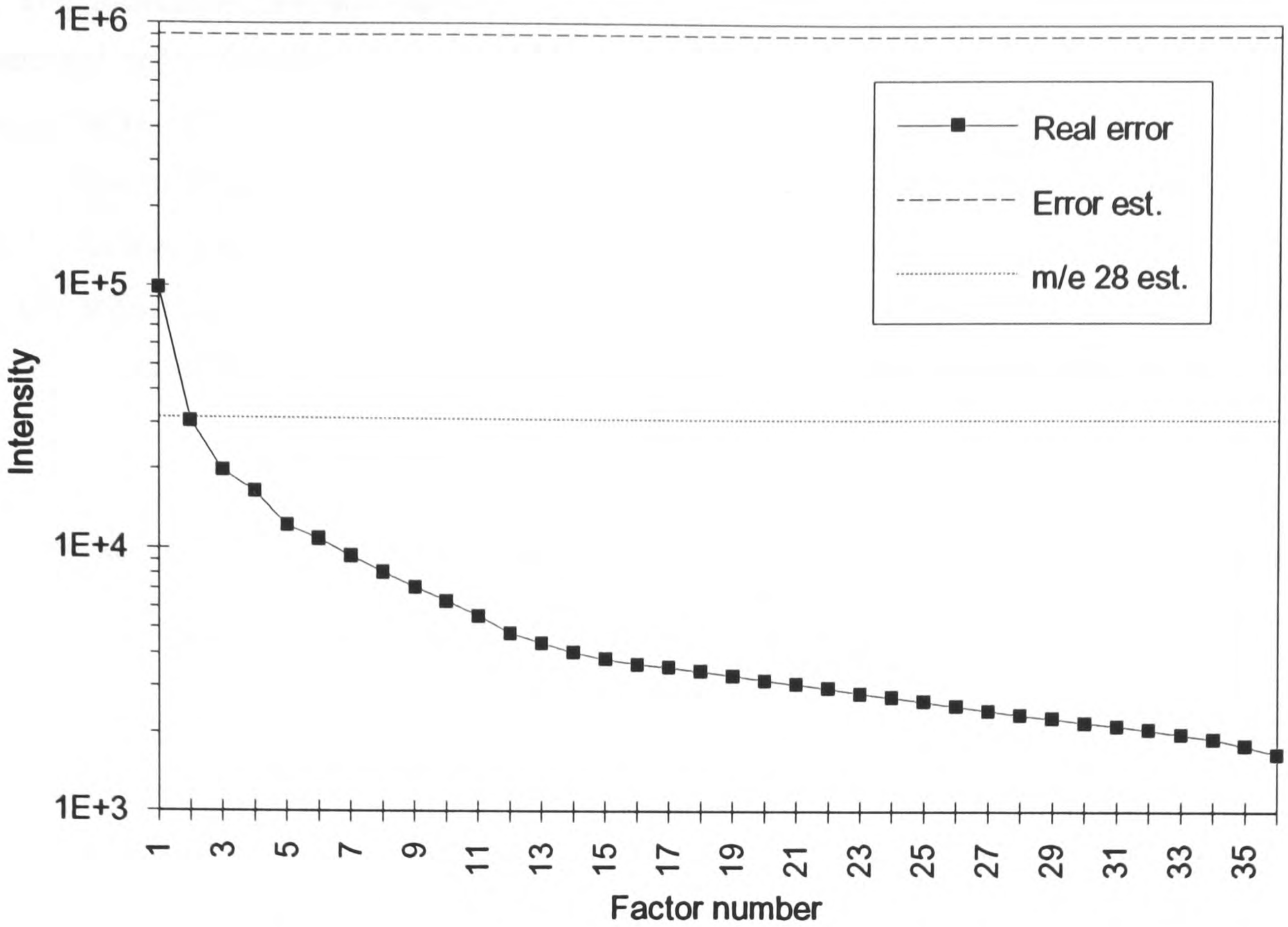


Figure 52: Plot of real error and estimated error for data set TB15 showing modified error estimate based on m/e 28 intensity.

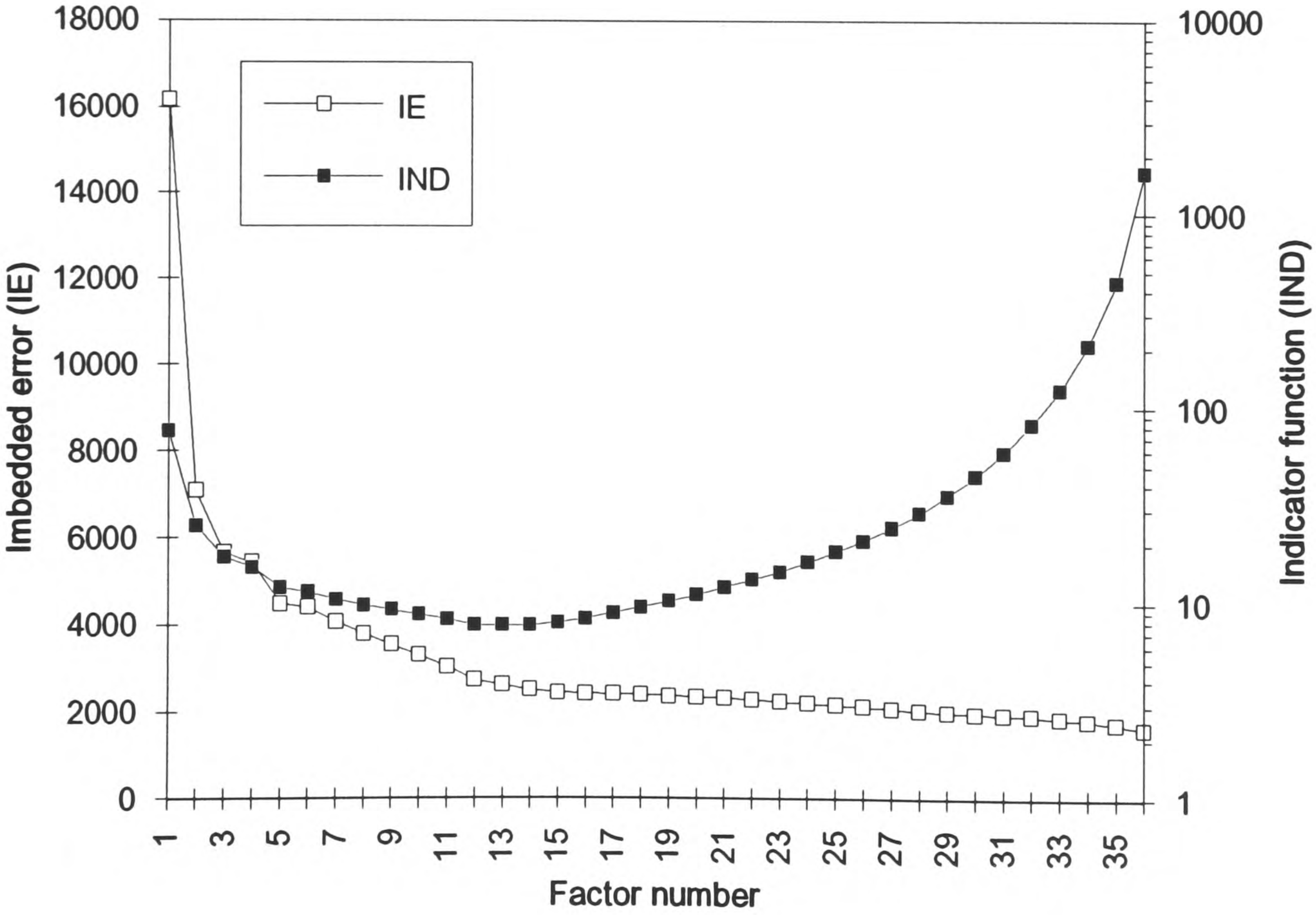


Figure 53: Plot of imbedded error (IE) and indicator function (IND) for the data set TB15.

The estimates for imbedded error and indicator function are shown in figure 53. The indicator function shows a distinct minimum around factor 13 and the imbedded error function, whilst not displaying a minimum, also exhibits an inflexion around factor 13.

Consideration of the percentage significance levels given in table 36 indicates that 12 factors are responsible for the data, as factor 13 has a value just above 10% and the significance threshold is between 5% and 10%. The values for IE and IND are very similar for factors 12 and 13 so it is concluded that 12 factors make up the data.

The error functions for TB15, like the functions for the earlier data sets, disagree as to the number of factors necessary to describe the data. It is believed that the estimates produced by the IE, IND and %SL functions accurately predict the change from primary to secondary eigenvectors. The single component samples, when analysed by this method, give rise to more than one factor due to multiple reaction pathways and complicated background factors. The error estimate, if it can be determined with sufficient accuracy, appears to give a better indication of the number of physical components in the sample.

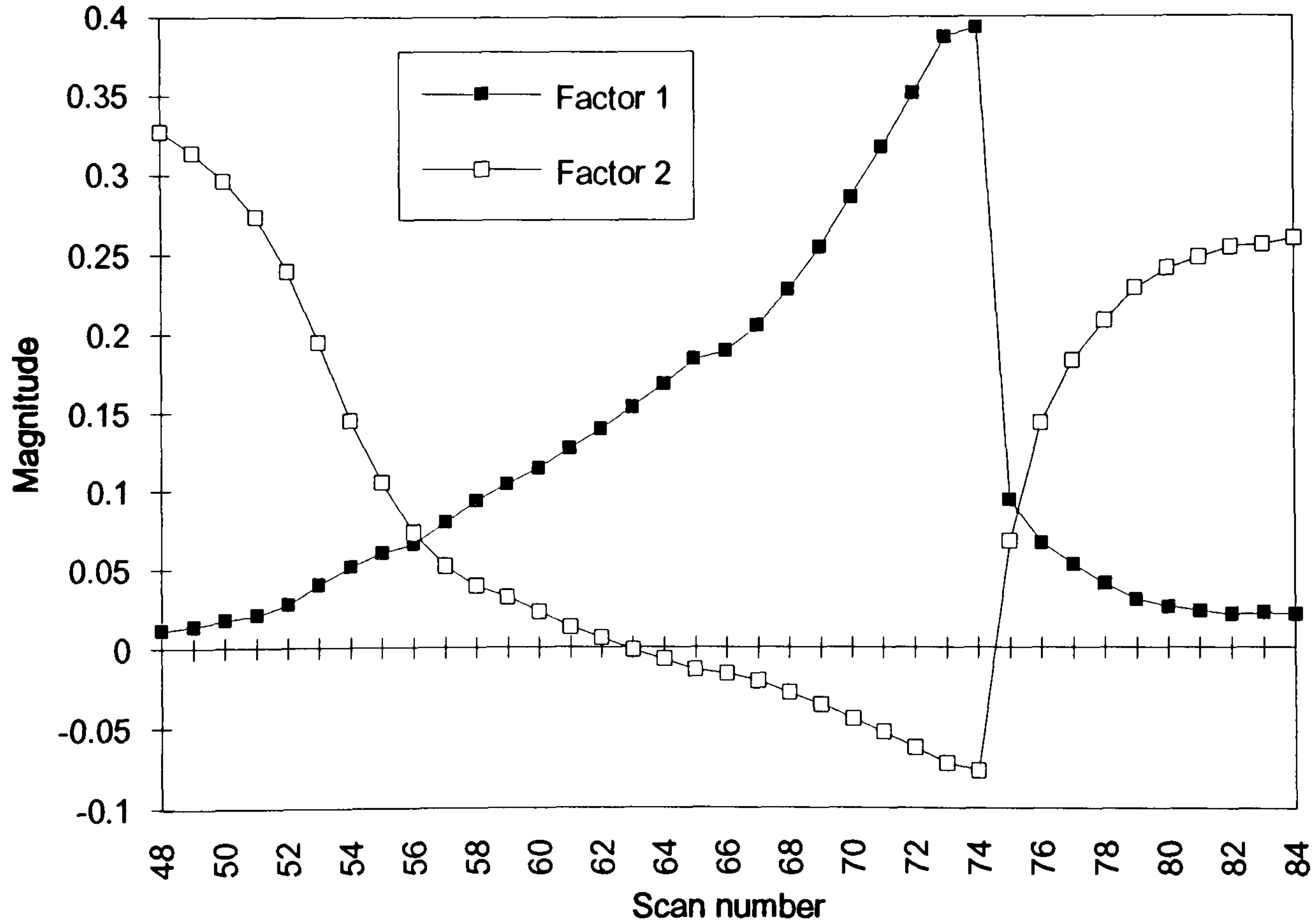


Figure 54: Plot of factors 1 and 2 for the data set TB15.

Abstract column matrix

The first two eigenvectors of data set TB15 are shown plotted against scan number in figure 54. Factor 1 shows clearly the TIC pyrogram and factor 2 is shaped like the background. This is opposite to the results found for TB11, this behaviour is explained when it is seen that the data set has had too many scans removed. Study of the single ion pyrograms indicates that sample evolution has started in scan 86 and has barely returned to zero at scan 94, this means that the sample spectrum, present in almost every scan, is the most dominant signal and the background therefore falls in importance to the data. The fall in background signal during the sample evolution is due to an increase in total pressure within the source reducing the partial pressure of the background signal.

Figure 55 shows the third and fourth factors of TB15. Their interpretation is considerably more complex than the earlier two. Both factors show sharp peaks at scan 75, which suggests that they describe bias in the data as did factor 3 for data set TB11; as both factors describe the same source of variance then the positioning of factors for this data set has not been as advantageous as it was for TB11 (allowing largely individual sources of variance to be seen) and the sources of variance are mixed together. Two conclusions may be drawn from this evidence; that the data is more complicated (i.e. has more sources of variance) and that the sources are of similar importance to the data (i.e. produce similar amounts of variance). The peak at the beginning of factor 3 may describe bias in the rising edge of the pyrogram, whilst the jagged beginning of factor four seems anomalous and will be looked at in more depth. Further factors show even more complicated behaviour.

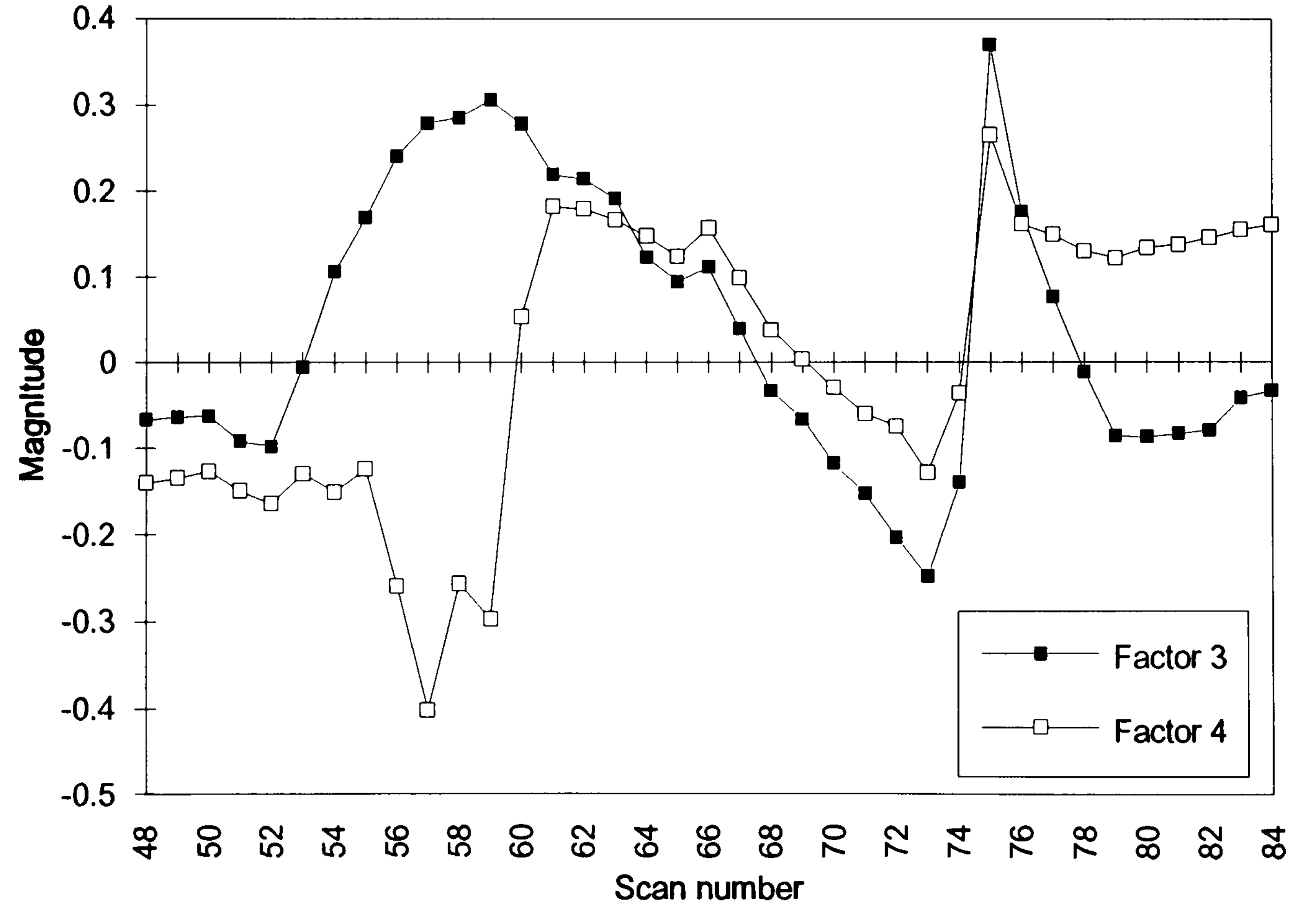


Figure 55: Plot of factors 3 and 4 for the data set TB15.

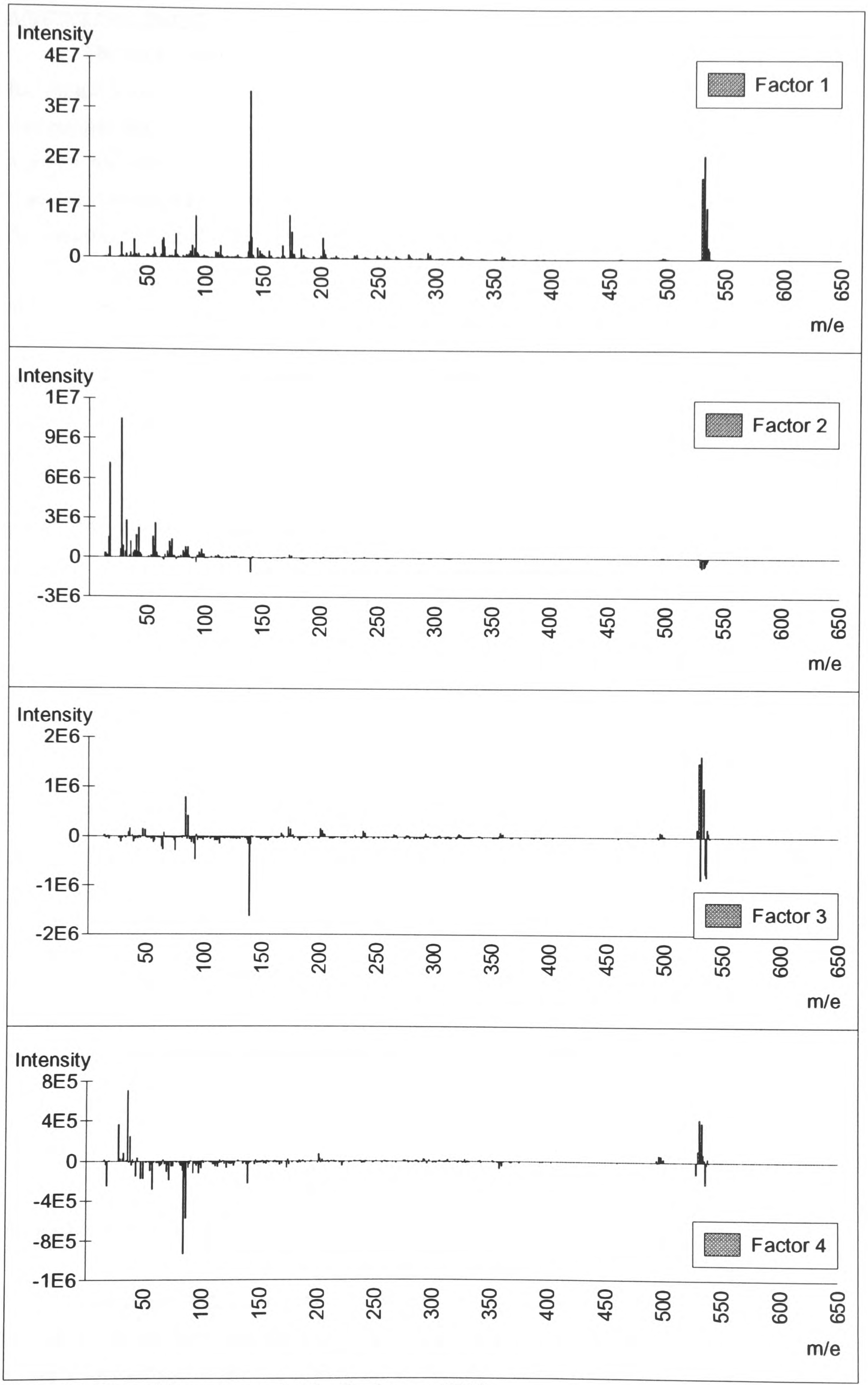


Figure 56: Plots of factor scores 1-4 for the data set TB15.

Abstract row matrix

The factor scores for the first four factors are given in figure 56. The scores for factor 1 show the spectrum of the sample. Factor 2 provides a separation between the sample and the background, the background being visible on the positive side and a much reduced sample spectrum on the negative. The background spectrum shows that the instrument was suffering from hydrocarbon contamination in the source whilst the sample was analysed and may explain why the number of factors necessary to describe the data has increased from TB11 and also the complicated nature of the abstract factors.

Factor 3 shows the results of bias in the fall of the pyrogram, with the molecular ion peak very large in comparison with the base peak and the smaller peaks. The spectrum also shows a strong weighting for m/e 84 and m/e 86, a weighting which is repeated even more strongly in factor 4. These peaks were not expected from the decomposition of 1-1'DCBF. To try and identify the source of the peaks, the single ion pyrogram for m/e 84 was plotted; this is shown with the TIC pyrogram in figure 57.

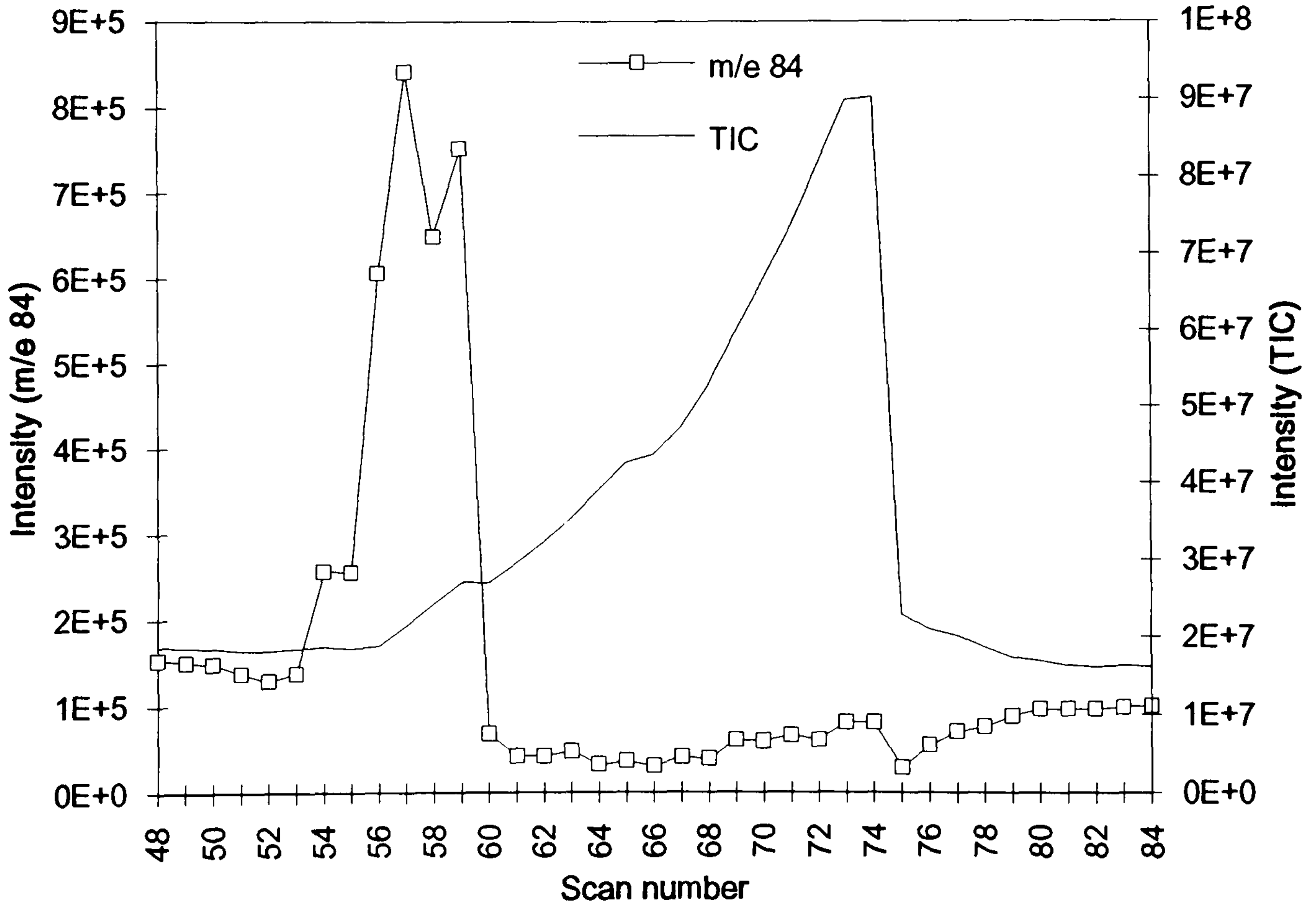


Figure 57: Single ion pyrogram for m/e 84 for data set TB15.

The graph shows that the component giving rise to m/e 84 occurs only at the beginning of the sample peak and is almost identical to the jagged beginning of factor 4. As a further aid in the identification of the component an iterative target test was performed using a uniqueness vector for m/e 84 and 10 iterations. The resulting spectrum showed peaks at m/e 84,86, at m/e 47,49 and 35,37 and also a very small peak at m/e 119.

These peaks are consistent with the fragmentation of $CDCl_3$ (shown in figure 58) allowing the component to be identified as due to deuterated chloroform that had been trapped in the crystals of 1-1'DCBF when it was recrystallised from the solution used for ^{1}H-NMR analysis.

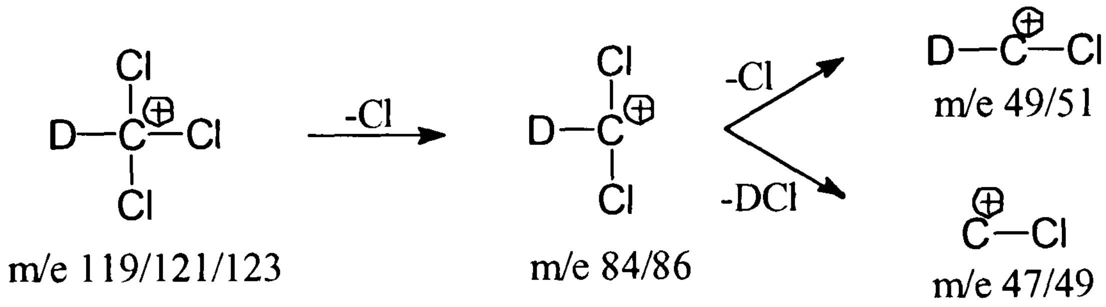


Figure 58: The fragmentation of deuterated chloroform in the mass spectrometer.

Factor 4, in addition to the negative m/e 84,86 signals also shows a strong positive signal for m/e 36 and 38 and consideration of the single ion pyrogram for m/e 36 (figure 59) shows similar behaviour to that observed for m/e 36 in TB11 with HCl being evolved both during and after the evolution of the sample.

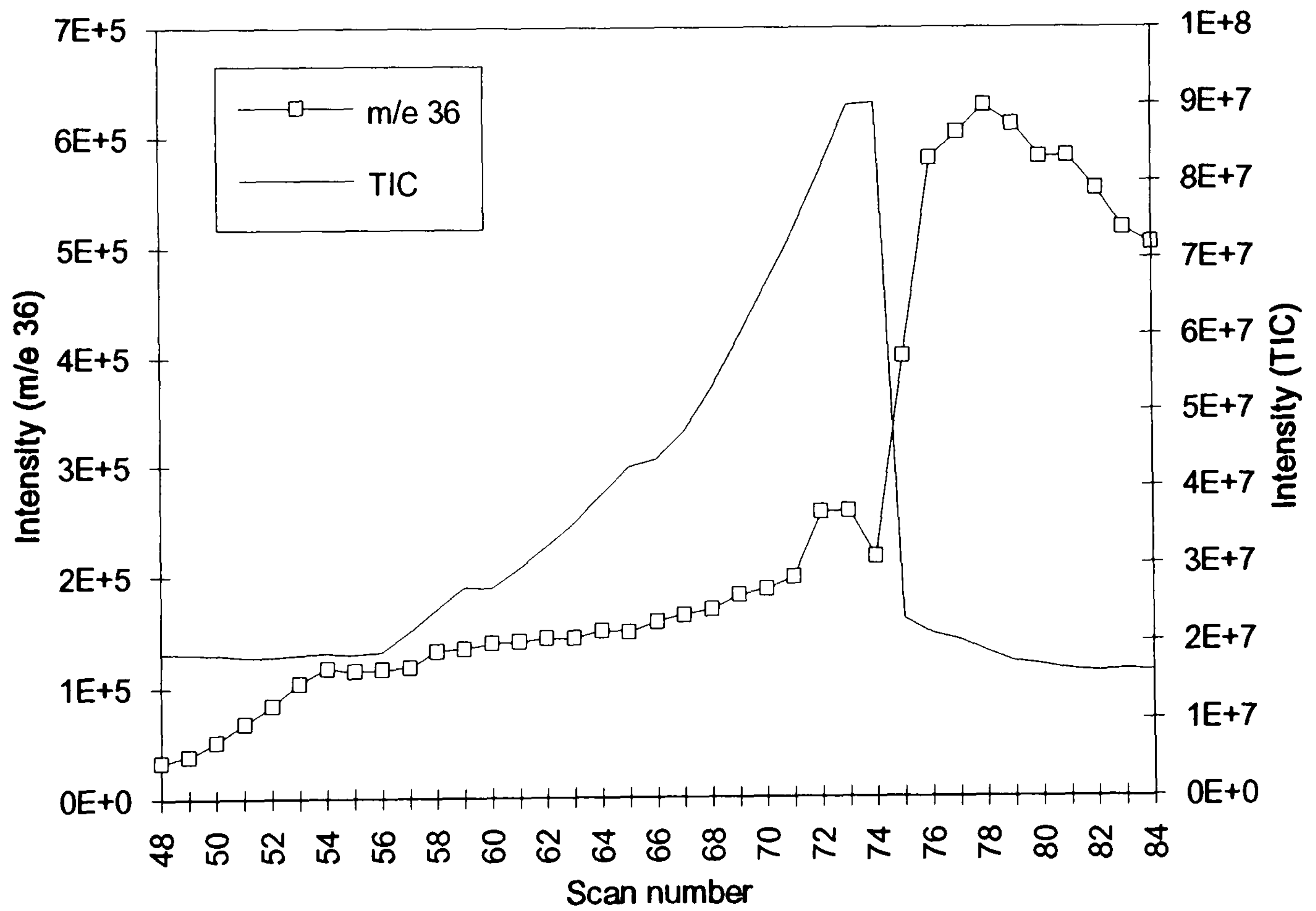


Figure 59: Single ion pyrogram for m/e 36 for data set TB15.

Factors 3 and 4 both show the base peak at m/e 139 and the molecular ion at m/e 572 to have weightings of opposite signs, but because the factors are mixed the way this arises is not obvious. To provide a better visualization of the variation in the data a plot of factor 2 against factor 4 was constructed. This plot shows that there are many more sources of variation than for the TB11 data set. The variance from the m/e 84 impurity can be seen as well as the direction of the background signal. The m/e 18 signal from H_2O does not align perfectly with the air peaks and appears to be grouped with a lot of other low m/e peaks, probably due to the source contamination. For the sample there can be seen independent variances for m/e 36/38, m/e 139 and m/e 532.

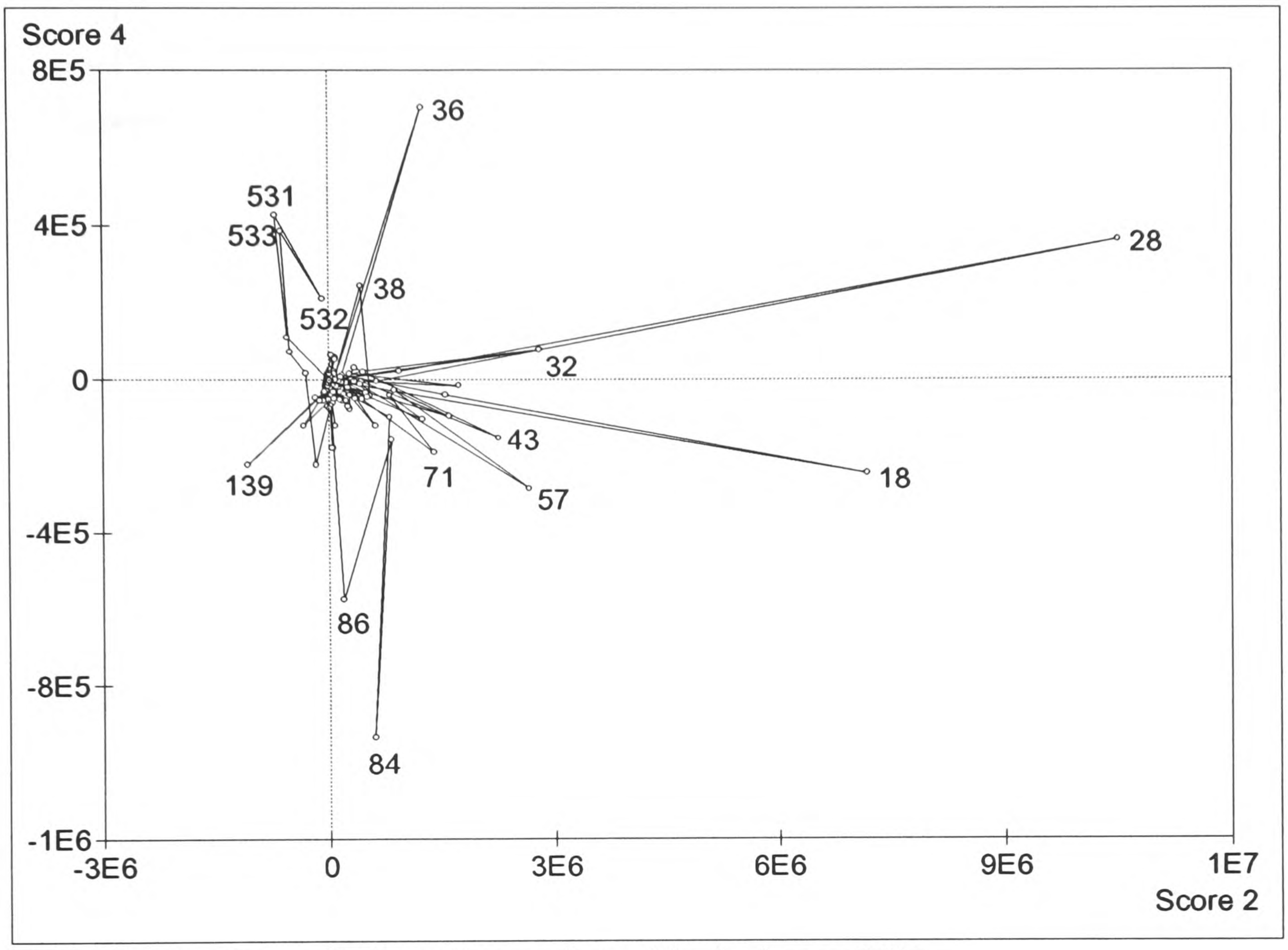


Figure 60: Plot of scores for factor 2 versus factor 4 for the data set TB15.

Ion source chemistry

To explain why the different sources of variance arise, their chemistry must be investigated. Consideration of the structure of the 1-1'DCBF produces the expected fragmentation in the source given in figure 61; the fragmentation is based upon a pathway given for diacetylferrocene by Sheley and Fishel[101].

All the fragments are evident in the mass spectrum of 1-1'DCBF except m/e 384, from which it is concluded that the ferrocene breaks apart so quickly that the fragment is not observed.

Similar to the FCA analysis, the base peak observed in the spectrum is not produced from the volatilized sample, so the decomposition reactions must be studied to explain the abundant m/e 139 base peak.

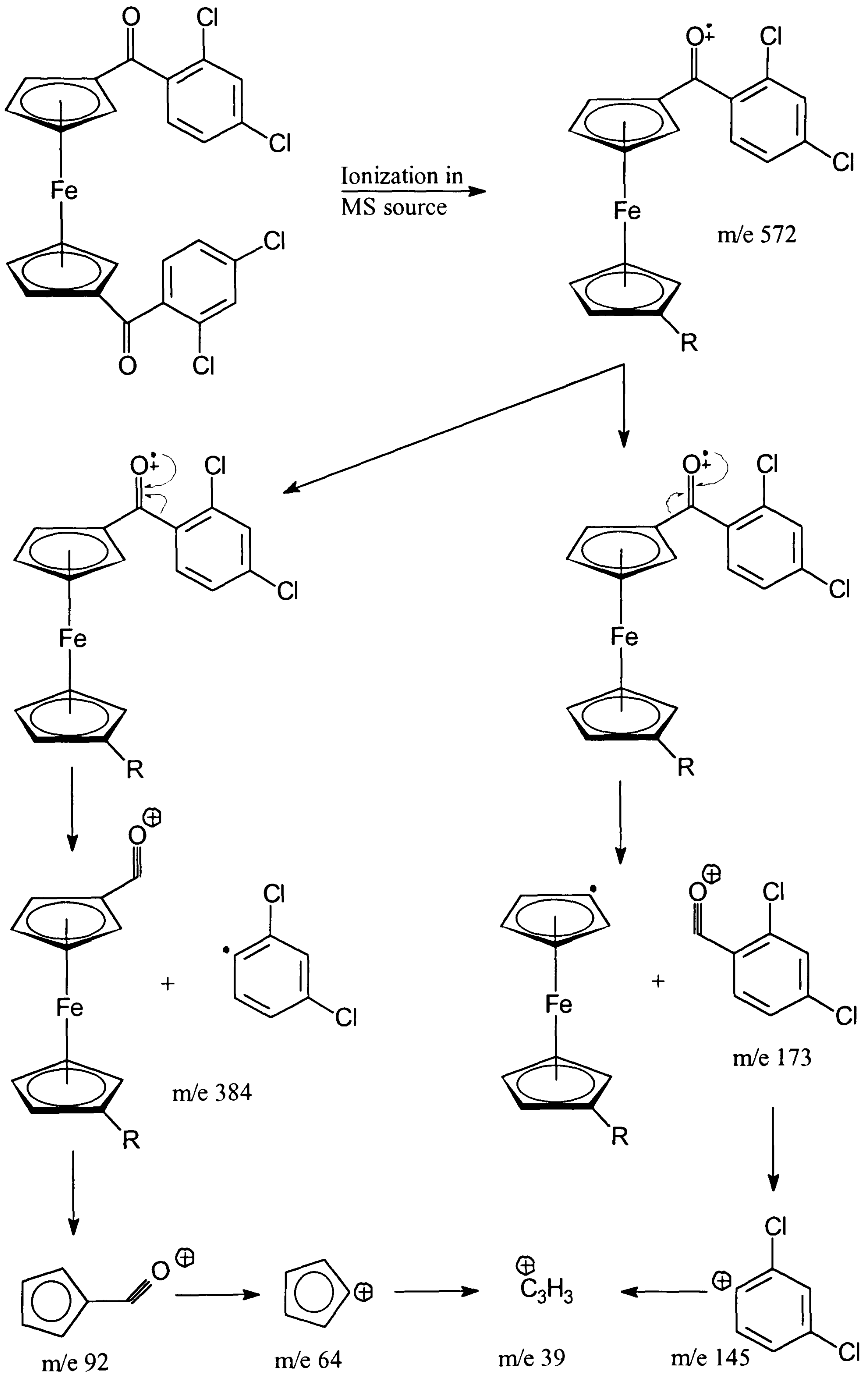


Figure 61: Proposed fragmentation pathway for 1-1'(2,4-dichlorobenzoyl) ferrocene.

Accurate mass analysis of m/e 139

The possible decomposition reactions for 1-1'DCBF are not as obvious as those for FCA and to assist in the identification of the mechanism an accurate mass analysis was performed on the base peak to unambiguously assign the composition of the fragment.

The mass spectrometer was tuned to an instrument resolution of 1500 and set up to perform a linear accelerating voltage scan to cover the range 125-175 m/e. The sample was then analysed as before but with perfluorokerosene being introduced via the liquid injection port to act as an internal mass standard.

Two peaks from the perfluorokerosene were selected which bracketed the sample peak at m/e 139; these were C_2F_5 at m/e 130.9921 and C_3F_7 at m/e 168.98882. Scans of the analysis were selected either side of the sample peak in the TIC pyrogram so that the perfluorokerosene signals were appreciable in comparison with the m/e 139 peak. For each scan the peak centroids' at 50% peak height were measured for the two reference peaks and the m/e 139 peak. The data system then calculated the exact mass by linear interpolation between the reference peaks. The results obtained are listed below.

Scan number	m/e 139 centroid
55	139.0596
56	139.0534
57	139.0510
73	139.0552
74	139.0564
75	139.0569
Average m/e	139.0554

Table 37: Table of calculated peak centroids for m/e 139 of 1-1'(2,4-dichlorobenzoyl) ferrocene.

Using the elemental composition calculation program in the MS data system the possible molecular formulae were calculated. Chlorine atoms were excluded from the calculation as no isotopic substitution was visible, iron was allowed though it was expected to be possible to see the isotopic substitution pattern as well, and carbon, hydrogen and oxygen were also allowed in the formula. The results showed that only two possible fragments could be formed from the available atoms, these were $C_8H_{11}O_2$ at m/e 139.07590, a discrepancy of 20 mmu (milli-mass units) and C_7H_{11} at m/e 139.05478, a discrepancy of 0.6 mmu. The 20 mmu difference is enormous in accurate mass analysis terms and thus the only possible fragment can be C_7H_{11}.

Decomposition reaction

The identification of the base peak as having the formula $C_{11}H_7{}^+$ meant that a possible reaction mechanism could be proposed, this is given in figure 62.

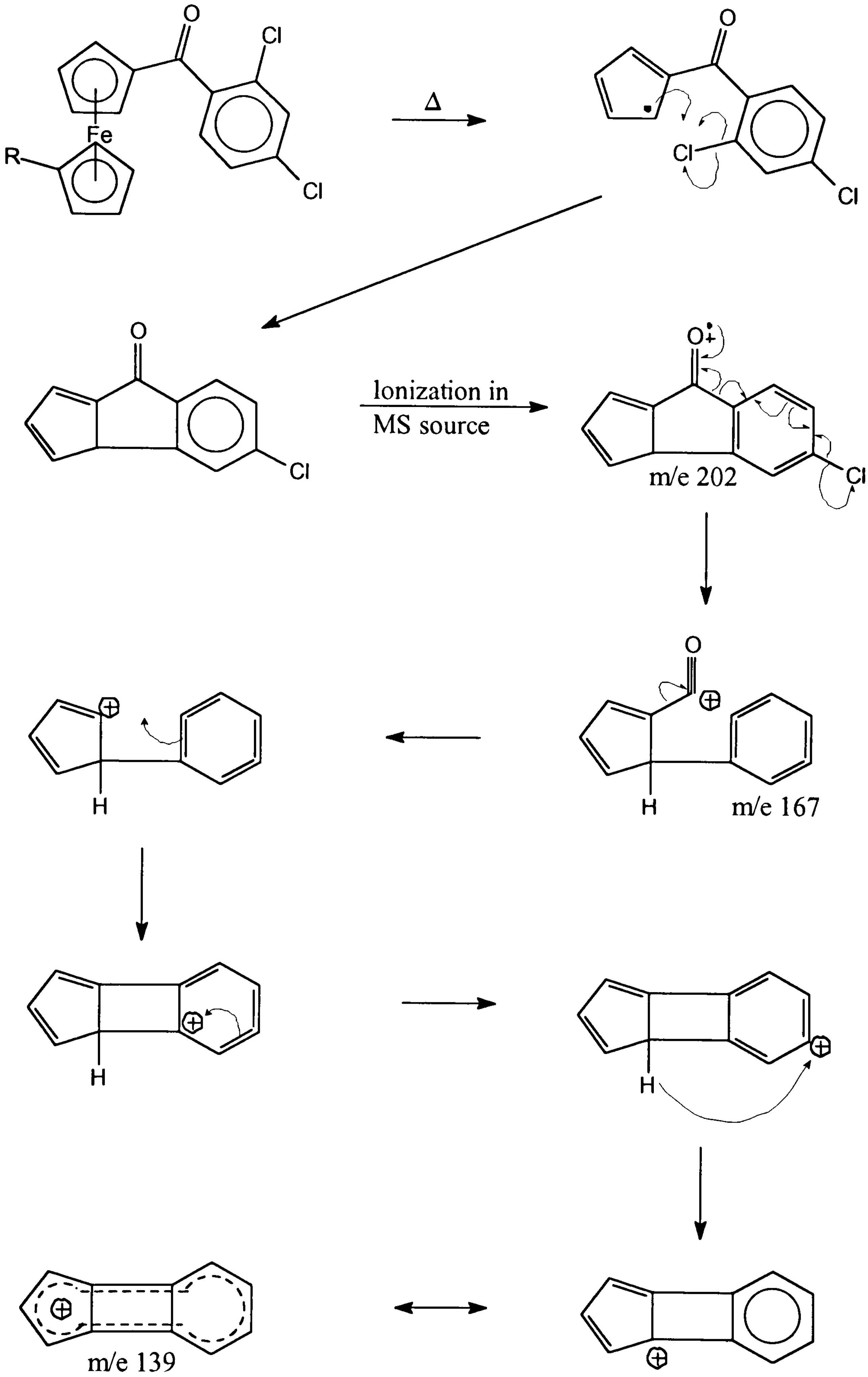


Figure 62: Proposed thermal decomposition mechanism for 1-1'(2,4-dichlorobenzoyl) ferrocene.

Supporting evidence for the mechanism can be seen in the spectrum at m/e 202/204 and at m/e 167. This mechanism does not appear to have been reported for ferrocenes in the literature and after consideration of the results obtained for 3,4-dichlorobenzoyl ferrocene it would be expected to occur in any di-chloro substituted benzoyl ferrocene. The intensity of the m/e 139 peak is much reduced without a chlorine in the ortho position, as shown in figure 33, but is still the favoured decomposition route.

There is evidence of another decomposition/fragmentation pathway in the mass spectrum as small peaks below the molecular ion peak indicate two losses of HCl identical to those for FCA. The loss of two molecules is in this case limited by the number of chlorine atoms available and not by the number of hydrogen's available as was the case with FCA. For 1-1'DCBF this reaction is not the only route to produce HCl as the m/e 145 fragment in figure 61 could further fragment to produce HCl rather than the proposed fragments.

Air peak removal

Figure 50 shows that subtracting the air peaks has reduced the magnitude of the background either side of the sample peak suggesting a reduction in background.

The error terms produced from the analysis are shown in table 38 and show the same characteristics as the unsubtracted data, including an estimate for the dimensionality of the data of 12 factors based upon %SL and IND values.

Factor no.	Eigenvalue	RE	IE	IND	SEE	%SL
1	2E+15	47663	7836	37	2E+13	0.0
2	3E+13	29776	6923	24	4E+12	0.0
3	1E+13	18920	5387	16	1E+13	0.0
4	3E+12	15560	5116	14	9E+12	0.6
5	2E+12	11868	4363	12	3E+12	0.1
6	7E+11	10561	4253	11	8E+12	4.0
7	6E+11	9052	3937	10	8E+12	2.0
8	4E+11	7946	3695	9	6E+12	3.5
9	3E+11	6950	3428	9	1E+13	3.6
10	2E+11	6103	3173	8	1E+13	4.2
11	2E+11	5282	2880	8	1E+13	3.5
12	2E+11	4444	2531	7	2E+13	2.4
13	6E+10	4128	2447	7	1E+13	11.9
14	4E+10	3922	2412	7	1E+13	17.1
15	3E+10	3724	2371	8	1E+13	17.5

Key: RE = real error, IE = imbedded error, IND = indicator function, %SL = percentage significance level.

Table 38: Table of error terms for the first 15 factors for data set TB15S.

Study of the eigenvectors produced from the factor analysis showed no significant change in any of the factors. The factor scores also appear essentially unchanged including the second score, which was related to the background in the untreated data. The two scores for TB15 and TB15S are shown in figure 63 and it can be seen that the large peaks due to N_2 and H_2O are gone leaving only the hydrocarbon contamination in the source, which has now become the new background with a similar intensity to the hydrocarbon background of TB15.

This indicates that the technique of air peak removal does not work as a method for reducing the number of factors in the data. In this particular case, due to the large amount of contamination in the source, the subtraction has been particularly ineffective. Even when the source is clean there are always more than just the air peaks producing background in the instrument so unless the extra peaks are subtracted as well (as for background subtraction using the mass spectrometer data system) then the benefits of air peak removal are limited.

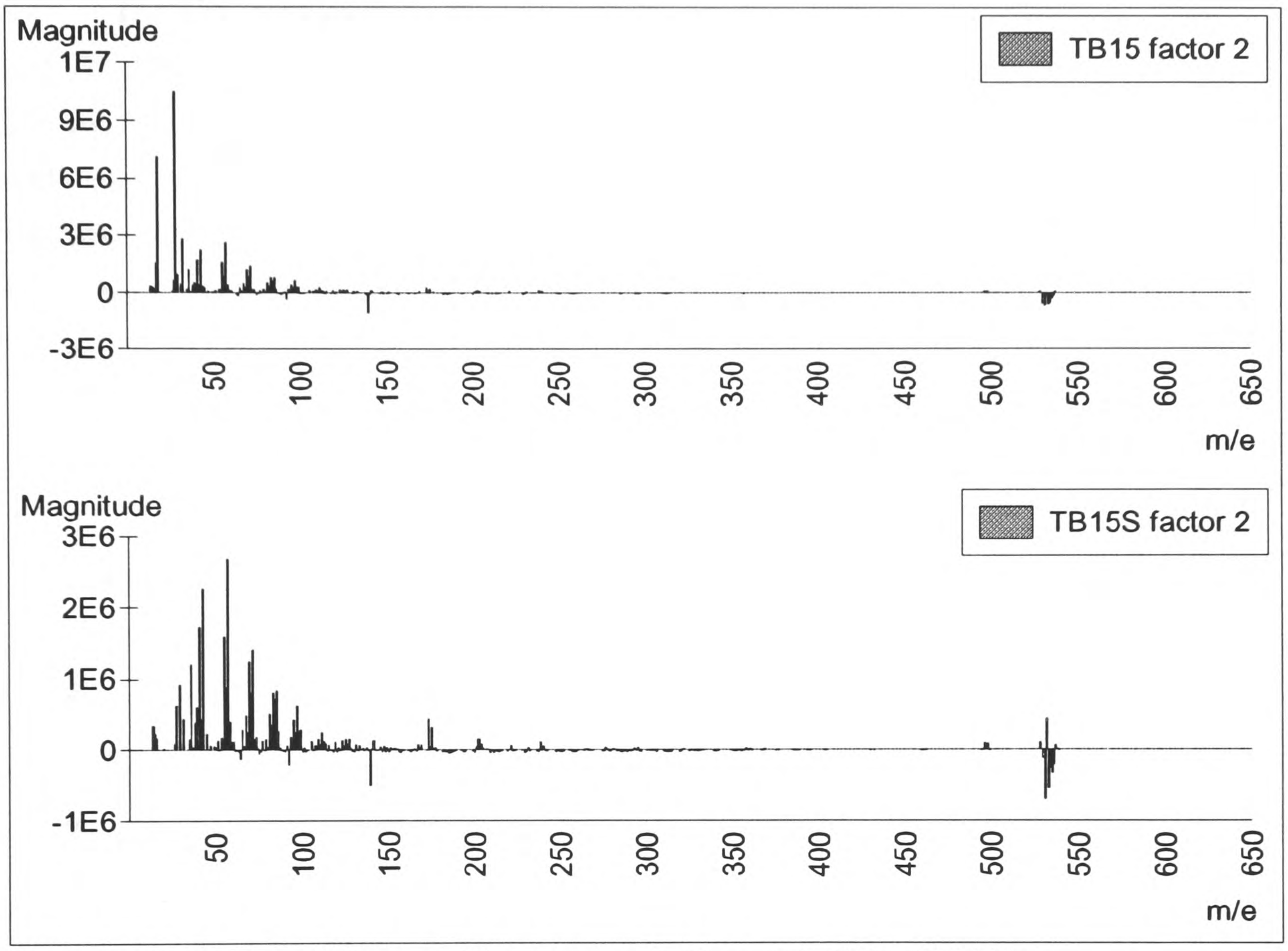


Figure 63: Comparison of the second factor scores for data sets TB15 and TB15S.

4.6.2. Two component data

4.6.2.1. Ferrocene derivative mixture

In a simulation of a system with two components, the ferrocene derivatives FCA and 1-1'DCBF, were analysed simultaneously by placing a crystal of each sample in a pyrolysis tube and analysing by temperature programmed pyrolysis mass spectrometry according to the method laid down in the experimental section.

The TIC pyrogram is shown in figure 64, two maxima are visible, at scans 63 and 78. The aim was to use two components that overlapped closely in the pyrogram but this was not possible. In this particular case, the presence of two components could be concluded from the two maxima, as the overlap of components is incomplete, but there is an overlapping portion between the two maxima that will complicate the analysis providing approximately real life conditions.

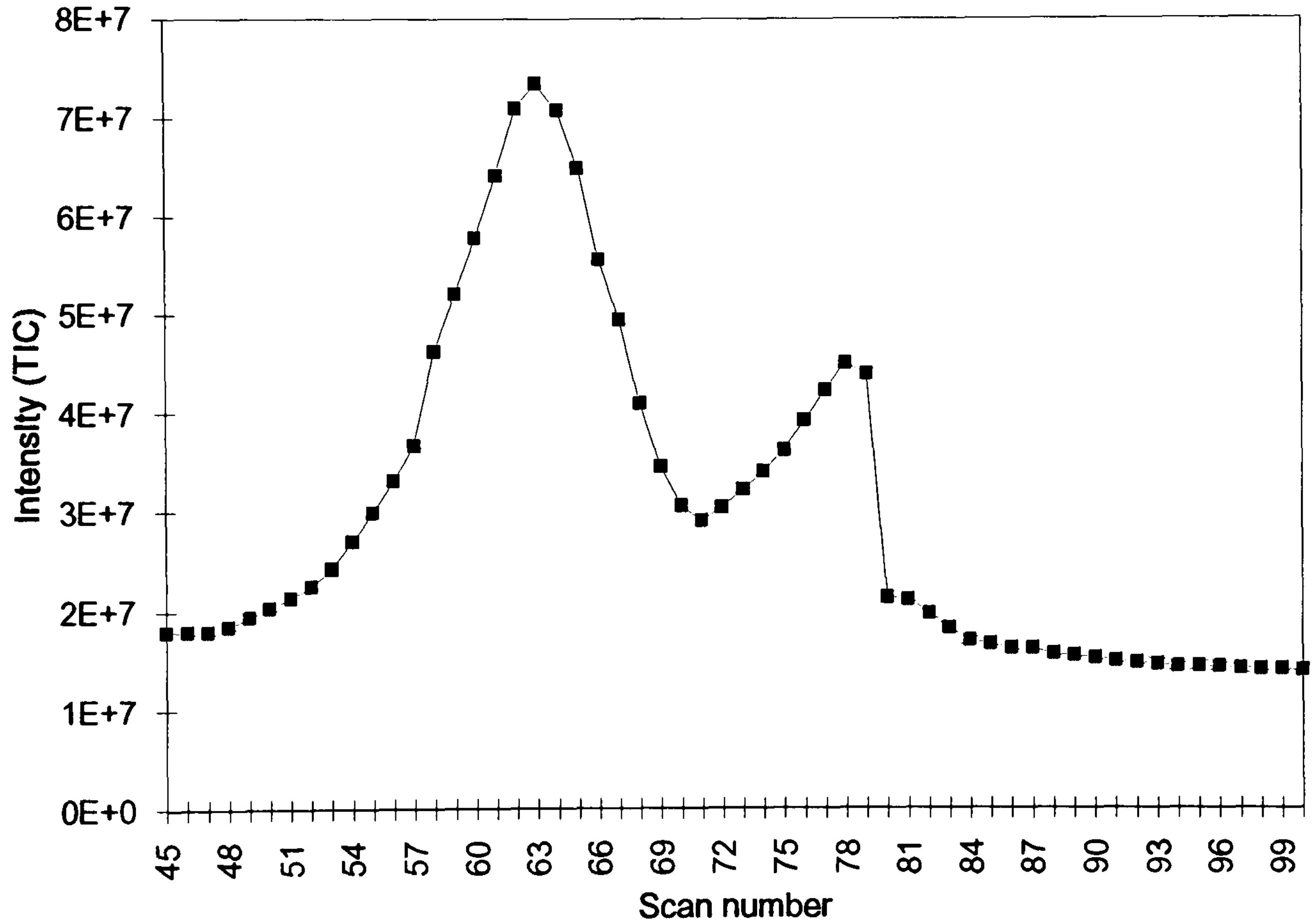


Figure 64: TIC pyrogram for data set TB19.

Data pre-treatment

The data produced from the analysis was converted to EPA format on the mass spectrometer data system and then transferred to the PC where it was converted using the program into a spreadsheet format file.

The size of the file was reduced by deletion of scans containing only background leaving scans 45-100.

Data analysis

The reduced data set was then analysed by the program using the following parameters.

Error estimate = 25495.

Minimum = 1773 (smallest base peak / 999).

Accuracy = 1×10^{-19} (default value).

The estimate for error in the data was determined by averaging the intensities for m/e 28 in the first and last scans of the data set (scans 45 and 100) and using 1% of this value as the error estimate.

The analysis ran to the maximum number of possible factors (56) indicating that the minimum value is not representative of the true minimum value in the data.

Determination of the rank of the data

The percentage variance and percentage significance level values are given in table 39.

The percentage variance figures show that the first three factors account for a substantial portion of the total variance (97%) the factors following these account for substantially smaller portions of the variance in the data. The values for percentage significance level indicate that the data is not adequately described until factor 18 or 19. These results are similar to the earlier analyses where the single components needed two factors to describe most of the variance and the percentage significance level indicated a larger number.

Factor no.	%variance	%SL	Factor no.	%variance	%SL
1	61.194	0.0	14	0.009	5.2
2	23.918	0.0	15	0.008	4.8
3	12.297	0.0	16	0.007	3.6
4	1.693	0.0	17	0.006	3.9
5	0.367	0.0	18	0.004	5.9
6	0.157	0.2	19	0.003	8.5
7	0.092	0.7	20	0.002	11.5
8	0.060	1.5	21	0.002	12.2
9	0.052	1.1	22	0.001	14.2
10	0.042	0.9	23	0.001	15.9
11	0.033	0.8	24	0.001	16.5
12	0.027	0.6	25	0.001	18.6
13	0.014	2.5			

Table 39: Table of percentage variances and percentage significance levels for the first 25 factors of the data set TB19.

The eigenvalues and standard error in eigenvalue are shown plotted in figure 65. The first three eigenvvalues appear close together and the fourth and further factors fall in importance rapidly. The standard error in eigenvalue indicates that factor 14 marks the boundary between primary and secondary eigenvectors contradicting the figure obtained from percentage significance level, previous estimates of dimensionality given by the standard error in eigenvalue have also been incorrect because of the estimate of error in the data set being too large and that is probably also the case here.

The values calculated for the real error and the estimate of error in the data are shown in figure 66 and clearly show that the error estimate gives a dimensionality of 4. Once again, the estimate of error in the data is too large and even though it is based upon the magnitude of the background signal, it doesn't identify the background at factor 3. This makes its use in determining the number of components in the system untrustworthy.

Figure 67 shows the imbedded error and indicator functions for TB19. The imbedded error fails to reach a minimum, similar to the single component cases. The indicator function does exhibit a shallow minimum at factor 19 in agreement with the value obtained from the percentage significance level values. From these two results the data can be said to be composed of 19 factors, a considerable increase over the number for the single components.

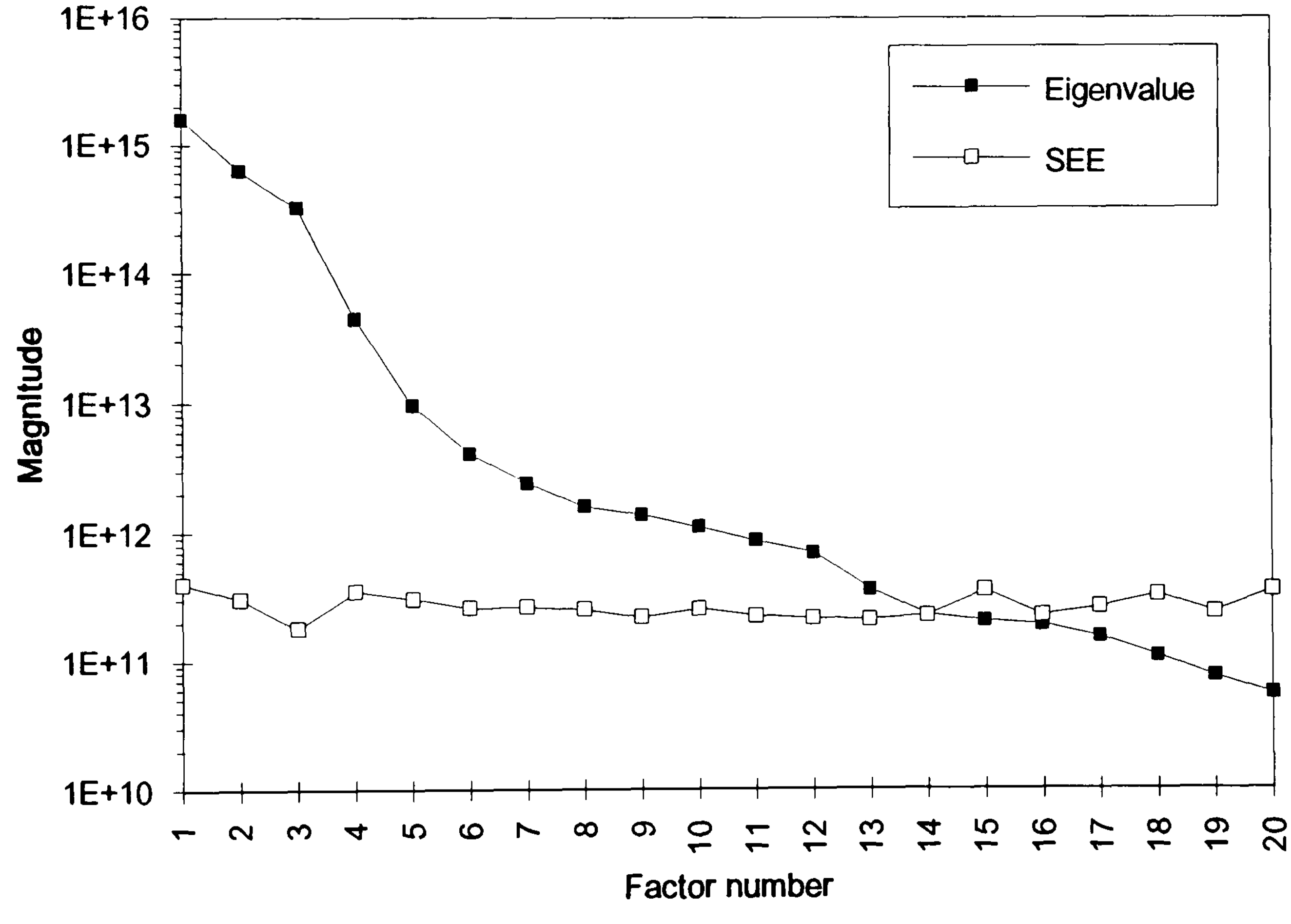


Figure 65: Plot of eigenvalue and standard error in eigenvalue (SEE) for the first twenty factors of data set TB19.

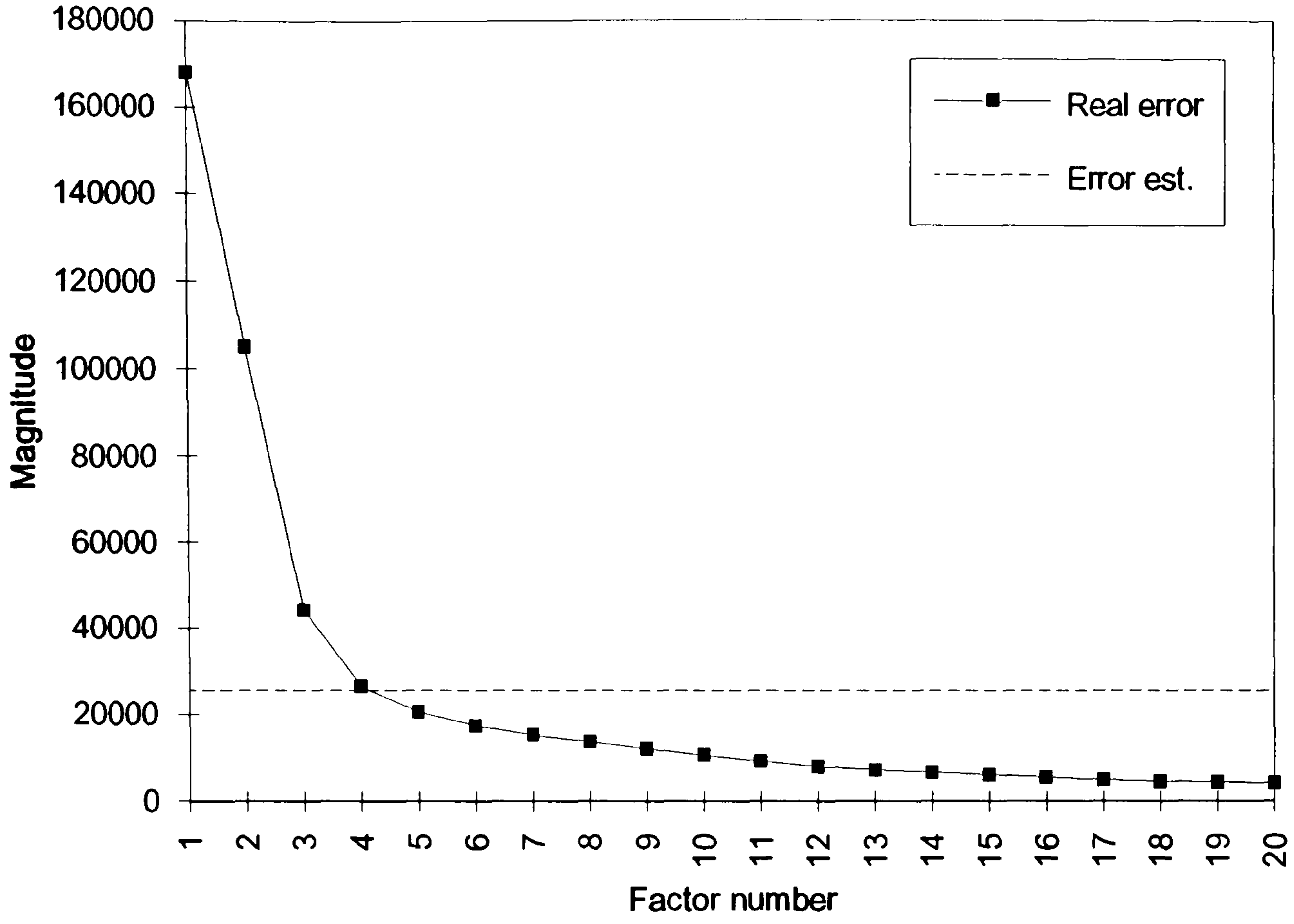


Figure 66: Plot of the real error function and the estimate of error for the first 20 factors of data set TB19.

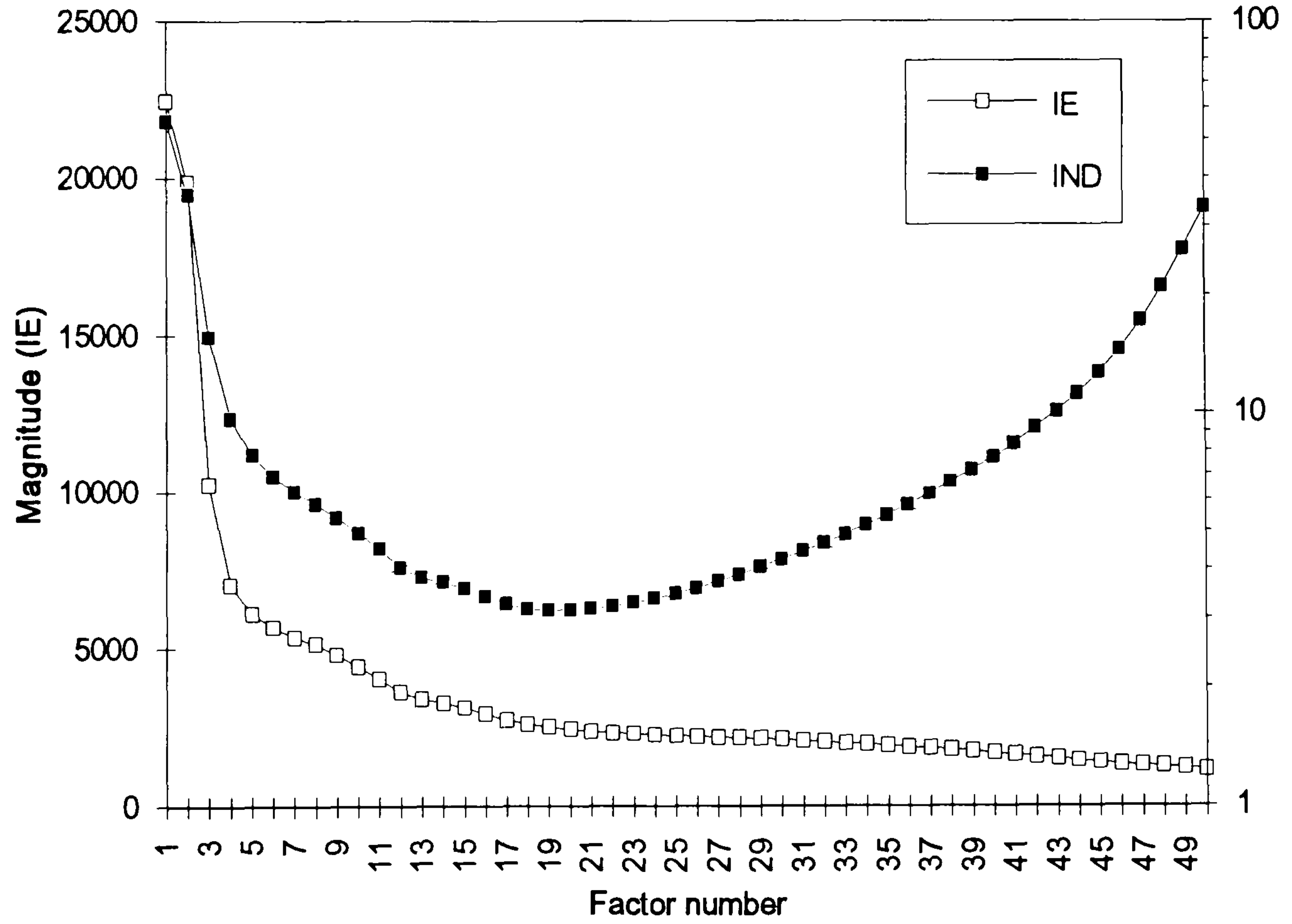


Figure 67: Plot of imbedded error (IE) and indicator function (IND) for factors 1-50 of data set TB19.

Abstract column matrix

The eigenvectors for factors 1 and 2 are shown in figure 68, from the shape of the factors it can be seen that factor 1 describes mostly the first peak in the TIC pyrogram, and factor 2 describes the second peak. This is unexpected as the first factor is always the average pyrogram, closer inspection shows that the first factor does contain a small second peak. The disproportionate size of the two peaks is explained by the amount of variance contributed to the total data set. The first peak in the TIC pyrogram is large and occurs over a number of scans. The second peak is quite small in comparison and occurs over fewer scans therefore contributing less to the data. The first factor reflects this by emphasizing the first peak in relation to the second peak.

In finding the second factor the largest source of residual variance is described, which in this case is the second peak as the first peak is well modelled by the first factor, this gives the resulting second factor showing the second peak.

Figure 69 shows the eigenvectors for factors 3 and 4. Factor 3 is related to the background due to its large constant value from scan 90 onwards. The slope during earlier scans is explained by the mixing of some of the background signal with factor 1.

Factor 4 exhibits some extremely curious behaviour that, from its position, is related to the first peak. Factors 5 and 6 (not shown) also follow a similar, erratic path. Factor 6 also has some significance for the second peak as well. The explanation for these factors can be divined with reference to their associated factor scores.

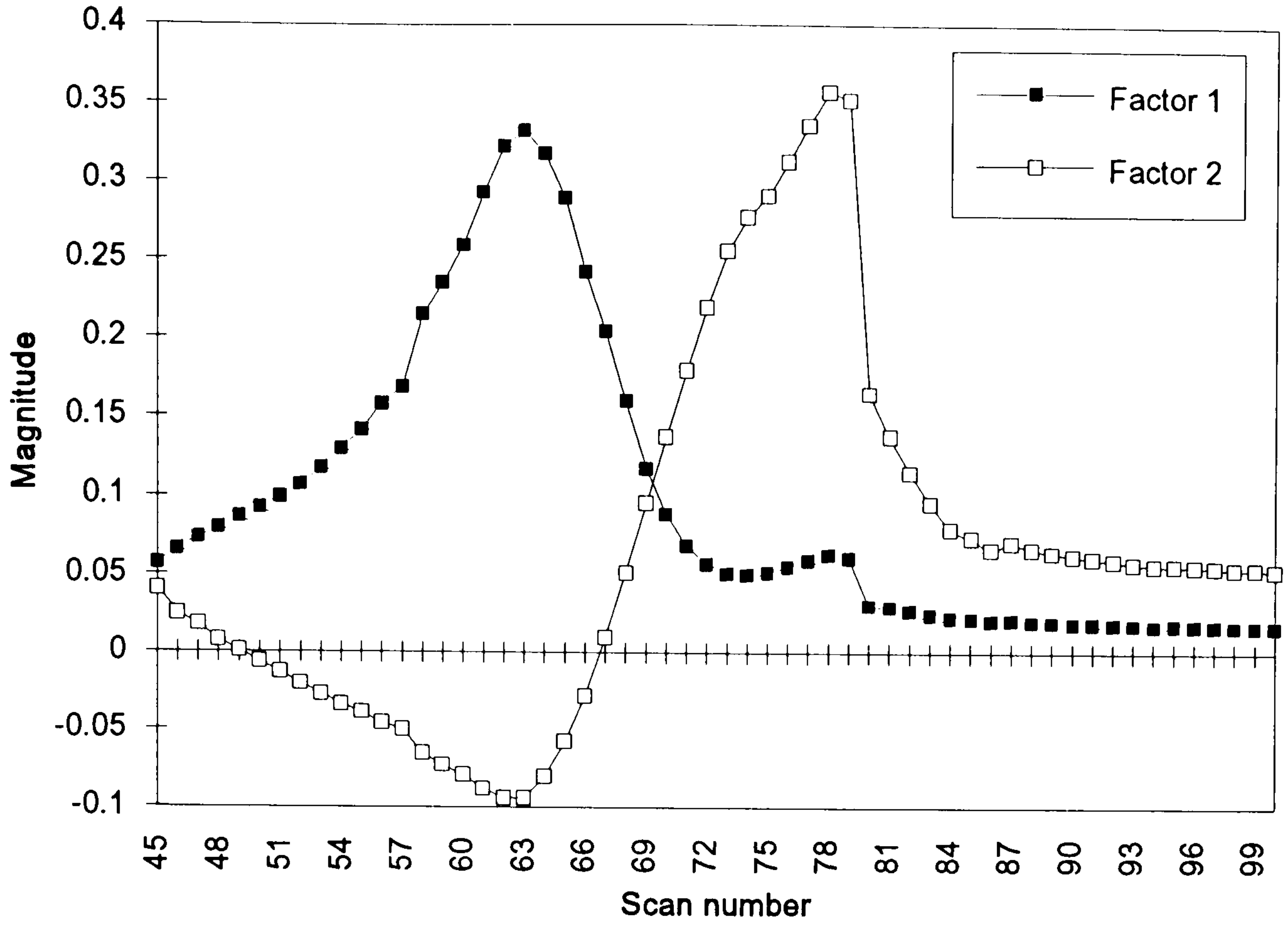


Figure 68: Plot of factors 1 and 2 for the data set TB19.

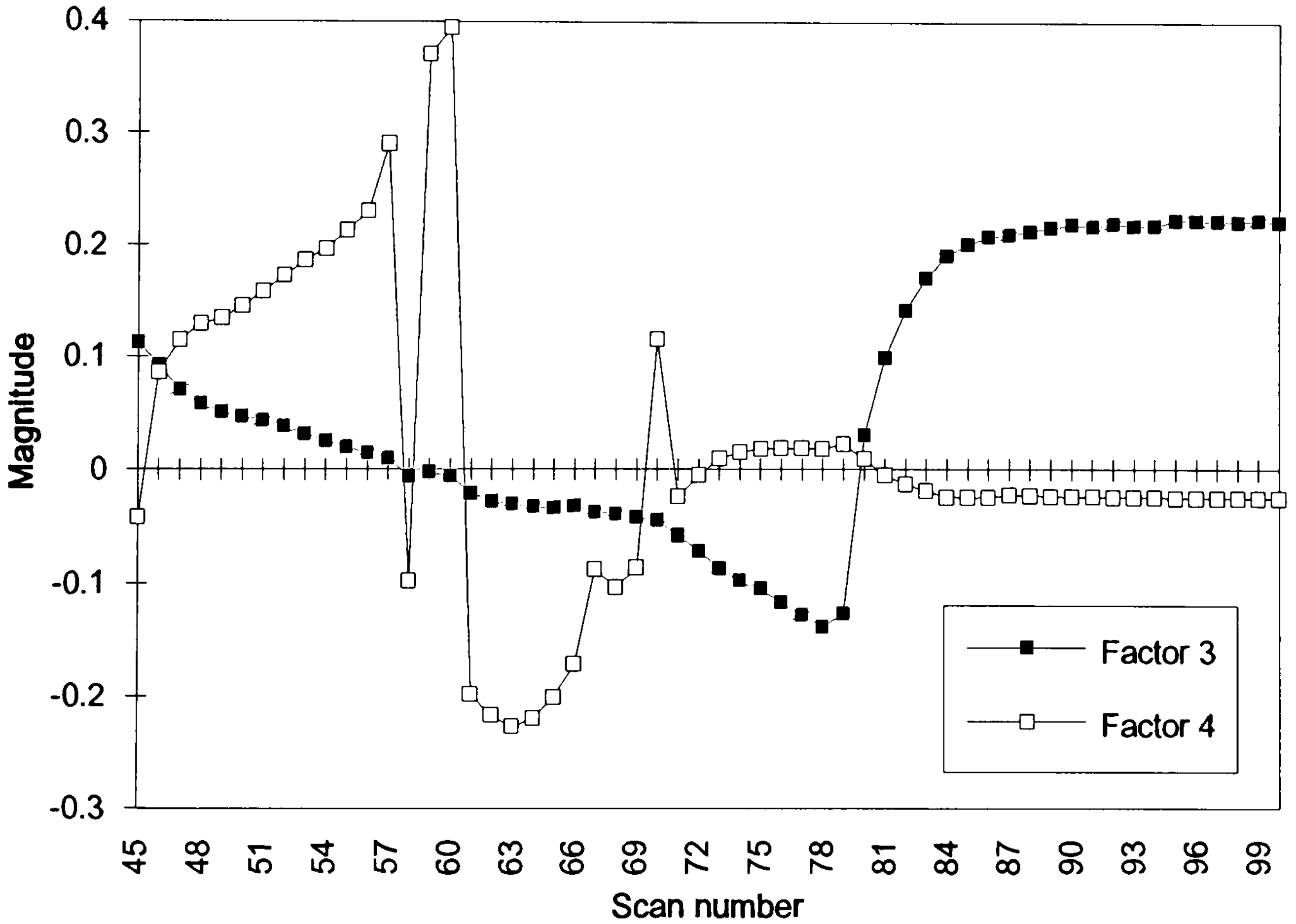


Figure 69: Plot of factors 3 and 4 for the data set TB19.

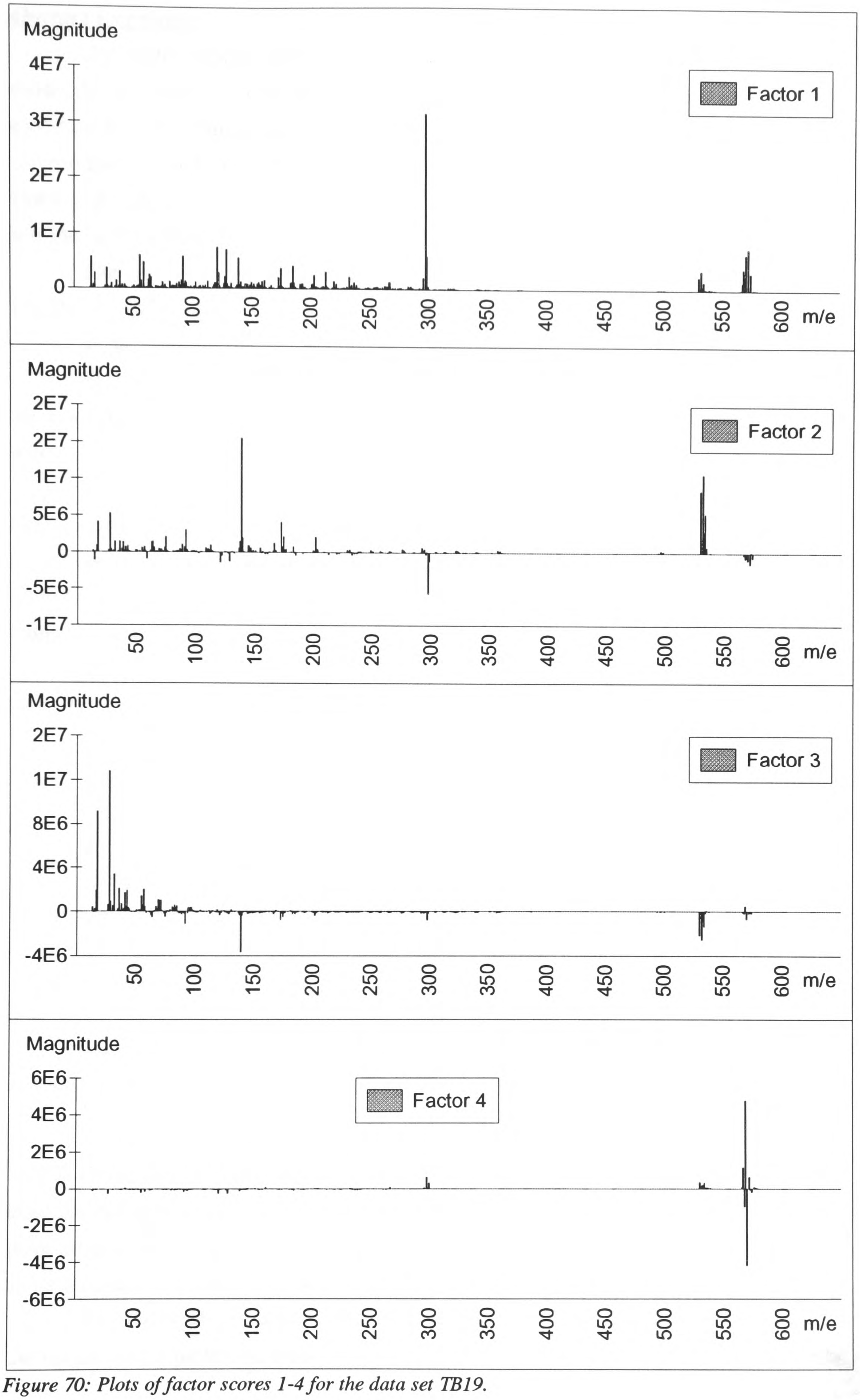


Figure 70: Plots of factor scores 1-4 for the data set TB19.

Abstract row matrix

The factor scores for the first four factors are shown in figure 70. Factor 1 shows the average spectrum that, as expected from the eigenvector, shows a strong spectrum for one component (FCA) and a weaker spectrum for the other (1-1'DCBF) and the background. The identity of the peaks in the TIC pyrogram was confirmed by looking at individual scans from the data set to show that the first peak is due to FCA and the second from 1-1'DCBF.

The second factor scores show a positive weighting for the spectrum of 1-1'DCBF and a negative weighting for the FCA. The background spectrum is also visible in the positive weightings but at a low intensity.

Factor 3 clearly shows the background spectrum including the contamination the instrument was suffering from at the time of analysis. Negative weightings for the spectra of both components are visible.

Consideration of factor four allows us to interpret the curious eigenvector observed earlier. The weightings show a strong positive peak for m/e 569 and a strong negative one for m/e 570. This indicates that they vary oppositely to each other (if one increases the other decreases, etc.) which is unexpected as both peaks arise from different isotopic substitutions of the molecular ion.

The relative intensities of the molecular ion isotope cluster were calculated using the mass spectrometer data system producing the following results.

Formula	C20 H14 O3 Fe Cl6				
Element	C	Cl	Fe	H	O
#atoms	20	6	1	14	3
#isotopes	2	2	4	3	2
Calculated spectrum					
Mass	%Intensity	Mass	%Intensity	Mass	%Intensity
566	3.07	571	24.73	576	8.97
567	0.70	572	80.34	577	2.08
568	54.56	573	19.62	578	1.29
569	13.56	574	35.27	579	0.28
570	100.00	575	8.44	580	0.09

Table 40: Table of calculated isotope abundances for the molecular ion cluster of FCA.

For data set TB11 all the isotope peaks were visible down to m/e 579 at 0.28% abundance but for data set TB19 the smaller sample size meant that the smallest isotope peak visible was m/e 574 at 35% abundance. This means that the peak at m/e 569 with the strong negative weighting in factor score 4 should not exist as its abundance is only 13.6%.

To investigate the source of the factor further, single ion pyrograms for both the masses were plotted and the results shown in figure 71. The pyrograms show that the signal at m/e 570 is incorrectly assigned to m/e 569 for some scans. The three major isotopes that make up m/e 570 and their exact masses are:

$C_{20}H_{14}O_3\,{}^{56}Fe\,{}^{35}Cl_5\,{}^{37}Cl = 569.8394$

$C_{20}H_{14}O_3\,{}^{54}Fe\,{}^{35}Cl_4\,{}^{37}Cl_2 = 569.8411$

$C_{20}H_{14}O_3\,{}^{58}Fe\,{}^{35}Cl_6 = 569.8407$

The identification of the actual m/e value for a peak is calculated by the mass spectrometer data system using the peak centroid. If, when the instrument was calibrated, the centroid for the standard was moved slightly higher than its true position so that the 569.8 peak would be artificially read at around 569.5, then depending upon which side of the 569.5 the centroid falls is whether the peak is registered as 569 or 570. The problem is particularly bad for the high m/e values because of the exponential down scan being used. Factor 4 can be assigned to describing the variance brought about by the poor calibration of the instrument.

Factor 5 (not shown) exhibits similar behaviour for m/e 567 and m/e 568 and factor 6 (not shown) describes the inaccuracies for many m/e values over the whole of the data set.

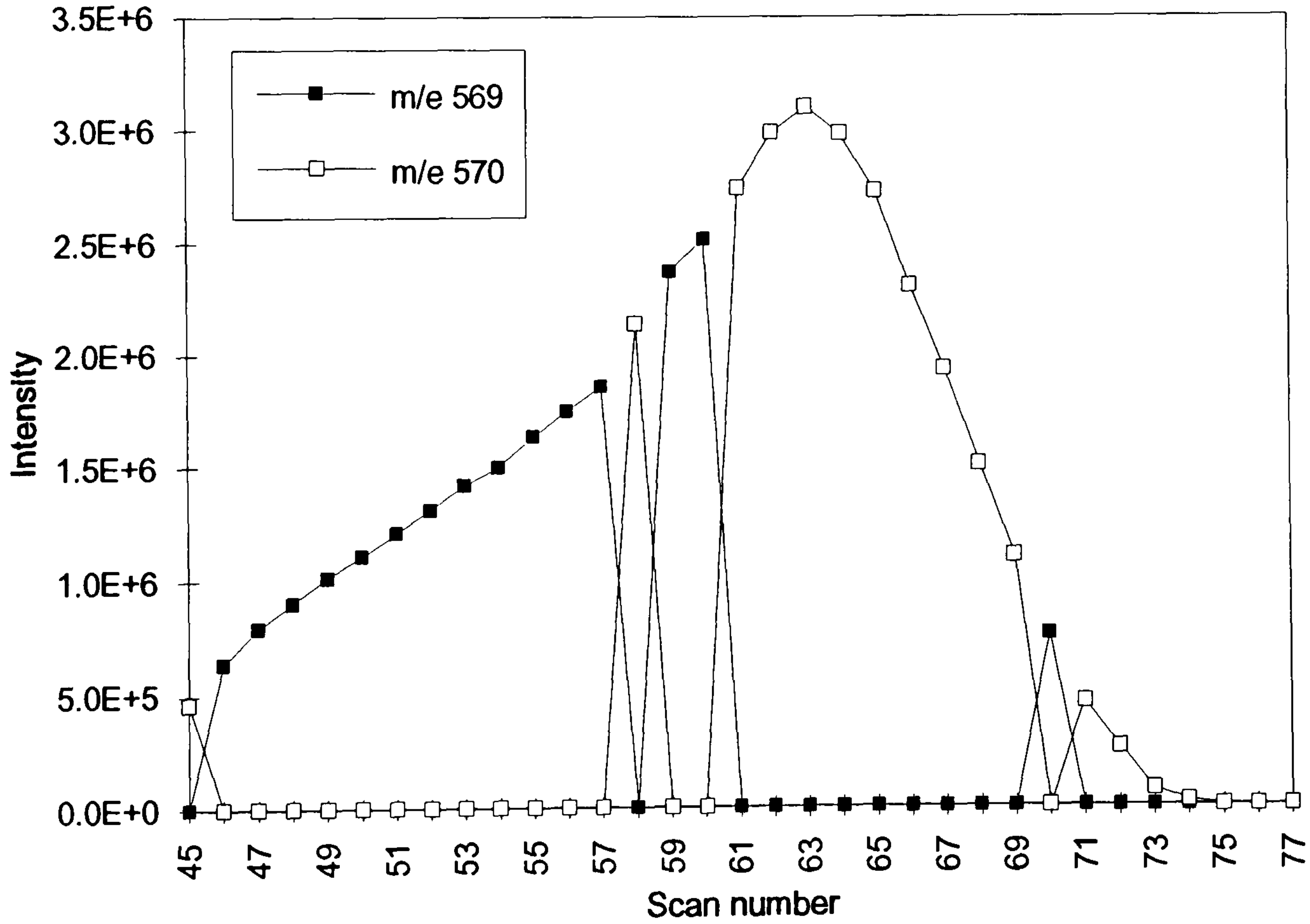


Figure 71: Single ion pyrograms for m/e 569 and m/e 570 for the data set TB19.

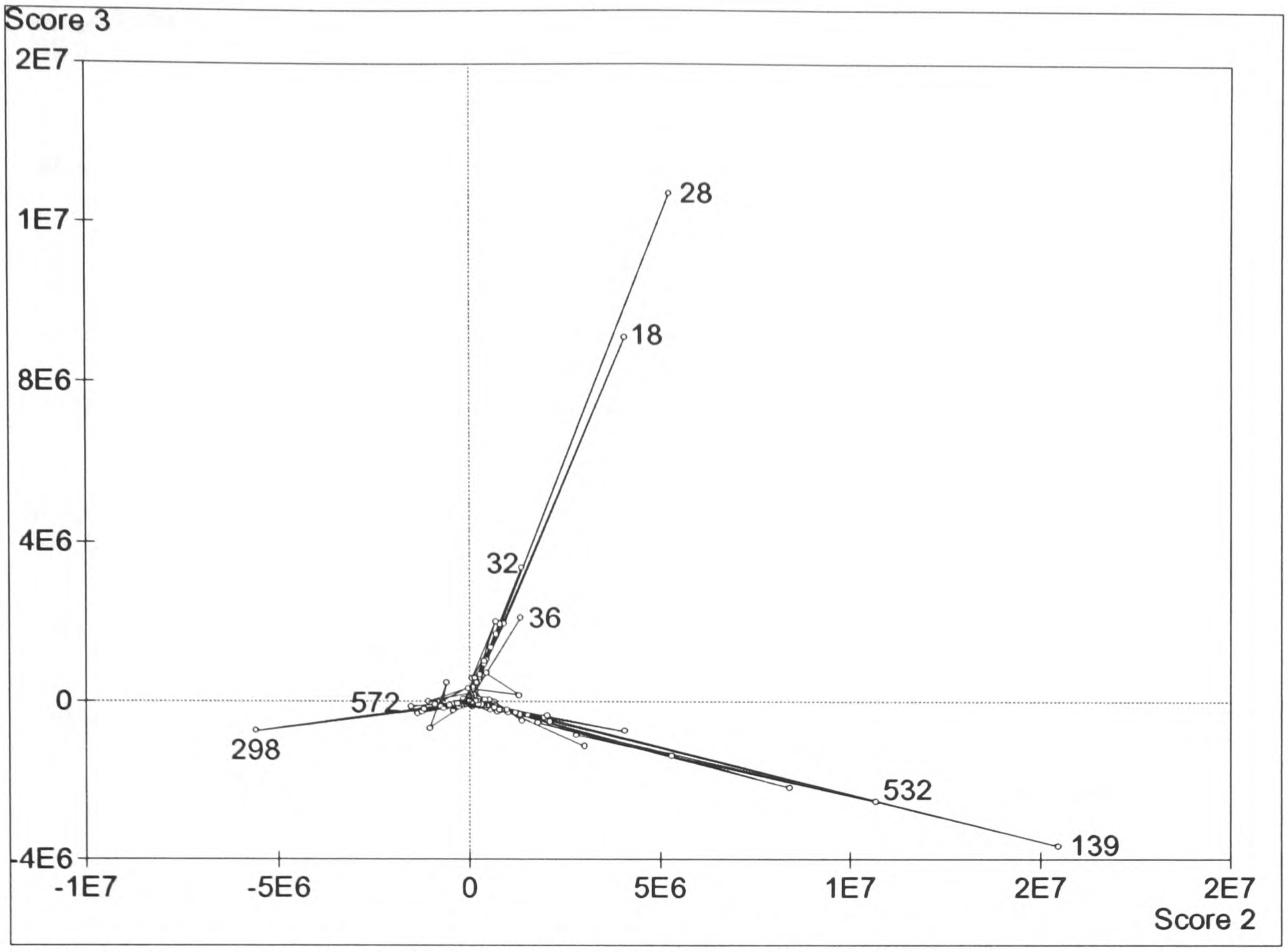


Figure 72: Plot of scores for factor 2 versus factor 3 for the data set TB19.

Despite the errors due to poor calibration and the extra factors introduced into the data by them, it is still possible to obtain a good differentiation between the sample components and the background. This can most clearly be seen when the scores for factors 2 and 3 are plotted against each other as shown in figure 72. The different directions for each of the two substituted ferrocenes and the background are clearly visible.

The effect of dimensionality on target testing

The effect of choosing a number of factors to describe the data upon target tests was studied by using a test vector of a known component (FCA, data set TB11S, scan 49) and target testing using different numbers of factors to model the data. The FCA vector was chosen as it was background subtracted, reducing the possibility of any background mixing in with it, and the available analyses of the other component (1-1'DCBF) were made when the instrument was suffering from source contamination, thus the vector used was the purest spectrum of FCA available.

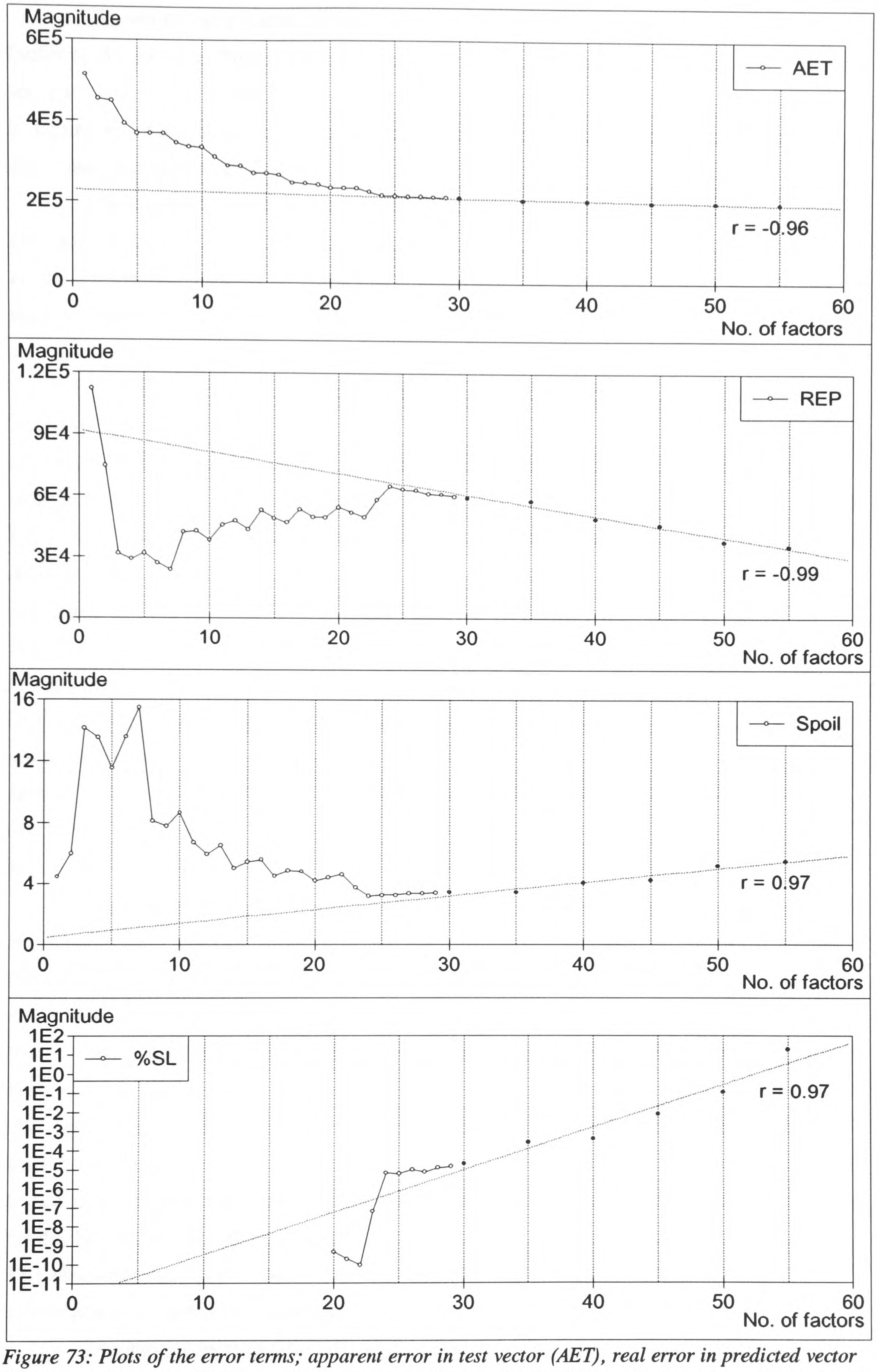


Figure 73: Plots of the error terms; apparent error in test vector (AET), real error in predicted vector (REP), spoil function (SPOIL) and percentage significance level (%SL) against number of factors used to model the data for data set TB19 with test vector TB11S scan 49.

The vector was tested using consecutive factors from 1-30 and then every fifth factor to 55 factors. The error estimate used for the vector was that used for the data set, i.e. 207200. The error terms from each test were collated and the results shown in figure 73. The regression lines were calculated using only the filled markers and are linear regression except for the %SL chart, which uses exponential regression.

The apparent error in the test vector (AET) falls as larger numbers of factors are used to describe the data until the curve levels out at factor 24. Two points of note are that when the curve flattens out into a straight line it still continues to fall, but slowly. The second point is that the curve becomes a straight line at factor 24 and not at factor 19. The AET term is an estimate of the error in the test vector based upon the difference between the test and predicted vectors. It would be expected that the function would reach a minimum at the correct number of factors necessary to describe the data and then be level from that point onwards. The observed behaviour is explained as follows; the test vector in this case, whilst being the purest spectra of FCA available, still contains some background signal and also, as it was taken from a different analysis, then there are slight variations between it and the spectra of FCA contained in data set TB19. This means that the fit produced at the theoretical 19 factors is not quite perfect and that the addition of a few excess factors describing random error allows a small amount of extra 'freedom' for the target test to model the discrepancies between the test and the true vector. The fact that the graph does not completely level out supports this hypothesis as the addition of extra factors of random error will allow a linear improvement in the fit of the vector for each factor added and also the improvement in fit will be small as only very small factors are involved.

The real error in the predicted vector (REP) is calculated using the real error estimate for the relevant number of factors and the transformation vector. It estimates the difference between the predicted vector and a theoretical pure vector containing no error. The graph shows that the function falls rapidly for the first three factors and then climbs slowly and erratically to a straight line, starting at 24 factors, which then falls slowly. The graph is interpreted as indicating that the predicted vector containing the least error is produced using seven factors and the addition of further factors introduces error into the predicted vector. When it is considered that the factor being tested is the major component in the data, this result is reasonable, it indicates that to model the behaviour of FCA only seven factors are needed out of the theoretical 19. The chart also indicates that after only three factors the largest portion of the information required to model the component has been found.

The spoil function (SPOIL) is calculated from the ratio between RET and REP. The RET function is almost identical to the AET plot resulting in the SPOIL function mirroring the REP function.

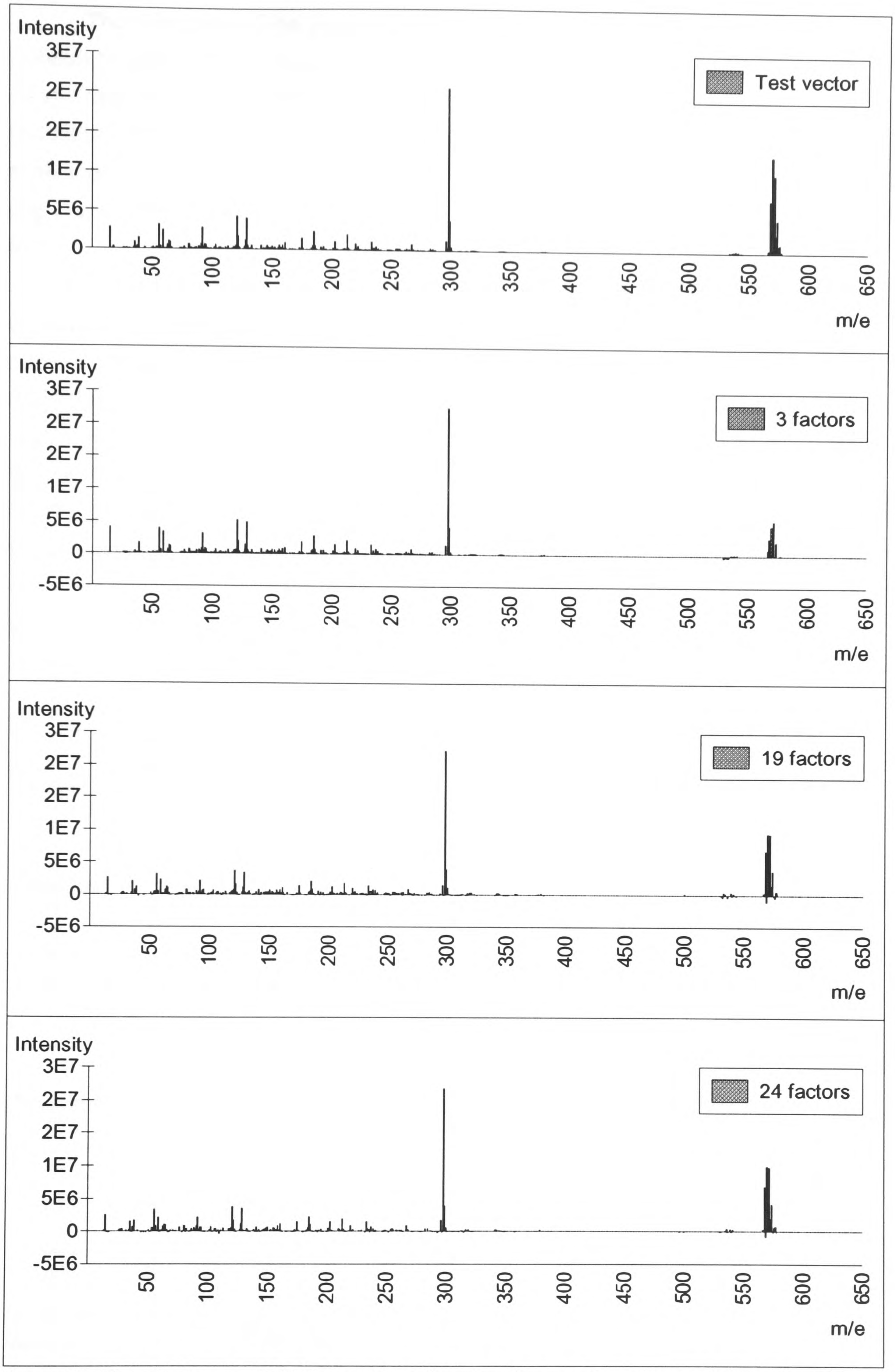


Figure 74: Plot of test vector (TB11S scan 49) and predicted vectors from data set TB19 using 3, 19 and 24 factors to describe the data.

The behaviour of the percentage significance level (%SL) is complex and difficult to interpret. No significant value is produced for the %SL until after factor 19 and then it behaves erratically until factor 24 after which it rises steadily. When 55 factors are being used to model the data it finally reaches a significant level. The explanation for this is not clear but is believed to be related to the large numbers of degrees of freedom in the test vector and assumptions made in the derivation of the F-test.

The results of target testing using varying numbers of factors to describe the data are shown in figure 74. It can clearly be seen that the spectrum is modelled extremely well even with only three factors, the molecular ion cluster is not very well fitted due to the problem with the calibration of the instrument but the rest of the spectrum is reproduced very well. The addition of the extra factors adds very little to the spectrum, the most obvious change is in the molecular ion cluster, which better fits the test vector and some small intensity changes in the lower m/e region are also visible, certainly the change from 19 to 24 factors is almost negligible and must be restricted to changes in the smallest peaks that may be from the FCA or from another source but are of little or no consequence for identification of the component.

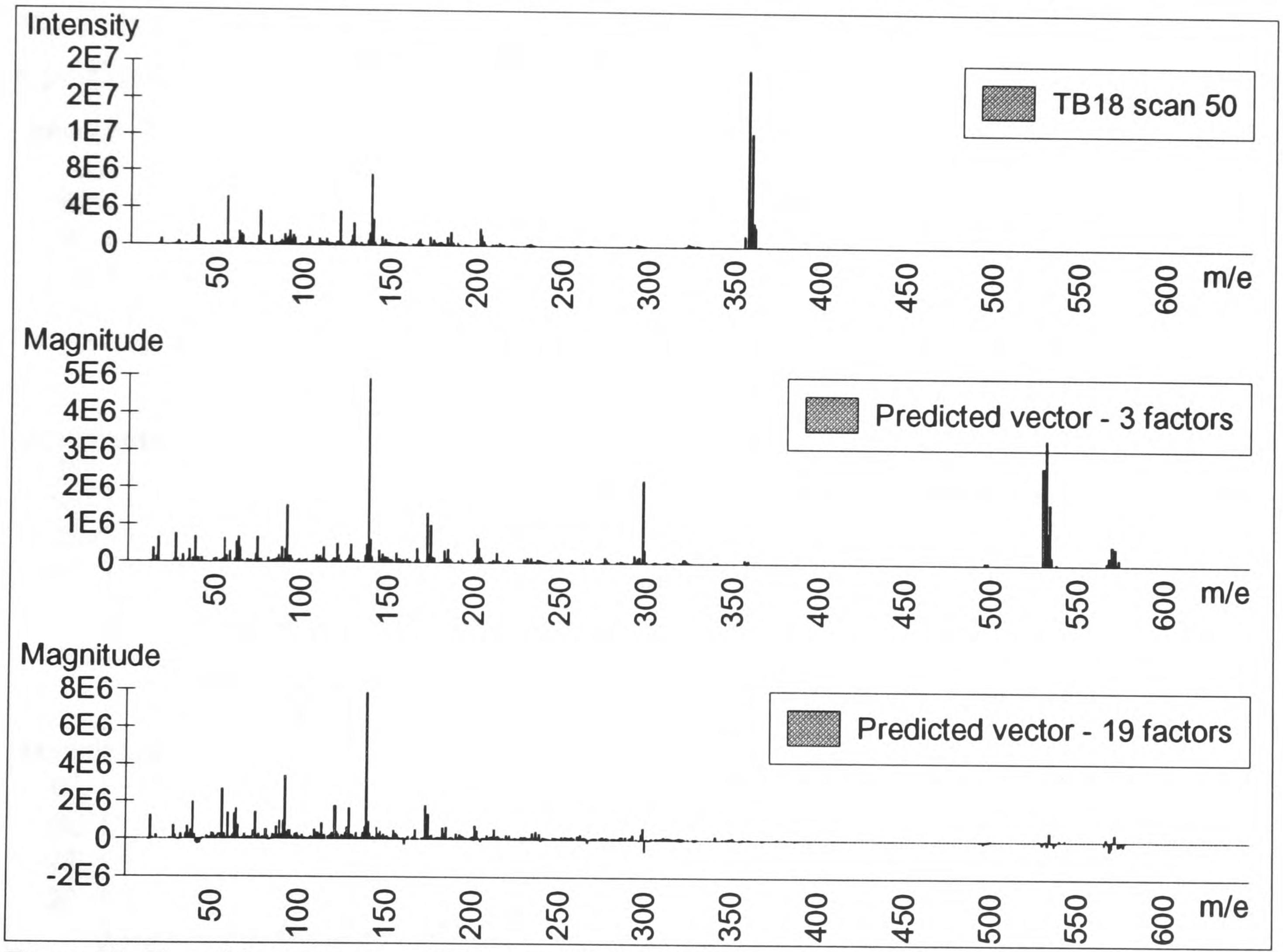


Figure 75: Plot of the test vector, TB18 scan 50 (3,4-DCBF), and the predicted vectors resulting from testing data set TB19 (1-1'DCBF & FCA) using 3 and 19 factors to model the data.

In order to investigate the results produced from testing for a component that was not present in the data, target tests were performed using data set TB18, scan 50 as a test vector. This data set, containing the pyrolysis of 3,4-DCBF, contains a lot of the peaks found in the other two substituted ferrocene spectra and should therefore constitute a worst case for testing. The results of these target tests are given in figure 75, which shows how poor the fits of the predicted vectors are. The peak at m/e 139 has been fitted as it is common to both 3,4 DCBF and 1-1'DCBF, as well as a few other common peaks, but the absence of any signal for the molecular ion clearly indicates that the test vector is not a component of the data set. The test using three factors includes peaks for the molecular ions of both the real components and the base peak for FCA at m/e 298. The test using 19 factors contains only the lower mass peaks, showing that the extra freedom allowed by a 19 dimension space, as opposed to the restricted 3 dimension space, has enabled a closer approximation to the test vector. It is apparent in both cases that the test vector is not a factor.

To determine the feasibility of using target testing as an investigative tool a test vector was created that contained m/e 139 only with an intensity of 1, the rest of the values being zero, this is referred to as a uniqueness vector and is normally used to test for any sample or variable of the data that behaves uniquely. The result of a uniqueness test is regarded as positive when the predicted vector contains unity for the selected value and zero or near zero for the other values. In this particular case the search is not for unique factors but to gain insight into the values that are linked with a particular m/e value.

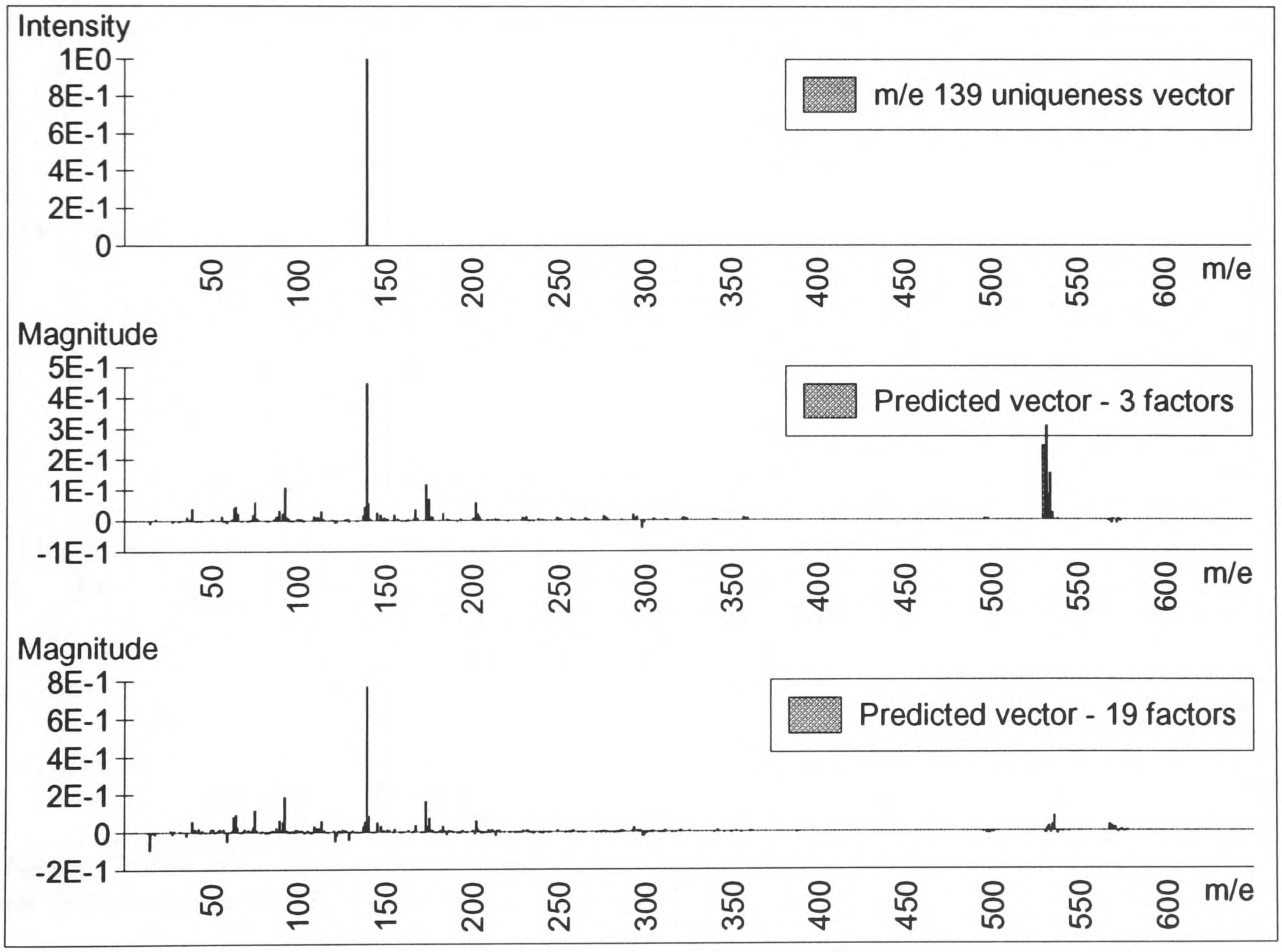


Figure 76: Plot of the test vector (m/e 139 uniqueness) and the predicted vectors resulting from testing data set TB19 using 3 and 19 factors to model the data.

The results from testing the uniqueness vector are contained in figure 76. The two predicted vectors show similar spectra in their low m/e portions, the 19 factor vector having fewer small peaks, but the molecular ion for 1-1'DCBF is visible only in the predicted vector using 3 factors. This seems to indicate that the accuracy of the test is diminishing with the increasing number of factors, in fact, what is happening is that the addition of the extra dimensions allows the test to produce only those peaks related to the m/e 139 peak. It was shown in the earlier discussion of the single component analysis of 1-1'DCBF that the m/e 139 peak is produced from the thermal decomposition of 1-1'DCBF and not via volatilization and the molecular ion peak, thus, the molecular ion peak does not appear in the predicted vector. In the case of the 3 factor test the entire spectrum from all sources appears because there are insufficient factors to differentiate between anything but the three major sources of variance, i.e. the two substituted ferrocenes and the background.

It is important to note that the peaks from FCA do not feature in either of the predicted vectors and this technique of uniqueness testing is therefore valuable in determining the individual components present in the data, though care must be taken in selecting how many factors to use in the test.

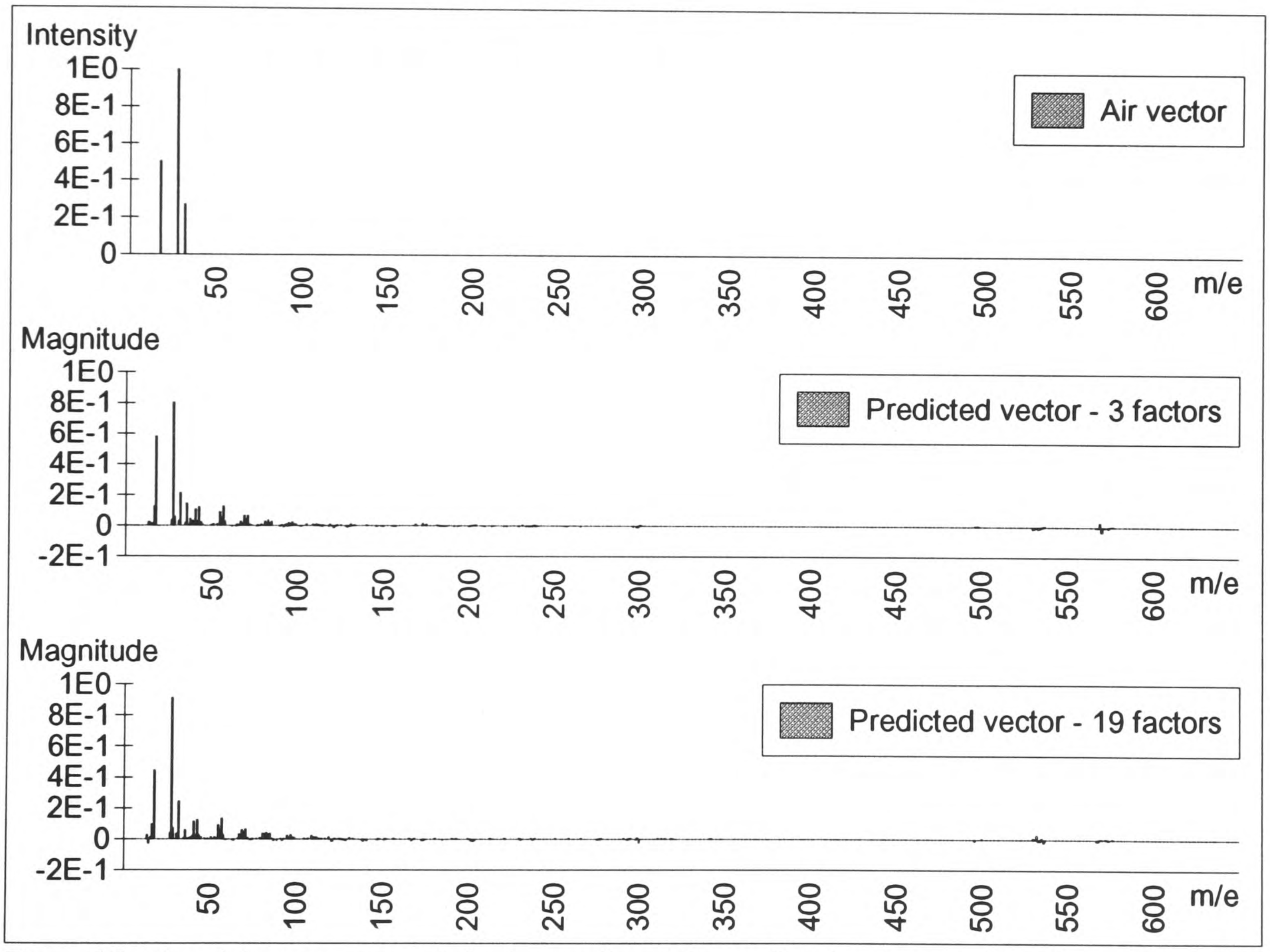


Figure 77: Plot of the test vector (air background) and the predicted vectors resulting from testing data set TB19 using 3 and 19 factors to model the data.

The background was also tested for, using a variation on the uniqueness test. The test vector was constructed using the three most abundant peaks from the air, m/e

28, m/e 18 and m/e 32, with the m/e 28 peak having an intensity of 1 and the others scaled to their normal relative intensities. This air background vector was tested as before, using 3 and 19 factors and the results are shown in figure 77.

The background spectrum can be seen to contain the hydrocarbon contaminant seen in the abstract factors. The use of 19 instead of 3 factors has produced very little change in the predicted vector. This indicates that one factor describes the background very well. The only peak that has changed is that for m/e 18, the signal for water was found to require a factor of its own when the abstract factors of FCA were studied and the change in intensity of the peak between the 3 factor and 19 factor results suggests that something similar is the case here.

4.6.2.2. Iterative target testing

The results from the uniqueness and air background test vectors used on data set TB19 indicated that testing for a single peak of a particular component resulted in a complete approximation to that component. There was evidence to suggest that the resulting vector was not completely pure, for example, peaks from the background and negative peaks were visible. This led to the idea that an iterative procedure might allow an analysis to start from a single mass value and to converge to a pure factor.

Re-submission of the result from a target test for testing again results in the production of a predicted vector that is identical to the test vector. This is obvious when it is seen that the predicted vector is the closest match to the test vector that can be made within the constraints of the original data. By resubmitting this best fit vector the data can be made to fit the vector perfectly as it was created from the data.

In order for the process to converge towards a pure vector the vector must be modified before re-submission. The best modification would be to delete any peak that does not belong to the true factor. This, however, requires a foreknowledge of the true factor that makes the search for the factor redundant. An alternative approach is to modify the vector according to some rule or rules that will perturb the vector sufficiently to allow the target testing process to converge towards the true factor.

Chemical selection rules

By far the best rules for modifying of the test vector are based upon chemically valid criteria. Work performed by Vandeginste et al.[102], looking at the elution profiles of components in HPLC, used two selection rules for modifying the test vector. The rules were that only one maxima was possible (a single component in the chromatogram) and values below a threshold were set to zero. In the case of the mass spectra there can be more than one maxima, excluding the use of that rule, but the setting of values below a threshold to zero is applicable to the mass spectra.

To investigate iterative target testing the program was modified so that the predicted vector from the target test could be modified and then re-submitted as the test vector for the next test. For the first tests the vector was modified by the removal of any negative components, as the mass spectra cannot contain any negative peaks.

In order to determine the progress of the convergence some measure of the similarity of the test vector and predicted vector is necessary. Convergence is deemed to have been achieved when the difference between the test and predicted vectors is constant and the predicted vector is not changing any further. Two measures of vector similarity were implemented to quantify this, the product-moment correlation coefficient and a vector difference.

The formula used to calculate the correlation coefficient is given in equation 48, where x_l is the l^th^ member of $\mathbf{x}$, the test vector and $\hat{x}_l$ is the l^th^ member of $\hat{\mathbf{x}}$, the predicted vector. The term r_{corr} is used instead of r to avoid confusion with the number of rows.

$$r_{\text{corr}} = \frac{\sum_{l=1}^{r}(x_l - \bar{x})(\hat{x}_l - \bar{\hat{x}})}{\sqrt{\left(\sum_{l=1}^{r}(x_l - \bar{x})^2\right)\left(\sum_{l=1}^{r}(\hat{x}_l - \bar{\hat{x}})^2\right)}} \tag{48}$$

The vector difference is calculated from equation 49 for all positive values of $\hat{\mathbf{x}}$ only.

$$\mathbf{x}_{\text{diff}} = \sum_{l=1}^{r} |\hat{x}_l - x_l| \tag{49}$$

The test vectors used in the course of the investigation and a description of their contents is given in table 41 below.

Name	Description
TV4	Uniqueness vector, m/e 139 = 1 - M$^+$ for 1-1'DCBF
TV5	Air background, m/e 18 = 0.5, m/e 28 = 1, m/e 32 = 0.27
TV6	Uniqueness vector, m/e 370 = 1 - peak not in either component
TV7	Uniqueness vector, m/e 142 = 1- weak FCA peak, not present in 1-1'DCBF
TV8	Uniqueness vector, m/e 542 = 1 - M$^+$ for FCA
TV9	Uniqueness vector, m/e 56 = 1 - common to both components
TV10	Uniqueness vector, m/e 139 = 1908700 - scan 68 intensity
TV11	Uniqueness vector, m/e 298 = 4803000 - scan 68 intensity
TV12	Air background, m/e 28 = 384625, m/e 32 = 96156, m/e 18 = 293276 - scan 68 intensity
TV13	Uniqueness vector, m/e 56 = 2015279 - scan 63 intensity

Table 41: Table of test vectors used in the investigation of iterative target testing.

For the first experiment, using only the zeroing of negative values, the test vectors TV4, TV6, TV7, TV8 and TV9 were tested to a maximum of 10 iterations and the results saved after each iteration. Four factors were used to model the data as this number of factors describes over 99% of the variance in the data. The vector difference is shown plotted against iteration number in figure 78. The results for test vector TV6 are not shown as the test, for a peak not present in either component, produced an all zero vector after the first test.

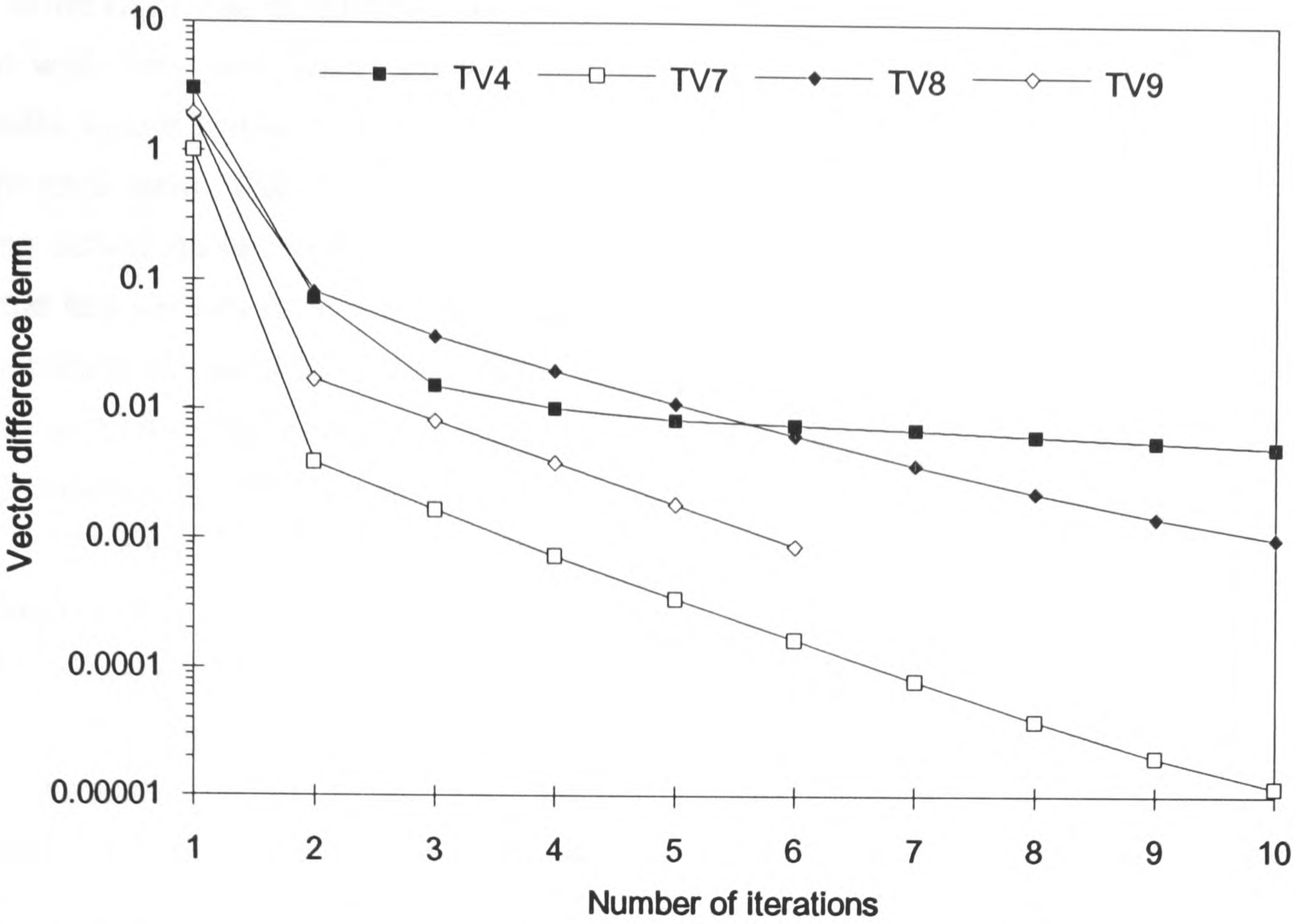


Figure 78: Plot of the vector difference term against iteration number for test vectors TV4, TV7, TV8 and TV9 resulting from iterative target testing modifying only negative values.

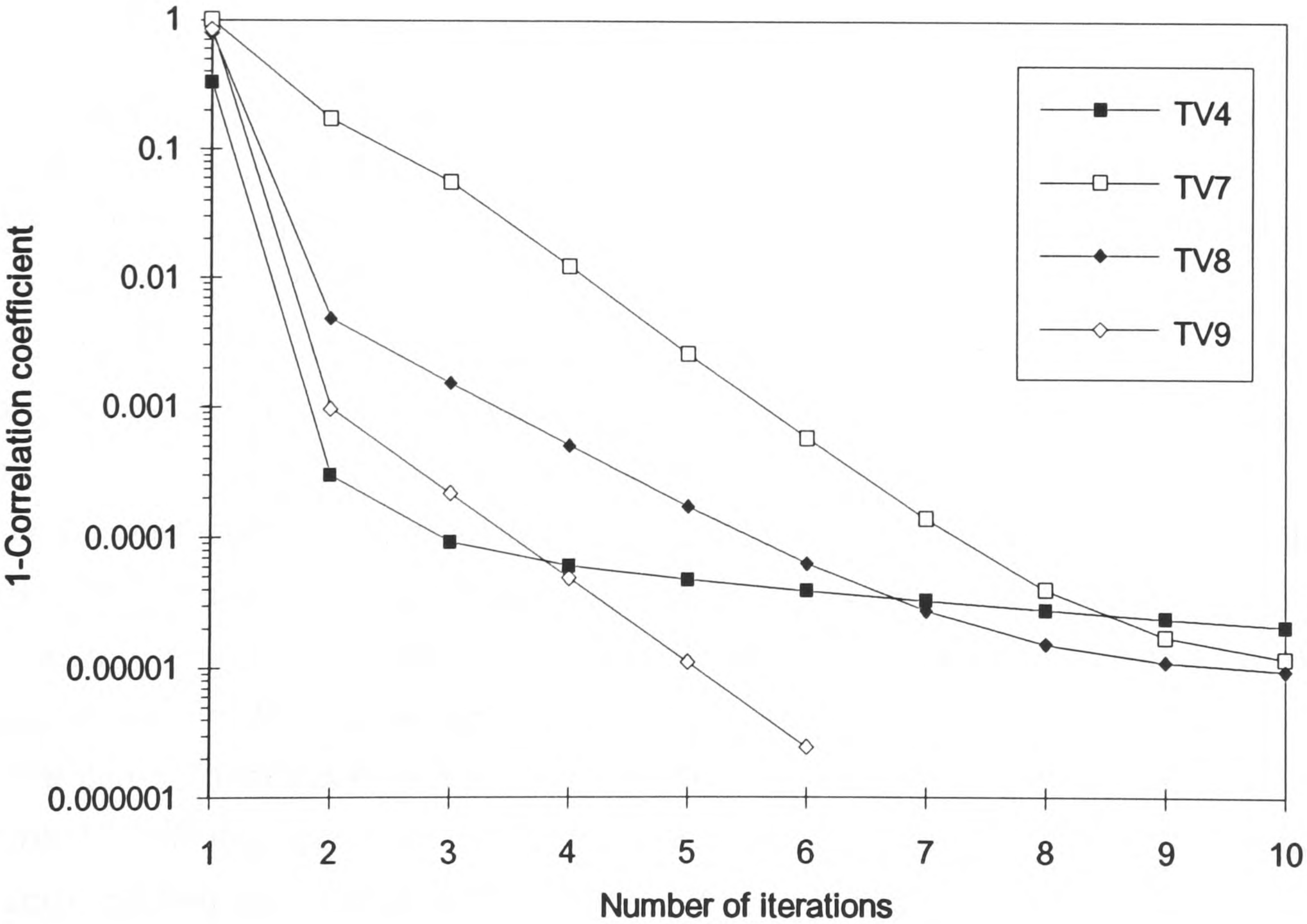


Figure 79: Plot of the correlation coefficient against iteration number for the test vectors TV4, TV7, TV8 and TV9 resulting from iterative target testing modifying only negative values.

The vector difference term shows that the test vectors do not all converge at the same rate. The steep fall after the first factor is expected as all the test vectors start with very few significant values, after the first test many of the m/e points contain values as the best fit vector is produced from the data, resulting in a large difference term. The changes following the first fall are smaller as the vector now being submitted is much closer to the true factor. From the second iteration onwards all the test vectors follow the same path except for TV4, which rapidly levels off to an almost horizontal line, the values for TV8 also show a tendency towards this behaviour as it approaches the tenth factor. The remaining two factors show no sign of abatement in their logarithmic fall.

The values for the correlation coefficient are shown plotted against iteration number in figure 79. The graph shows similar results to figure 78, especially for TV4 and TV9. TV8 now shows its fall levelling off more clearly and TV7 now also exhibits the beginning of similar behaviour. The difference is attributed to the neglect of any negative values in the predicted vector from inclusion in the calculation of the vector difference. This difference in behaviour, between the vector difference term and the correlation coefficient, indicates that for TV4 and TV9 both positive and negative values are changing by similar amounts, and that for the other two vectors the negative values are changing more than the positive ones. This might indicate that the vector is moving in the direction intended.

The target test is a least squares fit of the test vector to the data and will try to minimize the difference between the target and the predicted vectors. This can be visualized as the orientation of a vector within a space defined by the data. The orientation of the vector in a particular direction will result in a value for each point in the vector according to the projection of the data points on to the vector. The target test minimizes the differences between the projections and the test vector. In a vector containing only one non-zero value the least squares difference process tries to fit the data to all the vector points, including the zero values. The data was decomposed using covariance about the origin giving the zero value a special significance. If a test vector is composed of only zeros then the result is always an identical vector. This means that in the least squares process all the points in the vector fit perfectly except the non-zero point. The vector will move towards the orientation that will allow it to adopt the value in the test vector but in doing so will force other points in the vector to have non-zero values thus producing an error between the test and predicted vector. The final orientation produced from the least squares process will balance the 'pull' of the zero and non-zero components in the vector producing a vector that points close to the right direction but is not quite perfectly aligned.

The intention of the iterative target testing process is to modify the points that are causing the vector to point in the wrong direction so that the vector can orientate itself correctly. If the negative points are changing more quickly than the positive points then, as the true factor must have all positive values, so long as the negative values are reducing in size, the vector must be moving towards the true factor orientation.

Determination of satisfactory convergence was checked by calculating the difference between the last two iterations of TV4. This is shown plotted in figure 80. The difference spectrum shows on the positive side (being added to the vector) the spectrum of FCA and the background; the negative side shows m/e 139 and the molecular ion cluster for 1-1'DCBF. This suggests that the vector is not converging at all but is in fact moving further away from the pure 1-1'DCBF spectrum.

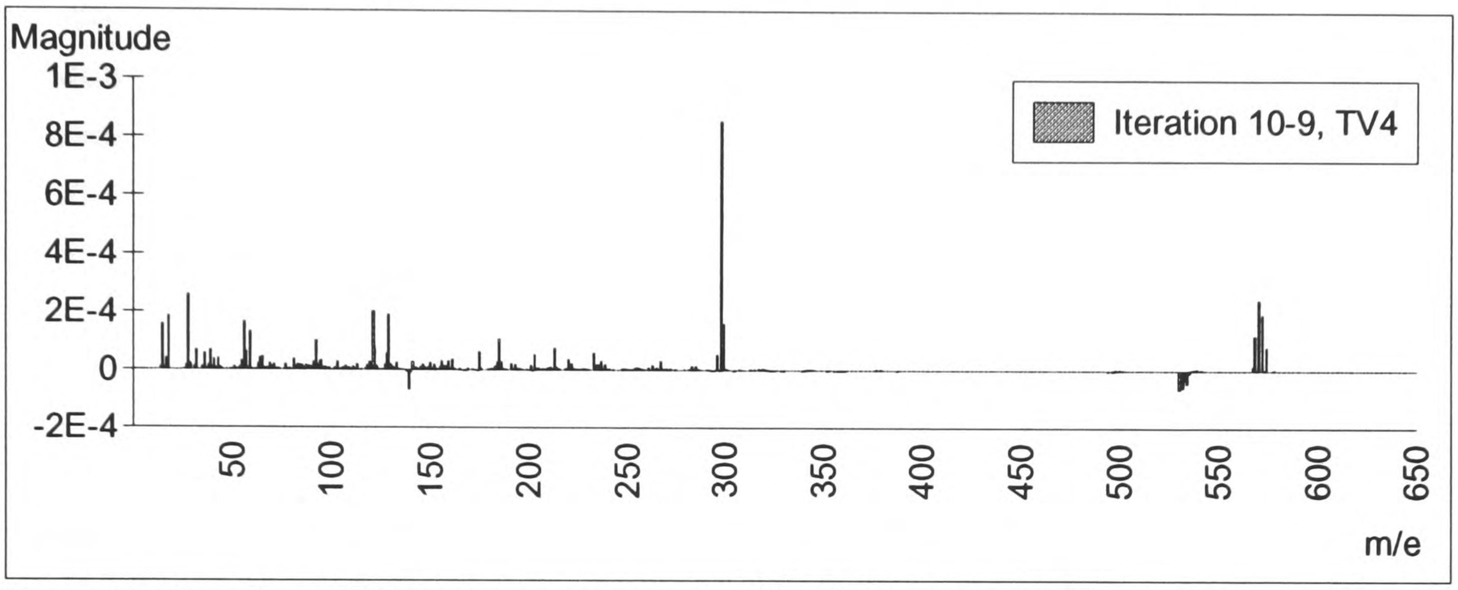


Figure 80: Plot of the difference between the tenth and ninth iterations of TV4.

ı ne apparent dıvergence of the test vector suggests that the zeroing of negative values is not sufficient to perturb the vector and allow it to move towards the true factor. A close consideration of the spectrum of FCA reveals that there is also a peak for m/e 139 present and so the addition of the chlorendic spectrum to the 1-1'DCBF spectrum is explained and the vector was converging on the true factor.

TV7 was made using a very weak peak in the FCA spectrum, m/e 142, that was not present at all in the 1-1'DCBF spectrum. The results show that its vector difference is the smallest of any of the tested vectors and that its correlation coefficient only just starts to level off over the ten factors of the experiment. The predicted vector from the tenth iteration shows the spectrum of FCA with only a very small amount of interference from the molecular ion cluster of 1-1'DCBF. The difference spectra show that the spectrum of FCA is still being added, but in very small amounts, and also that the molecular ion peak for 1-1'DCBF is still being added, also in small amounts. The addition is due to some slight mixing of the two components as only four factors are used instead of the theoretical 19 necessary to completely describe the data.

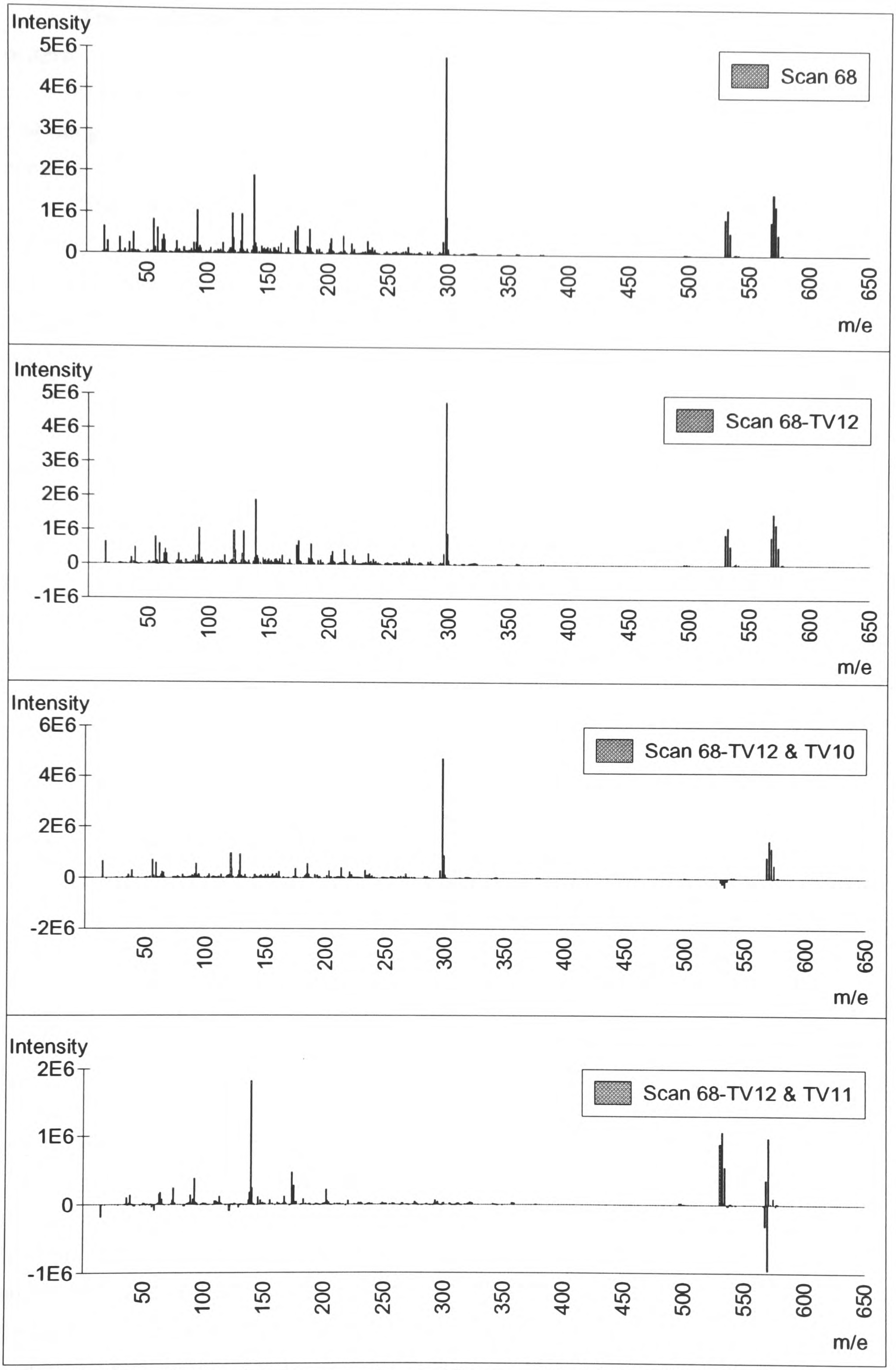


Figure 81: Plots of ITT produced vectors subtracted from scan 68 of data set TB19. Key to charts - Top to bottom: Scan 68 of data set TB19; Scan 68 minus iterated air background (TV12); Scan 68-TV12 minus iterated 1-1'DCBF spectrum; Scan 68-TV12 minus iterated FCA spectrum.

The results for TV8, also due only to FCA, show an almost exactly similar pattern.

TV9's results show a mixed spectrum of FCA and 1-1'DCBF and it appears that the proportion of each spectrum in the predicted vector depends upon the overall contribution of each component to that value.

The iterative target testing process using single data points seems to work well for unique points in the spectrum of a component. The selection of a peak that arises from more than one source results in a mixed predicted vector thus complicating interpretation. It must be remembered that the results, in this case, are from testing single m/e values and if more m/e points are added then the vector would be able to converge on the true factor and not a mixed one.

To test how well the predicted data fits the true data, a series of iterative target tests were performed using test vectors containing the values of the base peaks of each component in scan 68 of data set TB19 and the values for the air spectrum in the same scan. These test vectors, TV10, TV11 and TV12 were tested for ten iterations. After the last iteration the magnitude of the original peak was seen to have fallen in each vector so they were scaled back to the original peak intensity. The vectors were then subtracted from scan 68 and the results shown in figure 81. The background subtraction (-TV12) cleans up the lower portion of the spectrum considerably and the subtraction of the 1-1'DCBF predicted spectrum (-TV10) gives a close approximation to the FCA spectrum. The subtraction of the predicted spectrum of FCA (-TV11) gives a good spectrum of 1-1'DCBF in the lower portion of the spectrum but the molecular ion cluster is modelled poorly. This is due to only four factors being used to model the data and the calibration problem present during this analysis, the later, neglected, factors must describe the variation in the molecular ion peak and thus the predicted vector fits poorly.

The loss in magnitude observed in this experiment for the m/e value containing the intensity figure was also observed for the uniqueness vectors. Further tests indicate that the more factors used to describe the data the less the loss in original value, also, if more points were added to the vector then the reduction in magnitude of the original points reduced. These results show how the identification of the correct dimensionality of the data space is important, the addition of superfluous factors allows an artificially good fit to the factor thanks to the extra degrees of freedom introduced, they also indicate that to achieve a good result from a single target test, several points must be included in the test vector. The target testing procedure will produce a perfect result for any vector containing fewer points than the number of factors comprising the data. In the case of the uniqueness vectors only one point is given a significant value and thus might be expected to contradict this rule. In fact the test vector contains a full 650 points but most of them are zero, this affects the way the factor is aligned with the true factor and, because of the imperfect alignment, results in the reduction of magnitude observed. The addition of the extra points produces better and better alignment with the true factor until nearly 100% of the original value is returned.

In all of the vectors tested, the convergence appears logarithmic over the early iterations and then as it approaches the true factor, slows considerably. In order to make the convergence proceed at a faster rate a positive threshold, below which the vector point was zeroed, was investigated.

The algorithm that set any negative values to zero was modified to set any values below the minimum value set for the decomposition to zero. The minimum value was set, as before, using 1/999th of the smallest base peak in the data set. Test vectors containing the actual intensities of peaks taken from scans in TB19 were used for the tests; these were TV10, TV11, TV12 and TV13. The difference terms (figure 82) and correlation coefficients (figure 83) are shown below. The results for the difference term show a similar behaviour for TV10, TV11 and TV12 with a rapid fall after the first iteration and then a levelling out of the gradient, the flat portion of the curve shows some undulation not observed for the earlier experiment. TV13 continues to fall after the first drop and its gradient increases until at iteration 22 an all zero vector was returned. The results for the correlation coefficient tell a slightly different story, TV11 shows a sharp fall that levels out into a plateau for several iterations followed by a slight increase. TV10 and TV12 also have a sharp fall to a plateau, but it is much shorter and is followed by stepwise increases. TV13 shows this behaviour more strongly reaching a minimum at the third iteration and then increasing in a stepwise fashion until the all zero vector is returned.

This indicates that the difference term gives a misleading view of the convergence and that the correlation coefficient is a better measure. The results also show that the setting of values below a positive threshold to zero removes important information from the vector resulting, in the worst case, in the loss of all information in the vector. The problem is exacerbated by the reduction in magnitude of the values in the test vector as a result of the less than perfect test.

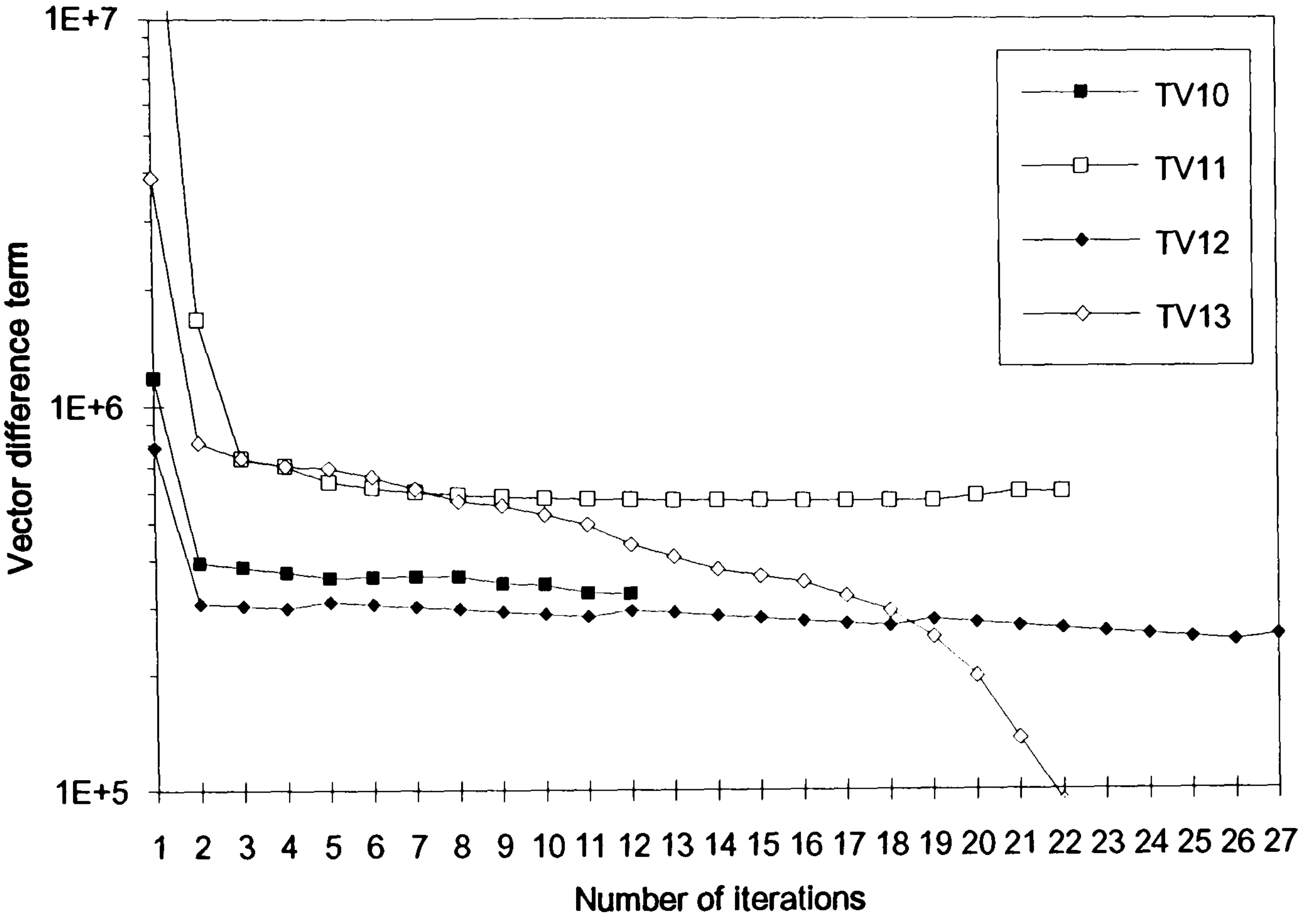


Figure 82: Plot of the vector difference term against iteration number for test vectors TV10, TV11, TV12 and TV13 resulting from iterative target testing with zeroing below a positive threshold.

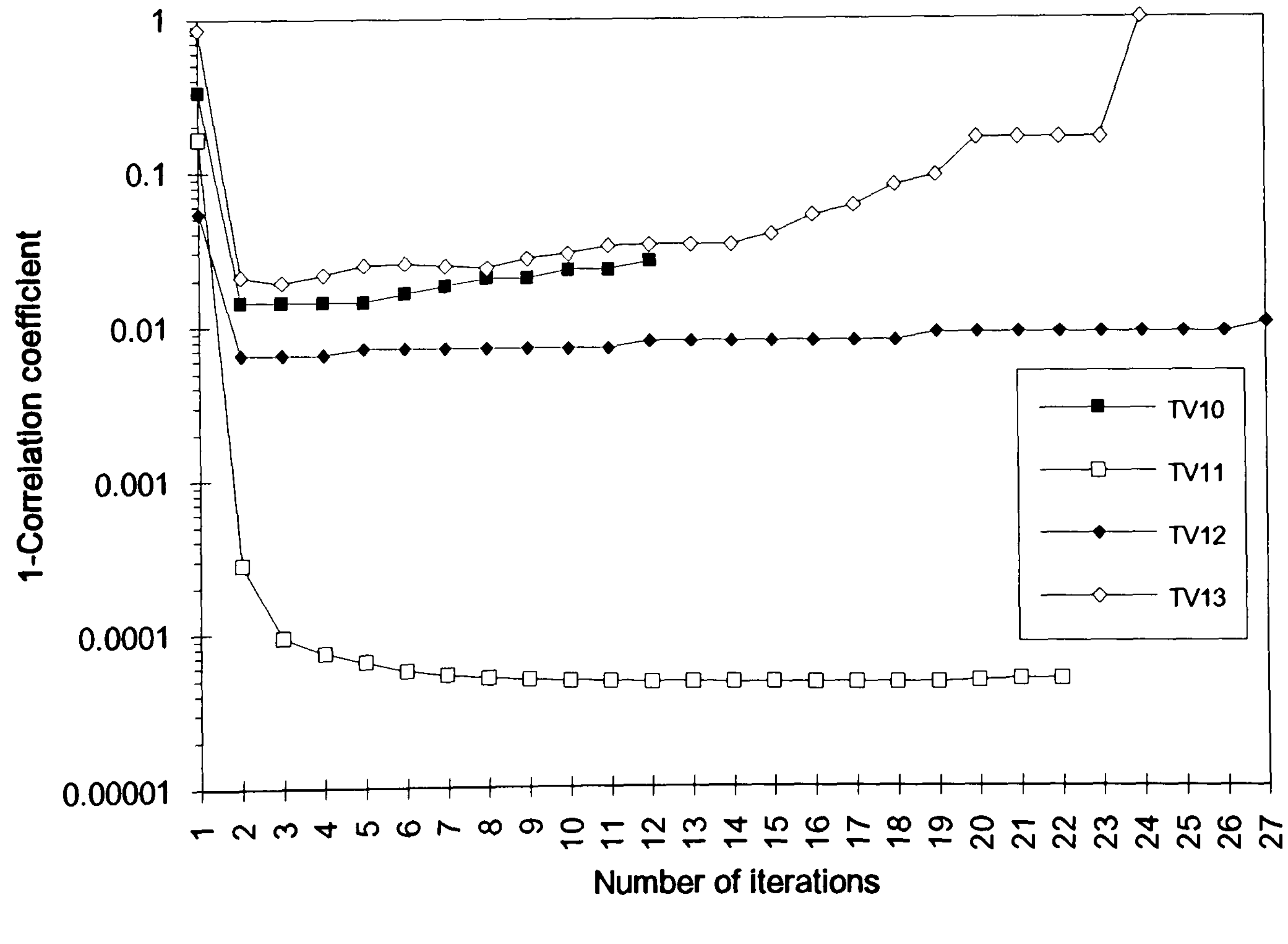


Figure 83: Plot of the correlation coefficient against iteration number for the test vectors TV10, TV11, TV12 and TV13 resulting from iterative target testing with zeroing below a positive threshold.

The lack of success of the two methods of iterative target testing led to the consideration of other methods of perturbing the vector based upon accelerating the convergence by prediction of the direction the vector is moving in.

Differential addition

The result from a target test differs from the test vector due to three possible changes in the values contained in the vector. The values may increase, decrease or remain the same. The values that remain the same must be correct for the true factor being searched for; the changing values change the orientation of the vector in the factor space to align it with the true factor. If the changing points in the vector are found and the amount they are changing increased then the vector should converge more rapidly.

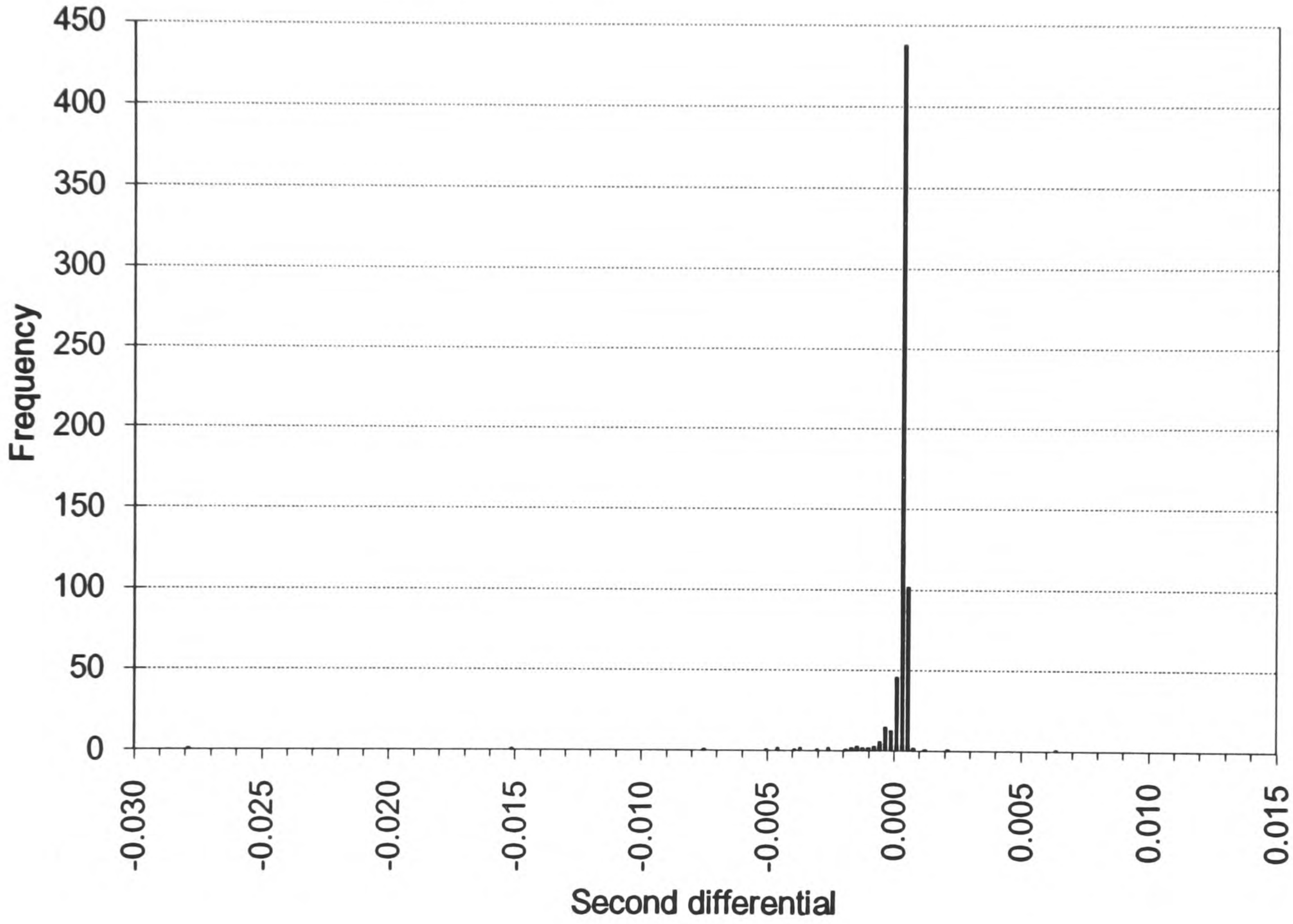


Figure 84: Frequency distribution of the 1st second differential of test vector TV4 for iterative target testing on data set TB19 using zeroing of negative values only.

To identify the changing points, the differential of the predicted vector may be considered. Unfortunately the values of the first differential are affected by the magnitude of the value in the test vector. To avoid having to determine which values are significant the second differential was used to find the most rapidly changing values in the predicted vector. To decide which values to modify the frequency distributions of the second differentials were studied, a typical differential is shown in figure 84. They indicated that most values in the vector changed very little with only a few points having significant changes. This led to the use of a standard deviation threshold to determine which values would be modified. The modification of the test vector was based upon a multiple of the previous change of that point. Provision was made that if the vector was seen to be 'bouncing' back and forth between two sets of values (either side of a true factor) then the differential modification would be stopped and convergence allowed to continue using the effect of zeroing negative points only.

The results from this method of modification were found to be disappointing. Initially the threshold for the standard deviation was set too low to affect the vector at all. When the value of the standard deviation threshold was set high enough to perturb the vector, the addition to the vector, however small, simply increased the values in the vector and no convergence was ever achieved, the vector increased in size continually.

Differential subtraction

Adding to the test vector did not produce the desired acceleration in convergence so the opposite approach was tested. Many of the points in the mass spectrum change very little from one iteration to the next. These points all have a bearing on which direction the vector points. Identification of the points and setting them to zero might allow the vector to align itself more rapidly. Unfortunately, as the convergence is approached, all of the points in the vector will change very little and would thus be set to zero. To avoid this problem the magnitude of the point is considered and if significant it is left unchanged. The significance of each point and its magnitude are checked by comparison with a standard deviation window. A value is chosen for the first differential, below which values are regarded as unimportant and a value set for the predicted vector above which values are regarded as important. Any point in the predicted vector that has a differential below the threshold and a magnitude below the threshold for the vector is set to zero to minimize its perturbation of the next test.

Numerous combinations of values for the two thresholds were tried between 2sd and 0.01sd by testing using TV4 and saving the vector after each iteration. Figure 85 shows the correlation coefficients for the iterations of some of the values tested. The results showed that use of values below 0.1sd invariably resulted in no modification to the vector after very few iterations. Values of 0.25sd and above, showed signs of the vector alternating between two sets of values and never approaching the true factor. The best results were obtained with threshold values of 0.1sd and 0.1sd. These values perturbed the vector for around five iterations before leaving it to converge affected only by the zeroing of negative components. This method appears to allow the vector to approach the true factor more closely than zeroing the negative values, but the difference is small and takes nearly twice the number of iterations and thus suffers a penalty in the time involved for analysis.

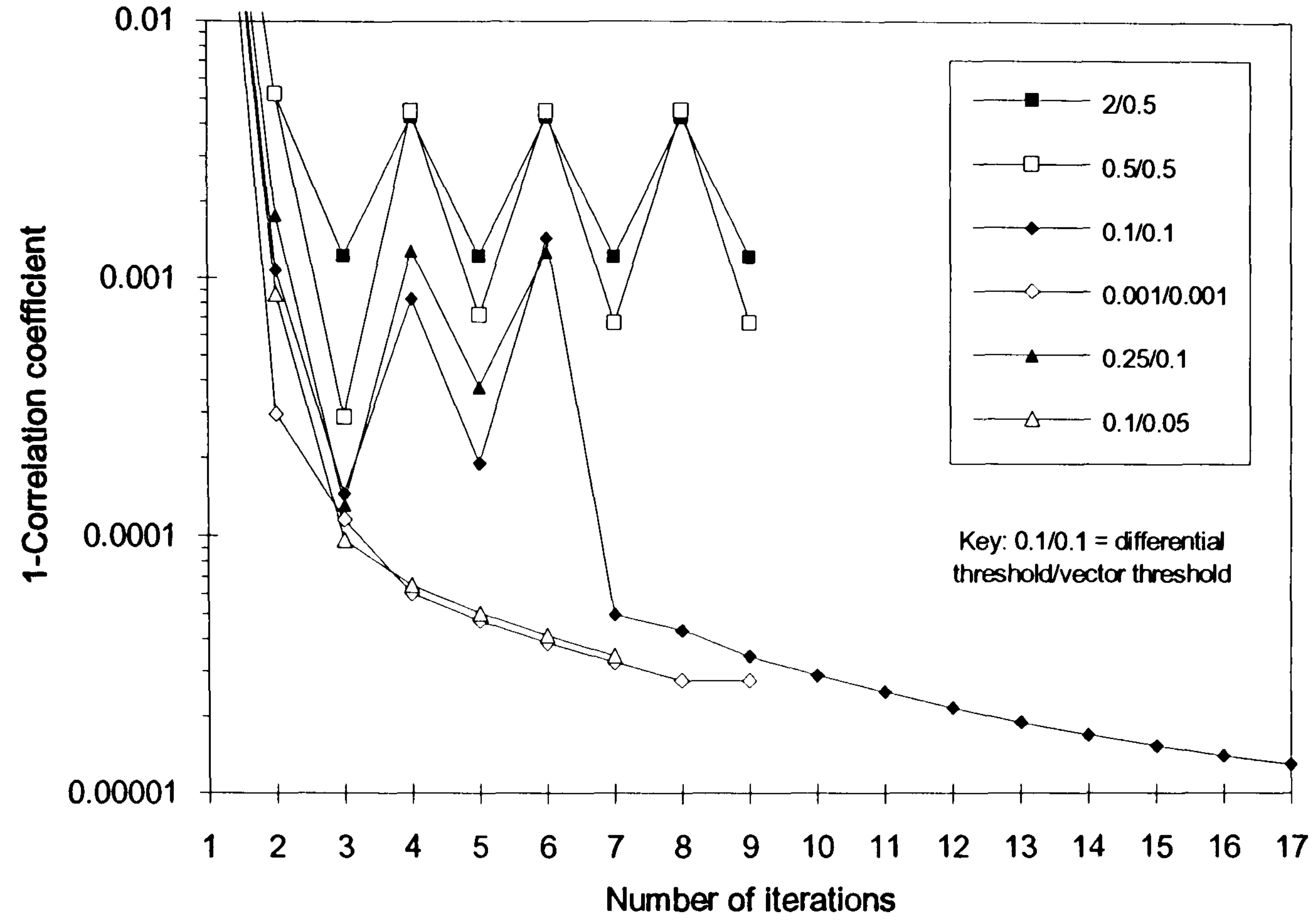


Figure 85: Plot of correlation coefficient versus iteration number for TV4 during iterative target testing on data set TB19 using differential subtraction and various values for the differential and vector threshold.

In conclusion it can be seen that the iterative approach starting from a single m/e value can produce spectra which at best would be identifiable using library search and at worst provide the spectroscopist with valuable extra information to aid in the identification of unknown components.

The differential addition technique proved unworkable as its modifications perturbed the vector away from the true factor. The differential subtraction technique shows some promise but offers, in its present form, little advantage over the simple chemical selection rule of zeroing negative peaks.

The problems of loss of information due to smaller values returning from each target test were observed and this loss of information limits the number of iterations possible for any vector. Reinforcing the new test vector with the original test vector after each iteration may help in increasing the number of possible iterations but the problem undoubtedly lies in the small number of significant data points available to orientate the test vector. An increase in the number of peaks attributable to the component being searched for gives much better results.

4.6.3. Multi-component systems

4.6.3.1. Sample history

The program and techniques developed were tested on real problems using the data from the analysis of two samples. The samples are referred to as MS1930 and MS1930A. Both were from similar origin but of different appearance and were submitted for identification. The analysis procedure used was identical to the procedure described in the experimental section.

In the original analysis sample MS1930 was found to be a polyurethane and sample MS1930A a chlorinated polymer using the normal techniques applied to the results of temperature programmed pyrolysis mass spectrometry. Both the samples contained complex regions early in the TIC pyrogram that were expected to correspond to evolution of plasticizers, stabilisers, etc. The results obtained from the standard analysis methods were that MS1930 and MS1930A contained two components each and MS1930 might also contain a third component. Assignments were made for all of the components believed to be present.

4.6.3.2. Suspected components

Of the components believed to be present in the samples, the first two listed below were believed to be in MS1930A and the rest in MS1930.

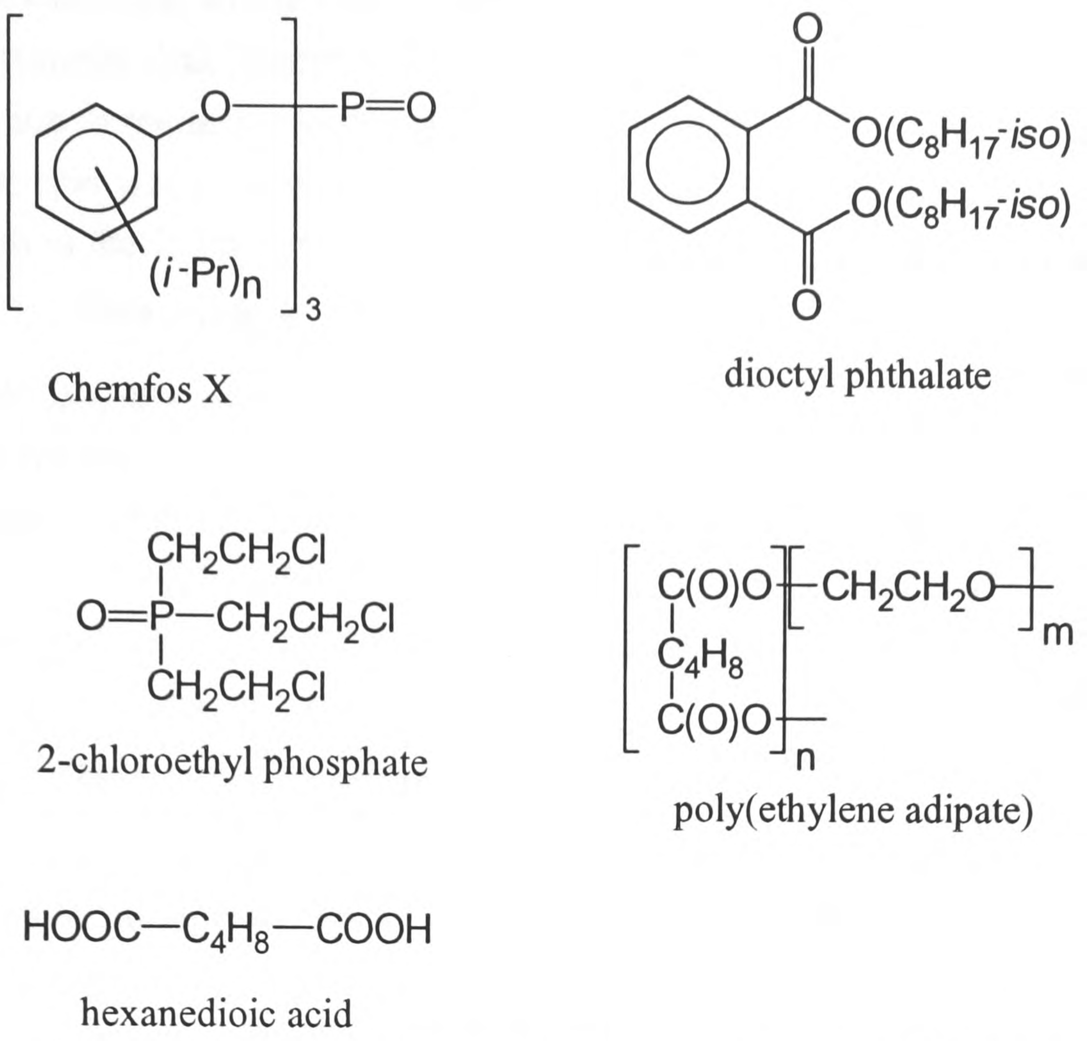
The components suspected from the standard analysis are as follows:
Chemfos X (propyl substituted phenyl phosphate),
dioctyl phthalate,
2-chloroethyl phosphate,
hexanedioic acid (adipic acid)
and poly(ethylene adipate).

The structures of the additives are given in figure 86 below.

Information about the mass spectra of the components was available from the following sources. Chemfos X, dioctyl phthalate, 2-chloroethyl phosphate and poly(ethylene adipate); reference spectra were obtained from the in-house library on the mass spectrometer data system. Hexanedioic acid; the reference spectrum for this component was found from the Eight Peak Index of Mass Spectra[103].

4.6.3.3. Data pre-treatment

The original analysis data files were converted to EPA format using the mass spectrometer data system and transferred to the PC where they were converted to spreadsheet format using the program. Both files contained 181 scans and their TIC pyrograms are shown in figure 87, also shown are the regions of the pyrograms that were selected for study using the program. For MS1930 this was scans 30-60 and for MS1930A scans 20-60 were used. The scans were extracted using a spreadsheet.



Chemfos X

dioctyl phthalate

2-chloroethyl phosphate

poly(ethylene adipate)

hexanedioic acid

Figure 86: The structures of the additives suspected to be present in samples MS1930 and MS1930A.

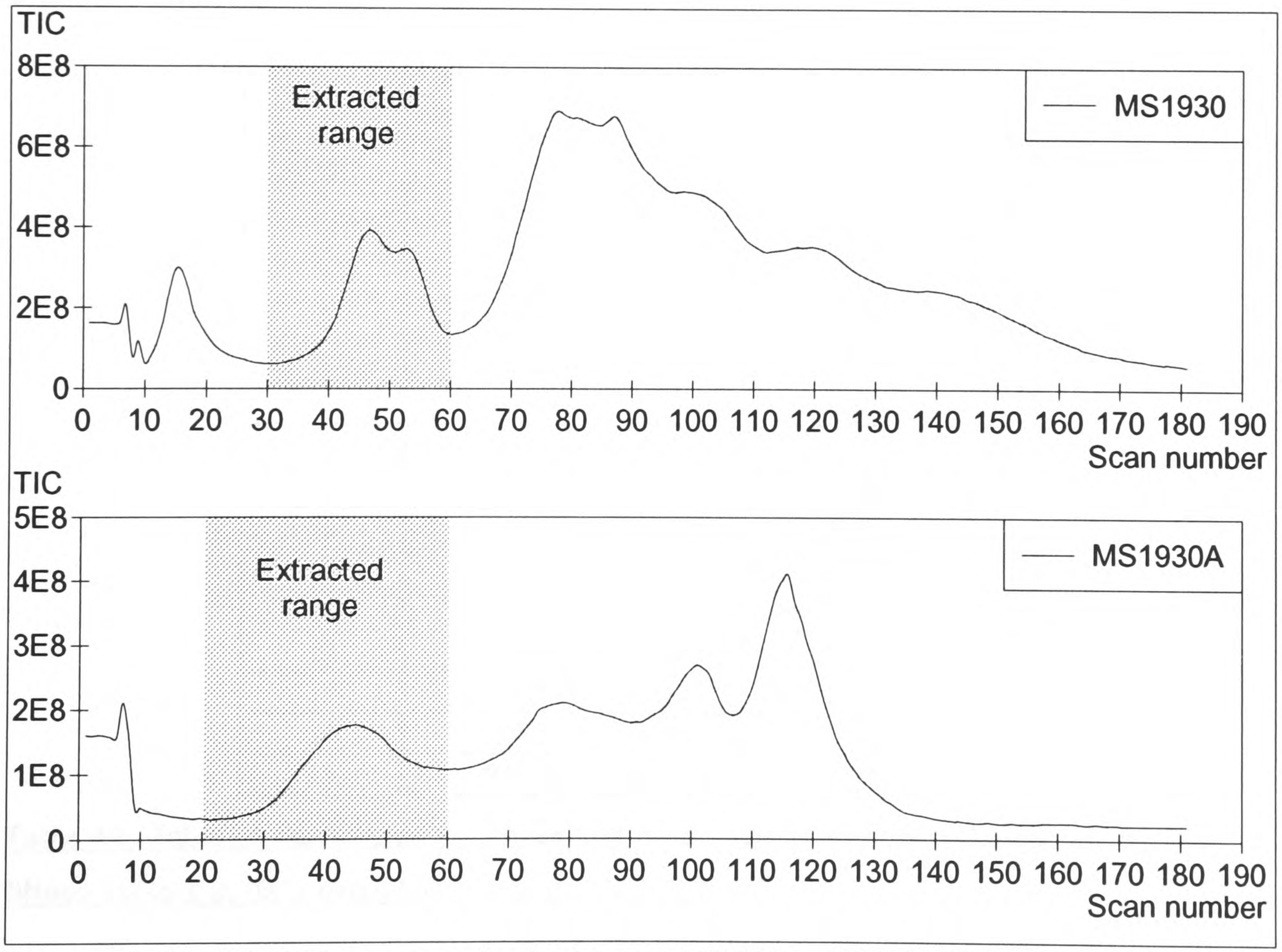


Figure 87: Plot of the TIC pyrograms for the data sets MS1930 and MS1930A showing the regions extracted for further analysis.

The additives, whose spectra were available in the in-house library on the mass spectrometer data system, were converted to EPA format files and transferred to the PC for use in the target testing procedures discussed later. The mass spectrum for hexanedioic acid was entered into a blank test vector created in a spreadsheet and the values given in the Eight Peak Index[103] used for the spectrum.

4.6.3.4. Data analysis

Both data sets were analysed using a minimum value of 100,000 and an error estimate of the same value. The values were arrived at using 1% of the m/e 28 peak at the beginning of the reduced series of scans, this is also the smallest base peak in the data and thus 1/999th of its value represents the smallest meaningful value.

Both analyses ran to the theoretical maximum number of factors, as has occurred with all previous data sets.

4.6.3.5. Determination of the rank of the data

The variance accounted for, in both data sets, by the first fifteen factors is given in table 42 along with the values for the percentage significance levels. The percentage variance values give very little information about the dimensionality in either data set; both account for more than 99% of the variance after only three factors. The values for percentage significance level indicate that for both data sets there are eight significant factors.

	Data set			
	MS1930		MS1930A	
Factor	%Variance	%SL	%Variance	%SL
1	83.41	0.0	87.53	0.0
2	11.36	0.0	10.98	0.0
3	4.36	0.0	1.10	0.0
4	0.54	0.0	0.24	0.0
5	0.26	0.0	0.10	0.0
6	0.05	0.0	0.02	0.1
7	0.01	0.2	0.01	0.8
8	3.71E-3	4.8	3.41E-3	6.1
9	1.84E-3	11.2	1.92E-3	12.1
10	1.30E-3	14.6	1.26E-3	17.6
11	9.85E-4	17.1	1.02E-3	20.0
12	8.32E-4	18.1	9.07E-4	20.9
13	6.60E-4	20.2	8.45E-4	21.1
14	6.30E-4	19.2	7.85E-4	21.4
15	4.92E-4	21.5	7.04E-4	22.2

Table 42: Table of percentage variances and percentage significance levels for the first fifteen factors in data sets MS1930 and MS1930A.

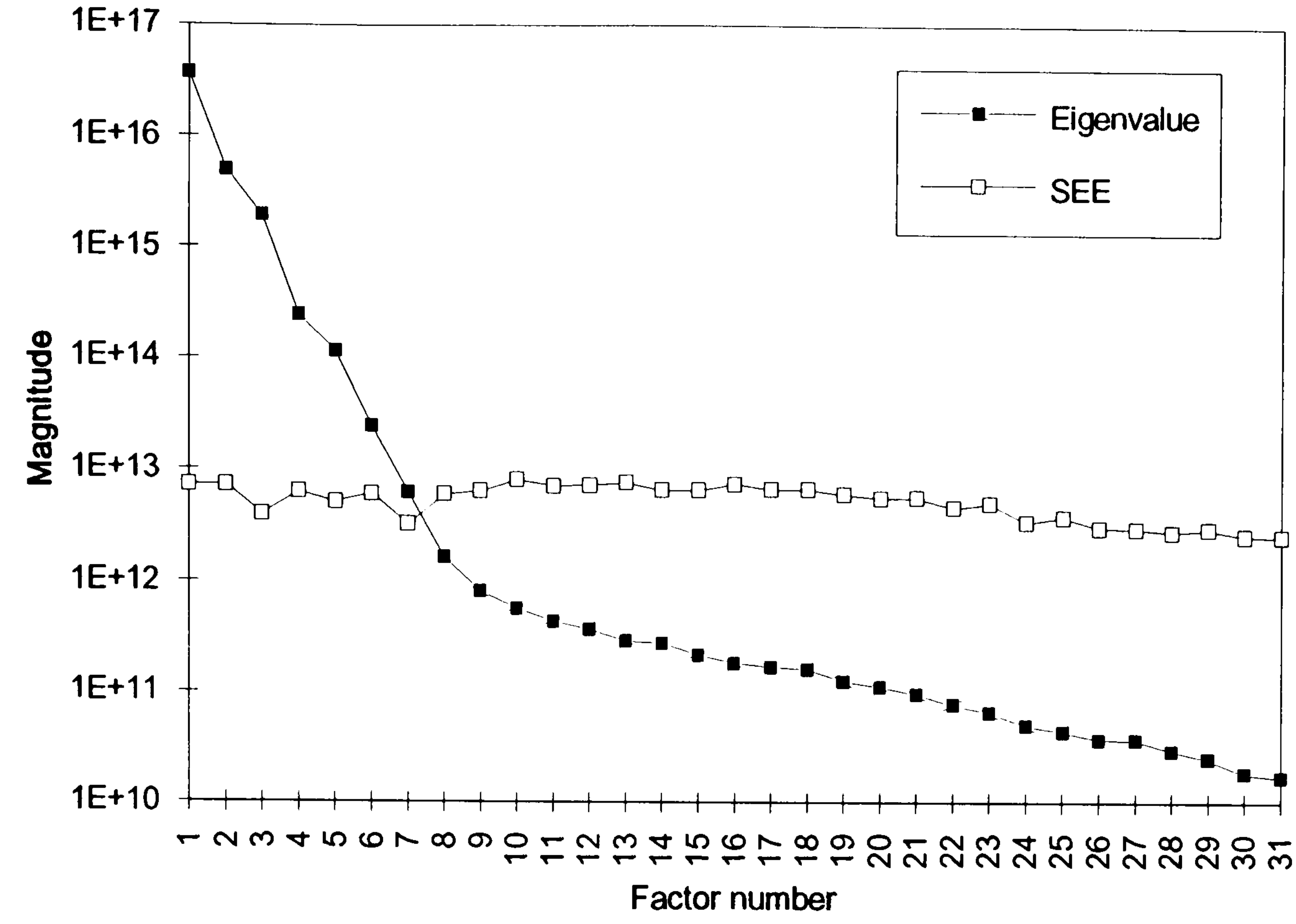


Figure 88: Plot of eigenvalue and standard error in eigenvalue for data set MS1930.

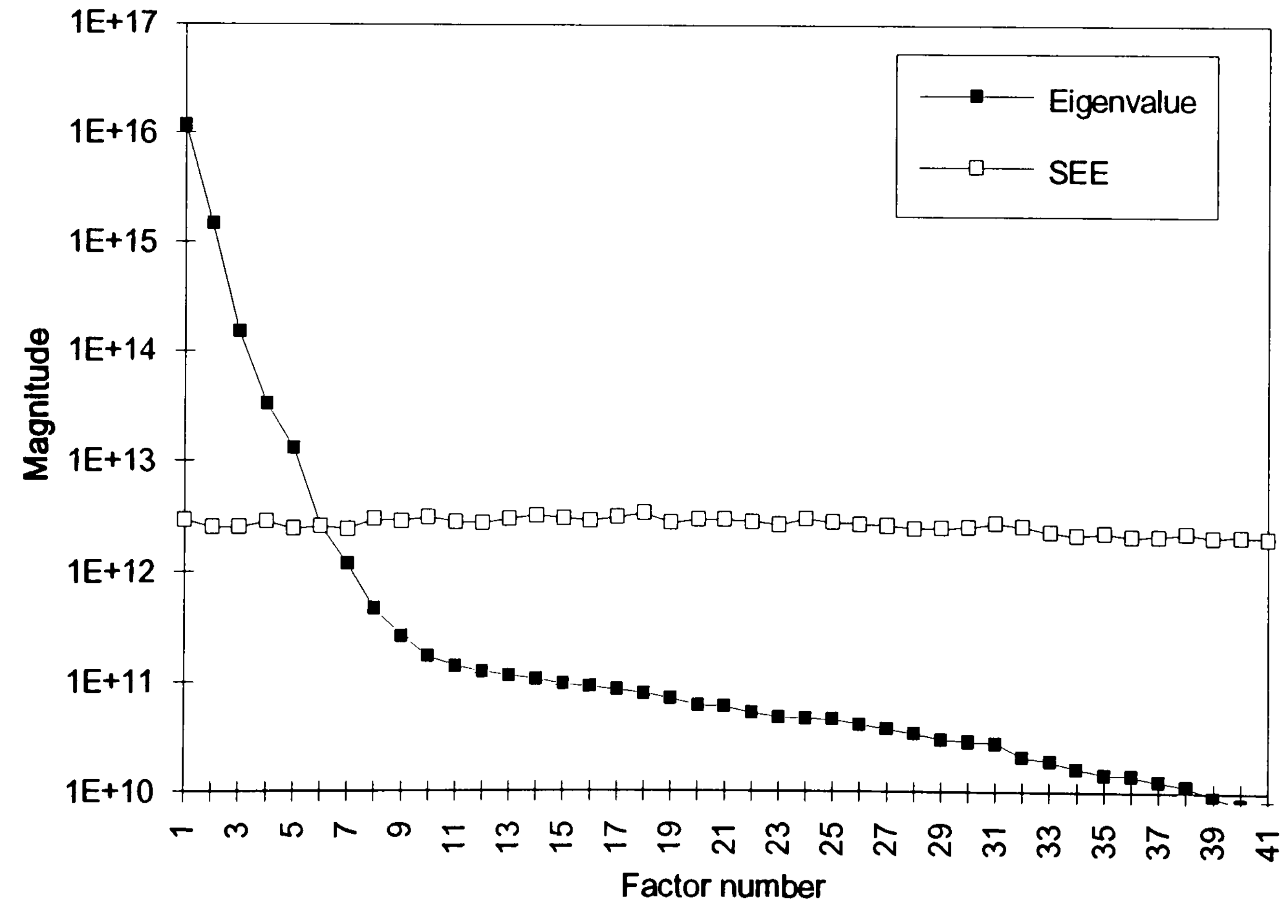


Figure 89: Plot of eigenvalue and standard error in eigenvalue for data set MS1930A.

The eigenvalues and standard error in eigenvalue for both data sets are given in figures 88 and 89. The SEE values indicate 7 and 5 factors for MS1930 and MS1930A respectively. The difference between the estimates produced from the use of this error term and the %SL has been noted before and is due to an over-estimate of the error in the data set.

The shape of the graphs of the eigenvalue for both data sets indicates a sharp change in gradient at around factor 9, close to the 8 factors suggested as the dimensionality of the data by the %SL term.

The real error terms calculated for the data (shown in figure 90) indicate that at 8 factors the error in data set MS1930 is 16,800 and MS1930A is 9,600. Consideration of the size of the data sets shows that the magnitude of MS1930A is around half that of MS1930 and in the same ratio as the RE values. The plots of the real error term also show an inflexion at around factor 8, adding weight to the selection of that number for the dimensionality of the data.

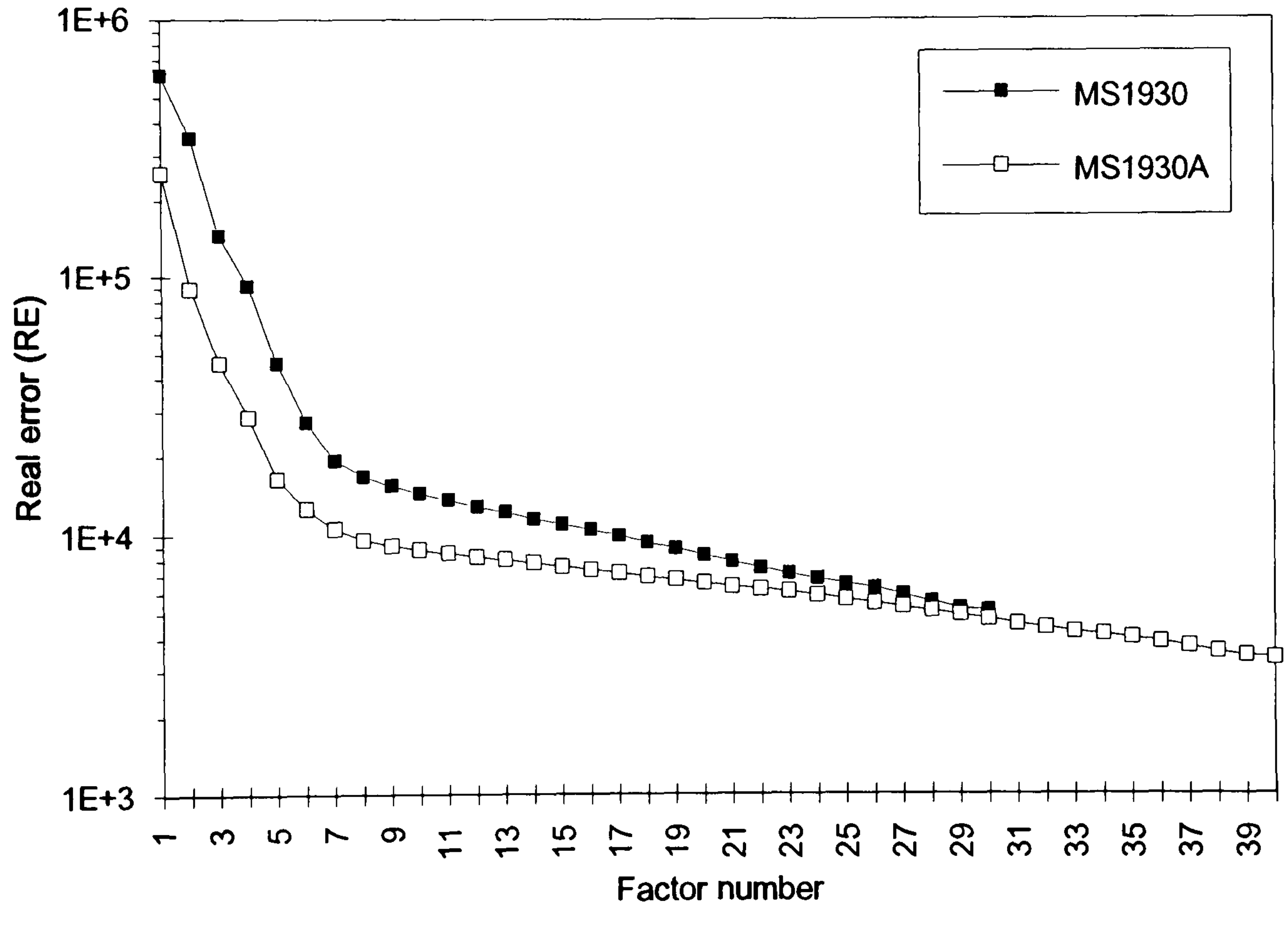


Figure 90: Plot of the real error (RE) terms against factor number for data sets MS1930 and MS1930A.

The results for the imbedded error and indicator function are shown for MS1930 and MS1930A in figures 91 and 92 respectively. The results for both data sets clearly indicate that there are eight primary factors in the data, both the indicator function and the imbedded error reach minima, though in the case of the imbedded error it is only a local minimum.

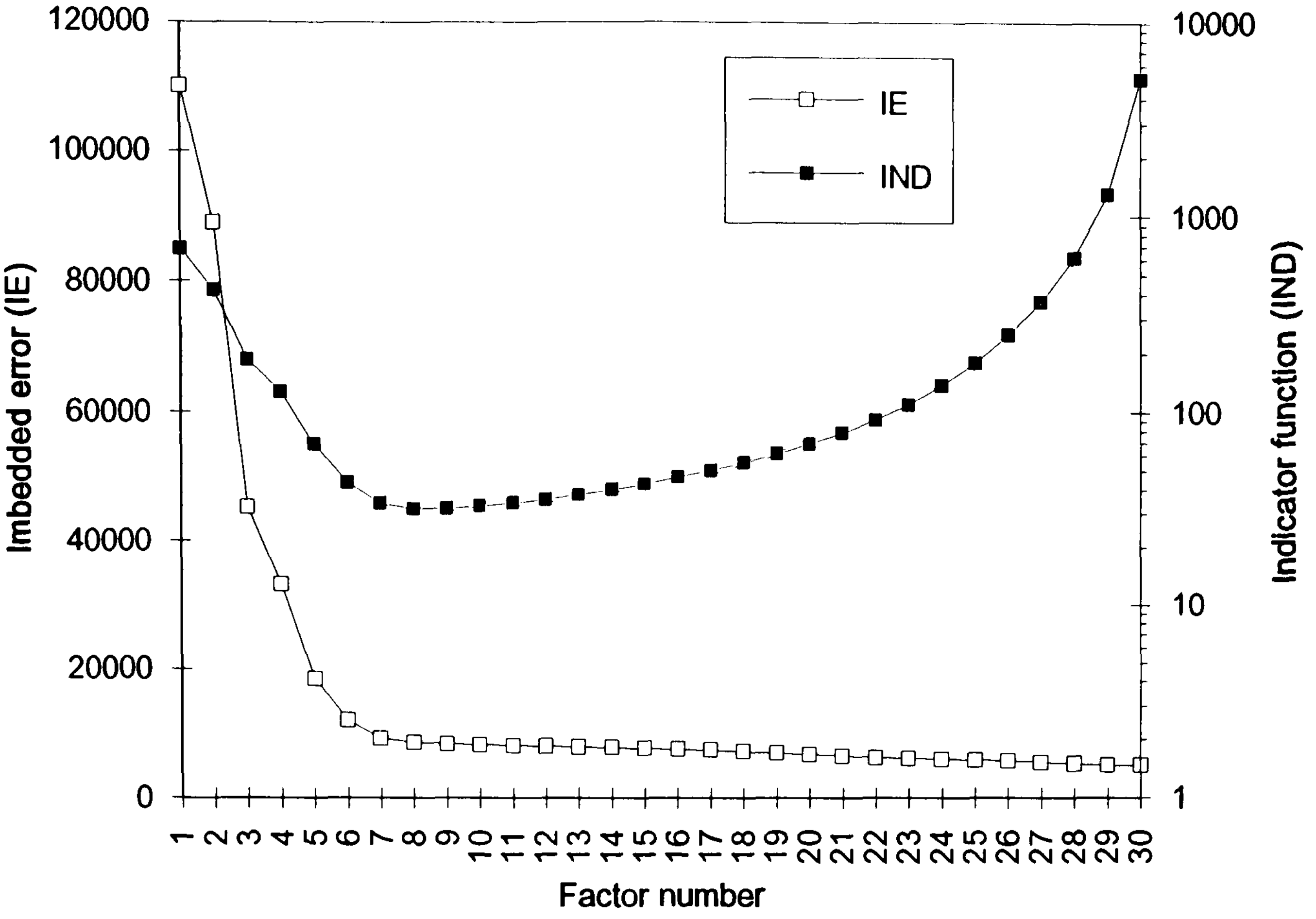


Figure 91: Plot of imbedded error (IE) and indicator function (IND) versus factor number for data set MS1930.

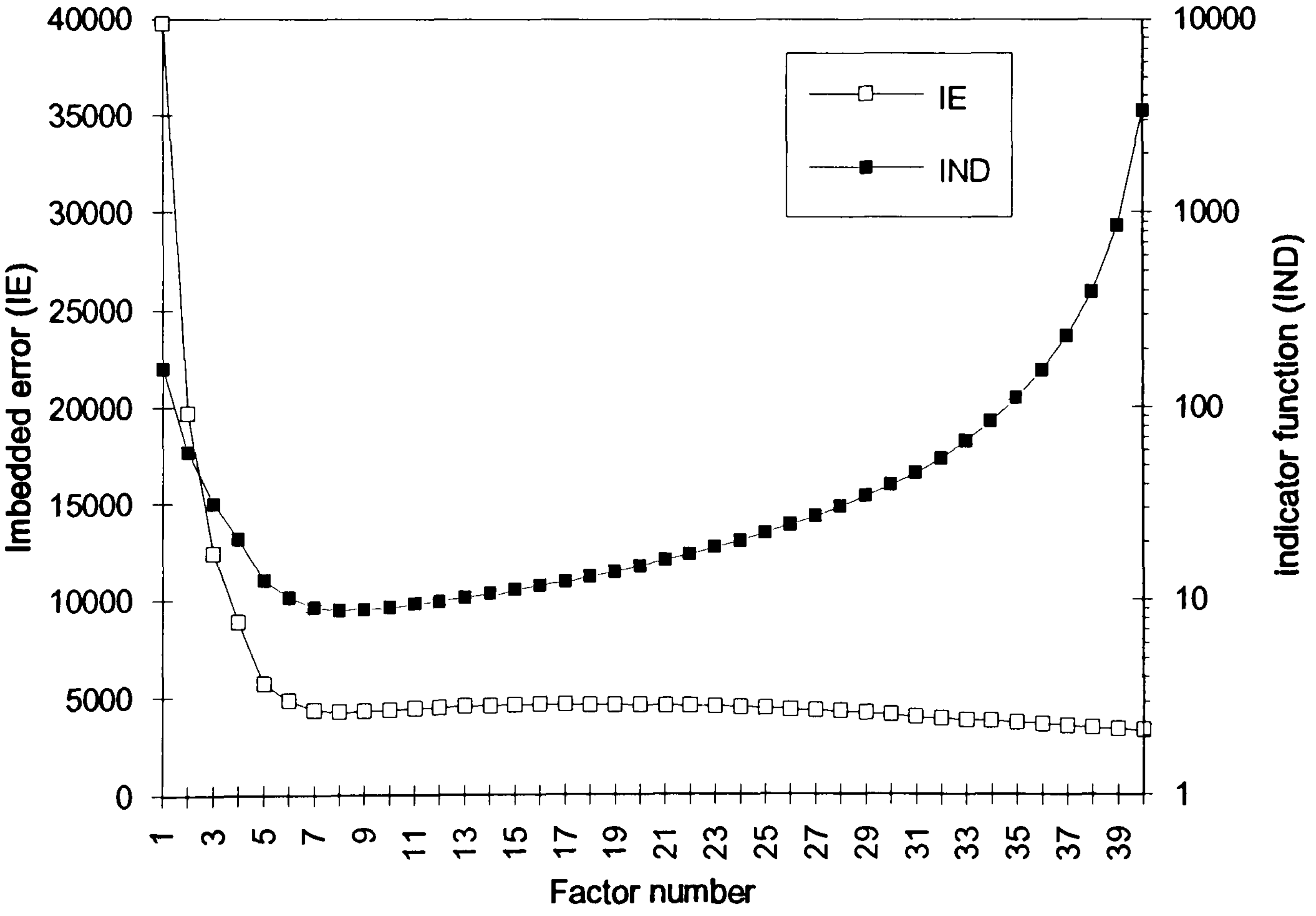


Figure 92: Plot of imbedded error (IE) and indicator function (IND) versus factor number for data set MS1930A.

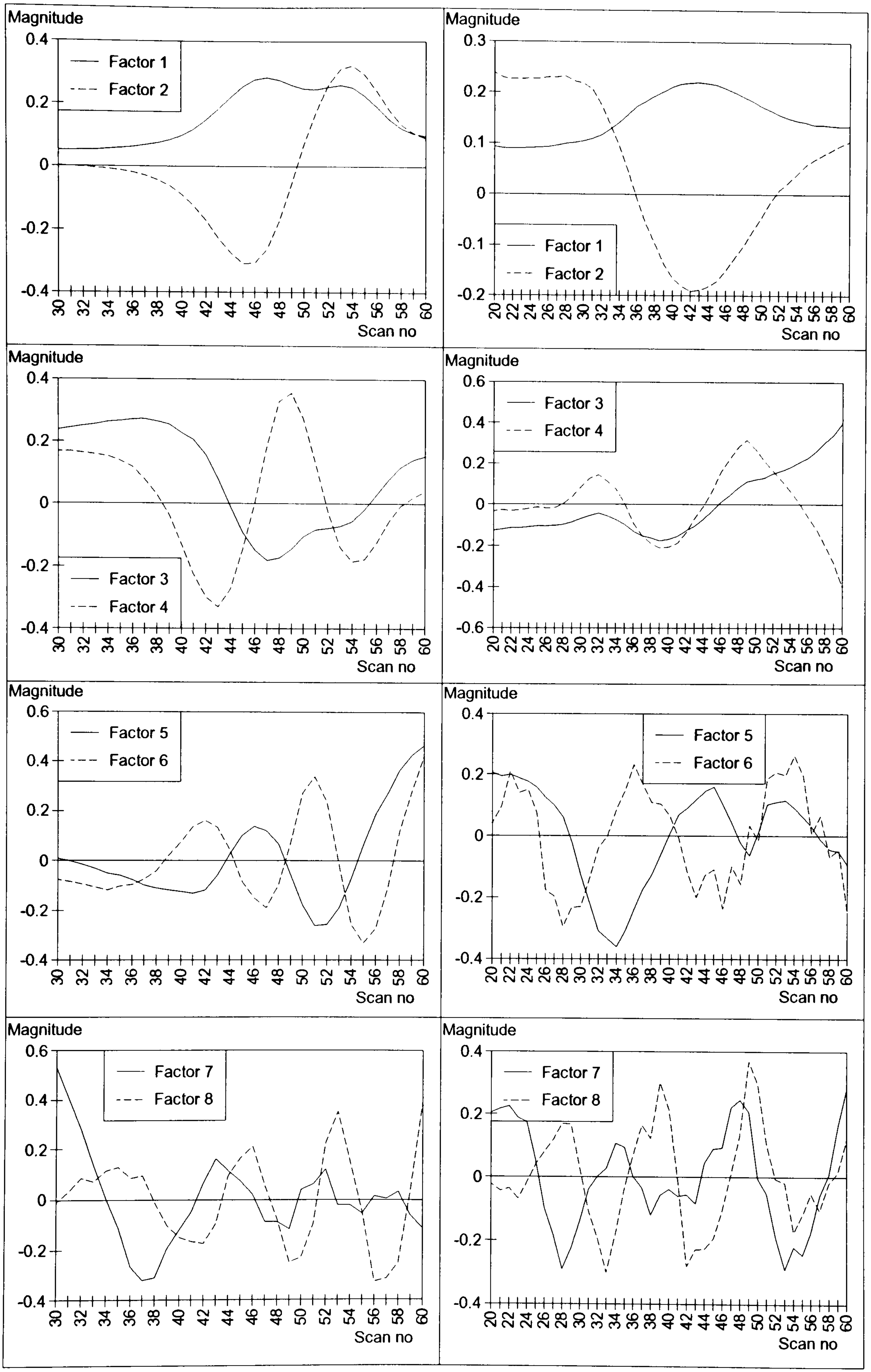


Figure 93: Plots of the loadings for factors 1 to 8 for the data sets MS1930 and MS1930A. Left hand charts are MS1930 and right hand charts MS1930A.

4.6.3.6. Abstract column matrix

The eigenvectors of all eight primary factors are shown in figure 93, the eigenvectors for data set MS1930 are on the left and those for MS1930A are on the right. In both cases the first factor shows the shape of the TIC pyrogram indicating that the variance from the sample is greater than the variance from the background over the range studied.

The second factors show, in MS1930, a separation of the two peaks apparent in the TIC pyrogram and in MS1930A, an inverted TIC pyrogram typical of the background. The background factor for MS1930 appears at factor 3.

Factor 3 for MS1930A shows a steady rise over the course of the data and probably describes the variance introduced by the large long peak seen in the TIC pyrogram (figure 87). A comparable factor is not observed for MS1930 and observation of the TIC pyrogram shows a smaller rise in TIC over the extracted range. This will reduce the variance introduced into the data by that factor and probably results in the variance being mixed into factors 5 and 6 that both start low and finish high.

Factor 4 for both data sets shows a maximum at scan 49 and a minimum that appears around scan 41. Factor 4 for MS1930A also shows a secondary maximum at scan 32 that is not seen in any of MS1930's factors.

Factors 5 and 6 for MS1930 show smooth profiles in comparison to the same factors for MS1930A that contain many jagged edges. This indicates that the factors for MS1930A are beginning to describe the unique behaviour of individual points. They still describe a more general source of variance in the data but it is beginning to become submerged beneath the many small variances produced from the experiment. The smooth curves of MS1930 describe, for factor 5, the general increase in TIC over the range of the data (mixed with some of the earlier factors), and, for factor 6, an event in the data at scan 51.

Factors 7 and 8 for both data sets contain many sharp features caused by much smaller perturbations of the data and do not readily yield useful information.

The factors certainly indicate that there are more components in the MS1930 data set than in the MS1930A data set, as suggested by the standard analysis.

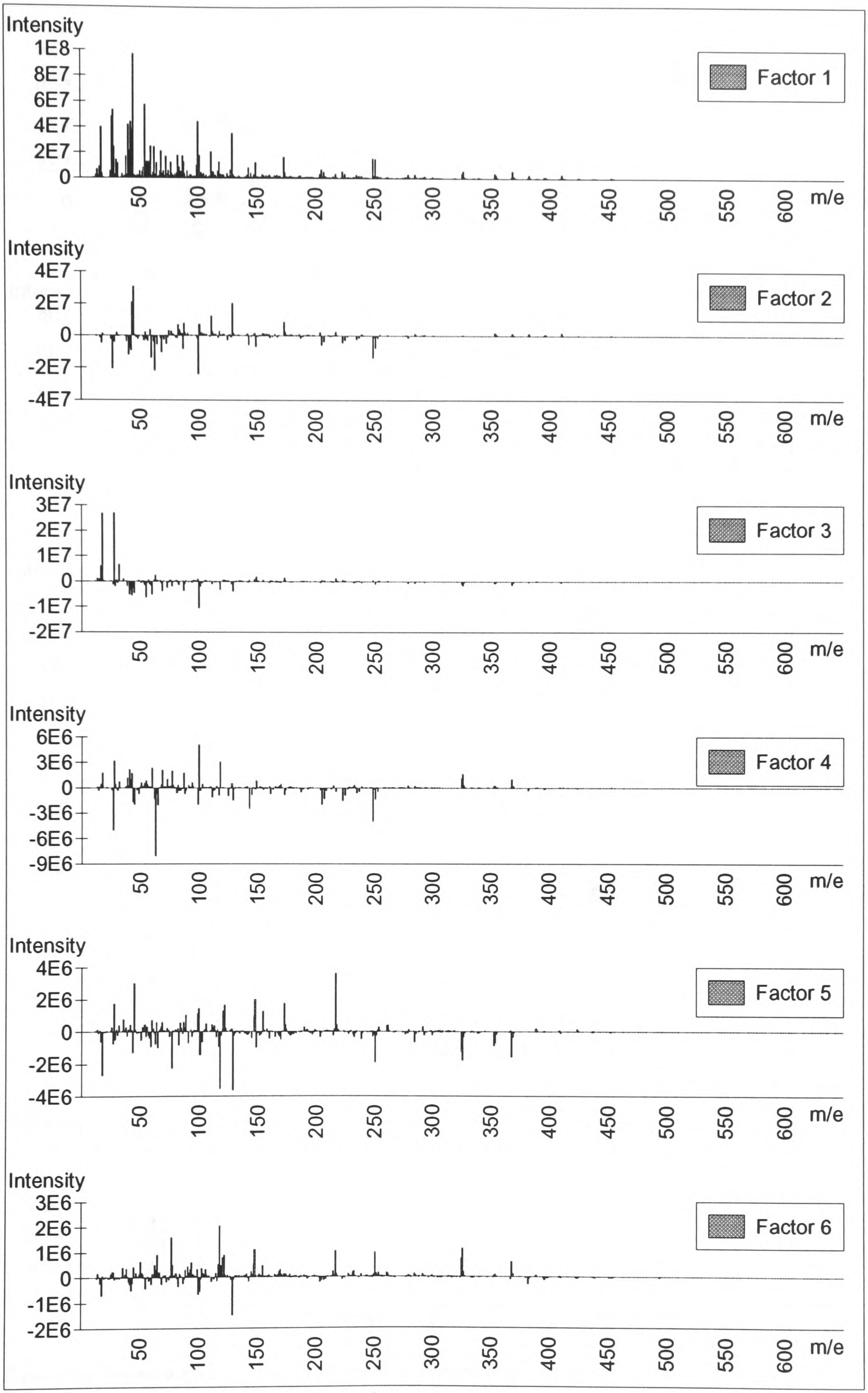


Figure 94: Plot of factor scores 1-6 for the data set MS1930.

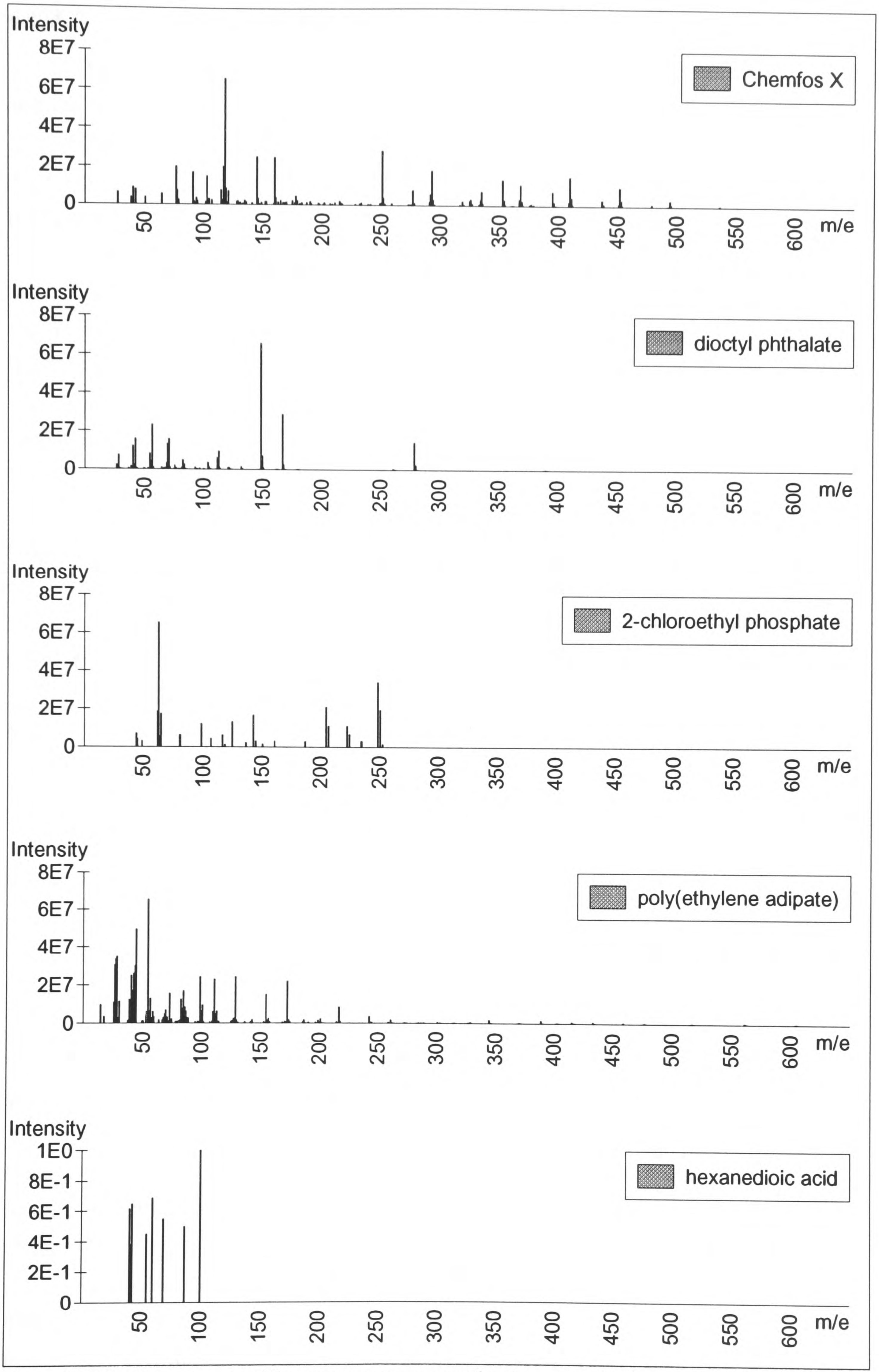


Figure 95: The mass spectra of the components suspected to be present in data sets MS1930 and MS1930A.

4.6.3.7. Abstract row matrix

The scores for the first six factors of data set MS1930 are shown in figure 94 and the mass spectra of the suspected components given in figure 95.

Factor 1 shows the average spectrum; factor 2 gives a separation between the two major sources of variance, in this case, the two visible peaks in the TIC pyrogram.

Factor 2 shows, on the positive side of the score, strong peaks at m/e 44,45 and 129. These peaks appear in the spectrum of poly(ethylene adipate). The negative side of the score shows peaks that exist in the spectra of 2-chloroethyl phosphate, hexanedioic acid and dioctyl phthalate.

Factor 3 clearly shows the background signal on the positive side of the factor with strong peaks at m/e 18,28 and 32. The negative side of the factor shows peaks found in hexanedioic acid and possibly Chemfos X.

Factor 4 provides a separation between 2-chloroethyl phosphate on the negative side and hexanedioic acid and possibly Chemfos X on the positive side. In this case the spectrum on the negative side is a close approximation to the true spectrum of 2-chloroethyl phosphate though some peaks from other components are still visible.

Factor 5 shows peaks not present in any of the suspected components on both sides. This indicates the presence of another unknown component, as the factor has already been ascribed to the increase in TIC over the course of the extracted data segment, then the peaks visible are probably from the source of that peak.

Factor 6 has so many other factors mixed in with it that interpretation is difficult.

The result of looking at the abstract factors gives an indication that the components might be present in the sample but it is difficult to point to a single factor containing only the spectrum of a component. To confirm the presence of any of the additives it will be necessary to target test. Before that stage is reached a further method that can be used for identifying which m/e values are associated is by use of a scores plot.

Figure 96 shows the scores for factors 2 and 3 plotted against each other. At the top of the chart are the m/e values for the background. Just below the horizontal to the right, are mass peaks visible in the poly(ethylene adipate) spectrum supporting the assumption that it is a component in the data.

Below the horizontal to the left are the peaks found in the spectrum of hexanedioic acid also pointing to the presence of that compound in the sample. It is noteworthy that m/e 55, a peak present in both poly(ethylene adipate) and hexanedioic acid is positioned midway between the two directions for the components.

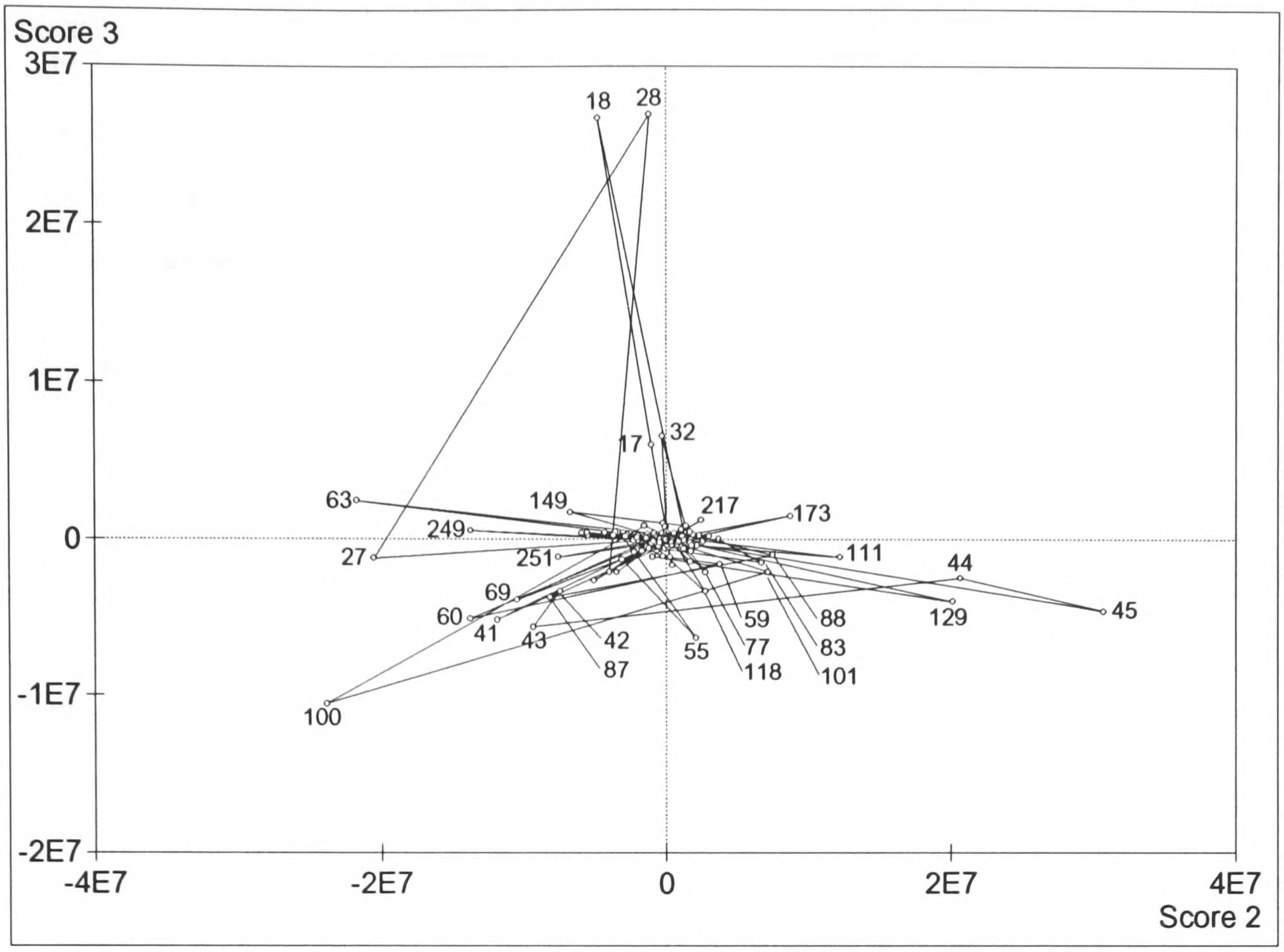


Figure 96: Plot of scores for factor 2 versus factor 3 for the data set MS1930.

Just above the horizontal to the left are peaks found in the spectra of 2-chloroethyl phosphate and dioctyl phthalate, the two factors chosen do not give a separation between the two components. This indicates that they contribute less to the variance in the data set than those that are separated, this leads to the conclusion that the components are present in lesser quantities than the two separated components.

Just visible at the bottom of the chart are m/e 77 and 118, these peaks are present in the Chemfos X spectrum but their small size and lack of any other substantial indication of their presence suggests that if a component at all, it will be at a very low concentration.

The first six factor scores for data set MS1930A are given in figure 97. Factor 1 shows the average spectrum over the whole of the extracted data range.

Factor 2, whose eigenvector (figure 93) was assigned to describing the background, can now be seen to provide a separation between the background, on the positive side, and what must be the major component of the peak on the negative side. The spectrum on the negative side of the factor appears very similar to that for dioctyl phthalate (figure 95).

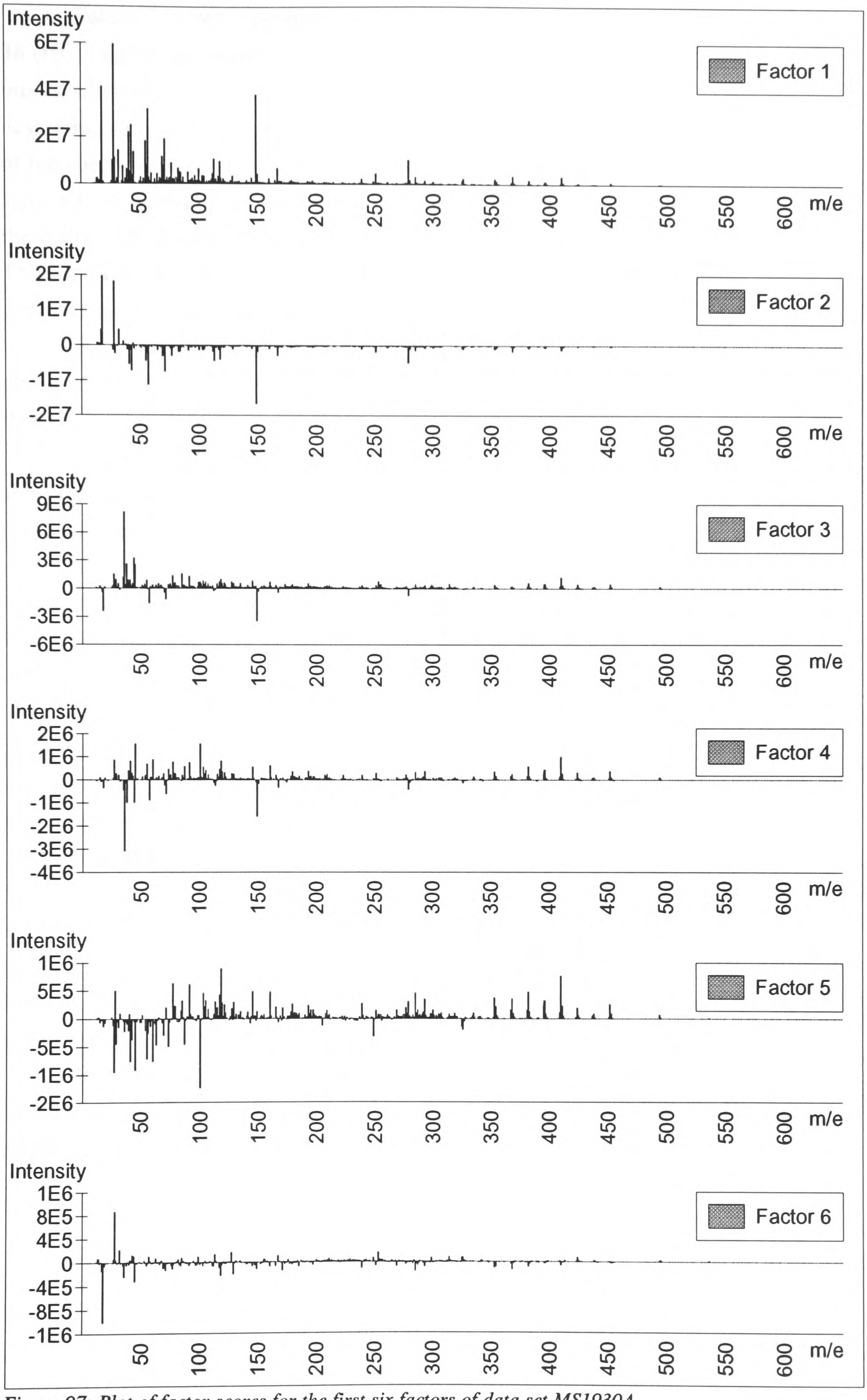


Figure 97: Plot of factor scores for the first six factors of data set MS1930A.

Factor 3 shows a positive spectrum containing a very strong m/e 36 and m/e 38 (HCl) signal that is probably from the chlorinated polymer rather than an additive, this conclusion is supported by the assignment of the associated eigenvector to describing the overall increase in TIC over the range of the data set. The negative side of the spectrum shows a strong signal at m/e 18 (H_2O) as well as some of the stronger signals from dioctyl phthalate, the phthalate signals are probably just mixed in with the factor, which describes a variation in m/e 18 that is not accounted for by factor 2. This kind of behaviour is typical of m/e 18 and has been observed in all of the previous samples analysed.

Factor 4 shows m/e 36 on its negative side with some peaks from dioctyl phthalate and a spectrum similar to that seen on factor 3 on the positive side. It is thought to describe some variance in the overall increase in TIC.

Factor 5 is more interesting as it shows evidence of Chemfos X on the positive side, though mixed with something else, and on the negative side shows a spectrum that looks similar to hexanedioic acid.

Factor 6 appears to be describing much smaller sources of variance in the data, the only two large peaks are for m/e 18 and m/e 28 so amongst other things this factor describes the variance between these background signals not accounted for by factors 2 and 3.

To assist further in visualising the sources of variance in the data, the scores plots for factor 2 versus factor 3 (figure 98) and factor 4 versus factor 5 (figure 99) are given. Figure 98 shows three distinct sources of variance, vertically from m/e 36, to the right from the background and to the left from m/e values found in dioctyl phthalate. Other sources of variance are indistinct and clustered about the origin.

To better show the other sources of variance factor 4 and 5 were plotted against each other in figure 99. This view of the data shows the earlier factors to the left and two further sources of variance at right top and bottom. Right top contains the m/e values seen in the spectrum of Chemfos X and right bottom contains the values associated with the hexanedioic acid spectrum.

In order to confirm the presence of the additives in the data it will be necessary to perform target tests and compare the predicted vectors with the suspected additives.

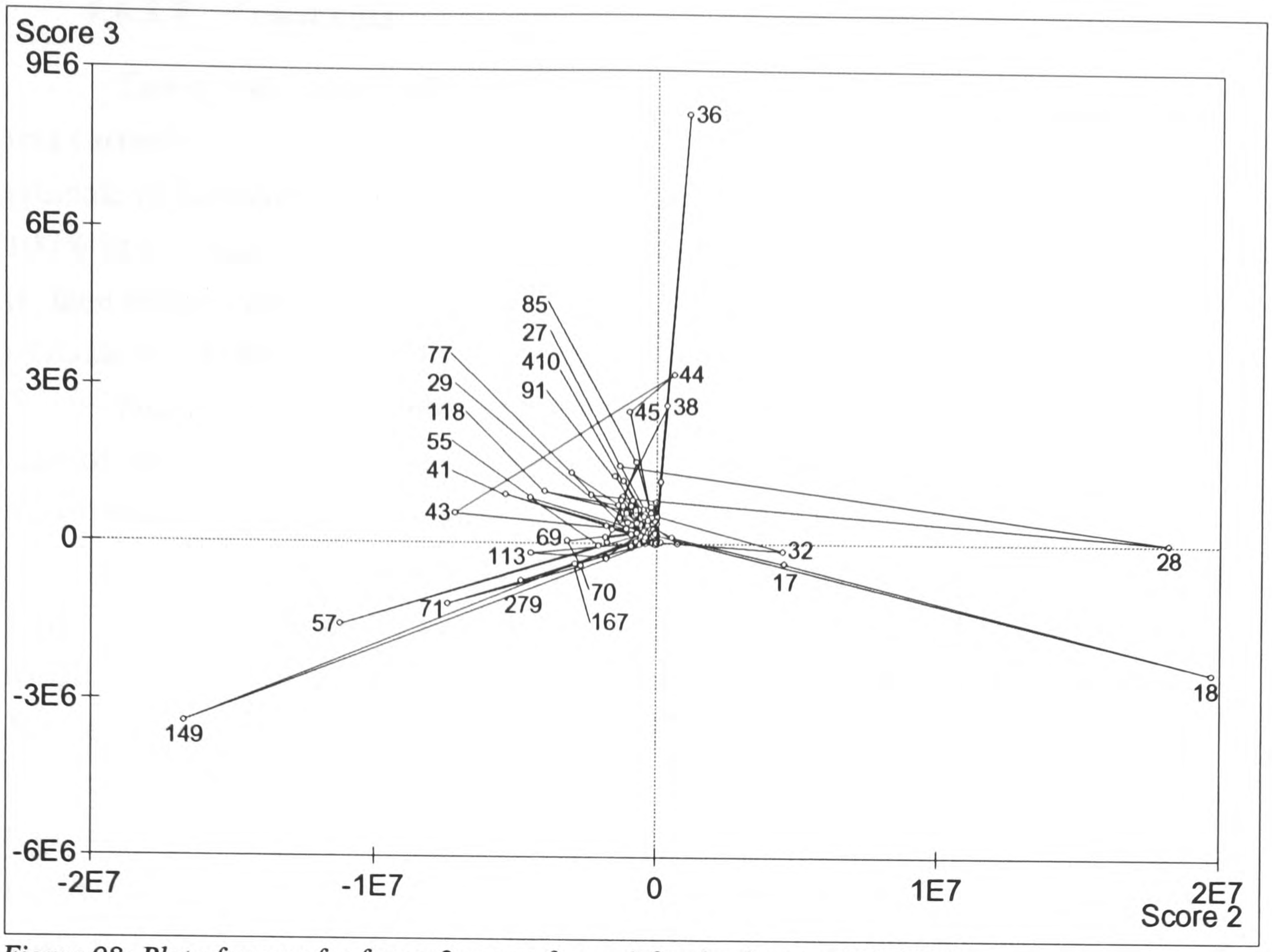


Figure 98: Plot of scores for factor 2 versus factor 3 for the data set MS1930A.

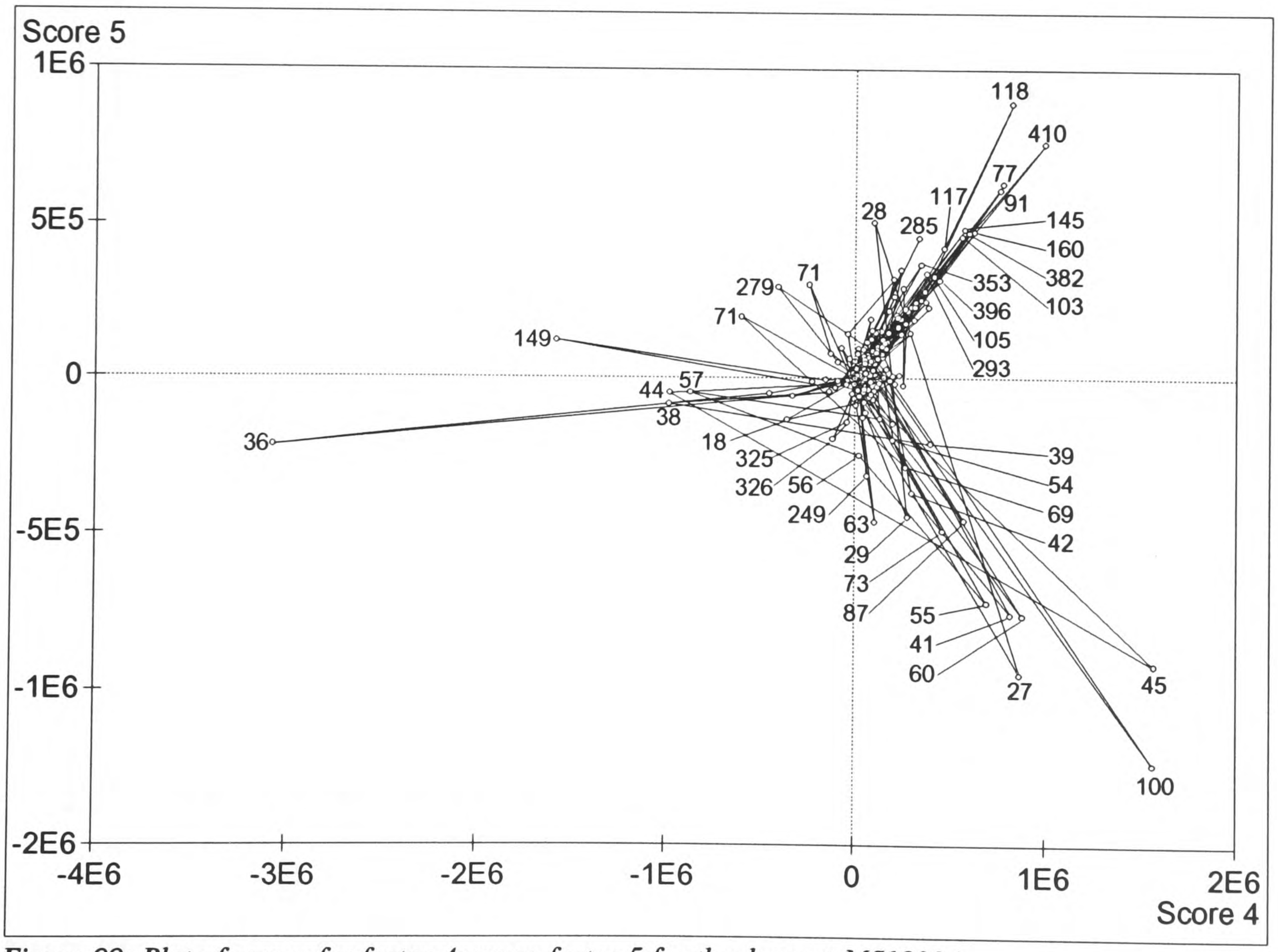


Figure 99: Plot of scores for factor 4 versus factor 5 for the data set MS1930A.

4.6.3.8. *Target tests*

Target tests were performed for each of the suspected components. The test was carried out using a test vector normalized to unit intensity, this allows a rough estimate of how close to the true factor the test vector is, as a true factor will return 100% of its original intensity and the further away from the true factor the test vector is, then smaller intensities are returned in the predicted vector. An arbitrary error estimate of 3% was entered for each test vector.

The results of the target tests for each of the five suspected components, on each of the data sets, are given in figures 100-104 on the following pages, along with the original test vectors for comparison.

Figure 100 shows the test for Chemfos X. The result of the test for MS1930 shows a good match for the base peak and one or two other peaks in the spectrum but mostly the fit is not very good. There is little evidence here to suggest that Chemfos X was a component in the sample. The test for MS1930A shows a good match between the test and predicted vectors. Most of the major peaks in the test vector are reproduced in the predicted vector and in roughly the same intensity ratios. The predicted vector appears to be considerably more complex and shows some strong peaks not present in Chemfos X. The result provides good evidence for the presence of Chemfos X or a similar compound in MS1930A.

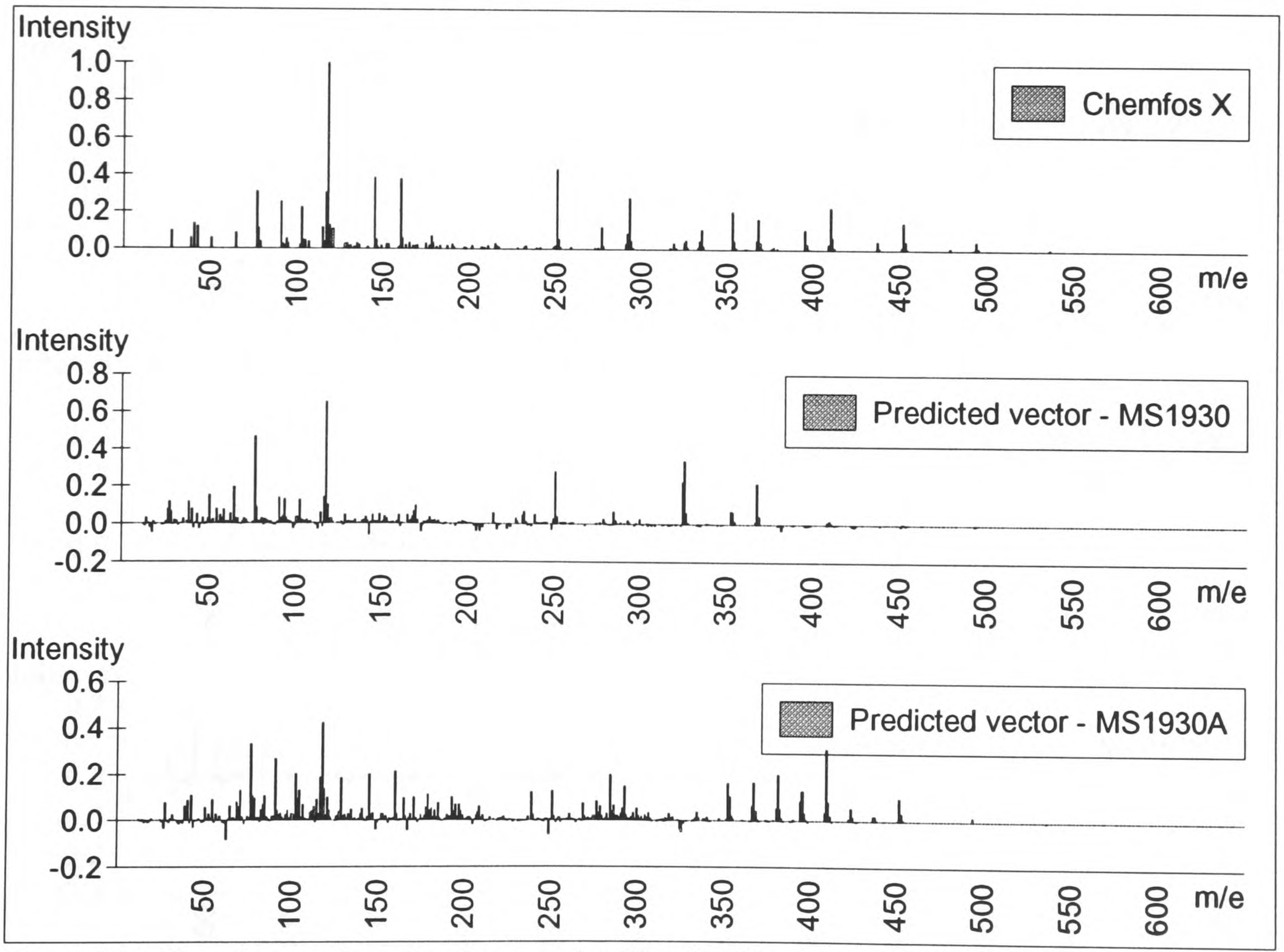


Figure 100: Plot of test and predicted vectors for Chemfos X in data sets MS1930 and MS1930A.

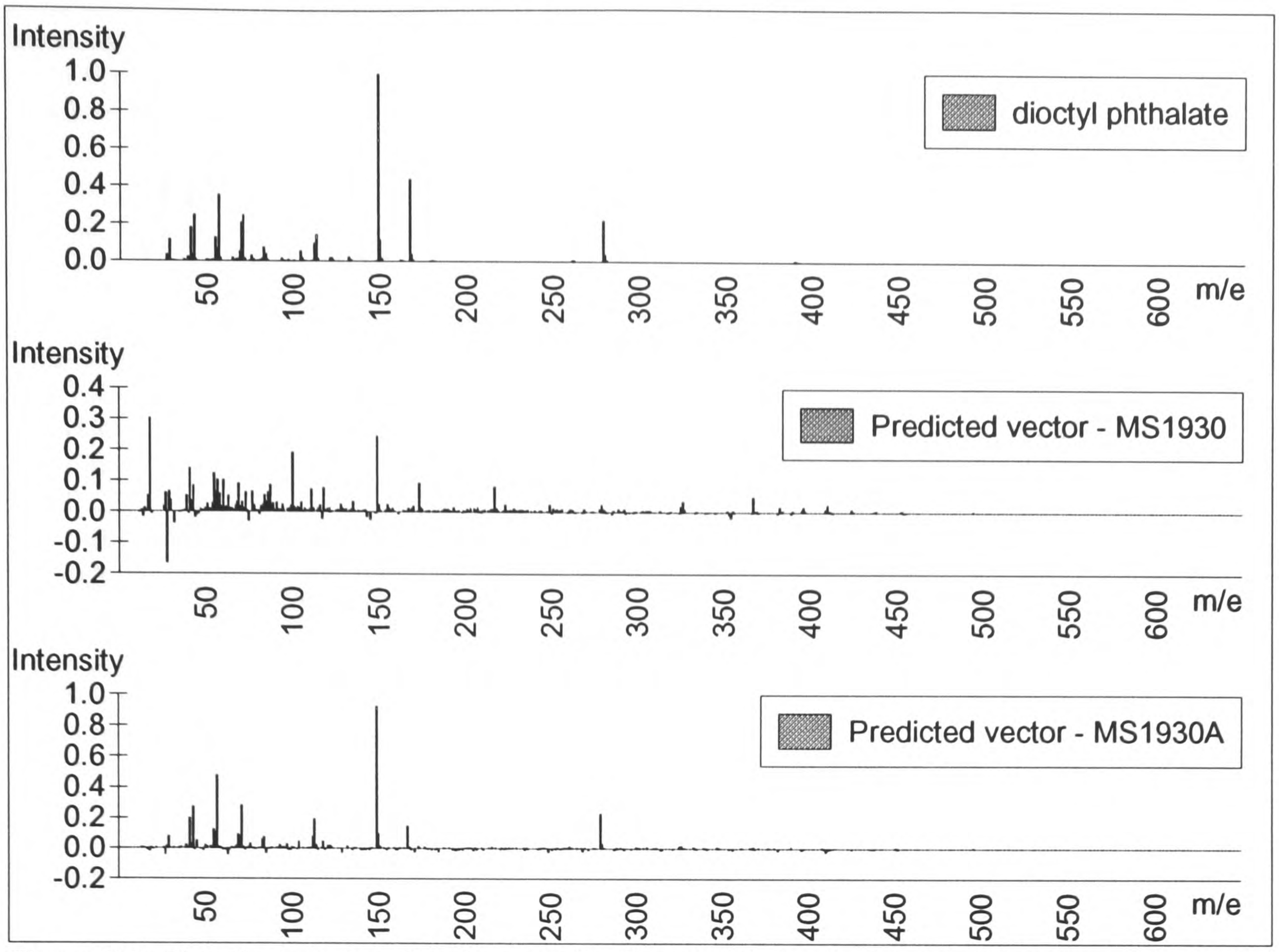


Figure 101: Plot of test and predicted vectors for dioctyl phthalate in data sets MS1930 and MS1930A.

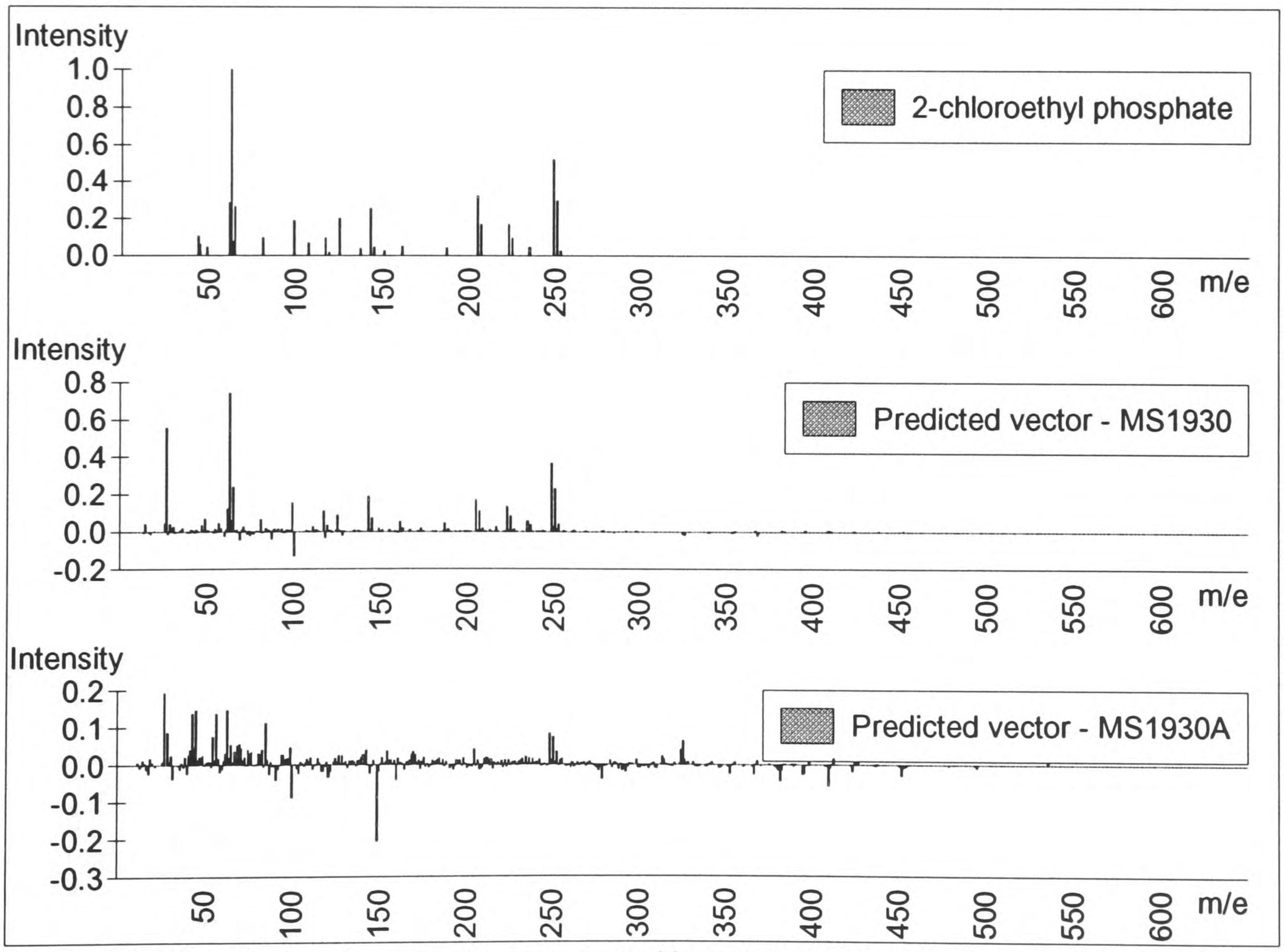


Figure 102: Plot of test and predicted vectors for 2-chloroethyl phosphate in data sets MS1930 and MS1930A.

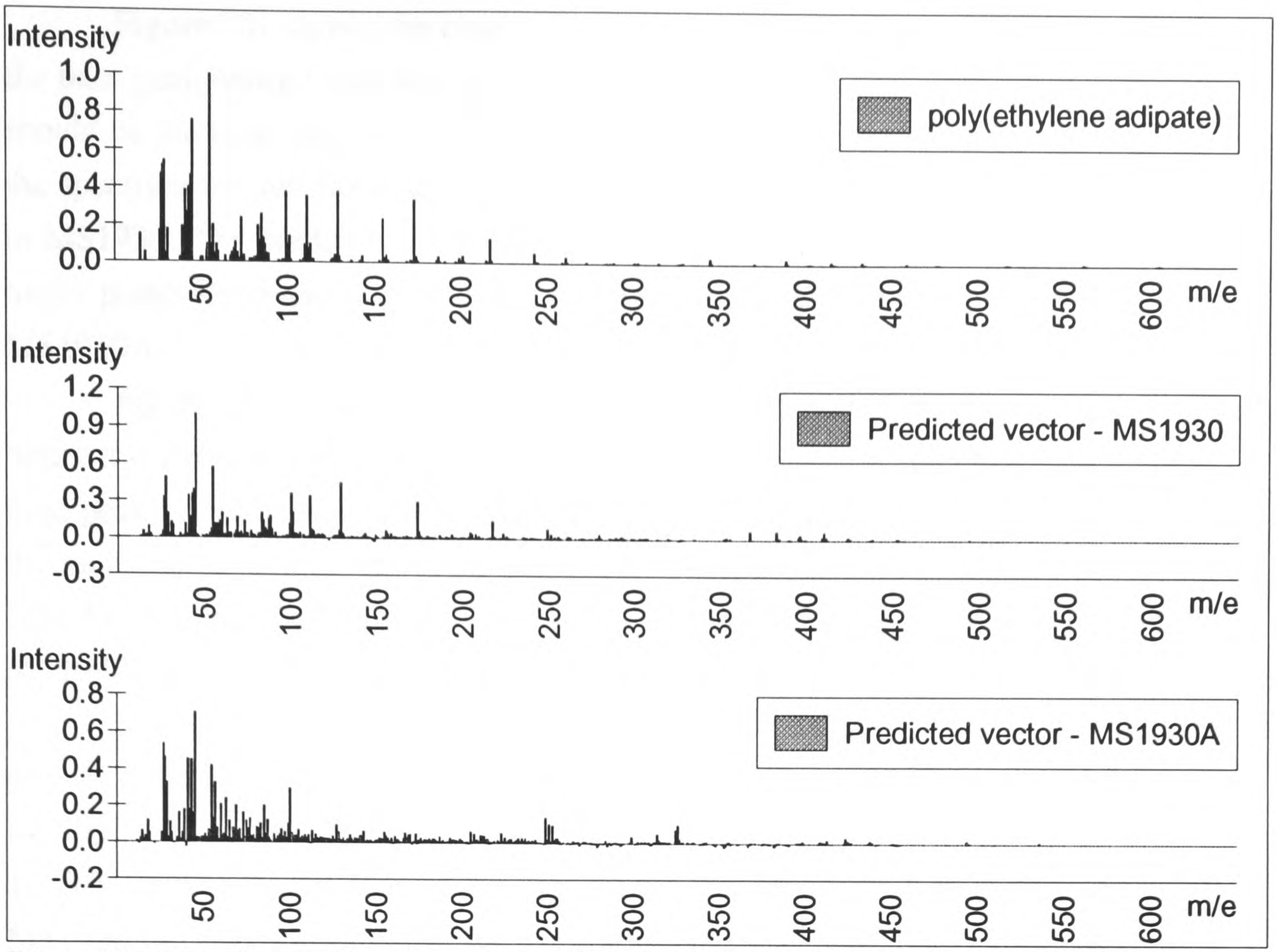


Figure 103: Plot of test and predicted vectors for poly(ethylene adipate) in data sets MS1930 and MS1930A

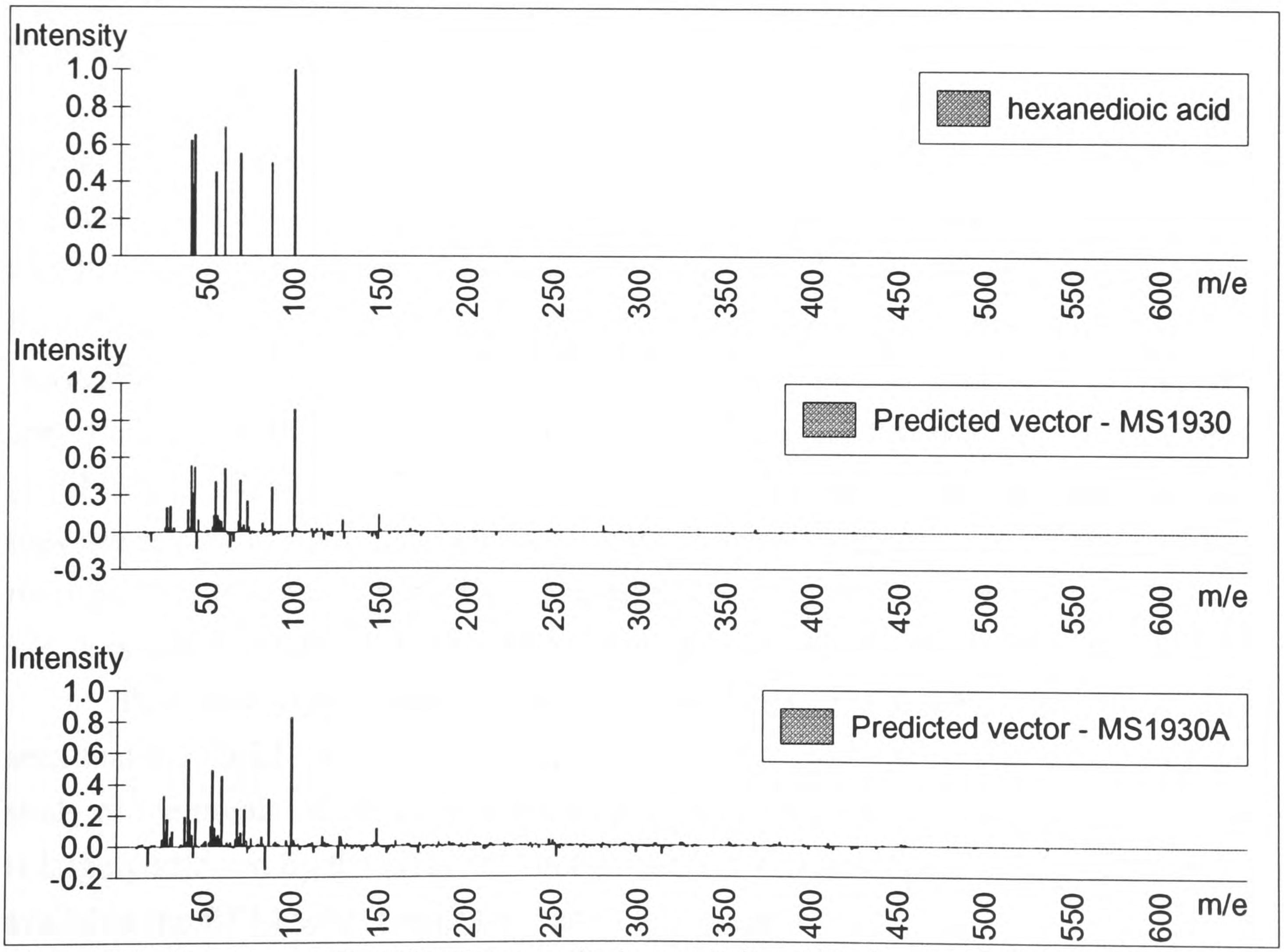


Figure 104: Plot of test and predicted vectors for hexanedioic acid in data sets MS1930 and MS1930A.

Figure 101 shows the results for dioctyl phthalate. The test for MS1930 shows the base peak being fitted wrongly to the m/e 18 peak, the m/e 149 peak, which should be the base peak only returns an intensity of 24%. Most of the other peaks in the spectrum are not fitted at all or fit poorly suggesting that this is not a component in MS1930. The predicted vector for MS1930A gives a very good match with all the major peaks fitted and in the correct ratios indicating that this is a component in MS1930A.

Figure 102 shows the results for 2-chloroethyl phosphate; this gives a much better test for MS1930. All the major peaks in the spectrum are fitted well and the base peak returns nearly 80% of its original intensity. This result certainly supports the inclusion of 2-chloroethyl phosphate as a component in MS1930. The results for MS1930A quite clearly show that 2-chloroethyl phosphate is not a component in MS1930A as almost none of the test vector peaks are fitted.

Figure 103 gives the results for poly(ethylene adipate). In this case the base peak is not fitted correctly in either result. Some of the peaks have a good fit for MS1930, but some major peaks are missing and the intensity pattern is not fitted at all. The results shown here do not support the inclusion of this as a component in MS1930 but they indicate that there is a component with similar spectral features to the test vector. The test for MS1930A shows that the predicted vector is trying to fit a completely different component to the major peaks of the test vector thus ruling out its inclusion in the components of MS1930A.

Figure 104 has the results from the target test for hexanedioic acid. This test differs from all the others as the test vector was taken from the Eight Peak Index[103] and therefore contains very little information. It can be seen that a very good fit for all the peaks in the test vector has been obtained for MS1930. All of the peaks are fitted and in the correct ratios (excluding an overly intense base peak). The test for MS1930A produces a spectrum very similar to the predicted vector for MS1930, there are some visible differences in the ratios of the peaks and two peaks at m/e 128 and m/e 149 are also visible. These extra peaks are also present in the MS1930 test; if they are related to the true factor then it cannot be hexanedioic acid, which only has a mass of 146. It would appear that something is a component of both data sets but further study is necessary to prove conclusively that it is hexanedioic acid.

The error terms produced from the target tests are shown in table 43. It can be seen that the RELI function shows a good match for all of the components though study of the predicted vectors has shown this is not the case. The calculation of RELI is largely affected by the error estimate in the test vectors and as no estimate was available the RELI values must be regarded as suspect.

The values for RET are generally above 0.03 for the vectors that give poor tests and below 0.03 for the vectors producing good tests. The best fitting test (dioctyl phthalate, MS1930A) has an RET value that approaches the 1% error estimate found in the mass spectrometer data. More interesting is the SPOIL value that is better for the poor tests than the good tests in MS1930 though in MS1930A it produces an acceptable value for dioctyl phthalate. The values calculated for %SL are all zero except for MS1930A and dioctyl phthalate, which produces a significance level of 6×10^{-8}. The failure of the error terms to provide a useful indication of true factors is ascribed to the large number of components in each test vector, few of which are significant to the identification of the component but all add to the error terms calculated, and the distribution of error in the data is not absolute but proportional, invalidating some of the assumptions used in the derivation of these error terms. It is clear from the result for dioctyl phthalate that when a perfect test is approached then the error terms start to behave normally but can be misleading until then.

	MS1930					MS1930A				
Test vector	A	B	C	D	E	A	B	C	D	E
Error est.	0.03	0.03	0.03	0.03	0.03	0.03	0.03	0.03	0.03	0.03
AET	0.038	0.044	0.029	0.036	0.027	0.038	0.016	0.049	0.047	0.041
REP	0.004	0.004	0.002	0.003	0.002	0.004	0.004	0.006	0.005	0.007
RET	0.038	0.044	0.029	0.036	0.027	0.038	0.015	0.049	0.047	0.040
SPOIL	10.2	10.9	15.7	10.4	15.9	10.0	3.8	8.4	9.5	6.0
RELI	79.3	68.7	104.3	83.3	111.8	80.2	192.3	62.0	64.7	75.3
%SL	0	0	0	0	0	0	6E-08	0	0	0

Key: A = Chemfos X; B = dioctyl phthalate; C = 2-chloroethyl phosphate;
D = poly(ethylene adipate); E = hexanedioic acid.

Table 43: Table of error terms resulting from target testing suspected components of MS1930 and MS1930A using 8 factors to model the data.

As a further attempt to quantify the similarity between the test and predicted vectors the product-moment correlation coefficient (equation 48) was calculated and also Bessel's inequality test[104] performed according to the formula in equation 50, where the coefficient of fit, *b*, is found from the ratio of the norms of the test and predicted vectors. The result of the function ranges between 0 and 1, with 1 being a perfect fit.

$$b=\frac{\|\hat{\mathrm{x}}\|^2}{\|\mathrm{x}\|^2} \tag{50}$$

The results from these calculations are given in table 44, the values have been converted to percentages for easier reading. It can be seen that for Chemfos X both score 75% for correlation and 57%,58% for Bessel inequality, but the results of the target tests are quite different (figure 100) and quite clearly indicate that these functions can give misleading values as well. The target test indicates quite clearly that Chemfos X is not a component of MS1930. The values found for dioctyl phthalate give good agreement, as do the values for 2-chloroethyl phosphate. Hexanedioic acid is clearly indicated in MS1930 and is also indicated to be in MS1930A though the Bessel inequality test is not as large as the correlation coefficient. The results also suggest that poly(ethylene adipate) is in both samples though the target tests suggest that this is not the case. It must be concluded from these answers that these measures also suffer from the same problem of being too sensitive to the small peaks in the spectra and therefore indicate the existence of components not present in the sample. The Bessel inequality test appears to be more sensitive to poorly fitting tests and may prove to be more valuable than the correlation coefficient. These two terms may be of more value in ruling out what is not present in the sample but as a unambiguous quantification of fit they appear unreliable.

Correlation coefficient	MS1930	MS1930A
Chemfos X	75%	75%
Dioctyl phthalate	48%	95%
2-chloroethyl phosphate	84%	38%
poly(ethylene adipate)	88%	79%
hexanedioic acid	92%	81%
Bessel inequality test	MS1930	MS1930A
Chemfos X	57%	58%
Dioctyl phthalate	24%	90%
2-chloroethyl phosphate	71%	16%
poly(ethylene adipate)	79%	65%
hexanedioic acid	85%	66%

Table 44: Table of correlation coefficients and Bessel inequality tests for the target tests of suspected components on data sets MS1930 and MS1930A.

The results from target testing positively identify 2-chloroethyl phosphate and hexanedioic acid in MS1930 and dioctyl phthalate in MS1930A. The results also indicate the presence of Chemfos X or a similar material in MS1930A. This is in agreement with the results obtained from the normal analysis. The target testing has also ruled out the presence of poly(ethylene adipate) in MS1930 and suggested the presence of hexanedioic acid in MS1930A, which was not observed at all in the normal analysis.

The target testing has raised question marks against the identity of the components identified as hexanedioic acid and Chemfos X from the normal analysis and these will be investigated further using iterative target testing.

4.6.3.9. Iterative target tests

To confirm the presence of each component in the data sets the strongest eight peaks of each suspected component were tested using iterative target testing (ITT). The iterative target testing procedure using differential subtraction, as developed earlier, was used. Each data set was tested for each component to confirm the negative as well as the positive results.

Eight peaks were chosen for the target testing as eight factors are responsible for the data. This complies with the requirement that the test vector must not contain fewer points than the number of factors comprising the data. This requirement is strictly only necessary when points in the vector are free floated, unlike the present case where a full test vector is being used but containing many zero values. By the use of eight factors it was hoped to converge towards the suspected component even if some of the masses selected arise from more than one component, as is almost certain to be the case for the lower mass peaks. Theoretically if a peak can be found which is unique to the component being looked for then only that mass number would be necessary for the test vector.

A problem arises with the test vector for hexanedioic acid, as this test vector was obtained from the Eight Peak Index[103] then only eight masses are available. It may be possible to use a subset of these masses if they are unique or are added to by different components but it is difficult to select these masses. All eight peaks are therefore used for the test vector and the resulting vector studied closely.

<u>Data set MS1930</u>

The results from target testing indicated that 2-chloroethyl phosphate and hexanedioic acid were present in this sample. The results of the iterative target testing for 2-chloroethyl phosphate are shown in figure 105. The resulting vector can be seen to match all the peaks of the test vector and follow their intensities reasonably well. The similarity between the unreduced component spectrum and the predicted vector is also marked. The only noticeable divergence is for m/e 28 and this is probably due to the subtraction of the background from the component spectrum. It can therefore be concluded that 2-chloroethyl phosphate is present in the sample.

Figure 106 shows the result of the ITT for hexanedioic acid. All of the eight peaks in the test vector are present in the predicted vector and are the eight strongest peaks in the spectrum; the intensities are also very similar. Little else may be said about the resulting spectrum without a complete spectrum of hexanedioic acid to compare it with. The conclusion, based upon this evidence is that hexanedioic acid is present in the sample.

Figure 107 shows the results for poly(ethylene adipate). The resulting vector has peaks for all of the eight peaks in the test vector but the intensities are wrong and the base peak in the predicted vector is not the same as the test vector. Consideration of the predicted vector with the complete spectrum shows that though some peaks appear, most do not. The component can be regarded as not being present in the sample.

Figure 108 gives the results for Chemfos X; this test gives a clear indication that the component is not present in the sample. Very few of the peaks in the test vector are fitted at all and the predicted vector looks nothing like the full spectrum of Chemfos X.

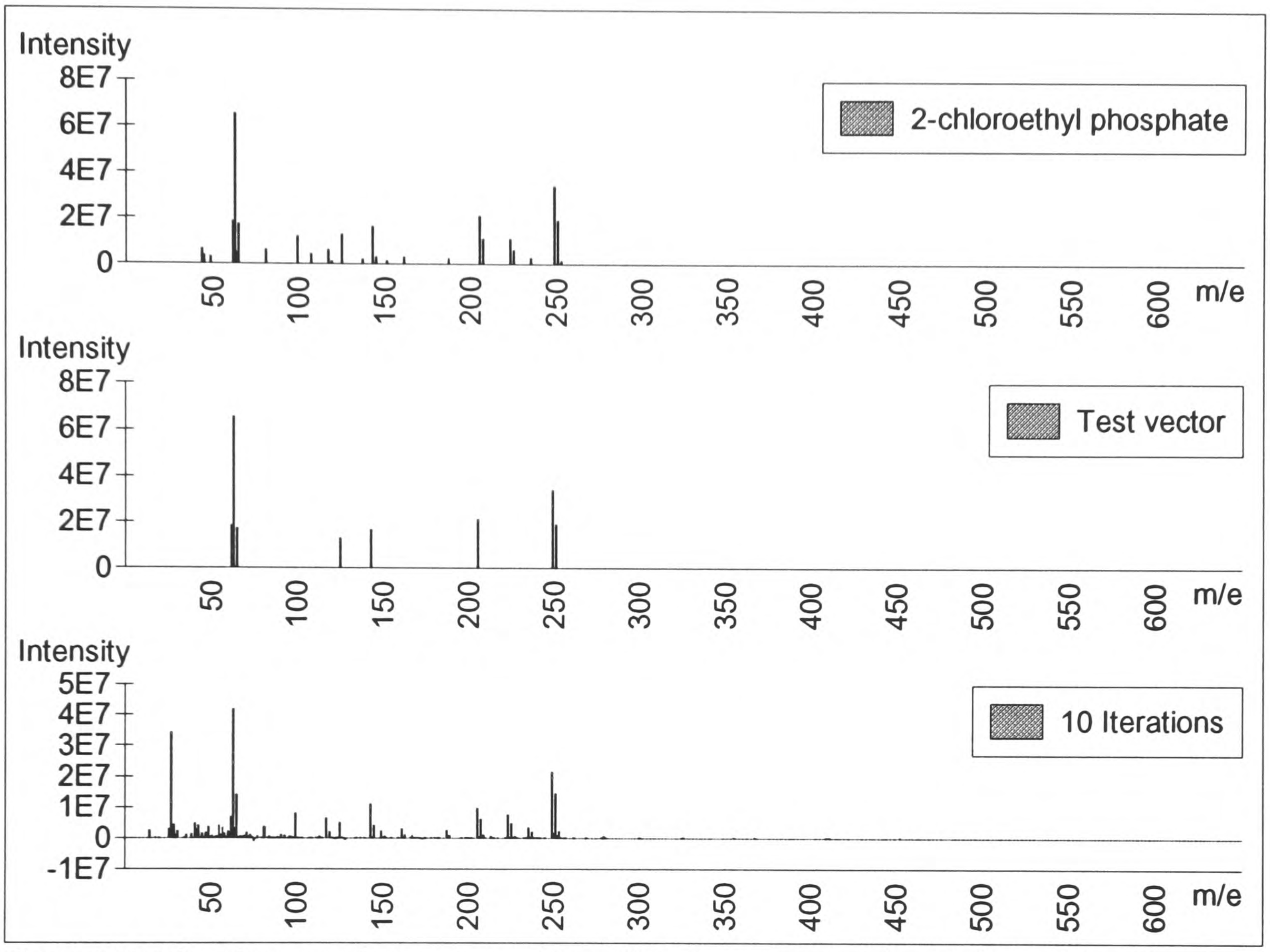


Figure 105: Plot of suspected component, reduced 8 peak test vector and result from 10 iterations for 2-chloroethyl phosphate in data set MS1930, exhibiting a positive result.

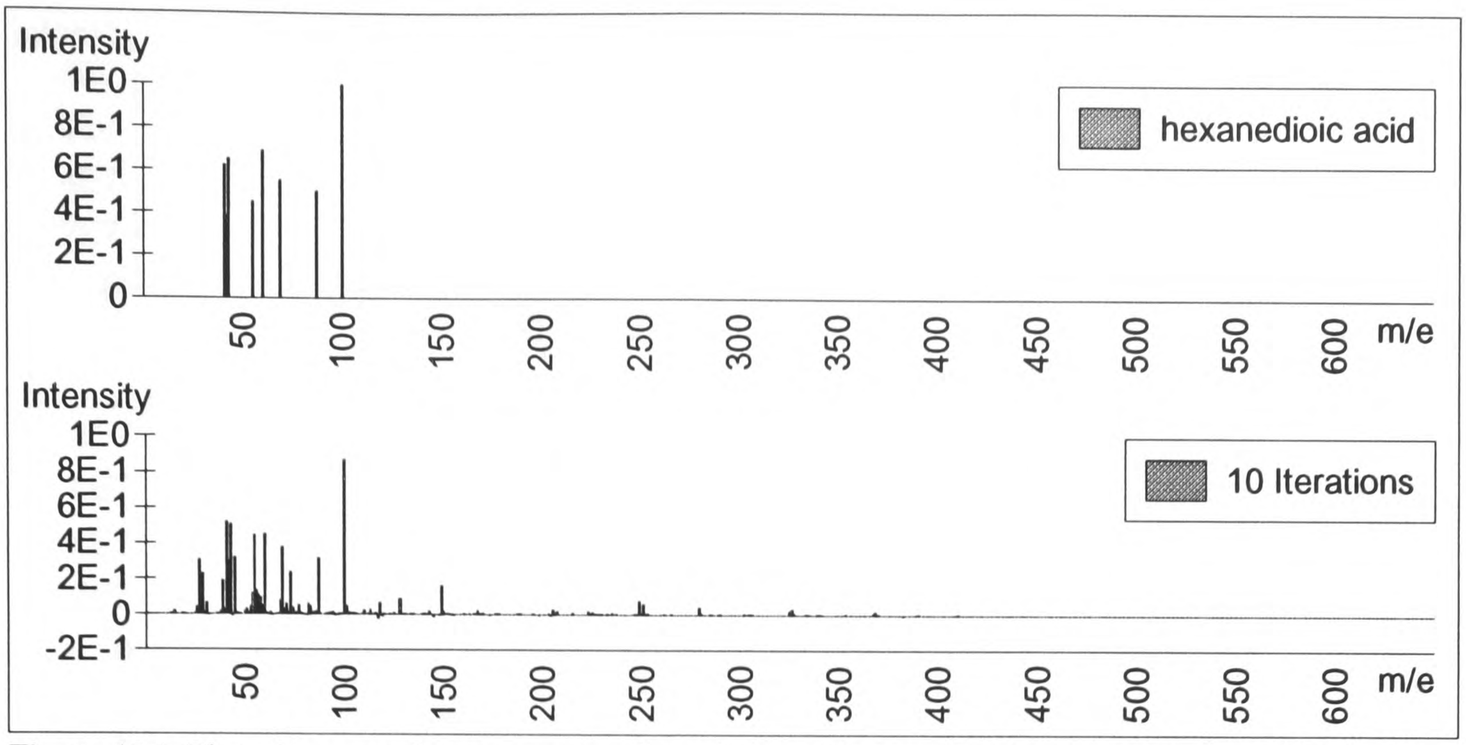


Figure 106: Plot of suspected component and result from 10 iterations for hexanedioic acid in data set MS1930, exhibiting a positive result.

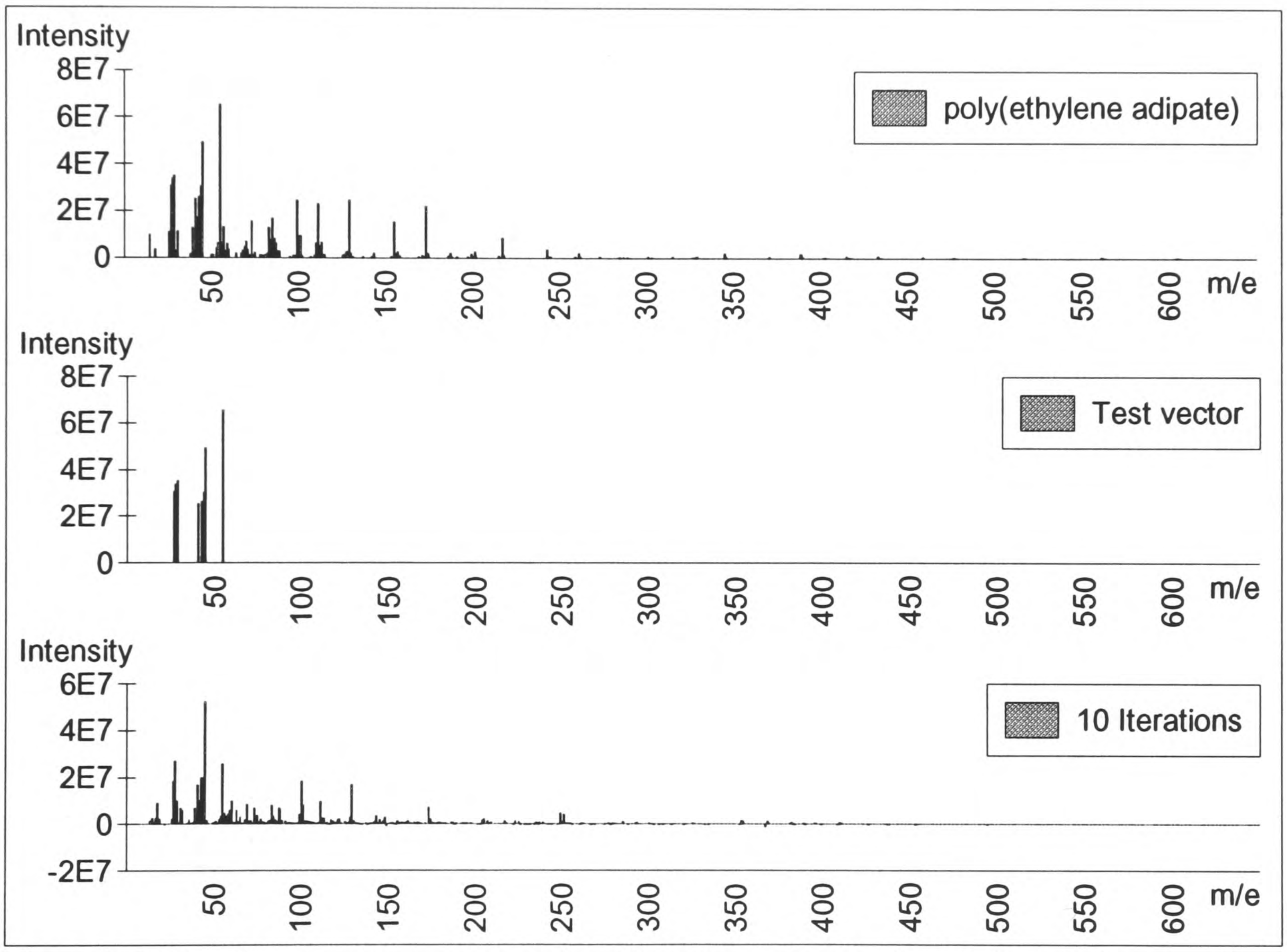


Figure 107: Plot of suspected component, reduced 8 peak test vector and result from 10 iterations for poly(ethylene adipate) in data set MS1930, exhibiting a negative result.

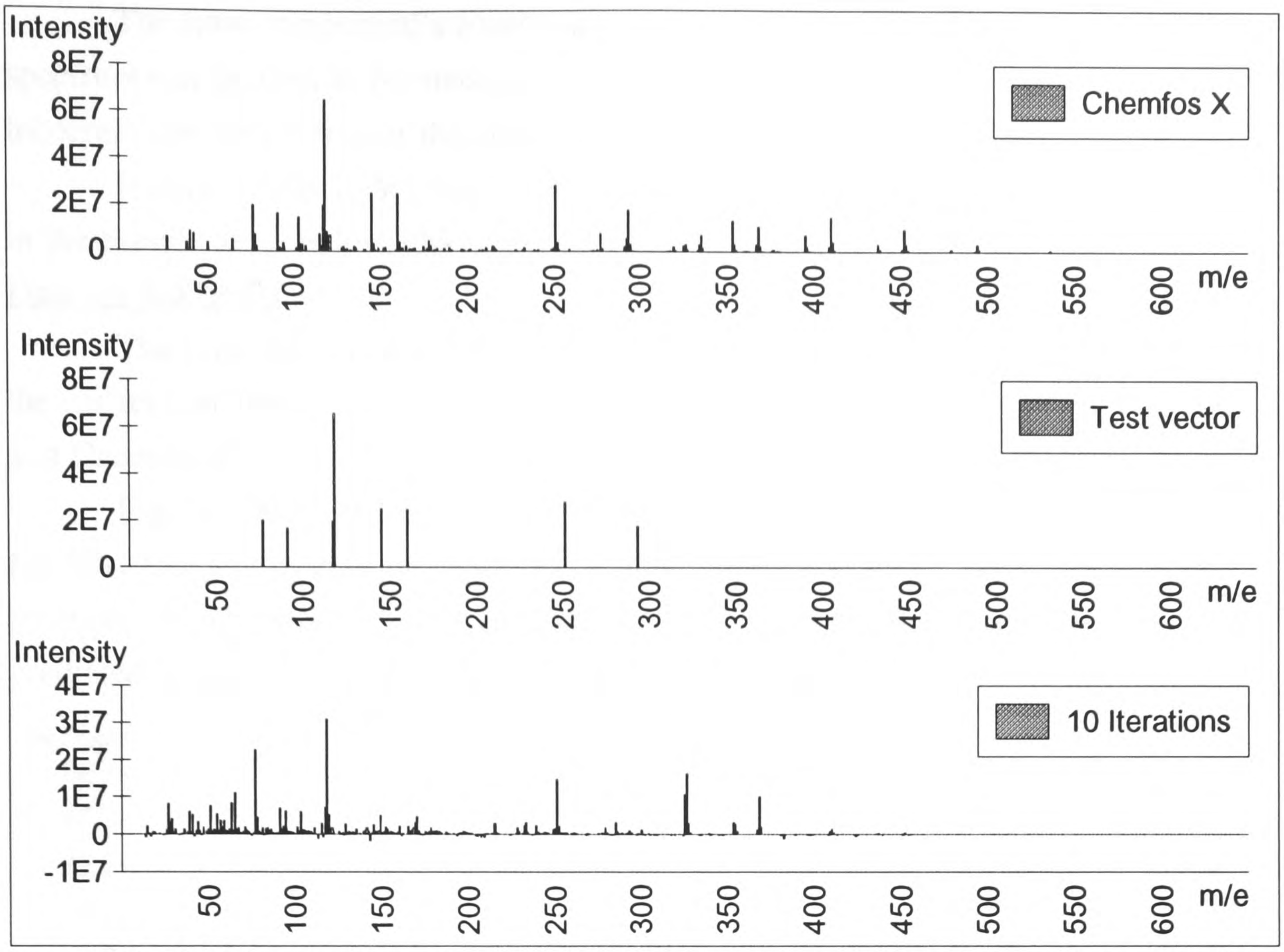


Figure 108: Plot of suspected component, reduced 8 peak test vector and result from 10 iterations for Chemfos X in data set MS1930, exhibiting a negative result.

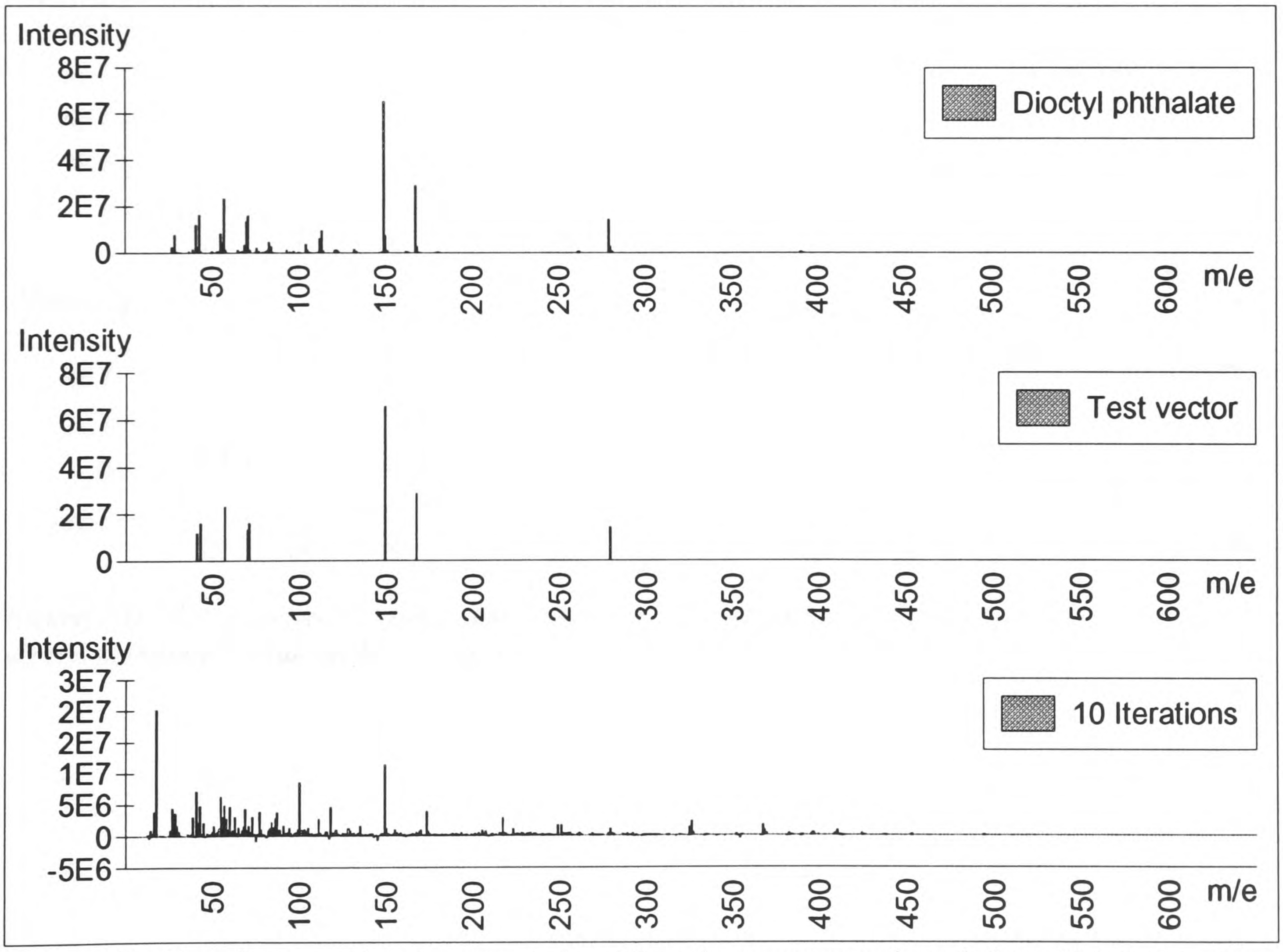


Figure 109: Plot of suspected component, reduced 8 peak test vector and result from 10 iterations for dioctyl phthalate in data set MS1930, exhibiting a negative result.

The final component's results are shown in figure 109. The dioctyl phthalate spectrum can be seen to be missing from the predicted vector. The base peak is incorrect and only a few of the peaks in the test vector have been fitted.

It may be concluded that out of the components tested; the only ones present in the sample are 2-chloroethyl phosphate and hexanedioic acid.

Data set MS1930A

The same test vectors were used on data set MS1930A in order to substantiate the earlier conclusion from target testing that the sample contained dioctyl phthalate and Chemfos X.

Figure 110 gives the result for dioctyl phthalate, unlike the predicted vector for MS1930, in this case the predicted vector fits the test vector very well in both position and intensity. The original spectrum of dioctyl phthalate also fits the predicted vector very well confirming the presence of dioctyl phthalate in the sample.

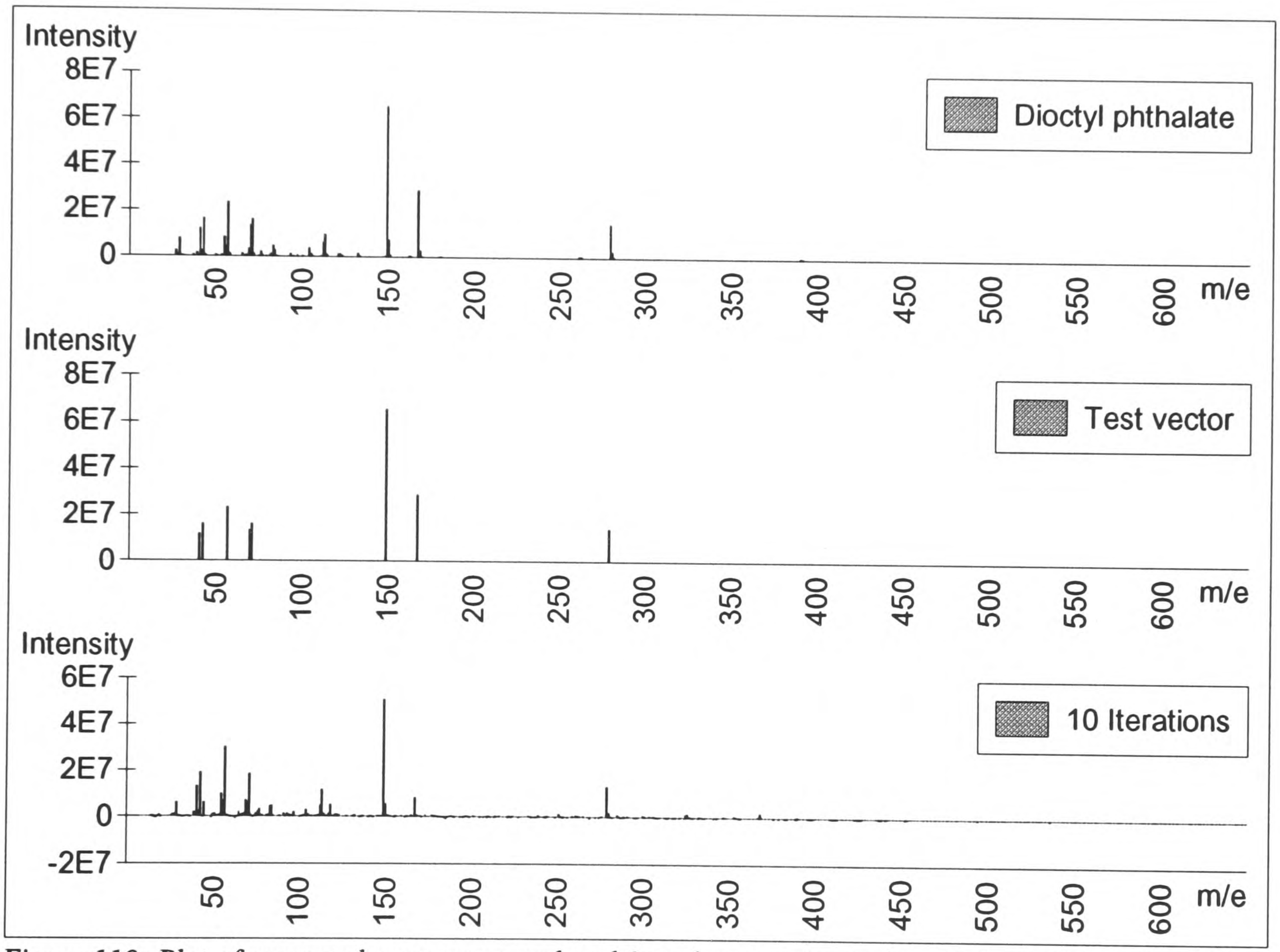


Figure 110: Plot of suspected component, reduced 8 peak test vector and result from 10 iterations for dioctyl phthalate in data set MS1930A, exhibiting a positive result.

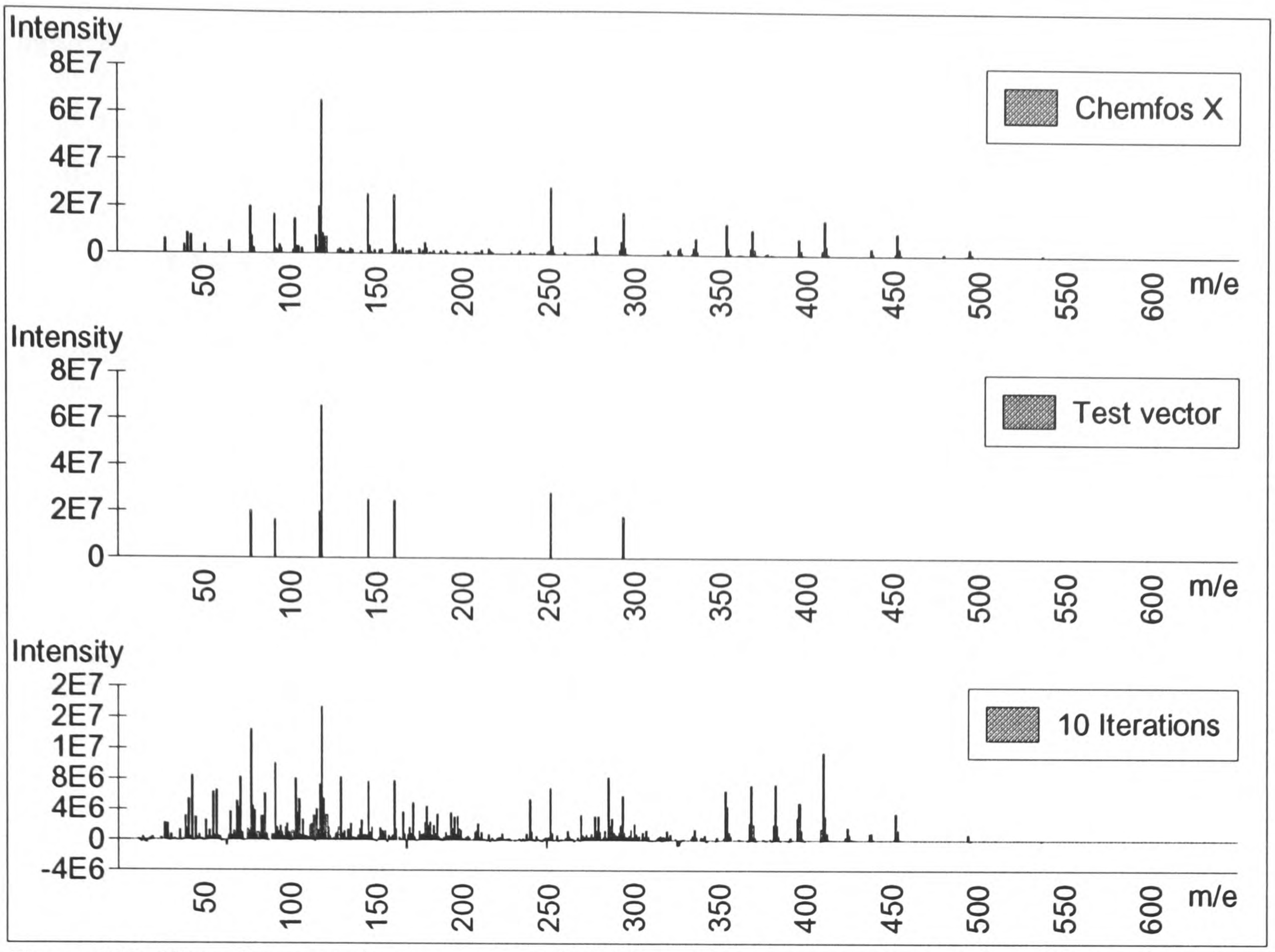


Figure 111: Plot of suspected component, reduced 8 peak test vector and result from 10 iterations for Chemfos X in data set MS1930A, exhibiting a positive result (see text).

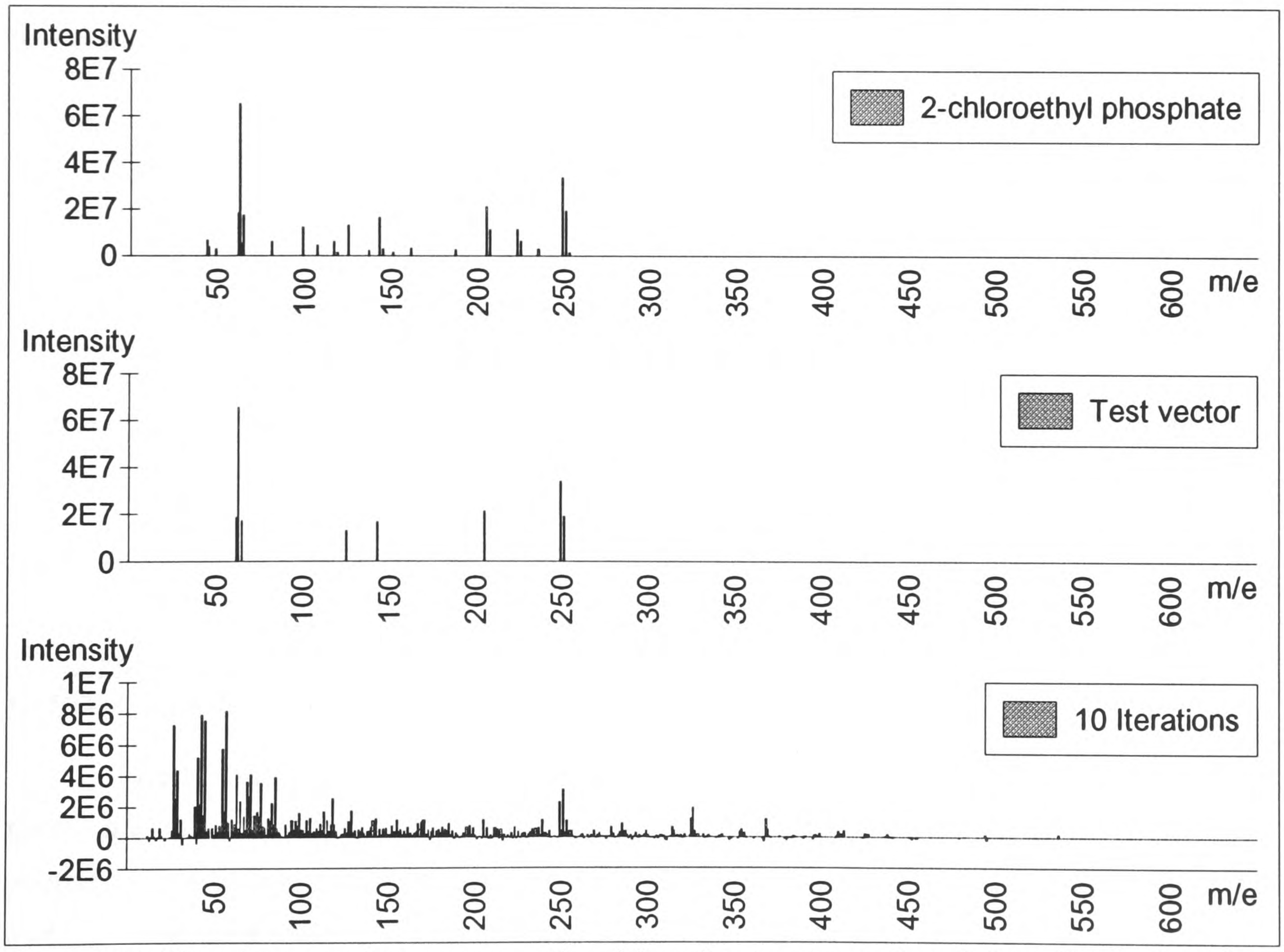


Figure 112: Plot of suspected component, reduced 8 peak test vector and result from 10 iterations for 2-chloroethyl phosphate in data set MS1930A, exhibiting a negative result.

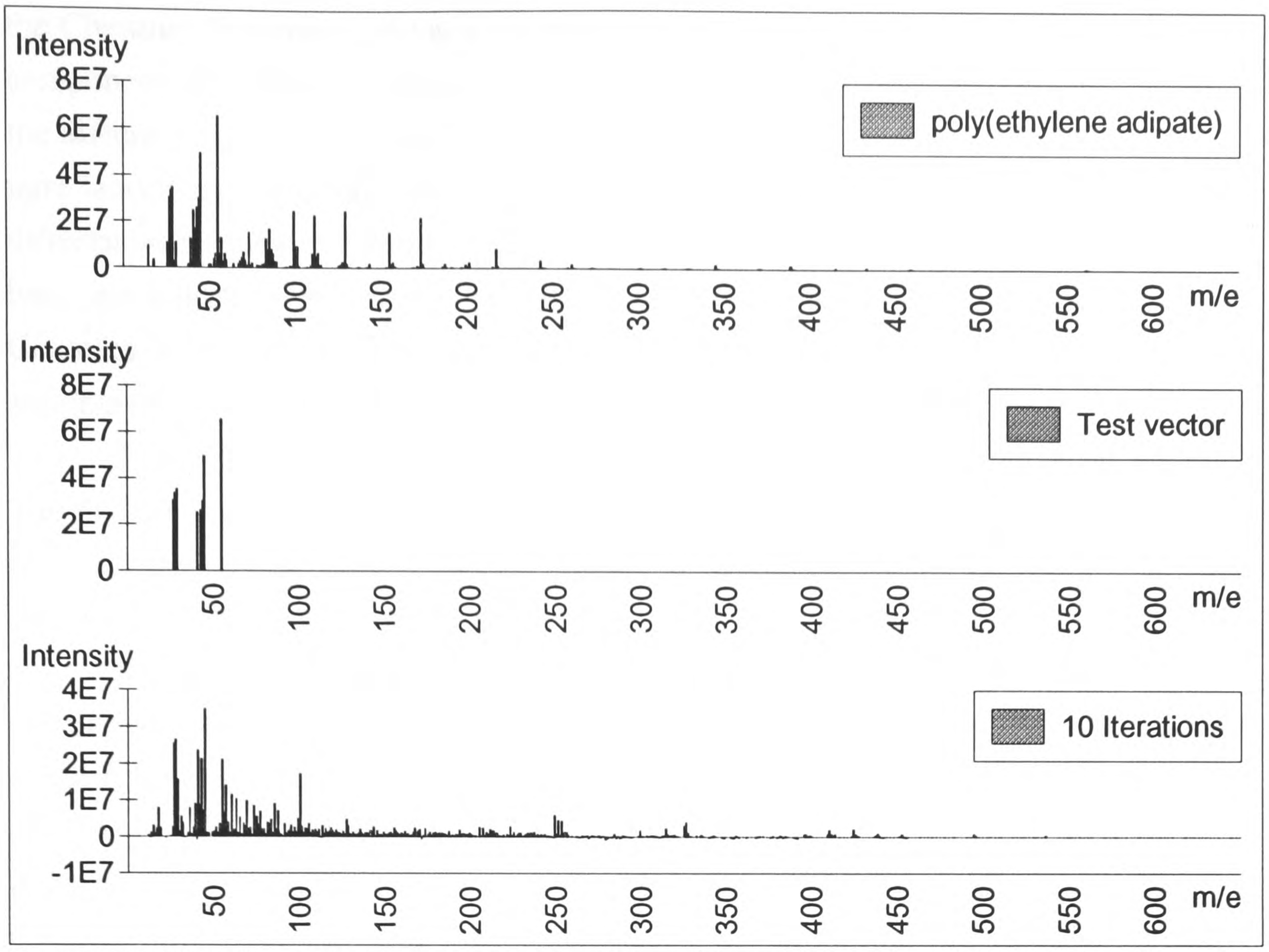


Figure 113: Plot of suspected component, reduced 8 peak test vector and result from 10 iterations of poly(ethylene adipate) in data set MS1930A, exhibiting a negative result.

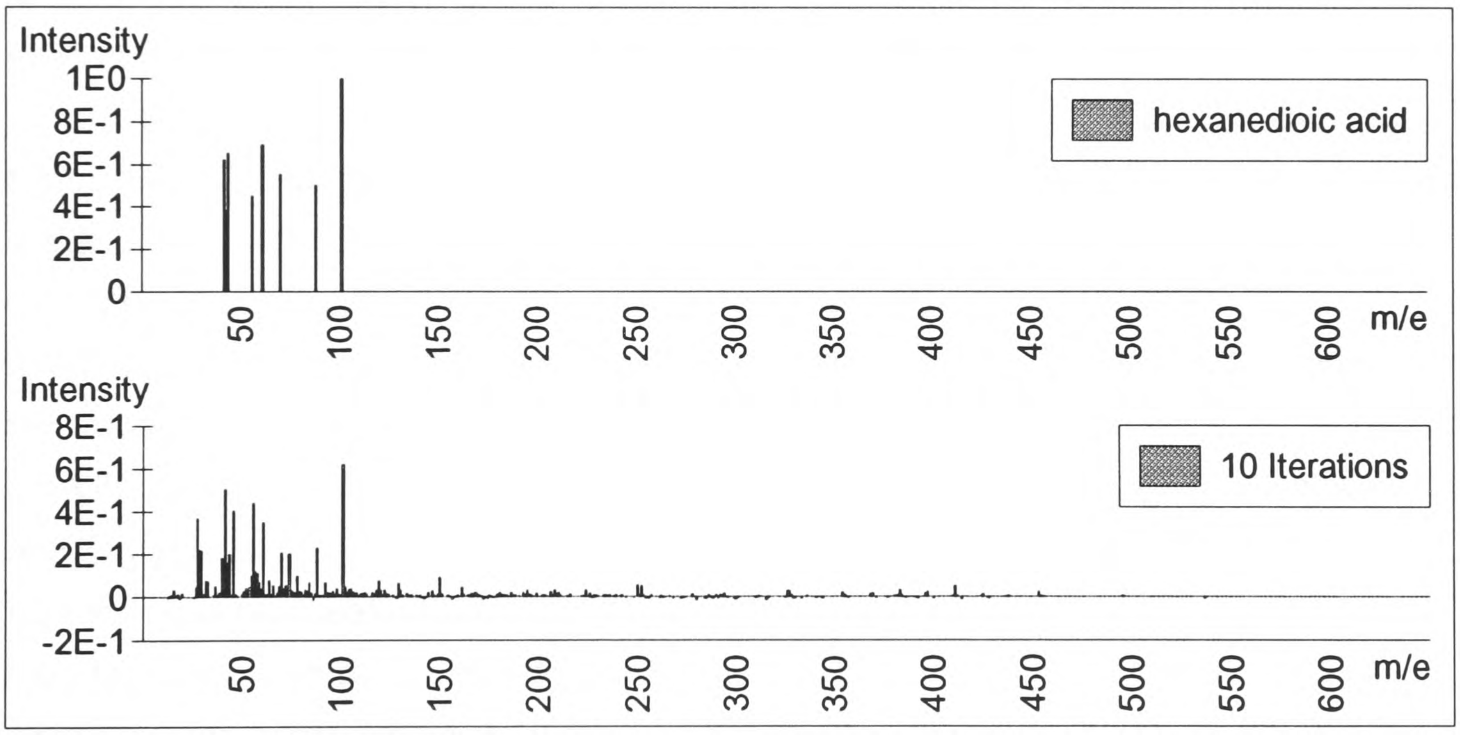


Figure 114: Plot of suspected component and result from 10 iterations for hexanedioic acid in data set MS1930A, exhibiting a positive result.

The results for Chemfos X are shown in figure 111. The predicted vector shows all the peaks in the test vector but the eight strongest peaks in the predicted vector do not correspond with the test vector. Comparison of the component spectrum and the predicted spectrum shows a very good match for all the major peaks in the spectrum but with intensity variations. The predicted spectrum is also considerably more complex than the sample spectrum. The differences may be due to mixing into the Chemfos X variance of variance from another component or a different component of similar chemical composition being present. Another explanation for the differences may be the sample itself, Chemfos X is a commercial product, not a pure laboratory chemical, and the observed differences may be due to production differences between the two samples. If the result of the iterative target testing had been less complicated it would have been possible to conclude the presence of Chemfos X, but the addition of so many extra peaks reduces confidence in that assignment.

The results from testing for 2-chloroethyl phosphate are shown in figure 112. The predicted vector shows that the base peak has not been fitted as well as many of the other peaks. The overall spectrum is not at all like that of the component spectrum and thus can be concluded as not being present in the sample.

Figure 113 gives the results for poly(ethylene adipate) for which similar conclusions to 2-chloroethyl phosphate must be drawn. The base peak in the spectrum is not fitted and the overall spectrum does not resemble the component spectrum at all. The low mass component of the test vector is fitted reasonably well which serves to highlight the complicated set of sources that produce peaks in this region. The different sources of the same mass peaks have allowed an approximation to the test vector to be produced by combining several sources, an approximation that falls down at the higher mass values. The evidence from tests using only a few low mass peaks must be treated sceptically in cases where several sources for the peaks may exist.

The last component to be tested is hexanedioic acid; the results are given in figure 114. The target test for this component suggested that it might be present in the sample, the predicted vector from ITT also suggests that hexanedioic acid is present on the basis of all the peaks being matched. Closer observation shows that the intensities are poorly matched and that the eight strongest peaks are not the same as the test vector, m/e 27 figuring strongly in the predicted vector. A study of the Eight Peak Index[103] revealed three entries for adipic (hexanedioic) acid, all different and one containing m/e 27 as a component. However, also included in the predicted vector is a strong peak at m/e 73 that was not present in any of the adipic acid spectra. Bearing in mind the previous paragraph's observations on the use of low mass peaks in a test vector and without a complete spectrum of hexanedioic acid it must be concluded that there is a component very similar to hexanedioic acid present in the sample but that it is not hexanedioic acid. A study of other possible candidates in the Eight Peak Index[103] produced no possible vectors worth consideration.

To further test the applicability of correlation coefficient and Bessel inequality test, these functions were calculated between the full spectrum of each component and the result from the iterative target test and the results are given in table 45 expressed as percentages. In this case the Bessel inequality test results are considerably smaller than for the target tests. The values for MS1930 indicate the presence of 2-chloroethyl phosphate, poly(ethylene adipate) and hexanedioic acid though at 44% from the Bessel inequality test the poly(ethylene adipate) could be neglected. The results for MS1930A indicate Chemfos X, dioctyl phthalate and hexanedioic acid though use of the Bessel inequality test would result in the neglect of ChemfosX. The assumption in all of these cases is that a true component would have a value greater than 50% though this is a arbitrary figure which could be modified on the basis of further experience.

Correlation coefficient	MS1930	MS1930A
Chemfos X	73%	73%
dioctyl phthalate	44%	94%
2-chloroethyl phosphate	82%	27%
poly(ethylene adipate)	86%	79%
hexanedioic acid	89%	76%
Bessel inequality test	MS1930	MS1930A
Chemfos X	34%	25%
dioctyl phthalate	15%	74%
2-chloroethyl phosphate	57%	7%
poly(ethylene adipate)	44%	38%
hexanedioic acid	78%	56%

Table 45: Table of correlation coefficients and Bessel inequality tests for the iterative target testing of suspected components of data sets MS1930 and MS1930A.

The results presented here indicate again the importance of visual comparison of the test and predicted vectors as the calculated values alone are misleading. Generally the conclusions reached by these values are in agreement with the conclusions from visual assesment. Particularly interesting is the very low value of the Bessel inequality test for Chemfos X in MS1930A which adds weight to the conclusion that it is a chemically similar compound but not Chemfos X.

The uncertainty about Chemfos X and hexanedioic acid in MS1930A was looked at further using the information contained in the scores plot (figure 99) to draw single ion pyrograms of some of the obvious sources of variance. Figure 115 below shows m/e 60 and 100, connected with hexanedioic acid and m/e 410 and 118, connected with Chemfos X. These pairs of m/e values appear to belong to the same components, to identify the components a different method of target testing was tried. The intensities of each m/e value were found at each components maximum in the single ion pyrogram and the largest value set to one, the other component was then scaled accordingly and the values entered in to test vectors for each component. The iterative target testing routine was then modified so that after each iteration values below a threshold were set to zero in an attempt to remove the influence of the small peaks on the convergence of the vector. When this had been tried previously the magnitude of values in the vector diminished with each iteration until an all zero vector resulted. To avoid this problem and also to hopefully speed the convergence, all non zero entries in the original test vector (in this case the two chosen m/e values) were reentered into the test vector after each iteration. This approach has the drawback of forcing the direction of the vector to comply with the applied constraints, but if true values are chosen then convergence should be accelerated. This technique of vector reinforcement was used for both of the pairs of m/e values, the threshold below which values were zeroed was set by starting with a large value (0.2), testing the vector and observing how many values had changed in the new test vector excluding the original m/e values. The size of the threshold was reduced until between ten and twenty values appeared in the new test vector. The process was then restarted with the new parameters and iterated until reasonably constant values were obtained for difference term and correlation coefficient.

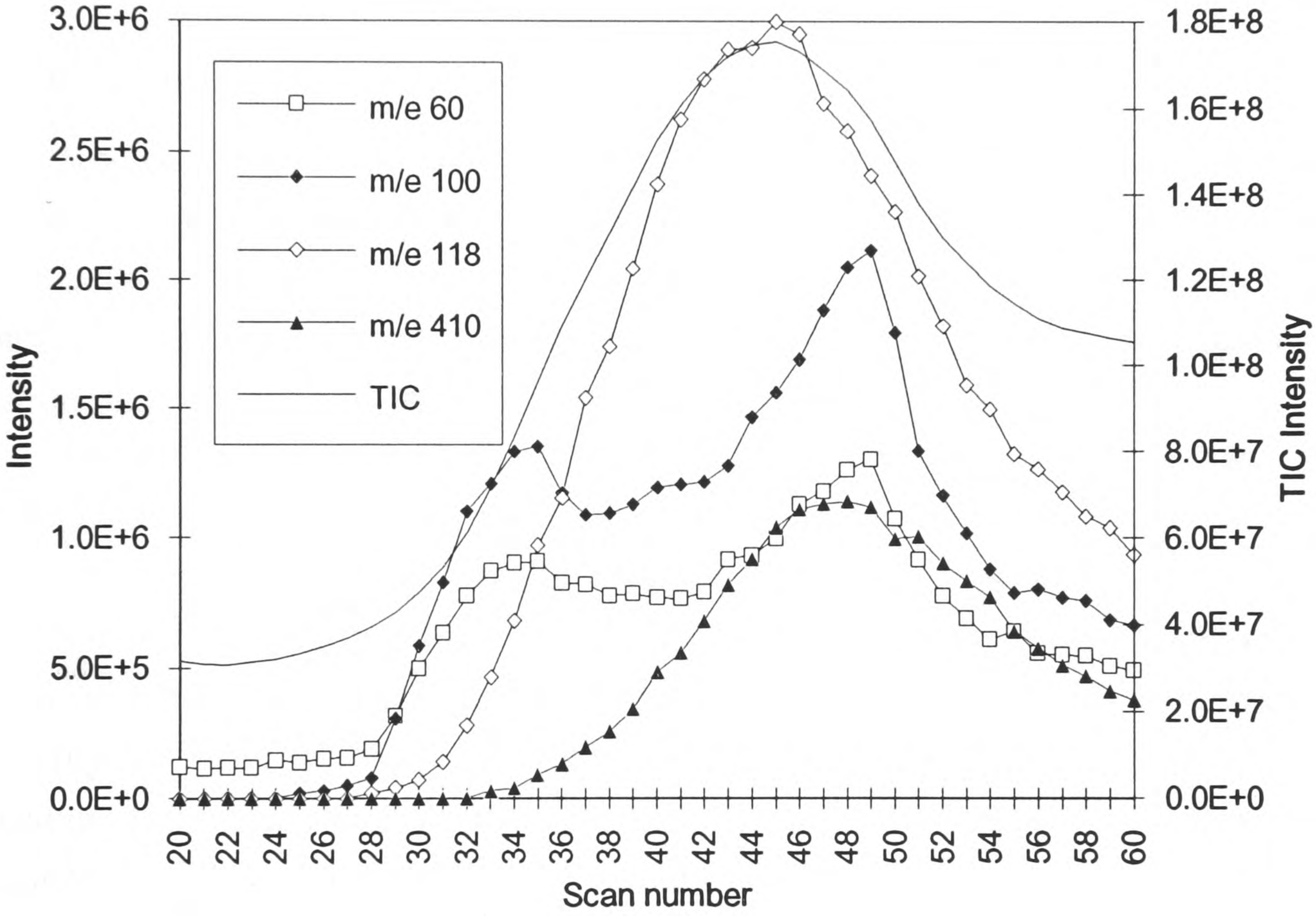


Figure 115: Single ion pyrograms for m/e 60, 100, 118 and 410 for data set MS1930A.

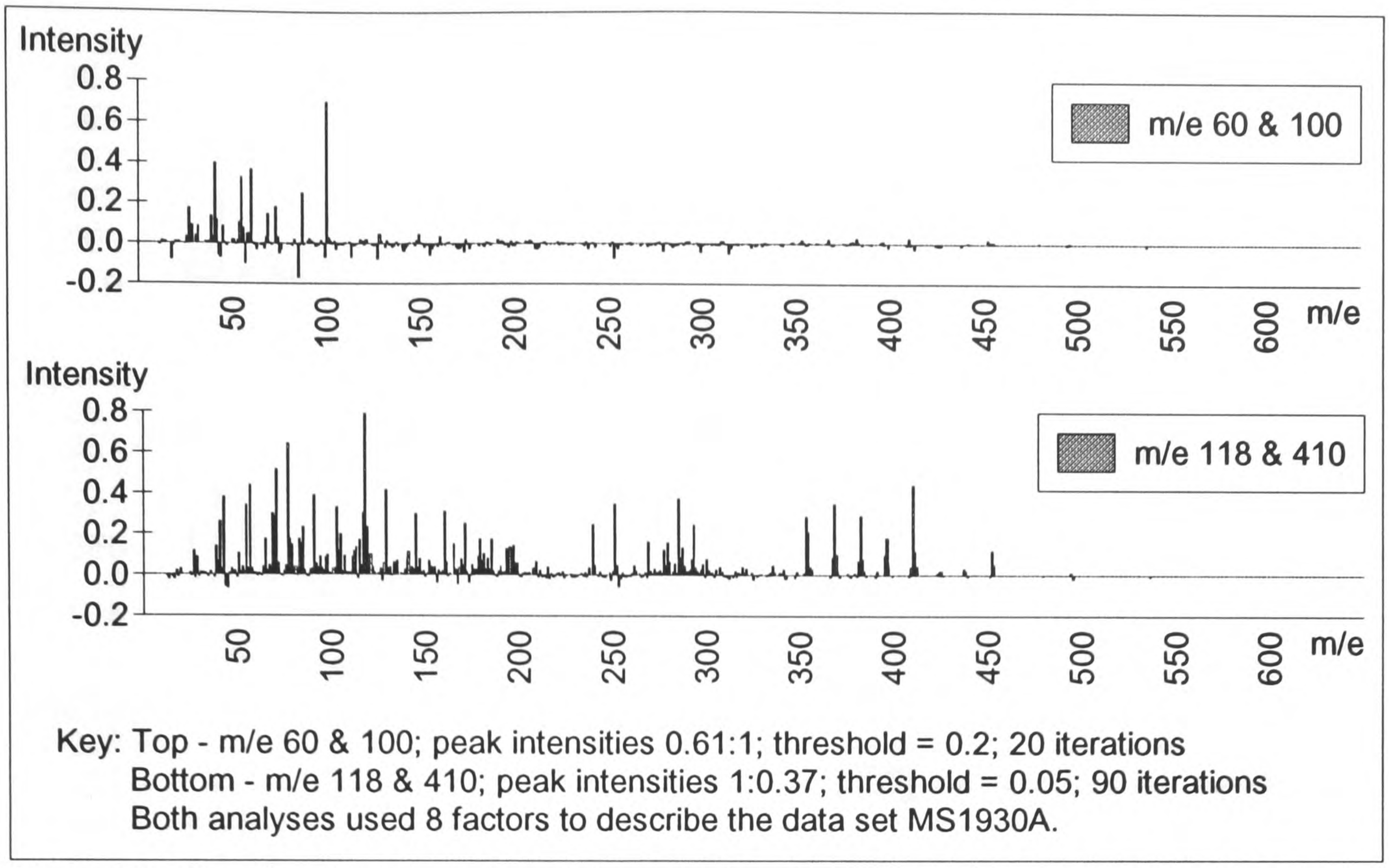


Figure 116: Plot of results from vector reinforcement iterative target testing of m/e 60 & 100 and m/e 118 & 410 in data set MS1930A.

The results from the iterative target testing are shown in figure 116. The top spectrum contains as its eight strongest peaks all the peaks in hexanedioic acid except for the addition of m/e 73. The results for m/e 118 & 410 show a far more complicated spectra than that of Chemfos X and the identification of that component from the standard analysis must now be regarded as suspect.

To resolve the ambiguity surrounding the identity of hexanedioic acid in MS1930A a complete spectrum was obtained by analysis of a sample of pure material[83]. The pure spectrum was then target tested against both MS1930 and MS1930A and the results shown in figure 117. The results from the tests both indicate the presence of hexanedioic acid. The values for correlation coefficient and Bessel inequality test were, MS1930 97%,95% and MS1930A 93%,87%; further strong indication of the component in both data sets. Present in the complete spectrum of hexanedioic acid is a strong peak at m/e 73 thus explaining the appearance of that peak in earlier tests.

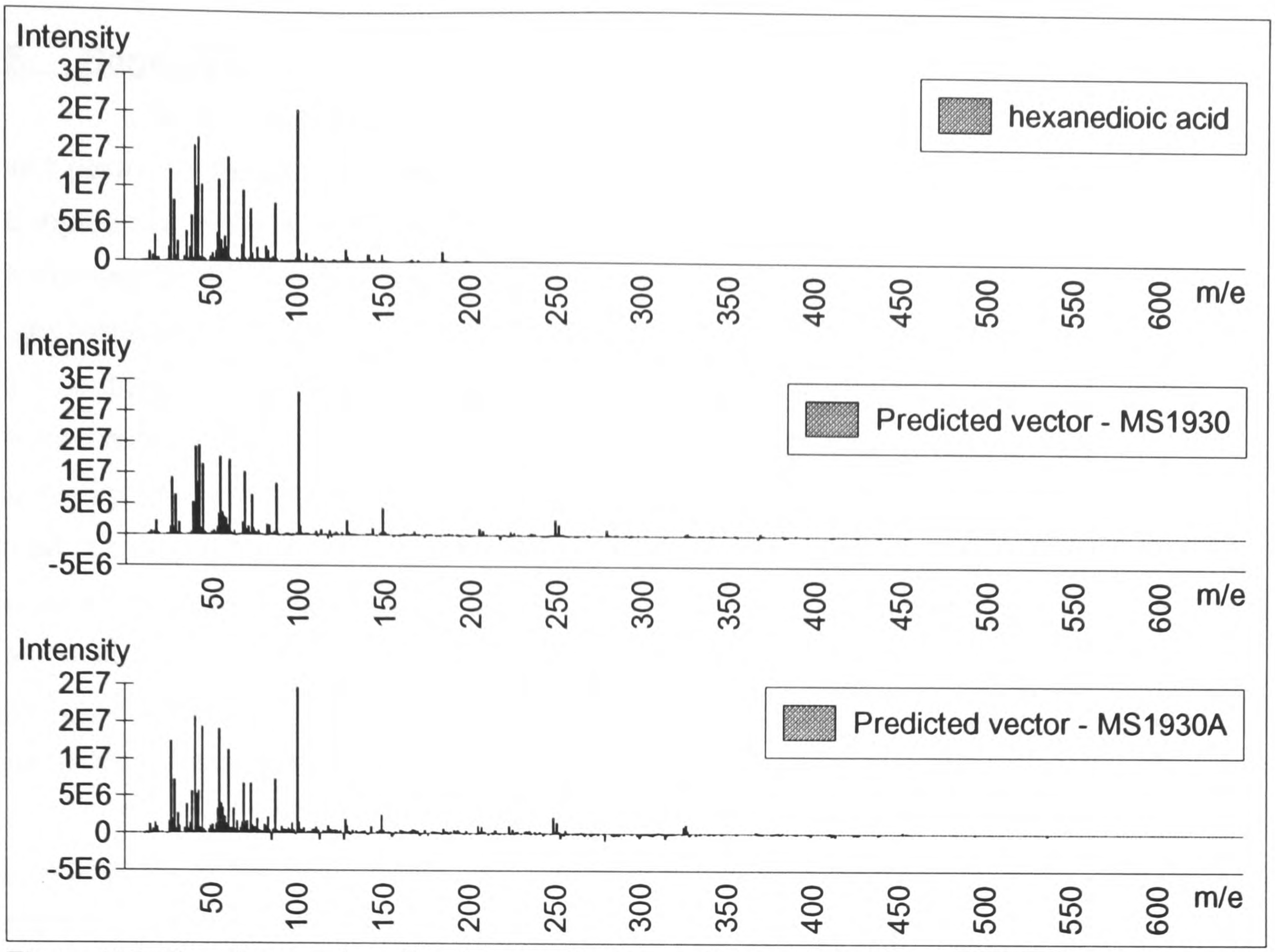


***Figure 117:** Plot of test and predicted vectors for hexanedioic acid in data sets MS1930 and MS1930A.*

In conclusion, the results from the data analysis are broadly in agreement with those found in the normal course of analysis for MS1930 except that it does not contain poly(ethylene adipate). Dioctyl phthalate was confirmed as being present in MS1930A but also hexanedioic acid is present and the component suggested to be Chemfos X, is not, but is a chemically similar material.

5. Conclusion

A computer program has been developed which will run on a standalone PC and performs factor analysis and target testing. The particular characteristics of temperature programmed pyrolysis - mass spectrometry (TPPy-MS) data have been accomodated within the program by the development of a method of reading the mass spectrometer data files and by developing a data structure which can store data sets of a size limited only by the available memory and disk space. Many modifications have been made to the program to allow the development of new techniques for investigating spectroscopic data. The program can perform principal component analysis and calculate a range of error terms for use in determining the number of components in the data. Target tests may be performed on the analysed data using any selected number of factors to model the data and several error terms calculated to assist in determination of the sucess of a test. New methods of iterative target testing have been developed and tested using the program.

A number of different methods have been tried to determine the number of components in the data. These are, percentage variance, eigenvalue plots, real error, imbedded error, indicator function, standard error in eigenvalue, misfit and percentage significance level. In all the data analysed as part of this work, successsful assignments of the number of factors responsible for the data has been possible using the evidence of several of the error terms. The physical and chemical origins of the factors giving rise to the data have been identified in single and multiple component systems and the technique shown to be a useful tool in the elucidation of fragmentation and pyrolysis mechanisms.

The interpretation of the origins of the components proved straightfoward for the UV-Vis data where extinction coefficient and concentration were the expected factors for the components, though the distortion which can be obtained from inadequate selection of sampling interval demonstrated that even simple systems can provide unexpected problems. The TPPy-MS data showed far more than the expected concentration and mass spectral information. The number of components found for even a single pure substance showed that for pyrolysis many different possibilities occur. The identification of different components for pyrolysis product and volatilization as well as different and competing routes for pyrolysis has shown this to be a powerful investigative technique for pyrolysis reactions. Other components identified in the data sets have been shown to describe source contamination, absorption of HCl onto 'cold' surfaces and artefacts introduced into the data from poor calibration and assumptions made by the data system.

The qualitative identification of components in the data using target testing was seen to be very effective for the UV-Vis data. The error terms calculated for the

UV-Vis target tests indicated clearly whether the component was present in the system or not. The normal error terms used in the quantification of fit of a test vector were found to be inadequate for use with mass spectrometer data. Alternative measures of fit were tested including correlation coefficient and Bessel's inequality function, these provided a much better indication of fit, particularly Bessel's inequality function. Most advantageously, the trained eye of the spectroscopist is easily able to discern the fit of the predicted vector and determine whether the test component is present or not.

Several methods of iterative target testing were developed and varying results were produced. The method using differential subtraction was seen to give the best fit of the data in the shortest number of iterations, and gives the advantage that nothing is assumed about the component, the data is fitted as well as possible to the test vector with no constraint placed upon the orientation of the vector so that the spectra produced are purely from the data. The technique of vector reinforcement restricts the orientation of the vector to that defined by the reinforced components, this allows ready confirmation of the presence of a component but injudicious selection of test values will result in a mixed spectrum from incorrect orientation of the vector thus complicating the interpretation of the results. Use of the two techniques has shown that it is possible to find components in the data without prior knowledge of the component or its spectrum and to then produce its spectrum in almost complete isolation.

Overall the program has shown itself to be a valuable aid in the interpretation of spectroscopic data and particularly beneficial in the identification of components present in TPPy-MS data.

6. Further work

In general, the program of work undertaken has fulfilled the aims defined at the outset of the project. With any piece of work there are always modifications which can be made and areas not investigated through lack of time or resources. The following outlines the most obvious areas for further investigation.

Since the inception of the program, widespread use has been made of the singular value decomposition (SVD) algorithm for the decomposition of the covariance matrix. This algorithm has advantages over the eigenanalysis used in this work in terms of accuracy and computational efficiency. The algorithm currently used suffers badly from needing extended numbers of iterations to find factors with similar eigenvalues. This adds a substantial overhead to the calculation times as well as adding error to the eigenvectors from the algorithm instability. The addition of the SVD algorithm would therefore aid substantially the use and accuracy of the program. The problems being experienced with interpretation of errors and mixing of factors may also be aided by the use of SVD as the algorithm is very accurate and able to extract very small eigenvalues succesfully.

One of the most difficult aspects of working with the mass spectrometric data has been found to be the nature of the error present in the data sets. The error introduced by the mass spectrometer is not absolute but is proportional to magnitude and may vary according to m/e ratio. Other workers in the field have used various data pre-treatment techniques in an attempt to overcome these difficulties but all suffer from distortion of the information in the data, either by changing its magnitude or its numerical origin. Both of these techniques have been acceptable in the case of related mixtures of similar intensities, but with the temperature programmed pyrolysis data the background is given an enormous boost of importance and all quantitative information is lost. A better understanding of the way error appears in the data is necessary for better interpretation of the data and also for the production of purer component spectra, which currently show mixing of other components due to the non-linearities introduced by the proportional error in the data.

The target testing has proved a valuable aid in the identification of components in the data. The interpretation of the test, for this work, has been largely a matter of observation of the resulting spectra. This approach has been necessary due to the currently developed criteria of fit and error being seen to be inadequate for this data and test vectors of this size. Further effort should be expended in finding improved measures suitable for determining the fit of predicted vectors that do not rely upon the un-quantitative eye of an experienced observer.

The adoption of SVD as the method of principal component analysis will also allow the use of row test vector as well as column test vectors. The use of row test vectors will enable particular peaks in the TIC pyrogram to be tested for and identified.

The inclusion of free floating of points in the test vector should be regarded as a very important extension of the target testing procedures. The work performed here has shown that though useful results can be obtained from the use of zeroed entries in test vectors, the constraints placed upon the vector by the null entries hamper the orientation of the vector in the optimum direction. By including free floated entries the alignment should be far less restricted. Some problems, which are inevitable in this work, arise from the necessity to include sufficient entries to span the data space thus precluding the perfect test that would otherwise result, and whether the inclusion of m/e values arising from more than one component will produce a mixed predicted vector. One interesting area of investigation would be in the use of m/e values that remain constantly zero throughout the analysis to make up the correct number of entries to span the data space. Consideration of the theory suggests that the inclusion of at least one such value should be acceptable thus reducing the burden upon the operator in selecting sufficient m/e values.

The use of iterative target testing (ITT) in this work has proved itself to be a powerful investigative tool. Further extensions to this work are suggested following the promising initial work with vector reinforcement, where non-zero values from the original test vector are rewritten over the values in the predicted vector, and also in the use of free floating of entries in the test vector. This may be approached from several directions, free floating of suspected major peaks to confirm their linkage or free floating of points below a threshold to remove their influence upon the convergence.

The inclusion of a target combination step would allow, in cases where sufficient components could be identified in the pyrogram, the generation of all the component spectra simultaneously as well as quantitative information from the pyrograms. This solution may be achieved using ITT of row test vectors to identify the position of component peaks in the TIC pyrogram followed by the target combination step using the predicted vectors to form the eigenvector matrix. This technique automatically produces a row matrix containing the spectra of the components.

The development of these procedures into iterative key set factor analysis may prove useful in the analysis of mass spectral data, though its computationally intensive nature may preclude its use on personal computers.

The computational aspects of the program place a heavy load upon the hardware and improvents could be realised by recoding the mathematics routines in assembly language. The performance gains from this would be noticeable but probably not worth the time involved, a better way forwards in this direction would be to re-code the application using a more portable language such as C, which would allow the program to be used on computers better suited to numerically intensive calculation, such as workstations. As a further possibility, recoding into FORTRAN would allow the use of transputer based add-in cards for the PC enabling parallel processing of the data to be performed.

The final recommendation for further work is in the area of improving the 'usability' of the program. The most obvious shortcoming in the program is the lack of graphics to visualize the data generated. If this area can be addressed and facilities for editing data incorporated into the program then the dependence upon spreadsheets for the interpretation of the data will be reduced and the biggest single restriction to the 'usability' of the program eliminated.

References

[1] Spearman, C. *Am. J. of Psychology* 1904, 15, 201-93.

[2] Harman, H.H. *Modern Factor Analysis,* 3rd ed.; The University of Chicago Press: Chicago, 1976, Chapter 1.

[3] Pearson, K. *Philos. Mag.*, Series 2 1901, 6, 559-72.

[4] Hotelling, H. *J. Ed. Psych.* 1933, 24, 417-41, 498-520.

[5] Malinowski, E.R. *Factor Analysis in Chemistry,* 2nd ed.; John Wiley: New York, 1991; Chapter 1.

[6] Rozett, R.W.; Petersen, M.E. *Anal. Chem.* 1975, 47, 1301-8.

[7] Stewart, G.W. *Introduction to matrix computations,* Academic Press: New York, 1973, Chapter 5.

[8] Cattell, R.B. *Multivariate Behav. Res.* 1966, 1, 245.

[9] Hugus, Z.Z.Jr.; El-Awady, A.A. *J.Phys.Chem.*, 1971, 75, 2954.

[10] Malinowski, E.R. *Anal. Chem.* 1977, 49, 606-12.

[11] Malinowski, E.R. *Anal. Chem.* 1977, 49, 612-17.

[12] Malinowski, E.R. *J.Chemometrics* 1988, 3, 49-60.

[13] Lorber, A. *Anal. Chem.* 1984, 56, 1004-10.

[14] Malinowski, E.R.; McCue, M. *Anal. Chem.* 1977, 49, 284-87.

[15] Westall, W.A.; Pidduck, A.J. *J. Anal. App. Pyrol.* 1987, 11, 3-14.

[16] Higman, B. *Applied Group-Theoretic and Matrix Methods*; Oxford University Press: Oxford, 1955.

[17] Brown, S.D. *Anal. Chem.* 1990, 62, 84R-101R.

[18] Malinowski, E.R. *Anal. Chim. Acta* 1980, 122, 327-30.

[19] Weiner, P.H.; Liao, H.L.; Karger, B.L. *Anal.Chem.* 1974, 46, 2182

[20] Roscoe, B.A.; Hopke, P.K. *Anal. Chim. Acta* 1981, 132, 89-97.

[21] Malinowski, E.R. *Anal. Chim. Acta* 1981, 133, 99-101.

[22] Eastment, H.T.; Krzanowski, W.J. *Tecnometrics* 1982, 24, 73-77.

[23] Wold, S. *Technometrics* 1978, 20, 397-405.

[24] Hopke, P.K.; Alpert, D.J.; Roscoe, B.A. *Computers and Chemistry* 1983, 7, 149-55.

[25] Hopke, P.K.; Dharmavaram, S. *Computers and Chemistry* 1986, 10, 163-4.

[26] Johansson, E.; Wold, S.; Sjödin, K. *Anal. Chem.* 1984, 56, 1685-88.

[27] Brayden, T.H.; Poropatic, P.A.; Watanabe, J.L. *Anal. Chem.* 1988, 60, 1154-8.

[28] Lorber, A.; Kowalski, B.R. *Anal. Chem.* 1989, 61, 1168-9.

[29] Little, R.J.A.; Rubin, D.B. *Statistical analysis with missing data*; Wiley: New York, 1987.

[30] Donahue, S.M.; Brown, C.W. *Anal. Chem.* 1991, 63, 980-5.

[31] Bulmer, J.T.; Shurvell, H.F. *J. Phys. Chem.* 1973, 77, 256-62.

[32] Lin, C.H.; Liu, S.H. *J. Chinese Chem. Soc.* 1978, 25, 167-77.

[33] Gemperline, P.J. *J. Chem. Inf. Comput. Sci.* 1984, 24, 206-12.

[34] Vandeginste, B.G.M.; Derks, W.; Kateman, G. *Anal. Chim. Acta* 1985, 173, 253-64.

References *continued*

[35]Maeder, M.; Zuberbüehler, A.D. *Anal. Chim. Acta* 1986, 181, 287-91.

[36]Gemperline, P.J. *Anal. Chem.* 1986, 58, 2656-63.

[37]Gampp, H.; Maeder, M.; Meyer, C.J.; Zuberbüehler, A.D. *Anal. Chim. Acta* 1987, 193, 287-93.

[38]Vandeginste, B.G.M.; Leyten, F.; Gerritsen, M.; Noor, J.W.; Kateman, G. *J. Chemometrics* 1987, 1, 57-71.

[39]Strasters, J.K.; Billiet, H.A.H.; De Galan, L.; Vandeginste, B.G.M.; Kateman, G. *J. Chromatogr.* 1987, 385, 181-200.

[40]Strasters, J.K; Billiet, H.A.H.; de Galan, L.; Vandeginste, B.G.M.; Kateman, G. *Anal. Chem.* 1988, 60, 2745-51.

[41]Schostack, K.J.; Malinowski, E.R. *Chemom. Intell. Lab. Syst.* 1991, 10, 303-24.

[42]Kankare, J.J. *Anal. Chem.* 1970, 42, 1322-6.

[43]McCue, M.; Malinowski, E.R., *Applied Spectroscopy*, 1983, 37, 463-9.

[44]Haldna,Ü.; Murshak, A.; *Computers and Chemistry*, 1984, 8, 201-4.

[45]Gemperline, P.J.; Boyette, S.E; Tyndall, K. *Applied Spectroscopy*, 1987, 41, 454-59.

[46]Malinowski, E.R. *ASTM Spectroscopic Technical Publication, 934 (Computerised Quantitative Infra-red Analysis)*, 1987, 155-68.

[47]Davis, J.E.; Shepard, A.; Stanford, N.; Rogers, L.B. *Anal. Chem.* 1974, 46, 821-5.

[48]Justice, J.B.; Isenhour, T.L. *Anal. Chem.* 1975, 47, 2286-8.

[49]Rozett, R.W.; McLaughlin Petersen, E. *Anal. Chem.* 1976, 48, 817-25.

[50]Ritter, G.L.; Lowry, S.R.; Isenhour, T.L.; Wilkins, C.L. *Anal. Chem.* 1976, 48, 591-5.

[51]Burgard, D.R.; Perone, S.P.; Wiebers, J.L. *Anal. Chem.* 1977, 49, 1444-6.

[52]Malinowski, E.R. *Anal. Chim. Acta* 1978, 103, 339-54.

[53]Rasmussen, G.T.; Hohne, B.A.; Wieboldt, R.C.; Isenhour, T.L. *Anal. Chim. Acta* 1979, 112, 151-64.

[54]Knorr, F.J.; Futrell, J.H. *Anal. Chem.* 1979, 51, 1236-41.

[55]Aries, R.E.; Gutteridge, C.S.; Macrae, R. *J. Chromatogr.* 1985, 319, 285-97.

[56]Windig, W.; Chakravarty, T.; Richards, J.M.; Meuzelaar, H.L.C. *Anal. Chim. Acta* 1986, 191, 205-18.

[57]Aries, R.E.; Gutteridge, C.S.; Ottley, T.W. *J. Anal. Appl. Pyrol.* 1986, 9, 81-98.

[58]Aries, R.E; Gutteridge, C.S; Evans, R. *J. Food Sci.* 1986, 51, 1183-6.

[59]Aries, R.E.; Gutteridge, C.S.; Laurie, W.A.; Boon, J.J.; Eijkel, G.B. *Anal. Chem.* 1988, 60, 1498-1502.

[60]Magee, J.T.; Hindmarch, J.M.; Bennett, K.W.; Duerden, B.I.; Aries, R.E. *J. Med. Microbiol.* 1989, 28, 227-36.

[61]Price, D.; Milnes, G.J.; Tayler, P.J.; Scrivens, J.H.; Blease, T.G. *Polymer Degradation and Stability* 1989, 25, 307-23.

[62]Lee, T.A.; Headley, L.M.; Hardy, J.K. *Anal. Chem.* 1991, 63, 357-60.

[63]Snyder, A.P.; Windig, W.; Toth, J.P. *Chemom. Intell. Lab. Syst.* 1991, 11, 149-60.

[64]Varmuza, K.; Davies, A.N. *Spectroscopy International* 1991, 3, 14-17.

References *continued*

[65]Howery, D.G. *ACS Symposium series , No.52 (Chemometrics: Theory and Applications Symposium)* 1976, 52, 73-9.

[66]Wold, S. *Tecnometrics* 1978, 20, 397-405.

[67]McReynolds, W.O. *J. Chromatogr. Sci.* 1970, 8, 685-91.

[68]Hirsch, R.F.; Gaydosh, R.J.; Chrétien, J.R. *Anal. Chem.* 1980, 52, 723-28.

[69]Benzécri, J.P., *l'Analyse des donnees*, Vol. 2; Dunod: Paris, 1973.

[70]Howery, D.G.; Williams, G.D.; Ayala, N. *Anal. Chim. Acta* 1986, 189, 339-51.

[71]Howery, D.G.; Soroka, J.M. *Anal. Chem.* 1986, 58, 3091-5.

[72]Zielinski, W.L.; Martire, D.E. *Anal.Chem.* 1976, 48, 1111-6.

[73]Howery, D.G.; Soroka, J.M. *J. Chemometrics* 1987, 1, 91-101.

[74]Howery, D.G.; Soroka, J.M. *J. Chromatogr. Sci.* 1987, 25, 149-53.

[75]Lochmüller, C.H.; Breiner, S.J.; Reese, C.E.; Koel, M.N. *Anal. Chem.* 1989, 61, 367-75.

[76]McCue, M.; Malinowski, E.R. *Anal. Chim. Acta* 1981, 133, 125-36.

[77]Roscoe, B.A.; Hopke, P.K. *Computers and Chemistry* 1981, 5, 1-7.

[78]Roscoe, B.A.; Chen, C.; Hopke, P.K. *Anal. Chim. Acta* 1984, 160, 121-34.

[79]Starks, T.H.; Fang, J.H.; Zevin, L.S. *J. Math. Geol.* 1984, 16, 351-67.

[80]Malinowski, E.R.; Howery, D.G. *Factor Analysis in Chemistry*; John Wiley: New York, 1980; Chapter 4.

[81]Howery, D.G.; Rubenstein, M. *Can. J. Chem.* 1987, 65, 1380-3.

[82]Kontron Instruments Ltd., Blackmoor lane, Croxley centre, Watford, Hertfordshire, WD1 8XQ.

[83]Merck Ltd., Broom Rd., Poole, Dorset, BH12 4NN.

[84]VG Analytical, Floats Road, Withenshawe, Manchester, M23 9L3.

[85]Grant, J.G. *Ferrocene-containing Smoke Suppressants and Flame Retardants for Flexible Polyvinyl Chloride*, PhD thesis, 1992.

[86]Report to DRA, QATS, Royal Arsenal East, *Mass spectrometer data sets used in the investigation of the applications of factor analysis to spectrometric methods*, August 1992.

[87]Malinowski, E.R.; Howery, D.G. *Factor Analysis in Chemistry*; John Wiley: New York, 1980; Chapter 5.

[88]Cartwright, H. *International Laboratory*, 1986, June, 18-27.

[89]Weiner, P.H.; Malinowski, E.R.; Levinstone, A.R. *J. Phys. Chem.* 1970, 74, 4537-42.

[90]Stenhagen, E.; Abrahamsson, S.; Mclafferty, F.W. *Atlas of Mass Spectral Data*; Wiley: New York, 1969.

[91]Dell Computer Corporation Ltd., Milbanke House, Western Road, Western Industrial Estate, Bracknell, Berkshire, RG12 1RW.

[92]Borland International (UK) Ltd., 8 Pavillions, Ruscombe, Twyford, Berkshire, RG10 9NN.

[93]TurboPower Software, P.O. Box 49009, Colorado Springs, CO 80949-9009.

[94]Kermit Distribution, Columbia University Centre for Computing Activities, 612 West 115th Street, New York, NY 10025, USA.

References *continued*

[95]Lotus Books *Lotus File Formats for 1-2-3, Symphony & Jazz. File Structure Descriptions for Developers*, Addison-Wesley Publishing Company, 1987.

[96]Wingz, Informix Software Inc., 16011 College Boulevard, Lenexa, Kansas 66219.

[97]Cooke, D.;Craven, A.H.;Clarke, G.M. *Statistical Computing in Pascal*; Edward Arnold: London, 1985; Chapter 8.

[98]Malinowski, E.R. *Factor Analysis in Chemistry*, 2nd ed.; John Wiley: New York, 1991; Chapter 5.

[99]Malinowski, E.R. *Factor Analysis in Chemistry*, 2nd ed.; John Wiley: New York, 1991; Chapter 4.

[100]Galloway, F.M.; Hirschler, M.M.; Smith, G.F. *Fire and Materials* 1991, 15, 181-89.

[101]Sheley, C.F.; Fishel, D.L. *Org. Mass Spectrom.* 1972, 6, 1131-37

[102]Vandeginste, G.M.; Derks, W.; Kateman, G. *Anal. Chim. Acta* 1985, 173, 253-64

[103]The Mass Spectrometry Data Centre *Eight Peak Index of Mass Spectra*, 3rd ed.; The Royal Society of Chemistry: Nottingham; 1983.

[104]Rasmussen, G.T.; Hohne, B.A.; Wieboldt, R.C.; Isenhour, T.L. *Anal. Chim. Acta.* 1979, 112, 151-64.

APPENDICES

Appendices

Table of contents of Appendices

Appendix 1 : Derivation of factor analysis

The following derivation is taken largely from Malinowski's excellent book, 'Factor analysis in Chemistry', second edition, 1991.

Principal component analysis is a mathematical technique that attempts to express a data point, d_{ik}, as a sum of product functions. Such a point can be expressed as a linear sum of n product terms called factors. Each factor is composed of a product of cofactors as follows,

$$d_{ik} = \sum_{j=1}^{n} r_{ij} c_{jk} \tag{1}$$

where r_{ij} is the j^{th} row cofactor and c_{jk} is the j^{th} column cofactor.

When a complete matrix of data points is considered the equation may be rewritten in matrix notation to give,

$$\mathbf{D} = \mathbf{RC} \tag{2}$$

where **D**, **R** and **C** are matrices with elements d_{ik}, r_{ij} and c_{jk} respectively.

The problem to be solved by principal component analysis is, starting with the data matrix, **D**, obtain the row and column cofactor matrices, **R** and **C**. This problem can be solved directly using algorithms like NIPALS but the approach taken in this work is via the covariance matrix.

The covariance is defined as the product of the data matrix pre-multiplied by its transpose.

$$\mathbf{Z} = \mathbf{D}'\mathbf{D} \tag{3}$$

where **D′** is the transpose of the data matrix.

The rank of the covariance matrix is determined by diagonalization. The rank identifies the dimensionality of the factor space and in the case of pure data exactly n factors would emerge. The effect of experimental error on the data produces a number of eigenvectors equal to the number of rows, r, or the number of columns, c, whichever is the smaller. Throughout this derivation it is assumed, for convenience, that the number of columns is less than the number of rows.

Let $\mathbf{f}_1, \mathbf{f}_2, \ldots, \mathbf{f}_c$ be the basis set of unit vectors that defines the factor space. i.e.,

$$\mathbf{f}_1 = \begin{bmatrix} 1 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}, \quad \mathbf{f}_2 = \begin{bmatrix} 0 \\ 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}, \quad \mathbf{f}_3 = \begin{bmatrix} 0 \\ 0 \\ 1 \\ \vdots \\ 0 \end{bmatrix}, \quad \text{etc.} \tag{4}$$

These factor axes are orthonormal (orthogonal and of unit length) so that,

$$\mathbf{f}'_j\mathbf{f}_k = \delta_{jk} \tag{5}$$

where δ_{jk} is the Kronecker delta which has the properties,

$$\delta_{jk} = \begin{cases} 0 & \text{if } j \neq k \\ 1 & \text{if } j = k \end{cases} \tag{6}$$

A data point can be viewed as a vector in factor space. Hence, instead of equation 1, a better representation is,

$$\mathbf{d}_{ik} = \sum_{j=1}^{c} r_{ij}\mathbf{f}_j c_{jk} \tag{7}$$

where $\mathbf{d}_{ik}$ is the data point vector.

The z_{ab} element of the covariance matrix is given by,

$$\begin{aligned} z_{ab} &= \sum_{i=1}^{r} \mathbf{d}'_{ia}\mathbf{d}_{ib} = \sum_{i=1}^{r}\left(\sum_{j=1}^{c} r_{ij}\mathbf{f}'_j c_{ja}\right)\left(\sum_{k=1}^{c} r_{ik}\mathbf{f}_k c_{kb}\right) \\ &= \sum_{i=1}^{r}\sum_{j=1}^{c} r_{ij}^2 c_{ja} c_{jb} \end{aligned} \tag{8}$$

the last step is true because of equation 5.

Equation 8 shows that the whole covariance matrix can be decomposed into a sum of product terms containing a dyad and the corresponding eigenvalue as follows,

$$\begin{aligned} \mathbf{Z} = &\sum_{i=1}^{n} r_{i1}^2 \begin{bmatrix} c_{11} \\ c_{12} \\ \vdots \\ c_{1c} \end{bmatrix} \begin{bmatrix} c_{11} & c_{12} & \cdots & c_{1c} \end{bmatrix} \\ &+ \sum_{i=1}^{n} r_{i2}^2 \begin{bmatrix} c_{21} \\ c_{22} \\ \vdots \\ c_{2c} \end{bmatrix} \begin{bmatrix} c_{21} & c_{22} & \cdots & c_{2c} \end{bmatrix} \\ &+ \cdots + \sum_{i=1}^{n} r_{ic}^2 \begin{bmatrix} c_{c1} \\ c_{c2} \\ \vdots \\ c_{cc} \end{bmatrix} \begin{bmatrix} c_{c1} & c_{c2} & \cdots & c_{cc} \end{bmatrix} \end{aligned} \tag{9}$$

When rewritten in terms of column cofactor vectors a simpler expression results,

$$\mathbf{Z} = \lambda_1 \mathbf{c}_1 \mathbf{c}_1' + \lambda_2 \mathbf{c}_2 \mathbf{c}_2' + \cdots + \lambda_c \mathbf{c}_c \mathbf{c}_c'$$

where

$$\lambda_{\mathrm{j}} = \sum_{i=1}^{c} r_{ij}^2 = \left\| r_j \right\|^2$$

and

$$\mathbf{c}_{\mathrm{j}}' = \begin{bmatrix} c_{j1} & c_{j2} & \cdots & c_{jc} \end{bmatrix} \tag{10}$$

λ_j is the eigenvalue associated with eigenvector (column cofactor) $\mathbf{c}_j$ and the term $\mathbf{c}_j \mathbf{c}_j'$ is referred to as a dyad.

We now have an expression relating the covariance matrix to the eigenvalues and eigenvectors (factor axes) of the data. Mathematically it is possible to define an infinite number of different sets of factor axes which will adequately describe the data. In principal component analysis two conditions are imposed upon the calculation of the factor axes. They are, that as much variance as possible is accounted for by each factor, and that each axis is orthogonal to the rest of the factor axes. The factor axes are calculated consecutively accounting for the maximum variance with each calculation. The contribution of the newly calculated axis to the data is then removed and the next axis calculated thus ensuring orthogonality.

Each factor axis describes an ever decreasing proportion of the variance in the data until the complete set of c eigenvectors have been found. The data can be reproduced using equation 1, where n is equal to c the values of the data points are reproduced perfectly. If a value of n less than c is used then the value of the data point is reproduced less accurately as some of its variance has been neglected. To keep track of the number of factors being considered the notation of equation 1 is modified as follows,

$$d_{ik}(m) = \sum_{j=1}^{m} r_{ij} c_{jk} \tag{11}$$

where d_{ik} is the reproduced data point in the i^{th} row and the k^{th} column calculated from the first m factors.

To find the first principal component (factor axis) we proceed in the following manner. The residual error, $e_{ik}(1)$, is defined to be the difference between the experimental data point, d_{ik}, and the reproduced data point, $d_{ik}(1)$, based upon one factor,

$$e_{ik}(1) = d_{ik} - d_{ik}(1) \tag{12}$$

inserting this into equation 11 gives,

$$e_{ik}(1) = d_{ik} - r_{i1} c_{1k} \tag{13}$$

In order to maximize the variance accounted for by the factor the residual error must be minimized. This is done by the application of the method of least squares. The derivative of the sum of squares of each residual error with respect to the column factors is taken, giving,

$$\sum_{i=1}^{r}\frac{de_{ik}^{2}(1)}{dc_{1k}} = 2c_{1k}\sum_{i=1}^{r} r_{i1}^{2} - 2\sum_{i=1}^{r} r_{i1}d_{ik} \tag{14}$$

According to the least squares principle the sum is set to zero, so,

$$\sum_{i=1}^{r} r_{i1}d_{ik} = c_{1k}\sum_{i=1}^{r} r_{i1}^{2} \tag{15}$$

as k varies from 1 to c there are c equations of this form which may be expressed in matrix form as,

$$\mathbf{r}_1'\mathbf{D} = \mathbf{c}_1'\mathbf{r}_1'\mathbf{r}_1 \tag{16}$$

As defined in equation 10 we can see that,

$$\lambda_1 = \mathbf{r}_1'\mathbf{r}_1 = \sum_{i=1}^{r} r_{i1}^{2} \tag{17}$$

Inserting this into equation 16 and transposing gives,

$$\mathbf{D}'\mathbf{r}_1 = \lambda_1\mathbf{c}_1 \tag{18}$$

It can be shown that the data matrix can be written as

$$\mathbf{D} = \mathbf{r}_1\mathbf{c}_1' + \mathbf{r}_2\mathbf{c}_2' + \cdots + \mathbf{r}_c\mathbf{c}_c' \tag{19}$$

Post-multiplying this by $\mathbf{c}_1$ and setting $\mathbf{c}_j'\mathbf{c}_j = \delta_{ij}$, so that the eigenvectors are orthonormal we get,

$$\mathbf{D}\mathbf{c}_1 = \mathbf{r}_1 \tag{20}$$

Inserting this into equation 18 gives,

$$\mathbf{D}'\mathbf{D}\mathbf{c}_1 = \lambda_1\mathbf{c}_1 \tag{21}$$

From the definition of the covariance matrix in equation 3 it can be seen that,

$$\mathbf{Z}\mathbf{c}_1 = \lambda_1\mathbf{c}_1 \tag{22}$$

thus giving an expression which can be used to calculate the first principal eigenvector and its associated eigenvalue.

The second principal component is found by considering the second residual error,

$$e_{ik}(2) = d_{ik} - d_{ik}(2) \tag{23}$$

which, from the definition given in equation 11 can be expressed as,

$$e_{ik}(2) = e_{ik}(1) - r_{i2}c_{2k} \tag{24}$$

The error in the second principal component is minimized by applying the method of least squares to $e_{ik}(2)$ while keeping $e_{ik}(1)$ constant. The resulting expression is analogous to equation 14 as follows,

$$\sum_{i=1}^{r} \frac{de_{ik}^2(2)}{dc_{2k}} = 2c_{2k}\sum_{i=1}^{r} r_{i2}^2 - 2\sum_{i=1}^{r} r_{i2}e_{1k}(1) \tag{25}$$

To minimize the error, the summation is set to zero, giving,

$$\sum_{i=1}^{r} r_{i2}e_{ik}(1) = c_{2k}\sum_{i=1}^{r} r_{i2}^2 \tag{26}$$

There are c equations of this type, which in matrix notation have the form,

$$\mathbf{r}_2'\mathbf{E}_1 = \mathbf{c}_2'\mathbf{r}_2'\mathbf{r}_2 \tag{27}$$

where $\mathbf{E}_1$ is an $r \times c$ error matrix composed of the first residual errors and λ_2 is defined as,

$$\lambda_2 = \mathbf{r}_2'\mathbf{r}_2 = \sum_{i=1}^{r} r_{i2}^2 \tag{28}$$

Combining equations 27 and 28 gives

$$\mathbf{E}_1'\mathbf{r}_2 = \lambda_2\mathbf{c}_2 \tag{29}$$

Matrix $\mathbf{E}_1$ can be written as,

$$\mathbf{E}_1 = \mathbf{D} - \mathbf{r}_1\mathbf{c}_1' = \mathbf{r}_2\mathbf{c}_2' + \mathbf{r}_3\mathbf{c}_3' + \cdots + \mathbf{r}_c\mathbf{c}_c' \tag{30}$$

post-multiplying this by $\mathbf{c}_2$ and maintaining the constraint of orthogonality gives,

$$\mathbf{E}_1\mathbf{c}_2 = \mathbf{r}_2 \tag{31}$$

inserting this into equation 29 gives,

$$\mathbf{E}_1'\mathbf{E}_1\mathbf{c}_2 = \lambda_2\mathbf{c}_2 \tag{32}$$

Using equations 16, 17 and 30 it can be shown that,

$$\mathbf{E}_1'\mathbf{E}_1 = \mathbf{D}'\mathbf{D} - \lambda_1\mathbf{c}_1\mathbf{c}_1' \tag{33}$$

and thus the first residual matrix is defined as,

$$\mathfrak{R}_1 = \mathbf{Z} - \lambda_1\mathbf{c}_1\mathbf{c}_1' \tag{34}$$

and it can be concluded from equations 32, 33 and 34 that,

$$\mathfrak{R}_1\mathbf{c}_2 = \lambda_2\mathbf{c}_2 \tag{35}$$

this expression, similar to equation 22, can be used to calculate the numerical values of the second principal eigenvector, $\mathbf{c}_2$, and its associated eigenvalue, λ_2.

To obtain further principal components the same principles may be applied to yield the following,

$$\Re_2 \mathbf{c}_3 = \lambda_3 \mathbf{c}_3 \qquad (36)$$

where,

$$\Re_2 = \mathbf{Z} - \lambda_1 \mathbf{c}_1 \mathbf{c}_1' - \lambda_2 \mathbf{c}_2 \mathbf{c}_2' \qquad (37)$$

so it can be seen that the procedure can be continued in this fashion until all the eigenvectors have been extracted.

The general equations for the principal component analysis are as follows,

$$\Re_m \mathbf{c}_{m+1} = \lambda_{m+1} \mathbf{c}_{m+1} \qquad (38)$$

where,

$$\Re_m = \mathbf{Z} - \sum_{j=1}^{m} \lambda_j \mathbf{c}_j \mathbf{c}_j' \qquad (39)$$

Once the complete set of principal components has been calculated the row cofactors can be calculated from,

$$\mathbf{R} = \mathbf{DC}' \qquad (40)$$

Appendix 2 : Derivation of target testing

The following derivation is taken mostly from Malinowski's book 'Factor analysis in Chemistry', second edition, 1991.

Transformations of the reference axes found by principal component analysis are accomplished by the following operation,

$$\hat{\mathbf{X}} = \overline{\mathbf{R}}\mathbf{T} \tag{41}$$

where $\mathbf{T}$ is the transformation matrix of dimensions $n \times n$, $\overline{\mathbf{R}}$ is the row cofactor matrix containing the first n significant factors and $\hat{\mathbf{X}}$ is the transformed row matrix in the new co-ordinate system.

The inverse of the transformation matrix is used to calculate the transformed column factor matrix, $\hat{\mathbf{Y}}$, as follows,

$$\hat{\mathbf{Y}} = \mathbf{T}^{-1}\overline{\mathbf{C}} \tag{42}$$

So that the data matrix based on n factors, $\overline{\mathbf{D}}$, may be reproduced by the following,

$$\overline{\mathbf{D}} = \hat{\mathbf{X}}\hat{\mathbf{Y}} \tag{43}$$

Many different forms of transformation can be applied to the principal components via this method, including the subject of this derivation, target testing.

Target testing individual factors is possible because of the mathematical operation defined in equation 41. This equation allows an individual column of the transformed row matrix to be calculated. Thus the l^{th} column of the newly transformed row matrix, $\hat{\mathbf{x}}_l$, is obtained by multiplying the l^{th} column of the transformation matrix, $\mathbf{t}_l$, by the row matrix $\overline{\mathbf{R}}$ as follows,

$$\hat{\mathbf{x}}_l = \overline{\mathbf{R}}\mathbf{t}_l \tag{44}$$

The individual columns are referred to as the predicted vector, $\hat{\mathbf{x}}_l$, and the transformation vector, $\mathbf{t}_l$. The transformation vector is calculated via a least squares procedure from the test vector, $\mathbf{x}_l$. The least squares procedure seeks to minimize the deviation between the test vector (called the 'target') and the predicted vector and is achieved via the following route.

The transformation vector, $\mathbf{t}_l$, has components $t_{1l}, t_{2l}, \ldots, t_{nl}$.

Each row of $\overline{\mathbf{R}}$ can be viewed as a row vector, thus the i^{th} row of $\overline{\mathbf{R}}$ is a vector $\mathbf{r}_{i.}$ having components $r_{i1}, r_{i2}, \ldots, r_{in}$. Note that $\mathbf{r}_{i.}$ is a row vector and $\mathbf{r}_{.i}$ is a column vector and care should be taken not to confuse the two.

When $\mathbf{r}_{i.}$ is multiplied by $\mathbf{t}_l$, the projection of the i^{th} row entity on the new transformed co-ordinate axis is obtained.

$$\hat{x}_{il} = \mathbf{r}'_{i.}\mathbf{t}_l = r_{i1}t_{1l} + r_{i2}t_{2l} + \cdots + r_{in}t_{nl} \tag{45}$$

Multiplying each row vector of the row matrix by $\mathbf{t}_l$ gives $\hat{x}_{1l}, \hat{x}_{2l}, \ldots, \hat{x}_{rl}$, which are the elements of $\hat{\mathbf{x}}_l$, the predicted vector. By comparing the elements of the predicted vector to the corresponding elements of the test vector a term describing the difference between the vector may be written.

$$\Delta x_{il} = \hat{x}_{il} - x_{il} = r_{i1}t_{1l} + r_{i2}t_{2l} + \ldots + r_{in}t_{nl} - x_{il} \tag{46}$$

where Δx_{il} is the difference between the value of $\hat{x}_{il}$ and x_{il}.

To determine the best $\mathbf{t}_l$, the deviation between the test and predicted vectors is minimized by setting the sum of the derivatives of the squares of the differences equal to zero. First the square of Δx_{il} is found,

$$\begin{aligned}(\Delta x_{il})^2 = {} & r_{i1}^2 t_{1l}^2 + r_{i2}^2 t_{2l}^2 + r_{in}^2 t_{nl}^2 + x_{il}^2 \\ & + 2r_{i1}t_{1l}r_{i2}t_{2l} + 2r_{i1}t_{1l}r_{in}t_{nl} + 2r_{i2}t_{2l}r_{in}t_{nl} \\ & - 2r_{i1}t_{1l}x_{il} - 2r_{i2}t_{2l}x_{il} - 2r_{in}t_{nl}x_{il}\end{aligned} \tag{47}$$

Then the derivative with respect to t_{1l} is determined,

$$\frac{d(\Delta x_{il})^2}{dt_{1l}} = 2r_{i1}^2 t_{1l} + 2r_{i1}r_{i2}t_{2l} + \cdots + 2r_{i1}r_{in}t_{nl} - 2r_{i1}x_{il} \tag{48}$$

Similar expressions may be obtained for each row designee, $\mathbf{r}_{i.}$. By summing over all the row designees and applying the least-squares criteria we obtain,

$$\sum_{i=1}^{r} \frac{d(\Delta x_{il})^2}{dt_{1l}} = 0 = t_{1l}\sum_i r_{i1}^2 + t_{2l}\sum_i r_{i1}r_{i2} + \cdots + t_{nl}\sum_i r_{i1}r_{in} - \sum_i r_{i1}x_{il} \tag{49}$$

This procedure is then repeated with respect to the remaining components of $\mathbf{t}_l$ to obtain expressions for $t_{2l}, t_{3l}, \ldots, t_{nl}$ giving the following set of simultaneous equations,

$$\begin{array}{lllll} \sum r_{i1}x_{il} = t_{1l}\sum r_{i1}^2 & + t_{2l}\sum r_{i1}r_{i2} & + \cdots & + t_{nl}\sum r_{i1}r_{in} \\ \sum r_{i2}x_{il} = t_{1l}\sum r_{i1}r_{i2} & + t_{2l}\sum r_{i2}^2 & + \cdots & + t_{nl}\sum r_{i2}r_{in} \\ \quad \vdots \qquad\quad \vdots & \quad \vdots & & \quad \vdots \\ \sum r_{in}x_{il} = t_{1l}\sum r_{i1}r_{in} & + t_{2l}\sum r_{i2}r_{in} & + \cdots & + t_{nl}\sum r_{in}^2 \end{array} \tag{50}$$

This equation can be written in matrix notation as follows,

$$\mathbf{a}_l = \mathbf{B}\mathbf{t}_l \tag{51}$$

where,

$$\mathbf{a}_l \equiv \begin{bmatrix} \sum r_{i1}x_{il} \\ \sum r_{i2}x_{il} \\ \vdots \\ \sum r_{in}x_{il} \end{bmatrix} \tag{52}$$

$$\mathbf{t}_l \equiv \begin{bmatrix} t_{1l} \\ t_{2l} \\ \vdots \\ t_{nl} \end{bmatrix} \tag{53}$$

$$\mathbf{B} \equiv \begin{bmatrix} \sum r_{i1}^2 & \sum r_{i1} r_{i2} & \cdots & \sum r_{i1} r_{in} \\ \sum r_{i1} r_{i2} & \sum r_{i2}^2 & \cdots & \sum r_{i2} r_{in} \\ \vdots & \vdots & & \vdots \\ \sum r_{i1} r_{in} & \sum r_{i2} r_{in} & \cdots & \sum r_{in}^2 \end{bmatrix} \tag{54}$$

Multiplying both sides of equation 51 by $\mathbf{B}^{-1}$ gives,

$$\mathbf{t}_l = \mathbf{B}^{-1}\mathbf{a}_l \tag{55}$$

Examination of equation 54 reveals that,

$$\mathbf{B} = \overline{\mathbf{R}}'\overline{\mathbf{R}} \tag{56}$$

and examination of equation 52 reveals that,

$$\mathbf{a}_l = \overline{\mathbf{R}}'\mathbf{x}_l \tag{57}$$

Combining equations 55, 56 and 57 we get,

$$\mathbf{t}_l = \left(\overline{\mathbf{R}}'\overline{\mathbf{R}}\right)^{-1}\overline{\mathbf{R}}'\mathbf{x}_l \tag{58}$$

It can be shown that equation 56 is equal to a diagonal matrix composed of the *n* primary eigenvalues (see equation 17 in appendix 1) and so equation 58 may be rewritten,

$$\mathbf{t}_l = \overline{\Lambda}^{-1}\overline{\mathbf{R}}'\mathbf{x}_l \tag{59}$$

where $\overline{\Lambda}^{-1}$ is the matrix containing the reciprocals of the eigenvalues along its major diagonal.

Equation 59 is the central equation of target testing, with it, the transformation vector is calculated and the predicted vector found using equation 44.

Appendix 3 : Program listing

The units comprising the target factor analysis program are arranged in the following hierarchical structure.

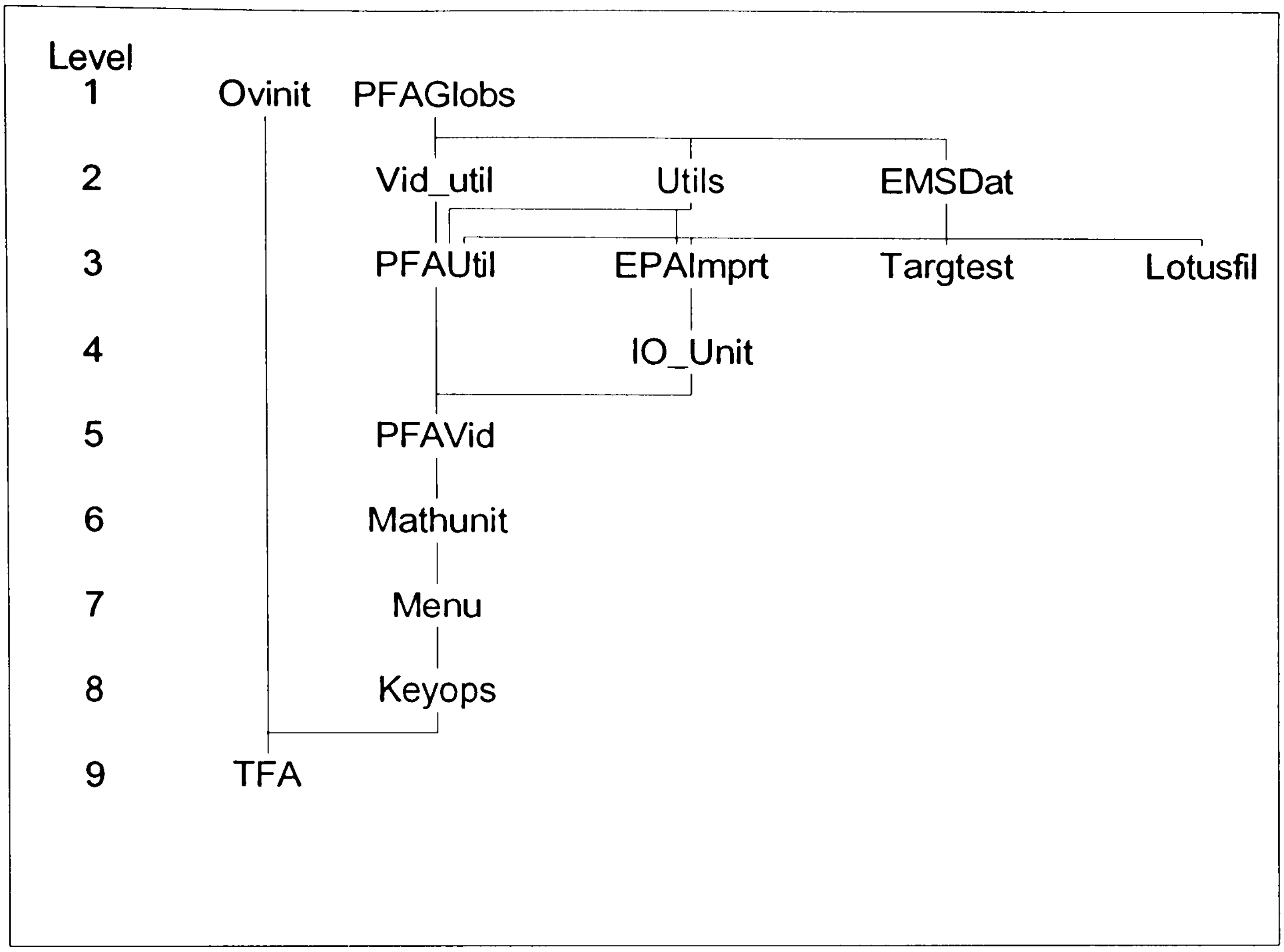


Figure 1: The unit hierarchy for program TFA.EXE.

The following two pages contain the procedural scope and dependency tables for the entire program, contained within the table are the unit scopes as determined by their uses clauses.

The source code listings following are arranged from the lowest level to the highest within the program and from left to right in the same level.

The first unit in the listings is DATSTRU2 which is the original sparse array data structure which was superseded by EMSDAT.

Procedural scope and dependancy

Top procedures

Unit	Utils													Vid util								EMSdat									TargTest						LotusFil		EPAimprt	PFAutils							PFAvid													
Dependent procedures	Convert_to_base_26	Convert_to_base_10	Get_date_time	Strip_trailing	Strip_leading	Strip	Power	Round_value	String_to_number	Read_kbd	Parse_coords	Check _for_esc	Is_mono	Set_border	Hide_cursor	Restore_cursor	Kbd_light_status	Colour_box	Fill_screen	Get_attribute	Initialise_matrix	Set_val	Get_val	Matrix_dimensions	Type_of_array	Delete_matrix	Save_file	Read_file	Free_EMS	Matrix_init	Calculate_TKON	Calculate_TVEC	Calculate_Pred_vector	Errors_in_test_vector	Test_vector	Lotus_import	Lotus_export	EPA_import	Value_to_string	Write_status	Change_current_matrix	Update_time	Update_lights	Write_location	Draw_cursor	Draw_screen	Draw_headings	Write_number_window	Clr_menu	String_input	Get_load_filename	Get_save_filename	Change_disp_matrix	Display_error_message	Write_text_window	Close_text_window	Write_msg_in_window	Write_factor_number	Update_iteration_number
Convert_to_base_26	×																																																										
Convert_to_base_10		×																																																									
Get_date_time			×																																																								
Strip_trailing				×																																																							
Strip_leading					×																																																						
Strip				×	×	×																																																					
Power							×																																																				
Round_value							×	×																																																			
String_to_number									×																																																		
Read_kbd										×																																																	
Parse_coords		×		×					×		×																																																
Check _for_esc										×		×																																															
Is_mono													×																																														
Set_border														×																																													
Hide_cursor															×																																												
Restore_cursor																×																																											
Kbd_light_status																	×																																										
Colour_box																		×																																									
Fill_screen																			×																																								
Get_attribute																				×																																							
Initialise_matrix																					×																																						
Set_val																						×																																					
Get_val																							×																																				
Matrix_dimensions																								×																																			
Type_of_array																									×																																		
Delete_matrix																										×																																	
Save_file																											×																																
Read_file																												×																															
Free_EMS																													×																														
Matrix_init																					×			×		×				×																													
Calculate_TKON																					×	×	×	×		×				×	×																												
Calculate_TVEC																						×	×									×																											
Calculate_Pred_vector																						×	×										×																										
Errors_in_test_vector																						×	×											×																									
Test_vector																																×	×	×	×																								
Lotus_import																					×	×														×																							
Lotus_export																							×	×													×																						
EPA_import									×												×	×																×																					
Value_to_string				×	×		×																×																×																				
Write_status																																								×																			
Change_current_matrix																								×																	×																		
Update_time			×																	×																						×																	
Update_lights																	×																										×																
Write_location	×					×														×			×																					×															
Draw_cursor																		×																										×	×														
Draw_screen																		×	×																					×		×		×		×													
Draw_headings	×					×												×	×	×																											×												
Write_number_window																				×																			×						×		×	×											
Clr_menu																																																	×										
String_input										×					×	×																								×		×	×							×									
Get_load_filename																				×																				×									×	×	×								
Get_save_filename																				×																				×									×	×		×							
Change_disp_matrix										×								×	×	×																				×	×	×	×										×						
Display_error_message																																								×														×					
Write_text_window																		×	×	×																																			×				
Close_text_window																																																								×			
Write_msg_in_window																																																									×		
Write_factor_number																																																										×	
Update_iteration_number																																																											×
Finished_factor																																																											
Write_data_file																											×																																
Read_data_file																												×																															
EPA_import																																						×																					
Import_file																										×										×				×	×								×		×		×	×					
Export_file																																					×			×	×								×			×	×	×					
Covariance_matrix																						×	×																																				
Set_Z_to_residual																						×	×																																				
Initialize_factor_matrix																						×																																					
Calc_intermediate_matrix																						×	×																																				
Calc_normalisation_const																							×																																				
Calc_eigenvector																						×	×																																				
Test_for_completed_factor																						×	×																																				
Rewrite_factor_matrix																						×	×																																				
Calc_residual_matrix																						×	×																																				
Residual_matrix_zero_test																							×																																				
Calc_abstract_row_matrix																						×	×																																				
Real_and_other_errors																						×	×																																				
Mean_and_sd_of_data_matrix																							×																																				
Misfits																						×	×																																				
Standard_error_in_eigenvalue																						×	×																																				
Errors_in_decomposition																																																									×		
Clear_matrices																										×																																	
Matrix_init																					×				×																																×		
Calc_variance																						×																																					
Pca												×												×																															×	×	×	×	×
Select_from_menu										×								×																								×	×																
Write_menu																																																											
Write_message																																																											
Decompose																																								×																			
Error_entry									×											×																				×									×	×									
Quit																																																	×	×									
Set_column_width									×											×																				×									×	×									
Change_column_headings																																																											
Get_abserr																																																											
Get_vecterr																																																											
Get_nofacs																																																											
Menu_system																															×				×														×				×						
Read_menu_file																																																											
Move_cursor_up																																													×														
Move_cursor_down																																													×														
Move_cursor_left																																													×														
Move_cursor_right																																													×														
Move_page_up																																																											
Move_page_down																																																											
Move_page_right																																																											
Move_page_left																																																											
Move_to_home																																																											
Move_to_end																																																											
Move_to_top																																																											
Move_to_bottom																																																											
Move_to_far_left																																																											
Move_to_far_right																																																											
Edit_cell									×											×			×																	×									×	×									
Goto_cell											×									×																				×									×	×									
Call_change_matrix																																																											
Handle_key																																																					×						

Procedural scope and dependancy

Unit	Dependent procedures	Top procedures																																																																																																														
		Convert_to_base_26	Convert_to_base_10	Get_date_time	Strip_trailing	Strip_leading	Strip	Power	Round_value	String_to_number	Read_kbd	Parse_coords	Check _for_esc	Is_mono	Set_border	Hide_cursor	Restore_cursor	Kbd_light_status	Colour_box	Fill_screen	Get_attribute	Initialise_matrix	Set_val	Get_val	Matrix_dimensions	Type_of_array	Delete_matrix	Save_file	Read_file	Free_EMS	Matrix_init	Calculate_TKON	Calculate_TVEC	Calculate_Pred_vector	Errors_in_test_vector	Test_vector	Lotus_import	Lotus_export	EPA_import	Value_to_string	Write_status	Change_current_matrix	Update_time	Update_lights	Write_location	Draw_cursor	Draw_screen	Draw_headings	Write_number_window	Clr_menu	String_input	Get_load_filename	Get_save_filename	Change_disp_matrix	Display_error_message	Write_text_window	Close_text_window	Write_msg_in_window	Write_factor_number	Update_iteration_number	Finished_factor	Write_data_file	Read_data_file	EPA_import	Import_file	Export_file	Covariance_matrix	Set_Z_to_residual	Initialize_factor_matrix	Calc_intermediate_matrix	Calc_normalisation_const	Calc_eigenvector	Test_for_completed_factor	Rewrite_factor_matrix	Calc_residual_matrix	Residual_matrix_zero_test	Calc_abstract_row_matrix	Real_and_other_errors	Mean_and_sd_of_data_matrix	Misfits	Standard_error_in_eigenvalue	Errors_in_decomposition	Clear_matrices	Matrix_init	Calc_variance	Pca	Select_from_menu	Write_menu	Write_message	Decompose	Error_entry	Quit	Set_column_width	Change_column_headings	Get_abserr	Get_vecterr	Get_nofacs	Menu_system	Read_menu_file	Move_cursor_up	Move_cursor_down	Move_cursor_left	Move_cursor_right	Move_page_up	Move_page_down	Move_page_right	Move_page_left	Move_to_home	Move_to_end	Move_to_top	Move_to_bottom	Move_to_far_left	Move_to_far_right	Edit_cell	Goto_cell	Call_change_matrix	Handle_key
	Finished_factor																																																										X																								X																																
IO_unit	Write_data_file																																																											X				X																			X																															
	Read_data_file																																																												X		X																																																									
	EPA_import																																																													X	X																																																									
	Import_file																																																														X																																	X																			
	Export_file																																																															X																																X																			
Mathunit	Covariance_matrix																																																																X																			X																															
	Set_Z_to_residual																																																																	X																		X																															
	Initialize_factor_matrix																																																																		X																	X																															
	Calc_intermediate_matrix																																																																			X																X																															
	Calc_normalisation_const																																																																				X															X																															
	Calc_eigenvector																																																																					X														X																															
	Test_for_completed_factor																																																																						X													X																															
	Rewrite_factor_matrix																																																																							X												X																															
	Calc_residual_matrix																																																																								X											X																															
	Residual_matrix_zero_test																																																																									X										X																															
	Calc_abstract_row_matrix																																																																										X									X																															
	Real_and_other_errors																																																																											X				X																																							
	Mean_and_sd_of_data_matrix																																																																												X	X																																														
	Misfits																																																																													X		X																																							
	Standard_error_in_eigenvalue																																																																														X	X																																							
	Errors_in_decomposition																																																																															X				X																															
	Clear_matrices																																																																															X			X																															
	Matrix_init																																																																																	X		X																															
	Calc_variance																																																																																		X	X																															
	Pca																																																																																				X				X																											
Menu	Select_from_menu																																																																																				X											X																			
	Write_menu																																																																																					X										X																			
	Write_message																																																																																						X									X																			
	Decompose																																																																																							X								X																			
	Error_entry																																																																																								X				X	X	X																						
	Quit																																																																																									X						X																			
	Set_column_width																																																																																										X					X																		X
	Change_column_headings																																																																																											X				X																		X
	Get_abserr																																																																																												X			X																			
	Get_vecterr																																																																																													X		X																			
	Get_nofacs																																																																																														X	X																			
	Menu_system																																																																																															X																			X
	Read_menu_file																																																																																																X																		
Keyops	Move_cursor_up																																																																																																	X																	X
	Move_cursor_down																																																																																																	X																X
	Move_cursor_left																																																																																																		X															X
	Move_cursor_right																																																																																																			X														X
	Move_page_up																																																																																																				X													X
	Move_page_down																																																																																																				X												X
	Move_page_right																																																																																																					X											X
	Move_page_left																																																																																																							X										X
	Move_to_home																																																																																																							X									X
	Move_to_end																																																																																																								X								X
	Move_to_top																																																																																																								X							X
	Move_to_bottom																																																																																																								X						X
	Move_to_far_left																																																																																																									X					X
	Move_to_far_right																																																																																																									X				X
	Edit_cell																																																																																																											X			X
	Goto_cell																																																																																																												X		X
	Call_change_matrix																																																																																																												X	X
	Handle_key																																																																																																														X

Unit scope

Uses	Unit											
	Utils	Vid_util	EMSdat	TargTest	LotusFil	EPAimprt	PFAUtils	PFAvid	IO_unit	Mathunit	Menu	Keyops
Utils												
Vid_util												
EMSdat												
TargTest			X									
LotusFil			X									
EPAimprt	X		X									
PFAutils	X	X	X									
PFAvid	X	X					X					
IO_unit			X		X	X	X	X				
Mathunit	X		X					X	X			
Menu	X	X	X	X			X	X	X	X		
Keyops	X	X	X				X	X			X	

3.1. Unit DATSTRU2

```
unit DATSTRU2; {Data_structure}

{The code contained in this unit is Copyright by T.G.Brockwell, 1989-92. All rights reserved.}

{$O+,F+}
                    { Creates and maintains a data structure of a }
                    { sparse array of records indexed by pointers. }
                    { Records are arranged as circularly linked lists }
                    { column by column with the top record of each }
                    { column being a row header which is part of a }
                    { circularly linked list of rows. Each sparse array }
                    { is anchored to a variable held in an index }
                    { containing maximum dimensions and a name for the }
                    { matrix. }

{********************************************************************}

interface

uses Overlay,PFAGlobs;

  function Write_to_matrix ( Name     : MatrixNameType;
                             Row,Col  : word;
                             Value    : Extended) : boolean;

  function Read_matrix ( Name    : MatrixNameType;
                         Row,Col : word;
                     var Value   : extended) : boolean;

  function Next_val (var Row,Col : word;
                     var Value   : extended) : boolean;

  function Matrix_dimensions ( Name      : MatrixNameType;
                           var Rows,Cols : word) : boolean;

  function Delete_matrix ( Name : MatrixNameType) : boolean;

{********************************************************************}

implementation

{********************************************************************}

type
  NodePtr = ^Node;
  Node = record
           NextColumn,LastColumn : NodePtr;
           case Header : boolean of
             False : (DataValue      : extended;
                      Row,Col        : word);
             True  : (RowNo          : word;
                      UpRow,DownRow  : NodePtr);
         end;
  IndexPtr = ^Index;
  Index = Record
            NextEntry : IndexPtr;
            Title     : MatrixNameType;
            Rows,Cols : word;
            Location  : NodePtr;
          end;

{********************************************************************}

var
  IndexHeader  : IndexPtr;            { Origin of record structure }
  CurrentEntry : NodePtr;             { Current entry in data array }

{********************************************************************}

function Start_data_array : NodePtr;

                             { Creates header for entire data array }

{-----------------------}

var
  DataArrayHeader : NodePtr;

{-----------------------}

begin
  New(DataArrayHeader);
  with DataArrayHeader^ do
  begin
    NextColumn := DataArrayHeader;
    LastColumn := DataArrayHeader;
    Header     := True;
    RowNo      := 0;
    UpRow      := DataArrayHeader;
    DownRow    := DataArrayHeader;
  end; {with}
  Start_data_array := DataArrayHeader;
end; {Start_data_array}

{********************************************************************}

function Find_val (var Srow,Scol              : word;
                   var MatrixHeader,TestPtr : NodePtr) : boolean;

                        { Searches entire data array for a particular }
                        { entry and returns it's position or the position }
                        { of the next entry (Find_val = False) }

{-----------------------}

var
  Found : boolean;          { True = successful search }

{-----------------------}

begin
  TestPtr := MatrixHeader^.UpRow;  { Set pointer to first row header }
  Found := False;
  while (TestPtr^.RowNo < Srow) and (TestPtr <> MatrixHeader) do
      TestPtr := TestPtr^.UpRow;
                                    { Search row headers till next largest }
                                    { row or array header reached }

  if TestPtr^.RowNo = Srow then
  begin
    TestPtr := TestPtr^.NextColumn;
    while (TestPtr^.Col < Scol) and not TestPtr^.Header do
        TestPtr := TestPtr^.NextColumn;
    if not TestPtr^.Header
      then Found := (TestPtr^.Col = Scol);
  end; {if}
                                     { Search ascending columns for a match }
                                     { till next largest column or header }
                                     { is reached if no match }
  CurrentEntry := TestPtr;
  Find_val := Found;
end; {Find_val}

{****************************************************************}

function Next_val (var Row,Col : word;
                   var Value   : extended) : boolean;

                      { Returns the next value in the data array. }

{----------------------}

begin
  CurrentEntry := CurrentEntry^.NextColumn;  { Point to next entry. }
  if CurrentEntry^.Header then               { Check for row header. }
  begin
    CurrentEntry := CurrentEntry^.UpRow;     { If header then move up a row. }
    if CurrentEntry^.RowNo = 0 then          { Check for MatrixHeader. }
      begin
        Next_val := false;         { If MatrixHeader then mark failiure }
        exit;                      { and leave. }
      end {if then}
    else
      CurrentEntry := CurrentEntry^.NextColumn; { Move to first data entry. }
  end; {if then}
  Row := CurrentEntry^.Row;                   { Return data. }
  Col := CurrentEntry^.Col;
  Value := CurrentEntry^.DataValue;
  Next_val := true;                           { Mark success. }
end; {Next_val}

{****************************************************************}

function New_row_header (var TheRow    : word;
                         var InsertPos : NodePtr) : NodePtr;

                                      { Creates a new node to act as }
                                      { a row header }

{----------------------}

var
  NewNode : NodePtr;

{----------------------}

begin
  New(NewNode);
  with NewNode^ do
  begin
    NextColumn       := NewNode;
    LastColumn       := NewNode;
    Header           := True;
    RowNo            := TheRow;
    UpRow            := InsertPos;
    DownRow          := InsertPos^.DownRow;
    DownRow^.UpRow   := NewNode;
    UpRow^.DownRow   := NewNode;
  end; {with}
  New_row_header := NewNode;
end; {New_row_header}

{****************************************************************}

procedure Create_node (var TheRow,TheCol : word;
                       var TheValue      : extended;
                       var InsertPos     : NodePtr);

                                      { Creates a new node to hold }
                                      { a data value and enters it }
                                      { in the data structure }

{----------------------}

var
  NewNode : NodePtr;

{----------------------}

begin
  New(NewNode);
  with NewNode^ do
  begin
    LastColumn              := InsertPos^.LastColumn;
    NextColumn              := InsertPos;
    NextColumn^.LastColumn  := NewNode;
    LastColumn^.NextColumn  := NewNode;
    Header                  := False;
    Row                     := TheRow;
    Col                     := TheCol;
    DataValue               := TheValue;
  end; {with}
end; {Create_node}

{****************************************************************}

function Delete_value (var Row,Col       : word;
                       var MatrixHeader : NodePtr) : boolean;

                                             { Delete selected node from }
                                             { data structure and tie up }
                                             { loose ends }
{----------------------}

var
  Found  : boolean;
  DelPtr : NodePtr;

{----------------------}

begin
  Delete_value := False;
  Found := Find_val(Row,Col,MatrixHeader,DelPtr);   { Find node }
  if not Found then exit;                    { Leave if no node }
  with DelPtr^ do                            { Delete node }
  begin
    NextColumn^.LastColumn := LastColumn;
    LastColumn^.NextColumn := NextColumn;
  end;
  if (DelPtr^.LastColumn = DelPtr^.NextColumn) then
    with DelPtr^.NextColumn^ do
    begin                                    { If node was only entry on }
      begin                                  { row then delete row header }
        DownRow^.UpRow := UpRow;
        UpRow^.DownRow := DownRow;
      end; {with}
      Dispose(DelPtr^.NextColumn);
    end; {if}
  Dispose(DelPtr);
  Delete_value := True;
end; {Delete_value}

{******************************************************************}

procedure Check_bounds(IndexEntry : IndexPtr);

var
  MaxRow,MaxCol : word;
  TestEntry     : NodePtr;
  Success       : boolean;

begin
  MaxCol := 0;
  TestEntry := IndexEntry^.Location^.DownRow;
  MaxRow := TestEntry^.RowNo;
  while not (TestEntry^.RowNo = 0) do
  begin
    if TestEntry^.LastColumn^.Col > MaxCol
    then MaxCol := TestEntry^.LastColumn^.Col;
    TestEntry := TestEntry^.DownRow;
  end; {while}
  If (MaxRow = 0) and (MaxCol = 0)
  then Success := Delete_matrix(IndexEntry^.Title);
  IndexEntry^.Rows := MaxRow;
  IndexEntry^.Cols := MaxCol;
end; {Check_bounds}

{******************************************************************}

function Put_val (var IndexEntry : IndexPtr;
                  var Row,Col    : word;
                  var Value      : extended) : boolean;

                                      { Changes a value in the data }
                                      { structure or adds a value creating }
                                      { the necessary nodes }

{----------------------}
var
  InsertPos,
  NewNode     : NodePtr;
  Found       : boolean;

{----------------------}

begin
  Put_val := False;             { Set function to 'unable to insert' }
  Found := Find_val(Row,Col,IndexEntry^.Location,InsertPos); { Find where to change or create new node }
  if not Found and (Maxavail < Sizeof(node)) then exit;
  if Found then
    if Value = 0 then
      begin
        Put_val := Delete_value(Row,Col,IndexEntry^.Location); { If value is zero then delete node }
        Check_bounds(IndexEntry);                              { Check the dimensions of the matrix }
        exit
      end {then}
    else
      begin
        InsertPos^.DataValue := Value;      { Edit an existing value }
        Put_val := True;                         { Mark success }
        exit                                   { Leave function }
      end {then}
  else
    if Value = 0 then                                 { Don't add zero entries. }
    begin
      Put_val := true;
      exit;
    end; {if}
    if InsertPos^.Header and (InsertPos^.RowNo <> Row) then  { Check if there is the correct }
    begin                                     { row header present }
      if Maxavail < (Sizeof(node) * 2) then exit;
      NewNode := New_row_header(Row,InsertPos);      { or create it }
      InsertPos := NewNode;
    end; {if}
  Create_node(Row,Col,Value,InsertPos);
  Put_val := True;
end; {Put_val}

{******************************************************************}

function Get_val (var MatrixHeader : NodePtr ;
                 var Row,Col      : word) : extended;
                                            { Finds a data value from }
                                            { the data structure and }
                                            { returns it }

{----------------------}
```

```
var
  FindNode : NodePtr;
  Found    : boolean;

{-----------------------}

begin
  Found := Find_val(Row,Col,MatrixHeader,FindNode);
  if Found then
    Get_val := FindNode^.DataValue
  else
    Get_val := 0;
end; {Get_val}

{****************************************************************}

procedure Start_index;

                       { Creates header for the data array index }

{-----------------------}

begin
  New(IndexHeader);                    { Create anchor variable. }
  with IndexHeader^ do
  begin
    NextEntry := IndexHeader;          { Initialise all fields. }
    Title     := NONE;
    Rows      := 1;
    Cols      := 1;
    Location  := nil;
  end; {with}
end; {Start_index}

{****************************************************************}

function Find_index_entry (var Name                  : MatrixNameType;
                           var PrevEntry,IndexTestPtr : IndexPtr) : boolean;

                                { Searches index to find a matrix. }
                                { Returns a pointer to the entry and the }
                                { one before it (Find_index_entry = true) }
                                { or points to the header and the last }
                                { entry of the list (Find_index_entry = }
                                { false). }

{-----------------------}

var
  Found : boolean;

{-----------------------}

begin
  IndexTestPtr := IndexHeader^.NextEntry;        { Start at first entry. }
  PrevEntry := IndexHeader;
  Found := False;
  while (IndexTestPtr <> IndexHeader) and (not Found) do
  begin
    Found := (Name = IndexTestPtr^.Title);       { Test for match. }
    if not Found
    then begin
           PrevEntry := IndexTestPtr;
           IndexTestPtr := IndexTestPtr^.NextEntry; { Look at next record. }
         end; {if}
  end; {while}
  Find_index_entry := Found;                  { Mark success or failure. }
end; {Find_index_entry}

{****************************************************************}

procedure Create_index_entry (var Name      : MatrixNameType ;
                              var PrevEntry : IndexPtr);

                              { Creates a new index entry an inserts it }
                              { at the end of the linked list. }

{-----------------------}

var
  HeaderAddress : NodePtr;
  NewNode : IndexPtr;

{-----------------------}

begin
  HeaderAddress := Start_data_array;      { Create new header for matrix. }
  New(NewNode);                           { Create new index record. }
  with NewNode^ do
  begin
    NextEntry := IndexHeader;             { Initialise record entries. }
    Title     := Name;
    Rows      := 0;
    Cols      := 0;
    Location  := HeaderAddress;           { Set pointer to new matrix }
  end; {with}
  PrevEntry^.NextEntry := NewNode;        { Insert entry into index. }
end; {Create_index_entry}

{****************************************************************}

function Delete_matrix ( Name : MatrixNameType) : boolean;

                             { Deletes every member of a matrix and removes }
                             { it's entry from the index to free memory. }

{-----------------------}

var
  TargetMatrix,PrevEntry : IndexPtr;
  DelPtr                 : NodePtr;
  Found                  : boolean;
  Success                : boolean;

{-----------------------}

begin
  Success := False;
  Found := Find_index_entry(Name,PrevEntry,TargetMatrix);
  if not Found then
  begin
    Delete_matrix := Success;
    exit;
  end; {if}
  DelPtr := TargetMatrix^.Location^.UpRow;     { Set to first row past }
                                               { matrix header. }
  while (DelPtr <> TargetMatrix^.Location) do  { For each row till the }
                                               { header is reached. }
  begin
    DelPtr := DelPtr^.NextColumn;              { Set to first column past }
    repeat                                     { row header. }
      DelPtr := DelPtr^.NextColumn;
      Dispose(DelPtr^.LastColumn);             { Delete all entries on row.}
    until DelPtr^.Header;
    DelPtr := DelPtr^.UpRow;                   { Move up a row. }
    Dispose(DelPtr^.DownRow);                  { Delete previous row. }
  end; {while}
  Dispose(TargetMatrix^.Location);             { Delete matrix header. }
  PrevEntry^.NextEntry := TargetMatrix^.NextEntry; { Take index entry out }
                                                   { of the index. }
  Dispose(TargetMatrix);                           { Delete index entry }
  Success := True;                                 { Mark succesful deletion. }
  Delete_matrix := Success;
end; {Delete_matrix}

{********************************************************************}

function Read_matrix ( Name    : MatrixNameType;
                       Row,Col : word;
                   var Value   : extended) : boolean;

                                            { Checks that the matrix }
                                            { exists and reads data. }

{-----------------------}

var
  Found : boolean;
  PrevEntry,IndexEntry : IndexPtr;

{-----------------------}

begin
  Read_matrix := False;
  Found := Find_index_entry(Name,PrevEntry,IndexEntry);
  if not Found then
  begin
    Value := 0;
    exit;
  end; {if}
  Value := Get_val(IndexEntry^.Location,Row,Col);
  Read_matrix := Found;
end; {Read_matrix}

{********************************************************************}

function Write_to_matrix ( Name    : MatrixNameType;
                           Row,Col : word;
                           Value   : Extended) : boolean;

                                           { Checks existence of matrix }
                                           { and creates it if necessary }
                                           { then updates the bounds and }
                                           { enters the value. }

{-----------------------}

var
  Found : boolean;
  PrevEntry,IndexEntry : IndexPtr;

{-----------------------}

begin
  Write_to_matrix := False;
  Found := False;
  while not Found do
  begin
    Found := Find_index_entry(Name,PrevEntry,IndexEntry);
    if not Found then
    begin
      if Maxavail < (Sizeof(Index) + (Sizeof(Node) * 3)) then exit;
      Create_index_entry(Name,PrevEntry);
    end; {if}
  end; {while}
  with IndexEntry^ do
  begin
    if Rows < Row then Rows := Row;
    if Cols < Col then Cols := Col;
  end; {with}
  Write_to_matrix := Put_val(IndexEntry,Row,Col,Value);
end; {Write_to_matrix}

{********************************************************************}

function Matrix_dimensions ( Name      : MatrixNameType;
                         var Rows,Cols : word) : boolean;

                                          { Finds matrix and returns }
                                          { dimensions. }

{-----------------------}

var
  Found : boolean;
  PrevEntry,IndexEntry : IndexPtr;

{-----------------------}

begin
  Matrix_dimensions := False;
  Found := Find_index_entry(Name,PrevEntry,IndexEntry);
  if not Found then exit;
  Rows := IndexEntry^.Rows;
  Cols := IndexEntry^.Cols;
  Matrix_dimensions := Found
```

```
end; {Matrix_dimensions}

{********************************************************************}

begin
  CurrentEntry := nil;
  Start_index;
end. {Data_structure}
```

3.2. Unit OVINIT

```
unit OvInit;

{The code contained in this unit is Copyright by T.G.Brockwell, 1989-92. All rights reserved.}


interface
implementation

uses Overlay;

begin
  OvrInit('TFA.OVR');
  OvrInitEMS;
  OvrSetBuf(30000);
end.
```

3.3. Unit PFAGLOBS

```
unit pfaglobs;

{The code contained in this unit is Copyright by T.G.Brockwell, 1989-92. All rights reserved.}

{$O+,F+}

interface

uses Overlay,Crt;

const
{ Key constants }
  NullChr    = #0;         { Null Character }
  Bell       = #7;         { Beep }
  BkSpc      = #8;         { Backspace }
  Tab        = #9;         { Tab }
  Shift_tab  = #15;        { Shift tab }
  CR         = #13;        { Carriage return }
  Esc        = #27;        { Escape }
  Space      = #32;        { Space character }
  Slash      = #47;        { Slash character }
  F1_key     = #59;        { F1 key scan code }
  F2_key     = #60;        { F2 key scan code }
  F3_key     = #61;        { F3 key scan code }
  F4_key     = #62;        { F4 key scan code }
  F5_key     = #63;        { F5 key scan code }
  F6_key     = #64;        { F6 key scan code }
  F7_key     = #65;        { F7 key scan code }
  F8_key     = #66;        { F8 key scan code }
  F9_key     = #67;        { F9 key scan code }
  F10_key    = #68;        { F10 key scan code }
  Ins_key    = #82;        { Insert key scan code }
  Del_key    = #83;        { Delete key scan code }
  Home_key   = #71;        { Home key scan code }
  Ctrl_home  = #119;       { Control home key scan code }
  End_key    = #79;        { End key scan code }
  Ctrl_end   = #117;       { Control end key scan code }
  Page_up    = #73;        { Page up key scan code }
  Page_down  = #81;        { Page down key scan code }
  Ctrl_pgup  = #132;       { Control page up scan code }
  Ctrl_pgdn  = #118;       { Control page down scan code }
  Cur_right  = #77;        { Right arrow scan code }
  Cur_left   = #75;        { Left arrow scan code }
  Cur_up     = #72;        { Up arrow key scan code }
  Cur_down   = #80;        { Down arrow key scan code }
  Ctrl_rght  = #116;       { Control right arrow key scan code }
  Ctrl_left  = #115;       { Control left arrow key scan code }

 type
  Coords = record
             Column,Row : word;
           end;
  StatusRecord = record
                   Text     : String[5];
                   Colour   : byte;
                   Blinking : boolean;
                 end;

  Str_2   = String[2];
  Str_3   = String[3];
  Str_4   = String[4];
  Str_5   = string[5];
  Str_6   = string[6];
  Str_8   = string[8];
  Str_9   = string[9];
  Str_10  = string[10];
  Str_12  = string[12];
  Str_13  = string[13];
  Str_20  = string[20];
  Str_30  = string[30];
  Str_80  = string[80];
  Str_255 = string[255];
  MatrixNameType = (NONE,DATA,COVA,RESI,LOAD,SCOR,
                    COMP,INTM,TKON,TVEC,TTST,DIFM);
  MatrixNameMap = array [MatrixNameType] of Str_30;
  StatusType = (RE,ER,WA,FI,ME,ED,LA,PO,VA);
  StatusDataType = array [StatusType] of StatusRecord; { Status flag specs. }
  TypeOfFile = (Native,ASCII,Lotus);


 var
  MonoSystem     : boolean;     { True if Mono card present }
  Found          : boolean;     { Flag for data structure procedures }
  EscPress       : boolean;     { Escape flag for procedures }
  CapsOn,NumOn,ScrollOn : boolean; { Keyboard light flags }
  NoOfCols       : byte;        { Number of columns possible in one screen }
  NoOfRows       : byte;        { Number of rows possible on one screen }

 type  {Set types}
  Any_char = set of char;

 const {Typed constants}
  FileChanged    : boolean = false;
  RedrawScreen   : boolean = true;  { Flag to redraw whole screen }
  Refresh        : boolean = true;  { Flag to redraw number area }
  EndSession     : boolean = false; { Flag to signal end of program }
  Headings       : boolean = true;  { Flag to show column headings or numbers }
  ColWidth       : byte = 10;       { Width of each number column }
  Status         : StatusType = RE; { Holds current status of spreadsheet }
  FacExTest      : extended = 1e-19;{ Factor extraction test value }
  ResNulTest     : extended = 1e-19;{ Completed decomposition test value }
  AbsError       : extended = 0;    { Estimate of error in the data matrix }
  VectErr        : extended = 0;    { Estimate of error in the test vector }
  NoFacs         : word = 0;        { Number of factors used to model data }
  CurrentMatrix  : MatrixNameType = None; { Name of displayed matrix }
  Letters        : Any_char = ['a'..'z','A'..'Z'];
  Numbers        : Any_char = ['0'..'9','.'];
  MaxInt         : longint = 32767;
  MaxLInt        : longint = 2147483647;
  MaxByte        : longint = 255;
  MaxWord        : longint = 65535;
  MaxReal        : extended = 1.7e38;
  MaxExt        : extended = 1.1e4932;

  MatrixDim     : Coords = (Column:1;Row:1);
                           { Dimensions of the current matrix }
  Origin        : Coords = (Column:1;Row:1);
                           { Coordinates of top left cell of screen }
  CursorPos     : Coords = (Column:1;Row:1);
                           { Current cursor screen row and column }
  OldCurPos     : Coords = (Column:1;Row:1);
                           { Old position of cursor on screen }
  MatrixNameStr : MatrixNameMap = ('No Matrix to display','Data Matrix',
                                   'Covariance Matrix','Residual Matrix',
                                   'Abstract Column Matrix',
                                   'Abstract Row Matrix','Composite Matrix',
                                   'Intermediate Matrix','Transform Constant',
                                   'Transform Vector','Target Test Matrix',
                                   'Vector Differential Matrix');
  MaxMatrixName : MatrixNameType = DIFM;        { Must be set to last name of }
                                                        { MatrixNameType }
  StatusData : StatusDataType = ((Text:'Ready';Colour:Cyan   ;Blinking:false),
                                 (Text:'ERROR';Colour:Red    ;Blinking:true ),
                                 (Text:'Wait ';Colour:Magenta;Blinking:true ),
                                 (Text:'Files';Colour:Cyan   ;Blinking:false),
                                 (Text:'Menu ';Colour:Magenta;Blinking:false),
                                 (Text:'Edit ';Colour:Cyan   ;Blinking:false),
                                 (Text:'Label';Colour:Cyan   ;Blinking:false),
                                 (Text:'Point';Colour:Magenta;Blinking:false),
                                 (Text:'Value';Colour:Cyan   ;Blinking:false));

implementation

end.
```

3.4. Unit VID_UTIL

```
unit Vid_util;

{The code contained in this unit is Copyright by T.G.Brockwell, 1989-92. All rights reserved.}


{$O+,F+}

interface

uses Overlay,Crt,DOS,PFAGlobs;

{ Video limits: }

const
  MaxRow       = 25;             { Maximum video rows. }
  MaxCol       = 80;             { Maximum video columns. }

{ Direct video access data structures: }

type
  PCChar       = record
                     character   : Char;
                     attribute   : Byte;
                   end;

  CharLine     = array[1..MaxCol] of PCChar;

  InpLines     = array[1..MaxRow] of CharLine;
  Curtype      = (Off, Big, Small);

var
  vid : Text;
  ColorLine  : InpLines absolute $b800:$0000;
  MonoLine   : InpLines absolute $b000:$0000;
  LineBuf    : InpLines; { Video line buffer. }

function  Is_mono: Boolean;

procedure BIOSCursor(Size : Curtype);

procedure Set_border(Border: Byte);

procedure Hide_cursor;

procedure Restore_cursor;

procedure Kbd_light_status;

procedure Colour_box (TopX,TopY,LwrX,LwrY,Fore,Back : byte;
                                           Blinking  : boolean);

procedure Fill_screen (TopX,TopY,LwrX,LwrY : byte;
                       Character           : char);

function Get_attribute (X,Y : byte) : byte;

implementation

function Is_Mono : Boolean; { Determines presence of monochrome video adaptor }

     { IBM-PC equipment flag location is $0040:0010. Bits 4 and 5
       set (00110000) indicate monochrome video adapter. }

begin
  if ((Mem[$0040:$0010] and $30) = $30)
    then Is_Mono := TRUE
    else Is_Mono := FALSE;
end; { Is_Mono }

procedure BIOSCursor(Size : Curtype);  { Sets the size of the IBM hardware }
                                       { text mode cursor }
var
  Regs : registers;

begin
  with Regs do
  begin
    AH := $01;
    CL := $07;           { Scan line to stop drawing cursor at }
    case Size of
     Off   : CH := $20;  { Sets bit 5 to hide cursor }
     Big   : CH := $00;  { Scan line to start drawing cursor at }
     Small : CH := $06;  { May not work correctly on MDA adapters }
    end; {case}
    Intr($10, Regs)
  end; {with}
end; {BIOSCursor}

procedure Set_Border(border: Byte);                     { Sets screen border }

var
  ms_reg : Registers;

begin
  Fillchar(ms_reg,SizeOf(ms_reg),0);
  with ms_reg do
  begin
    ah := $0d;
    bl := border;
  end; {with}
  intr($10,ms_reg);
end; { Set_Border }

procedure Hide_Cursor;                                 { Turns off BIOS cursor }

var
  ms_reg : Registers;

begin
  ms_reg.ah  := $01;
  ms_reg.ch  := $20;
  ms_reg.cl  := $00;
  Intr($10,ms_reg);
```

```
end; { Hide_Cursor }

procedure Restore_Cursor;                              { Turns BIOS cursor on }

var
  ms_reg : Registers;

begin
  ms_reg.ah := $01;
  if MonoSystem then
  begin
    ms_reg.ch := $0c;
    ms_reg.cl := $0d;
  end {if then}
  else
  begin
    ms_reg.ch := $06;
    ms_reg.cl := $07;
  end; {if else}
  Intr($10,ms_reg);
end; { Restore_Cursor }

procedure Kbd_light_status;                     { Shows when lock keys are set }

var
  KbdFlag               : byte absolute $40:$17; { BIOS keyboard flag byte 0 }

const                  { 7654 3210  bit No. }       { Binary masks for testing }
  CapsLock   = $40;    { 0100 0000 ,bit 6 set indicates on }
  NumLock    = $20;    { 0010 0000 ,bit 5 set indicates on }
  ScrollLock = $10;    { 0001 0000 ,bit 4 set indicates on }

begin                                  { Using masks test which locks are set }
  CapsOn := ((KbdFlag and CapsLock) = CapsLock);
  NumOn := ((KbdFlag and NumLock) = NumLock);
  ScrollOn := ((KbdFlag and ScrollLock) = ScrollLock);
end; {Kbd_light_status}

procedure Colour_box (TopX,TopY,       { Draws a coloured box between corners }
                      LwrX,LwrY,
                      Fore,Back : byte;
                      Blinking  : boolean);

var
  I,J,Attrib : byte;

begin
  Attrib := Back*16 + Fore;                                      { Calculate }
  if Blinking then Attrib := Attrib + Blink;                    { colour byte }
  for I := TopY to LwrY do                                     { For each row }
  begin
    for J := TopX to LwrX do                                { For each column }
    begin
      if MonoSystem then
        MonoLine[I,J].Attribute := Attrib      { Change colour on mono adaptor}
      else
        ColorLine[I,J].Attribute := Attrib  { Change colour on colour adaptor }
    end; {for J}
  end; {for I}
end; {Colour_box}

procedure Fill_screen (TopX,TopY,LwrX,LwrY : byte;    { Fills a specified area }
                       Character           : char);   { area of screen with a }
                                                                 { character }
var
  I,J : byte; { Local loop control }

begin
  for I := TopY to LwrY do                                     { For each row }
  begin
    for J := TopX to LwrX do                                { For each column }
    begin
      if MonoSystem then
        MonoLine[I,J].Character := Character     { Write char on mono adaptor }
      else
        ColorLine[I,J].Character := Character; { Write char on colour adaptor }
    end; {for J}
  end; {for I}
end; {FillScreen}

function Get_attribute (X,Y : byte) : byte;  { Returns current text attribute }
                                            { at location specified by coords }
begin
  If MonoSystem then
    Get_attribute := MonoLine[Y,X].Attribute
  else
    Get_attribute := ColorLine[Y,X].Attribute;
end; {Get_attribute}

begin {Vid_util initialisation}
  MonoSystem := Is_Mono;                          { Determine if mono adaptor }
end. {Vid_util}
```

3.5. Unit UTILS

```
unit Utils;

{The code contained in this unit is Copyright by T.G.Brockwell, 1989-92. All rights reserved.}

{$O+,F+}

interface

uses Overlay,Crt,DOS,PFAGlobs,Opdate;

type
  SwitchType =(On,Off);

procedure Convert_to_base_26 (Column : word;
                          var CodeString : Str_255);

procedure Convert_to_base_10 (CodeStr  : Str_255;
                          var Column   : word);

procedure Get_date_time (var DayString : Str_30);

procedure Elapsed_time_timer( Switch : SwitchType;
                          var ETime  : Str_80);

procedure Strip (var inp_str     : Str_255;
                     strip_set   : Any_char);

procedure Strip_Leading (var inp_str     : Str_255;
                             strip_set   : Any_char);

procedure Strip_trailing (var inp_str     : Str_255;
                              Strip_set   : Any_char);

function Power( X : Extended; Y : integer) : extended;

function String_to_number (var InpStr,Value,Min,Max;
                           var Code          : integer;
                               Int,AllowZero : boolean) : boolean;

procedure Read_kbd (var inchr,inctl : char);

function Parse_coords (CellAddress : Str_255;
                   var Row,Col     : word) : boolean;

procedure Check_for_esc;

function F_prob( f : real; k1,k2 : integer) : real;

implementation

procedure Convert_to_base_26 (Column : word;         { Takes a column No. and }
                          var CodeString : Str_255);  { converts it to letter }
                                                  { format spreadsheet style }
const
  Multiplier : array[1..5] of longint = (1,26,676,17576,456976);

var
  Position,Mm,Limit,I : byte;  { Note I masks Global I for local loop control }
  IntPart             : word;
  FracPart            : real;

begin
  Position := 0;                          { Indicates current column of code }
  FillChar(CodeString,256,' ');                          { Contains the code }
  CodeString[0] := Chr(Pred(SizeOf(CodeString)));
  repeat                                          { This loop finds the first }
    Position := Succ(Position);              { multiplier larger than column }
  until Multiplier[Position] > Column;
  Mm := Pred(Position);         { Stores the position of the largest multiple }
  for I := Pred(Position) downto 1 do
  begin                    { From the first divisible multiple down to units }
    IntPart := Column div Multiplier[I];       { Find the number of multiples }
    CodeString[(Length(CodeString)+1 - I)] := Chr(IntPart+64);
                                              { Add the relevant code letter }
    Column := Column mod Multiplier[I];             { Calculate the remainder }
  end; {for}
  I := Length(CodeString);              { Set loop length to CodeString length }
  Limit := I - (Mm -1);                 { Points to the most significant digit }
  repeat
    If (Ord(CodeString[I]) < 65) and (I <> Limit) then
                                        { If code is less than A and not MSD }
    begin
      CodeString[I] := Chr(Ord(CodeString[I]) + 26);            { Set A to Z }
      CodeString[Pred(I)] := Pred(CodeString[Pred(I)]);{ Subtract 1 from next }
    end; {if}                                                  { highest bit }
    I := I - 1;                         { Move down code string right to left }
  until (I <= Limit);                                      { Until reach MSD }
  If (Ord(CodeString[I]) < 65) then CodeString[I] := Chr(32);  { If MSD is <A }
end; {Convert_to_base_26}                                     { Set to space }

procedure Convert_to_base_10 (CodeStr : Str_255;      { Converts a column code }
                          var Column  : word);          { to its decimal value }

const
  Multiplier : array [1..5] of longint = (456976,17576,676,26,1);

var
  I,Ln : byte;

begin
  Column := 0;
  Ln := Length(CodeStr);
  for I := 1 to Ln do
    Column := Column + (Multiplier[(5-Ln)+I] * Ord(UpCase(CodeStr[I]))-64);
end; {Convert_to_base_10}

procedure Get_date_time (var DayString : Str_30);

var
DateStr : DateString;
TimeStr : DateString;
```

```
begin
  FillChar(DayString,SizeOf(DayString),' ');
  DayString[0] := #30;
  DateStr := TodayString('wwwwwwwww dd/mm/yy');
  TimeStr := CurrentTimeString('hh:mm:ss');
  DayString := DateStr + '  ' + TimeStr;
end; {Get_date_time}

procedure Elapsed_time_timer( Switch : SwitchType;
                          var ETime  : Str_80);

const
  DT1 : DateTimeRec = (D : 0;T : 0);
  DT2 : DateTimeRec = (D : 0;T : 0);

var

  Days : word;
  Secs : longint;
  Hours, Minutes, Seconds : byte;
  DayStr,HourStr,MinStr,SecStr : Str_80;

begin
  case Switch of
    On : begin
           DT1.D := Today;
           DT1.T := CurrentTime;
           ETime := 'Elapsed time = 0';
         end;
    Off : begin
            DT2.D := Today;
            DT2.T := CurrentTime;
            DateTimeDiff(DT1, DT2, Days, Secs);
            TimeToHMS(Secs,Hours,Minutes,Seconds);
            Str(Days,DayStr);
            Str(Hours,HourStr);
            Str(Minutes,MinStr);
            Str(Seconds,SecStr);
            ETime := 'Elapsed time = '+DayStr+' Days, '+
                     HourStr+' Hours, '+
                     MinStr+' Minutes, '+
                     SecStr+' Seconds';
          end;
  end; {case}
end; {Elapsed_time_timer}

procedure Strip_trailing (var inp_str    : Str_255;
                              Strip_set  : Any_char);

var                   { Strips specified characters from the end of inp_str }
  len : Byte;

begin
  len := Length(inp_str);
  inp_str[0] := Chr(0);                              { Set inp_str length to ZERO. }
  while (inp_str[len] in strip_set) and (len > 0) do
    len := Pred(len);
  inp_str[0] := Chr(len);                                    { Set inp_str to len. }
end; {Strip_trailing}

procedure Strip_Leading (var inp_str     : Str_255;
                             strip_set   : Any_char);

var                        { Strips specified characters from the beginning }
  i,len    : Byte;         { of inp_str }

begin
  len := Length(inp_str);
  i := 1;
  While (inp_str[i] in strip_set) and (i <= len) do
    i := Succ(i);
  inp_str := Copy(inp_str,i,len);
end; {Strip_leading}

procedure Strip (var inp_str     : Str_255;    { Removes specified characters }
                     strip_set   : Any_char);  { from the start and finish }
                                               { of inp_str }

begin
  Strip_Trailing(inp_str,strip_set);
  Strip_Leading(inp_str,strip_set);
end; {Strip}

function Power( X : Extended; Y : integer) : extended;  { Calculates X raised }
                                                        { to the power Y where }
                                         { may be a positive or negative integer }
var
  Z : extended;
  I : integer;

begin
  Z := 1.0; I := Abs(Y);
  while I > 0 do
  begin
    if Odd(I) then Z := Z * X;
    I := I div 2;
    X := Sqr(X);
  end; {while}
  if Y < 0 then Z := 1/Z;
  Power := Z;
end; {Power}

procedure Round_value (var Value  : extended;      { Rounds an extended number }
                           Places : integer);        { to the specified number }
                                                                  { of places }
var
  Factor : extended;

begin
  Factor := Power(10,Places);
  Value := Value * Factor;
  Value := Round(Value);
  Value := Value / Factor;
end; {Round_value}

function String_to_number (var InpStr,Value,Min,Max;   { Converts a string to }
                           var Code      : integer;     { a real or integer }
                               Int,AllowZero : boolean) : boolean;


{ NOTES: }
{ Input string may be any size or string type. }
{ Value must be be either longint or extended. If appropriate range checking }
{ is performed the result may be typecast to any of the real or integer types }
{ without a range check error. }
{ Max and Min values must be either longint or extended values only. }
{ If Int is true an integer value is returned or else a real value is. }
{ The function may be false if the number is out of range or an unsuccessful }
{ conversion took place. Code will be non zero if the conversion went wrong. }
{ The original value is unchanged if Ok is false. }
{ For extended numbers 0 is valid only if AllowZero is true. }
{ Range checking for extended numbers covers both positive and negative }
{ numbers in the range specified by Min/Max. }
{ Range checking for integers may include negative numbers by supplying a }
{ negative Min value. }

var
  Ok     : boolean;                 { Flag for in-range operation }
  AbsStr : Str_255 absolute InpStr; { Input string - may be any size }
  AbsInt : longint absolute Value;  { Variable for integer output }
  AbsExt : extended absolute Value; { Variable for real output }
  TmpInt : longint;                 { Temporary integer for conversion }
  TmpExt : extended;                { Temporary extended for conversion }
  ExtMax : extended absolute Max;   { Real maximum }
  IntMax : longint absolute Max;    { Integer maximum }
  ExtMin : extended absolute Min;   { Real minimum }
  IntMin : longint absolute Min;    { Integer maximum }

begin
{$R-}
  case Int of
    true  : begin
              Val(AbsStr,TmpInt,Code);
              Ok := (Code = 0) and (TmpInt >= IntMin) and (TmpInt <= IntMax);
              if Ok then AbsInt := TmpInt;
            end; {true}
    false : begin
              Val(AbsStr,TmpExt,Code);
              Ok := (Code = 0) and (Abs(TmpExt) >= ExtMin)
                               and (Abs(TmpExt) <= ExtMax)
                 or (Code = 0) and AllowZero and (TmpExt = 0);
              if Ok then AbsExt := TmpExt;
            end; {true}
  end; {case}
  String_to_number := Ok;
{$R+}
end; {String_to_number}

procedure Read_kbd (var inchr,inctl : char);           { Reads keyboard buffer }

begin
  inctl := NullChr;
  inchr := ReadKey;
  if inchr = NullChr then                { If control key read next character }
    inctl := ReadKey
  else
    if (inchr in [#1..#31,Slash]) then            { Makes keys 1-31 and slash }
      inctl := inchr;                                          { control keys }
end; {Read_kbd}

function Parse_coords (CellAddress : Str_255;          { Takes a coord string }
                  var Row,Col     : word) : boolean;   { and converts it to a }
                                                      { row and column number }
const
  Delimiter : Any_char = [',',' ']; { Coordinate separators }
  Max       : longint = 65535;      { Range check constants }
  Min       : longint = 1;          { for string conversion }
  Int       : boolean = true;       { Flag for integer conversion }
  AllowZero : boolean = false;      { Flag to disallow a zero entry }

var
  I             : integer;  { Loop control }
  RowStr,ColStr : Str_255;  { Coordinate strings }
  Ok,Convert    : boolean;  { Logic flags }
  SplitPos      : byte;     { Division between coords }
  Code          : integer;  { Conversion error code }

begin
  Convert := false;                                          { Initialization }
  Ok := false;                                                         { code }
  for I := 1 to Length(CellAddress) do
  begin
    if CellAddress[I] in Delimiter                            { Find division }
      then SplitPos := I;                                { between row & col }
    if CellAddress[I] in Letters then
    begin
      SplitPos := I;                                 { Note if col is letters }
      Convert := true;
    end; {if}
  end; {for}
  ColStr := Copy(CellAddress,1,SplitPos);                      { Separate col }
  Strip_trailing(ColStr,Delimiter);
  RowStr := Copy(CellAddress,Succ(SplitPos),Length(CellAddress)-SplitPos);
  if Convert then                                              { Separate row }
  begin
    Convert_to_base_10(ColStr,Col);               { Convert letters to numbers }
    Ok := true;
  end {if then}
  else
  begin
    Col := longint(Col);                                     { String to word }
    Ok := String_to_number(ColStr,Col,Min,Max,Code,Int,AllowZero); { conversion routine }
    if Ok and (Code = 0)
      then Col := word(Col);
  end; {if else}
  Row := longint(Row);                         { String to word conversion }
  Convert := String_to_number(RowStr,Row,Min,Max,Code,Int,AllowZero);
  Row := word(Row);
  Parse_coords := Ok and Convert;                  { Mark success or failiure }
end; {Parse_coords}

procedure Check_for_esc;

var
  Inchr,Inctl : char;

begin
  If KeyPressed then Read_kbd(Inchr,Inctl);
  if (Inctl in [Esc]) then EscPress := true
  else EscPress := false;
```

```
end; {Check_for_esc}

function F_prob( f : real; k1,k2 : integer) : real;

{ This function is reproduced from Cooke, Craven & Clarke, }
{ Statistical Computing in Pascal, 1985, Edward Arnold, London }

{ The distribution function of the F distribution based on }
{ k1 and k2 degrees of freedom; uses functions Ln_gamma }
{ and Beta_ratio. }

var
  h1,h2  : real; { Modified degrees of freedom }
  LnBeta : real; { log of complete beta function with parameters }
                 { h1 and h2. }
  x      : real; { Argument of incomplete beta function. }

  function Ln_gamma( w : real) : real;

  { Calculates the logarithm of the gamma function; }
  { w must be such that 2*w is an integer > 0. }

  const
    a = 0.57236494; { ln(sqrt(pi)) }

  var
    Sum : real; { temporary store for summation of values. }

  begin {Ln_gamma}
    Sum := 0;
    w := w-1;
    while w > 0.0 do
    begin
      Sum := Sum + ln(w);
      w := w-1;
    end; {while}
    if w < 0.0
      then Ln_gamma := Sum + a
      else Ln_gamma := Sum;
  end; {Ln_gamma}

  function Beta_ratio( x,a,b,LnBeta : real) : real;

  { Calculates the incomplete beta function ratio with }
  { parameters a and b. LnBeta is the logarithm of the }
  { complete beta function with parameters a and b. }

  const
    Error = 1.0e-7;

  var
    c,                 { c = a + b }
    Factor1,
    Factor2,
    Factor3 : real;    { Factors multiplying terms in series. }
    i,j     : integer; { Counters. }
    Sum     : real;    { Current sum of series. }
    Temp    : real;    { Temporary store for exchanges. }
    Term    : real;    { Term of series. }
    xLow    : boolean; { Status of x which determines the end }
                       { from which the series is evaluated. }
    y       : real;    { Adjusted argument. }

  begin { Beta_ratio }
    if (x=0) or (x=1)
    then Sum := x
    else begin
      c := a + b;
      if a < c*x
      then begin
        xLow := true;
        y := x;
        x := 1 - x;
        Temp := a;
        a := b;
        b := Temp;
      end {if then}
      else begin
        xLow := false;
        y := 1-x;
      end; {if else}
      Term := 1;
      j := 0;
      Sum := 1;
      i := trunc(b+c*y)+1;
      Factor1 := x/y;
      repeat
        j := j+1;
        i := I-1;
        if I > 0
        then begin
          Factor2 := b-j;
          if i = 0 then Factor2 := x;
        end; {if}
        Term := Term*Factor2*Factor1/(a+j);
        Sum := Sum + Term;
      until (abs(Term) <= Sum) and (abs(Term) <= error*Sum);
      Factor3 := exp(a*ln(x) + (b-1)*ln(y) - LnBeta);
      Sum := Sum*Factor3/a;
      if xLow then Sum := 1 - Sum;
    end; {if else}
    Beta_ratio := Sum;
  end; { Beta_ratio }

begin { F_prob }
  h1 := 0.5 * k1;
  h2 := 0.5 * k2;
  x := h2/(h2+h1*f);
  LnBeta := Ln_gamma(h1) + Ln_gamma(h2) - Ln_gamma(h1+h2);
  F_prob := 1 - Beta_ratio(x,h2,h1,lnBeta);
end; { F_prob }

{*******************************************************************}


end.
```

3.6. Unit EMSDAT

```
unit EMSDAT;  { EMS Data Structures }

{The code contained in this unit is Copyright by T.G.Brockwell, 1989-92. All rights reserved.}

{$O+,F+}

interface

uses PFAGlobs,OpRoot,OpLarray;

const
  EXVN : AutoPriority = (lEmsArray,lXMSArray,lVirtualArray,lNoArray);
  REXV : AutoPriority = (lRamArray,lEmsarray,lXMSArray,lVirtualArray);
  DefHeapToUse : longint = 16384;  { 16K space for one page of matrix }
  EigRow   = 1;                     { Constants for rows in COMP matrix }
  PVarRow  = 2;
  PCVarRow = 3;
  RERow    = 4;
  IERow    = 5;
  INDRow   = 6;
  MisRow   = 7;
  SEERow   = 8;
  SLRow    = 9;
  CompRows = 9;                   { Total no of rows in COMP matrix }

procedure Initialise_matrix( MatrixName : MatrixNameType;
                             Rows,Cols  : word;
                             HeapToUse  : longint;
                             Priority   : AutoPriority;
                             Filename   : string);

function Set_val( MatrixName : MatrixNameType;
                  Row,Col    : word;
                  Value      : extended
                  ) : boolean;

function Get_val( MatrixName : MatrixNameType;
                  Row,Col    : word;
              var Value      : extended
                  ) : boolean;

function Matrix_dimensions( MatrixName : MatrixNameType;
                        var Rows,Cols  : word
                            ) : boolean;

function Type_of_array( MatrixName    : MatrixNameType;
                    var MatrixTypeStr : string
                        ) : boolean;

function Delete_matrix( MatrixName : MatrixNameType ) : boolean;

procedure Save_file( MatrixName : MatrixNameType;
                     Filename   : string);

procedure Read_file( MatrixName : MatrixNameType;
                     HeapToUse  : longint;
                     Priority   : AutoPriority;
                     Filename   : string);

procedure Free_EMS;

implementation

type
  ExtndMatrixPtr = ^ExtndMatrix;
  ExtndMatrix = object(OpArray)
                  constructor Init( Rows,Cols : word;
                                    HeapToUse : longint;
                                    Priority  : AutoPriority;
                                    Filename  : String );
                  procedure   SetVal( Row,Col : word;
                                      Value   : extended );
                  procedure   GetVal( Row,Col : word;
                                  var Value   : extended );
                  procedure   LoadA( Filename : String );
                end;
  IndexNodePtr = ^IndexNode;
  IndexNode = object(SingleListNode)
                Name     : MatrixNameType;
                MatxAddr : ExtndMatrixPtr;
                constructor Init( MatrixName : MatrixNameType;
                                  Rows,Cols  : word;
                                  HeapToUse  : longint;
                                  Priority   : AutoPriority;
                                  Filename   : string);
                procedure   Entry_name( var MatrixName : MatrixNameType);
                function    Set_val( Row,Col    : word;
                                     Value      : extended
                                     ) : boolean;
                function    Get_val( Row,Col    : word;
                                 var Value      : extended
                                     ) : boolean;
                function    Matrix_dimensions( var Rows,Cols  : word
                                               ) : boolean;
                function    Type_of_array : ArrayType;
                procedure   Save_file( Filename   : String );
                procedure   Read_file( Filename   : String );
                destructor  Done; virtual;
              end;

  MatIndexPtr = ^MatIndex;
  MatIndex = object(SingleList)
               function    Search_index( MatrixName : MatrixNameType;
                                     var Location   : IndexNodePtr
                                         ) : boolean;
               procedure   New_entry( MatrixName : MatrixNameType;
                                      Rows,Cols  : word;
                                      HeapToUse  : longint;
                                      Priority   : AutoPriority;
                                      Filename   : string);
             end;
```

```
var
  DataIndex : MatIndexPtr;
  Err : word;

{-------------------------------------------------------}
{ ExtndMatrix's method implementations                  }
{-------------------------------------------------------}

constructor ExtndMatrix.Init( Rows,Cols : word;
                              HeapToUse : longint;
                              Priority  : AutoPriority;
                              Filename  : string);

const
  ArrayOptions = lRangeCheck;

var
  Value : extended;

begin
  if (Rows = 0) and (Cols = 0)
  then begin
    if not OpArray.LoadA( Filename,
                          HeapToUse,
                          ArrayOptions,
                          Priority)
    then Fail;
  end {if then}
  else begin
    if not OpArray.Init( Rows,Cols,
                         SizeOf(extended),
                         Filename,
                         HeapToUse,
                         ArrayOptions,
                         Priority)
    then Fail;
    Value := 0;
    ClearA(Value, ExactInit);
  end; {if else}
  Err := ErrorA;
end; {ExtndMatrix.Init}

procedure ExtndMatrix.SetVal( Row,Col : word;
                              Value   : extended);

begin
  SetA(Row,Col,Value);
  Err := ErrorA;
end; {ExtndMatrix.SetVal}

procedure ExtndMatrix.GetVal( Row,Col : word;
                          var Value   : extended);

begin
  RetA(Row,Col,Value);
  Err := ErrorA;
end; {ExtndMatrix.GetVal}

procedure ExtndMatrix.LoadA( Filename : String);

const
  priority   : AutoPriority = (lXmsArray,lEmsArray,lVirtualArray,lNoArray);
  HeapToUse = 0;

begin
  OpArray.LoadA( Filename, HeapToUse, 0, priority);
  Err := ErrorA;
end; {ExtndMatrix.LoadA}

{-------------------------------------------------------}
{ IndexNode's method implementations                    }
{-------------------------------------------------------}

constructor IndexNode.Init( MatrixName : MatrixNameType;
                            Rows,Cols  : word;
                            HeapToUse  : longint;
                            Priority   : AutoPriority;
                            Filename   : string);

begin
  if not SingleListNode.Init then Fail;
  Name := MatrixName;
  new( MatxAddr,Init( Rows,Cols,HeapToUse,Priority,Filename));
end; {IndexNode.Init}

procedure IndexNode.Entry_name( var MatrixName : MatrixNameType);

begin
  MatrixName := Name;
end; {IndexNode.Entry_name}

function IndexNode.Set_val( Row,Col     : word;
                            Value       : extended ) : boolean;

begin
  MatxAddr^.SetVal( Row,Col,Value);
  Set_val := true;
end; {IndexNode.Set_val}

function IndexNode.Get_val( Row,Col     : word;
                        var Value       : extended ) : boolean;

begin
  MatxAddr^.GetVal( Row,Col,Value);
  Get_val := true;
end; {IndexNode.Get_val}

function IndexNode.Matrix_dimensions( var Rows,Cols  : word ) : boolean;

begin
  MatxAddr^.ArrayDimensions( Rows,Cols);
  Matrix_dimensions := true;
end; {IndexNode.Matrix_dimensions}

function IndexNode.Type_of_array : ArrayType;

begin
  Type_of_array := MatxAddr^.TypeOfArray;
```

```
end; {IndexNode.Type_of_array}

procedure IndexNode.Save_file( Filename   : String );

begin
  MatxAddr^.StoreA( Filename);
  Err := MatxAddr^.ErrorA;
end; {IndexNode.Save_file}

procedure IndexNode.Read_file( Filename   : String );

begin
  MatxAddr^.LoadA( Filename);
end; {IndexNode.Read_file}

destructor IndexNode.Done;

begin
  MatxAddr^.Done;
end; {IndexNode.Done}

{----------------------------------------------------------}
{ MatIndex's method implementations                        }
{----------------------------------------------------------}

function MatIndex.Search_index( MatrixName : MatrixNameType;
                            var Location   : IndexNodePtr ) : boolean;

var
  Found    : boolean;
  Name     : MatrixNameType;
  IndexPtr : IndexNodePtr;

begin
  Search_index := false;
  Found := false;
  IndexPtr := IndexNodePtr( DataIndex^.Head);
  if (IndexPtr = nil) then exit;
  while (IndexPtr <> nil) and (not Found) do
  begin
    IndexPtr^.Entry_name( Name);
    Found := (Name = MatrixName);
    If Found then Location := IndexPtr;
    IndexPtr := IndexNodePtr( DataIndex^.Next(IndexPtr));
  end; {while}
  Search_index := Found;
end; {MatIndex.Search_index}

procedure MatIndex.New_entry( MatrixName : MatrixNameType;
                              Rows,Cols  : word;
                              HeapToUse  : longint;
                              Priority   : AutoPriority;
                              Filename   : string);

var
  NewNode : IndexNodePtr;

begin
  new( NewNode, Init( MatrixName,Rows,Cols,HeapToUse,Priority,Filename));
  DataIndex^.Append( NewNode);
end; {MatIndex.New_entry}

{----------------------------------------------------------}
{ Procedures that are not methods                          }
{----------------------------------------------------------}

procedure Initialise_matrix( MatrixName : MatrixNameType;
                             Rows,Cols  : word;
                             HeapToUse  : longint;
                             Priority   : AutoPriority;
                             Filename   : string);

var
  Found    : boolean;
  IndexPtr : IndexNodePtr;

begin
  Found := DataIndex^.Search_index( MatrixName,IndexPtr);
  if Found then exit;
  DataIndex^.New_entry( MatrixName,Rows,Cols,HeapToUse,
                        Priority,Filename);
end; {Initialise_matrix}

function Set_val( MatrixName : MatrixNameType;
                  Row,Col    : word;
                  Value      : extended
                  ) : boolean;

var
  Found    : boolean;
  IndexPtr : IndexNodePtr;

begin
  Found := DataIndex^.Search_index( MatrixName,IndexPtr);
  if Found then
    Found := IndexPtr^.Set_val( Pred(Row),Pred(Col),Value);
  Set_val := Found;
end; {Set_val}

function Get_val( MatrixName : MatrixNameType;
                  Row,Col    : word;
              var Value      : extended
                  ) : boolean;

var
  Found    : boolean;
  IndexPtr : IndexNodePtr;

begin
  Found := DataIndex^.Search_index( MatrixName,IndexPtr);
  if Found then
    Found := IndexPtr^.Get_val( Pred(Row),Pred(Col),Value);
  Get_val := Found;
end; {Get_val}

function Matrix_dimensions( MatrixName : MatrixNameType;
                        var Rows,Cols  : word
                            ) : boolean;
```

```
var
  Found    : boolean;
  IndexPtr : IndexNodePtr;

begin
  Found := DataIndex^.Search_index( MatrixName,IndexPtr);
  if Found then
  begin
    Found := IndexPtr^.Matrix_dimensions( Rows,Cols);
  end; {if}
  Matrix_dimensions := Found;
end; {Matrix_dimensions}

function Type_of_array( MatrixName    : MatrixNameType;
                    var MatrixTypeStr : string
                        ) : boolean;

var
  IndexPtr   : IndexNodePtr;
  Found      : boolean;
  MatrixType : ArrayType;

begin
  Found := DataIndex^.Search_index( MatrixName,IndexPtr);
  if Found
  then begin
    MatrixType := IndexPtr^.Type_of_array;
    case MatrixType of
      lRamArray : MatrixTypeStr := 'in RAM';
      lEmsArray : MatrixTypeStr := 'in EMS';
      lVirtualArray : MatrixTypeStr := 'a Virtual array';
      lXmsArray : MatrixTypeStr := 'in XMS';
      lNoArray : MatrixTypeStr := 'not allocated';
    end; {case}
  end; {if}
  Type_of_array := Found;
end; {Type_of_array}

function Delete_matrix( MatrixName : MatrixNameType ) : boolean;

var
  Found    : boolean;
  IndexPtr : IndexNodePtr;

begin
  Found := DataIndex^.Search_index( MatrixName,IndexPtr);
  if Found then
    IndexPtr^.Done;
  DataIndex^.Delete( IndexPtr);
  Delete_matrix := Found;
end; {Delete_matrix}

procedure Save_file( MatrixName : MatrixNameType;
                     Filename   : string);

var
  Found    : boolean;
  IndexPtr : IndexNodePtr;

begin
  Found := DataIndex^.Search_index( MatrixName,IndexPtr);
  if Found then
    IndexPtr^.Save_file( Filename);
end; {Save_file}

procedure Read_file( MatrixName : MatrixNameType;
                     HeapToUse  : longint;
                     Priority   : AutoPriority;
                     Filename   : string);

begin
    DataIndex^.New_entry( MatrixName,0,0,HeapToUse,Priority,Filename);
end; {Read_file}

procedure Free_EMS;

begin
  DataIndex^.Done;
end; {Free_EMS}

begin {Unit initialisation}
  new( DataIndex, Init);
end.
```

3.7. Unit PFAUTILS

```
unit PFAUtils;

{The code contained in this unit is Copyright by T.G.Brockwell, 1989-92. All rights reserved.}

{$O+,F+}

interface

uses Overlay,Crt,DOS,PFAGlobs,Utils,Vid_util,EMSDat;

procedure Value_to_string (    ColWidth  : byte;
                               Row,Col   : word;
                           var CellValue : Str_255);

procedure Write_status (Status : StatusType);

procedure Change_current_matrix (Name : MatrixNameType);

procedure Update_time;

procedure Update_lights;

procedure Write_location;

procedure Draw_cursor;

implementation

procedure Value_to_string ( ColWidth  : byte;
                            Row,Col   : word;
                        var CellValue : Str_255);

var                                             { Obtains a value from the data }
  Value,Min,Max : extended;                   { structure, converts it to a string }
  Places        : byte;                                  { for the column width }
  Result,Expnt  : integer;
  Mantissa      : Str_255;
  Exponent      : Str_255;
  Spaces        : Str_255;

  procedure Split_extended (Value : extended;      { Breaks an extended number }
                        var Mantissa,Exponent : Str_255);  { into two strings }

  var
    ValueStr : Str_255;
    OpPos : byte;
    Sign : char;

  begin
    Str(Value,ValueStr);                            { Convert number to string }
    OpPos := Pos('E',ValueStr);                                 { Find the E }
    Sign := ValueStr[Succ(OpPos)];                  { Store sign of exponent }
    Exponent := Copy(ValueStr,(OpPos+2),4);        { Extract the exponent part }
    Mantissa := Copy(ValueStr,1,Pred(OpPos));      { Extract the mantissa part }
    Strip_leading(Exponent,['0']);                     { Remove extra zeros }
    if Length(Exponent) <> 0
    then Insert(Sign,Exponent,1);                     { Add sign to Exponent }
  end; {Split_extended}

  procedure Strip_Nsig_Zeros (var Str : Str_255); { Strips non significant }
                                                  { zeros and hanging dp from }
                                                  { strings }
  var
    Point : byte;

  begin
    Point := Pos('.',Str);
    if Point > 0
    then begin
      Strip_trailing(Str,['0']);        { Remove trailing zeros }
      Strip_trailing(Str,['.']);        { Remove decimal point }
    end {if}                                  { if hanging }
  end; {Strip_Nsig_Zeros}

begin {Value_to_string}
  if (Row <= MatrixDim.Row) and (Col <= MatrixDim.Column) then
  begin
    Min := Power(10,-(ColWidth-4));
    Max := Power(10,(Colwidth-3));
    Found := Get_val(CurrentMatrix,Row,Col,Value);             { Read value }
    If Found then
    begin
      Split_extended(Value,Mantissa,Exponent);
      case (Abs(Value) >= Min) and (Abs(Value) < Max) of
        true : begin
                 Val(Exponent,Expnt,Result);
                 if Expnt < 0
                 then Str(Value:ColWidth:(ColWidth-4),CellValue)
                 else Str(Value:ColWidth:(ColWidth-4-Expnt),CellValue);
                 Strip_Nsig_Zeros(CellValue);
               end; {true}
        false : begin
                  Val(Mantissa,Value,Result);     { Convert mantissa to number }
                  Places := ColWidth-3-(Length(Exponent));  { Calc dec places }
                  Str(Value:Places:(Places-2),Mantissa); { Mantissa to string }
                  Strip_Nsig_Zeros(Mantissa);
                  if not (Exponent = '')
                  then Insert('E',Exponent,1);
                  CellValue := Mantissa + Exponent;    { Add str to CellValue }
                end; {false}
      end; {case}
      Strip_leading(CellValue,[' ']);                 { Remove leading spaces }
      if CellValue[1] <> '-'                     { Add a space if not negative }
        then Insert(' ',CellValue,1);
      FillChar(Spaces,Succ(ColWidth-Length(CellValue)),' ');
      Spaces[0] := Char(ColWidth-Length(CellValue));
      CellValue := CellValue + Spaces;
    end {if then}
    else
    begin
      FillChar(CellValue,Succ(ColWidth),' ');
      CellValue[0] := Char(ColWidth);
    end; {if else}
    end {if then}
    else CellValue := '';
end; {Value_to_string}

procedure Write_status (Status : StatusType);      { Draws status flag in top }
                                                    { right hand corner of }
var                                                 { screen }
  Attrib     : byte;
  PrevAttrib : byte;
  X,Y        : byte;

begin
  X := WhereX;
  Y := WhereY;
  PrevAttrib := TextAttr;
  with StatusData[Status] do          { Using the relevant status record }
  begin
    Attrib := Colour*16+White;                     { Calculate byte to }
    if Blinking then Attrib := Attrib + Blink;     { set attributes }
    TextAttr := Attrib;                            { Set attribute appropriately }
    GotoXY(76,1);
    Write(Text);                                   { Write flag to screen }
  end; {with}
  GotoXY(X,Y);
  TextAttr := PrevAttrib;
end; {Write_status}

procedure Change_current_matrix(Name : MatrixNameType);       { Resets global }
                                                          { variables for a new }
var                                                       { matrix and displays it }
  Ok : boolean;

begin
  CurrentMatrix := Name;
  CursorPos.Row := 1;
  CursorPos.Column := 1;
  Origin.Row := 1;
  Origin.Column := 1;
  OldCurPos.Row := 1;
  OldCurPos.Column := 1;
  Ok := Matrix_dimensions(Name,MatrixDim.Row,MatrixDim.Column);
  RedrawScreen := true;
end; {Change_current_matrix}

procedure Update_time;           { Writes date and time to bottom line }
                                 { of screen }
var
  DateString : Str_30;
  OldAttr    : byte;
  X,Y        : byte;

begin
  OldAttr := TextAttr;
  X := WhereX; Y := WhereY;
  GotoXY(1,25);
  TextAttr := Get_attribute(1,25);
  Get_date_time(DateString);        { Get date & time in string 30 chars long }
  Write(DateString);
  TextAttr := OldAttr;
  GotoXY(X,Y);
end; {Update_time}

procedure Update_lights;                             { Writes lock lights to screen }

const
  On = Green*16+White;           { Colour combination bytes}
  Off = Black*16+LightGray;
  CapsStr : array [boolean] of Str_5 = ('     ',' CAPS');
  NumStr  : array [boolean] of Str_5 = ('     ',' NUM ');
  ScrStr  : array [boolean] of Str_6 = ('      ','SCROLL');
  Attrib  : array [boolean] of byte = (Off,On);

var
  OldAttr : byte;
  X,Y     : byte;

begin
  Kbd_light_status;
  OldAttr := TextAttr;
  X := WhereX; Y := WhereY;
  GotoXY(64,25);
  TextAttr := Attrib[CapsOn];
  Write(CapsStr[CapsOn]);
  TextAttr := Attrib[NumOn];
  Write(NumStr[NumOn]);
  TextAttr := Attrib[ScrollOn];
  Write(ScrStr[ScrollOn]);
  TextAttr := OldAttr;
  GotoXY(X,Y);
end; {Update_lights}

procedure Write_location;      { Writes cursor location on top line of screen }

var
  CodeString : Str_255;     { String to hold contents of screen line }
  Value      : extended;    { Cell value from data structure }
  Row,Col    : word;        { Cell coordinates }

begin                                                        { Calculate cell }
  Row := Origin.Row + Pred(CursorPos.Row);                      { coordinates }
  Col := Origin.Column + Pred(CursorPos.Column);       { Clear contents of string }
  FillChar(CodeString,Succ(75),' ');
  CodeString[0] := Chr(75);
  GotoXY(1,1);
  TextAttr := Get_attribute(1,1);                        { Erase old line contents }
  Write(CodeString);                                  { Write numbers or letters ? }
  if headings then                                                 { Get letters }
    Convert_to_base_26(Col, CodeString)
  else
    begin                                               { Convert number to string }
      Str(Col,CodeString);                                { Add a comma separator }
      CodeString := CodeString + ',';
    end; {if else}
  Strip(CodeString,[' ']);                        { Remove spaces from CodeString }
  Found := Get_val(CurrentMatrix,Row,Col,Value);                      { Get value }
  if not found then Value := 0;
  GotoXY(1,1);                                                     { Write string }
  Write(CodeString,Row,': ',Value);
```

```
end; {Write_location}

procedure Draw_cursor;       { Draws a cell cursor and row and column cursors }

var
  X,Y : byte; { 25 X 80 coords }

begin
  X := (OldCurPos.Column - 1) * ColWidth + 6;              { Erase old cursor }
  Y := OldCurPos.Row + 4;
  Colour_box(X,4,(X + Pred(ColWidth)),4,White,Blue,False);
  Colour_box(X,Y,(X + Pred(ColWidth)),Y,White,LightGray,False);
  Colour_box(1,Y,5,Y,White,Blue,False);
  X := (CursorPos.Column - 1) * ColWidth + 6;              { Draw new cursor }
  Y := CursorPos.Row + 4;
  Colour_box(X,4,(X + Pred(ColWidth)),4,White,Cyan,False);
  Colour_box(X,Y,(X + Pred(ColWidth)),Y,White,Cyan,False);
  Colour_Box(1,Y,5,Y,White,Cyan,False);
  OldCurPos := CursorPos;
  Write_location;                                          { Update location info }
end; {Draw_cursor}

end.
```

3.8. Unit EPAIMPRT

```
unit EPAimprt;

{The code contained in this unit is Copyright by T.G.Brockwell, 1989-92. All rights reserved.}


{$O+,F+}

interface

uses DOS,PFAGlobs,Utils,EMSDat;

procedure EPA_import (Filename : PathStr;
                      Name     : MatrixNameType;
                  var Err      : boolean;
                  var ErrMsg   : Str_80);

implementation

procedure EPA_import (Filename : PathStr;
                      Name     : MatrixNameType;
                  var Err      : boolean;
                  var ErrMsg   : Str_80);
                                          { Imports an EPA data format }
                                          { file from VG mass spec data }
                                          { system. }

type
  BufferType = array [1..82] of byte;

var
  { Program control variables }
  Code         : integer;             { String/number conversion code }
  ScanNo       : integer;             { Number of current scan in file }
  ScaleFactor  : extended;            { Constant to scale intensity figures }
  LineNo       : longint;             { Line number of data in scan record }
  InFile       : file of BufferType;  { Input file }
  Buffer       : BufferType;          { Contains one line of file }
  BufferStr    : string[80];          { Contains one line of file - CR & LF }
  ReadHeader   : boolean;             { Flag to read header record }
  ScanFinished : boolean;             { Flags the end of each set of scan data }
  Ok           : boolean;
  Int          : boolean;
  AllowZero    : boolean;
  Found        : boolean;
  Max          : longint;
  Min          : longint;
  Rows,Cols    : word;
  Row,Col      : word;
  Value        : extended;

  { EPA file header data fields & run info variables }
  OrigFileName : string[12];   { Name of original data file. }
  Date         : string[8];    { Date of data acquisition. }
  RunStartTime : string[5];    { Start time of run. }
  SampleId     : string[64];   { Sample identification. }
  InstName     : string[6];    { Instrument Name. }
  RunConds     : string[64];   { Run Conditions. }
  SecsPerScan  : real;         { Seconds per scan for this file. }
  Analyst      : string[8];    { Analyst name. }
  SubmittedBy  : string[8];    { Submitted by. }
  AccountNo    : string[8];    { Account number. }
  Formula      : string[20];   { Formula. }
  MinMass      : word;         { Lowest mass possible. }
  MaxMass      : word;         { Highest mass possible. }
  NoOfScans    : word;         { Number of scans in file. }

  {--------------------}

  procedure Error (ErrNum : byte);
                                  { Passes appropriate error messages }
                                  { to Display_error_message. }
  type
    MsgType = array [1..15] of Str_80; { Array to contain messages }

  const                                                      { Message list }
    ErrorMsg : MsgType = ('Scan header not found at start of file',
                     'Scan number out of sequence - file is corrupted',
                 'Mass/Intensity data list too long ( > 77 lines)',
             'LineCount and no. of lines in Mass/Intensity list do not match',
         'String to number conversion error for Mass value',
      'String to number conversion error for Intensity value',
    'Unable to write to data matrix',
    'String to number conversion error for LineCount in Mass/Intensity list',
    'Conversion error for scan number',
    'Conversion error for largest peak intensity',
    'Scan header does not match File header - file is corrupted',
    'Conversion error for Seconds per scan',
    'Conversion error for lowest Mass',
    'Error reading file',
    'File contains incomplete lines - file is corrupted');

  begin
    Err := true;
    ErrMsg := ErrorMsg[ErrNum];
    Close( InFile);
    Exit;
  end; {Error}

  {--------------------}

  procedure Read_line;

  var
    I : byte;

  begin
    Read(InFile,Buffer);
    Code := IOResult;
    if (Code <> 0)
    then begin
      Error(14);
      exit;
    end; {if}
  for I := 1 to 80 do
  begin
    BufferStr[I] := Chr( Buffer[I]);
  end; {for}
  BufferStr[0] := Chr(80);
end; {Read_line}

{--------------------}

procedure Detatch_line_number;
                        { Removes the line number from the  }
                        { end of the buffer. }
var
  LineStr         : Str_2;   { String for LineNo }

begin
  LineStr := Copy(BufferStr,Pred(Length(BufferStr)),2);  { Copy line number }
  Int := true; AllowZero := true; Min := 0; Max := 77;
  Ok := String_to_number(LineStr,LineNo,Min,Max,Code,Int,AllowZero);
  if (Code <> 0) or not Ok then Error(8);
  Delete(BufferStr,Pred(Length(BufferStr)),2);           { Delete line number }
end; {Detatch_line_number}

{--------------------}

procedure Dissect_scan_header;
                             { Breaks scan header line into its components }
var
  tOrigFileName : Str_12;  { Original data file name }
  ScanNoStr     : Str_5;   { String containing scan number }
  tDate         : Str_8;   { Date of run }
  tRunStartTime : Str_5;   { Time at start of run }
  TimeOfScan    : Str_6;   { Time after start time at end of run }
  MassLgePeak   : Str_4;   { Nominal mass of the largest peak }
  LgeIntensity  : Str_9;   { Intensity of the largest peak }
  iLgeIntensity : longint; { Integer intensity of the largest peak }
  Tic           : Str_10;  { Reconstructed ion current }

begin
  tOrigFileName := Copy(BufferStr,1,12);         { Read the relevant sections }
  ScanNoStr := Copy(BufferStr,14,5);            { of the scan header into the }
  tDate := Copy(BufferStr,21,8);                         { appropriate string }
  tRunStartTime := Copy(BufferStr,30,5);
  TimeOfScan := Copy(BufferStr,40,6);
  MassLgePeak := Copy(BufferStr,52,4);
  LgeIntensity := Copy(BufferStr,57,9);
  Tic := Copy(BufferStr,71,10);
  case ReadHeader of
   true : if (tOrigFileName <> OrigFileName)      { Compare the file header }
            or (tDate <> Date)                    { to the scan header. }
            or (tRunStartTime <> RunStartTime)
          then begin
            Error(11);
            Exit;
          end; {true if then}
   false :  begin
              OrigFileName := tOrigFileName;          { Set the values for }
              Date := tDate;                          { the file header. }
              RunStartTime := tRunStartTime;
            end; {false}
  end; {case}
  Int := true; AllowZero := true; Min := 0;   { Convert scan no to integer }
  Ok := String_to_number(ScanNoStr,ScanNo,Min,MaxInt,Code,Int,AllowZero);
  if (Code <> 0) then Error(9);
  if (LgeIntensity = '         ')
    then ScaleFactor := 0
    else
    begin
      Max := 999999999;                    { Convert intensity to integer }
      Ok := String_to_number(
            LgeIntensity,iLgeIntensity,Min,Max,Code,Int,AllowZero);
      if (Code <> 0) then Error(10);
    end; {if else}
  ScaleFactor := iLgeIntensity/999;           { Calculate scaling factor }
end; {Dissect_scan_header}

{--------------------}

procedure Find_max_scan;

var
  LinesInFile : longint;
  BufferSize  : longint;
  SizeOfFile  : longint;
  CurrentPos  : longint;
  TargFile    : file of byte;

begin
  Assign( TargFile, Filename);
  Reset( TargFile);
  CurrentPos := FilePos( InFile);
  BufferSize := longint(SizeOf(Buffer));
  SizeOfFile := FileSize( TargFile);
  LinesInFile := SizeOfFile mod BufferSize;
  Close( TargFile);
  if LinesInFile <> 0
  then begin
    Error(15);
    Exit;
  end; {if}
  LinesInFile := SizeOfFile div BufferSize;
  Seek( Infile, Pred(LinesInFile));
  Read_line;
  if Err then Exit;
  Detatch_line_number;
  if Err then Exit;
  Seek( Infile, (Pred(LinesInFile)-LineNo));
  Read_line;
  if Err then Exit;
  Dissect_scan_header;
  if Err then Exit;
  NoOfScans := ScanNo;
  Seek( InFile, CurrentPos);
end; {Find_max_scan}

{--------------------}

procedure Read_file_header;
```

```
                              { Reads additional data from the remaining }
                              { three lines of the file header. }
  {--------------------}

  procedure Second_header_line;

  begin
    Read_line;
    if Err then Exit;
    SampleId := Copy(BufferStr,1,64);
    InstName := Copy(BufferStr,74,6);
  end; {Second_header_line}

  {--------------------}

  procedure Third_header_line;

  var
    SecScanStr : string[6];
    Min        : extended;

  begin
    Read_line;
    if Err then Exit;
    RunConds := Copy(BufferStr,1,64);
    SecScanStr := Copy(BufferStr,75,6);
    Min := 0; Int := false; AllowZero := false;
    Ok := String_to_number(
          SecScanStr,SecsPerScan,Min,MaxReal,Code,Int,AllowZero);
    if (Code <> 0) or not Ok
    then begin
      Error(12);
      exit;
    end; {if}
  end; {Third_header_line}

  {--------------------}

  procedure Fourth_header_line;

  var
    StripStr    : Str_255;
    MinMassStr  : string[3];
    MaxMassStr  : string[3];

  begin
    Read_line;
    if Err then Exit;
    Analyst := Copy(BufferStr,5,8);
    SubmittedBy := Copy(BufferStr,19,8);
    AccountNo := Copy(BufferStr,33,8);
    Formula := Copy(BufferStr,47,20);
    StripStr := Copy(BufferStr,75,3);
    Strip( StripStr,[' ']);
    MinMassStr := StripStr;
    StripStr := Copy(BufferStr,78,3);
    Strip( StripStr, [' ']);
    MaxMassStr := StripStr;
    Min := 0; Int := true; AllowZero := true;
    Ok := String_to_number(
          MinMassStr,MinMass,Min,MaxWord,Code,Int,AllowZero);
    if (Code <> 0) or not Ok
    then begin
      Error(13);
      exit;
    end; {if}
    Ok := String_to_number(
          MaxMassStr,MaxMass,Min,MaxWord,Code,Int,AllowZero);
    if (Code <> 0) or not Ok
    then begin
      Error(14);
      exit;
    end; {if}
  end; {Fourth_header_line}

  {--------------------}

begin {Read_file_header}
  Second_header_line;
  if Err then Exit;
  Third_header_line;
  if Err then Exit;
  Fourth_header_line;
  if Err then Exit;
  ReadHeader := true;
end; {Read_file_header}

{--------------------}

procedure Read_scan_data;
                          { Reads the data from each scan stored in the file }
var
  PrevMass          : longint; { Last mass number read }
  PrevIntensity     : longint; { Corresponding or sum intensity }
  PrevScanNo        : integer; { Last scan number read, for error checking }
  LineCount         : byte;    { Number of lines of mass/intensity data read }


  {--------------------}

  procedure Dissect_scan_data;
                          { Splits the file lines into data pairs and }
                          { writes them to the data structure }
  var
    Found          : boolean; { Success flag for data structure writes }
    MassStr        : Str_3;   { Storage for mass number }
    IntenStr       : Str_3;   { Storage for intensity }
    DataPair       : Str_6;   { Storage for mass/intensity pair }
    Mass           : longint; { Storage for mass number }
    Intensity      : longint; { Storage for peak intensity }

    {--------------------}

    procedure Detatch_MI_pair;
                                  { Splits a DataPair from the buffer }
    begin
      DataPair := Copy(BufferStr,1,6);           { Copy first six chars }
      Delete(BufferStr,1,6);              { Remove six chars from buffer }
    end; {Detatch_MI_pair}

    {--------------------}

    procedure Split_MI_pair;
                                  { Splits a DataPair into two strings }
    begin
      MassStr := Copy(DataPair,1,3);
      IntenStr := Copy(DataPair,4,3);
    end; {Split_MI_pair}

    {--------------------}

    procedure Convert_MI_pair;
                                  { Converts data pair to numbers }
                                  { and tests for end of scan }
    var
      eIntensity : extended; { Intensity of peak as real variable }

    begin
      Max := 999; Min := 0; Int := true; AllowZero := true;
      Ok := String_to_number(MassStr,Mass,Min,Max,Code,Int,AllowZero);
      if not (Code = 0) or not Ok
      then begin
        Error(5);
        Exit;
      end; {if}
      Ok := String_to_number(IntenStr,Intensity,Min,Max,Code,Int,AllowZero);
      if not (Code = 0) or not Ok
      then begin
        Error(6);
        Exit;
      end; {if}
      if (Mass = 0) and (Intensity = 0) then    { Test for end of data list }
        begin
          ScanFinished := true;                       { Flag end of data }
          BufferStr := '';                             { Clear BufferStr }
        end {if then}
      else
        begin
          eIntensity := Intensity * ScaleFactor; { Calculate true intensity }
          Intensity := Round(eIntensity);
        end {if else}
    end; {Convert_MI_pair}

    procedure Write_point_to_matrix;
                                  { Enters values into the data structure }
                                  { Sums successive intensities of the }
                                  { same unit mass }
    begin
      if PrevMass = Mass then
        PrevIntensity := PrevIntensity + Intensity
      else
        begin
          PrevIntensity := Intensity;
          PrevMass := Mass;
        end; {if else}
      Found := Set_val(Name,PrevMass,ScanNo,PrevIntensity);  { Write value }
      if not Found
      then begin
        Error(7);
        Exit;
      end; {if}
    end; {Write_point_to_matrix}

    {--------------------}

  begin {Dissect_scan_data}
    Detatch_line_number;
    if Err then Exit;
    if (Succ(LineCount) <> LineNo)
    then begin
      Error(4);
      Exit;
    end; {if}
    while (Length(BufferStr) <> 0) do       { Loop till end of m/i data }
    begin
      Detatch_MI_pair;
      Split_MI_pair;
      Convert_MI_pair;
      if not ScanFinished
      then Write_point_to_matrix;
      if Err then Exit;
    end; {while}
  end; {Dissect_scan_data}

  {--------------------}

begin {Read_scan_data}
  while not Eof(InFile) do
  begin
    PrevScanNo := ScanNo;
    Read_line;
    if Err then Exit;
    Dissect_scan_header;
    if Err then Exit;
    if (ScanNo = 0) or not (ScanNo > PrevScanNo)
    then begin
      Error(2);          { Error trap for non-incremental scan numbers }
      Exit;
    end; {if}
    LineCount := 0;      { Variable initialization for reading MI data }
    PrevMass := 0;
    PrevIntensity := 0;
    ScanFinished := false;
    while not ScanFinished do          { Read all mass/intensity data }
    begin
      Read_line;
      if Err then Exit;
      Dissect_scan_data;
      if Err then Exit;
      Inc(LineCount);                              { Count lines read }
    end; {while}
    if LineNo > 77
    then begin
      Error(3);
      Exit;
      end; {if}
      if LineCount <> LineNo
      then begin
        Error(4);  { Compare lines read & NoOfLines }
        Exit;
      end; {if}
    end; {while}
  end; {Read_scan_data}

  {--------------------}

begin {EPA_import}
  ReadHeader := false;
  Assign( InFile,Filename);
  Reset( InFile);
  Find_max_scan;
  if Err then Exit;
  Read_line;
  if Err then Exit;
  Dissect_scan_header;
  if Err then Exit;
  if ScanNo = 0                                      { Check for file header }
    then Read_file_header
    else Error(1);
  if Err then Exit;
  Initialise_matrix( DATA,MaxMass,NoOfScans,DefHeapToUse,EXVN,'DATARRAY.$$$');
  Read_scan_data;
  if Err then Exit;
  Close( InFile);
end; {EPA_import}

end. {EPAimprt}
```

3.9. Unit TARGTEST

```
unit TargTest;

{The code contained in this unit is Copyright by T.G.Brockwell, 1989-92. All rights reserved.}

interface

uses PFAGlobs,Utils,EMSDat;

procedure Calculate_TKON;

procedure Test_vector;

procedure ITTInit;

procedure ITT( var Finished : boolean);

implementation

const {typed}
  Rows : word = 0;
  Cols : word = 0;

const
  TestCol  = 1;  { Column containing test vector }
  PredCol  = 2;  { Column containing predicted vector }
  ErrCol   = 3;  { Column containing error information }
  ErrRow   = 1;  { Estimated error in test vector }
  AetRow   = 2;  { Aparrent error in test vector }
  RepRow   = 3;  { Real error in predicted vector }
  RetRow   = 4;  { Real error in test vector }
  SpoilRow = 5;  { Spoil function }
  ReliRow  = 6;  { Reliability estimate }
  FRow     = 7;  { F-test for vector }
  IterRow  = 8;  { Iteration number for ITT }
  DiffRow  = 9;  { Difference term row }
  CorrRow  = 10; { Correlation coefficient for test & pred vectors }

procedure Matrix_init;

var
  Found : boolean;
  Row,Col : word;

begin
  Found := Delete_matrix( TVEC);
  Initialise_matrix( TVEC,NoFacs,1,DefHeapToUse,EXVN,'TVECARAY.$$$');
  Found := Matrix_dimensions( TTST,Row,Col);
  if (Row <> Rows) or (Col <> 3)
  then begin
    Write(Bell);
    Exit;
  end; {if}
end; {Matrix_init}

procedure Calculate_TKON;

var
  I,K          : word;
  Found        : boolean;
  Tmp          : word;
  Eig,Kon,
  Score        : extended;

  procedure Leave;

  begin
    Write(Bell);
    Exit;
  end; {Leave}

begin
  Found := Matrix_dimensions( COMP,Tmp,Cols); { Uses the composite matrix }
  if not Found then Leave;                   { to determine the no.of cols. }
  Found := Matrix_dimensions( SCOR,Rows,Tmp); { Uses the scores matrix to }
  if not Found then Leave;                    { find the no. of rows. }
  Found := Delete_matrix( TKON);             { Delete any existing constant. }
  Initialise_matrix( TKON,NoFacs,Rows,DefHeapToUse,EXVN,'TKONARAY.$$$');
  Matrix_init;
  for I := 1 to NoFacs do
  begin
    Found := Get_val( COMP,EigRow,I,Eig);
    Eig := 1/Eig;
    for K := 1 to Rows do
    begin
      Found := Get_val( SCOR,K,I,Score);
      Kon := Eig * Score;
      Found := Set_val( TKON,I,K,Kon);
    end; {for K}
  end; {for I}
end; {Calculate_TKON}

procedure Calculate_TVEC;

var
  Found          : boolean;
  I,K            : word;
  Kon,Test,Trnv  : extended;

begin
  for I := 1 to NoFacs do
  begin
    Trnv := 0;
    for K := 1 to Rows do
    begin
      Found := Get_val( TKON,I,K,Kon);
      Found := Get_val( TTST,K,1,Test);
      Trnv :=  Trnv + Kon * Test;
    end; {for K}
    Found := Set_val( TVEC,I,1,Trnv);
  end; {for I}
end; {Calculate_TVEC}
```

```
procedure Calculate_Pred_vector;

var
  Found      : boolean;
  I,K        : word;
  Score,
  Trnv,Pred : extended;

begin
  for I := 1 to Rows do
  begin
    Pred := 0;
    for K := 1 to NoFacs do
    begin
      Found := Get_val( SCOR,I,K,Score);
      Found := Get_val( TVEC,K,1,Trnv);
      Pred := Pred + Score * Trnv;
    end; {for K}
    Found := Set_val( TTST,I,2,Pred);
  end; {for I}
end; {Calculate_Pred_vector}

type
  TestFocusType = (Row,Column);

procedure F_test_pred_vector( TestFocus : TestFocusType;
                              Rows,Cols,NoFacs : word;
                              var SigLevel : real);


var
  J           : word;     { Loop control variable for weight calculation. }
  S           : word;     { Minimum of Rows and Cols. }
  R,C         : integer;  { Integer values of Rows and Cols for calculation. }
  F           : integer;  { Focus dependent variable. }
  B           : integer;  { Number of missing points in vector. Set to 0. }
  K1,K2       : integer;  { Degrees of Freedom for F-test. }
  Fvalue      : real;     { F ratio for F-test. }
  REVPool     : real;     { Reduced eigenvalue and error eigenvalue pool. }
  VarT        : real;     { Variance between test and predicted vectors. }
  Weight      : real;     { Eigenvalue pool weighting value. }
  Eig         : extended; { Intermediate storage of eigenvalues. }
  Tval,Pval   : extended; { Intermediate storage of vector values. }
  Trval       : extended; { Intrmd strge of transformation vector values. }
  TotVar      : extended; { Total variance in data matrix. }
  PVar        : extended; { Percentage variance of eigenvector. }
  Sum         : real;     { Sum of error eigenvalues. }
  TSq         : real;     { Sum of squares of transformation vector. }
  Prob        : real;     { Area in tail of F distribution. }
  Found       : boolean;


begin
  Fvalue := 0; SigLevel := 0;    { Variable initialization. }
  Weight := 0; Sum := 0;
  R := Rows; C := Cols; B := 0;
  case (Rows > Cols) of  { Sets the maximum number of factors possible }
    true  : S := Cols;   { for the data set. }
    false : S := Rows;
  end; {case}
  case TestFocus of          { Sets the value of the focus dependent variable. }
    Row : F := Cols;         { If the Column vectors are being tested then the }
    Column : F := Rows;      { vector will contain R points and vice versa. }
  end; {case}
  K1 := (F - NoFacs - B); K2 := (S-NoFacs);                    { D of F. }
  Found := Get_val(COMP,EigRow,1,Eig);
  Found := Get_val(COMP,PVarRow,1,PVar);
  TotVar := (Eig/PVar)*100;                     { Calculate total variance. }
  for J := 1 to NoFacs do
  begin
    Found := Get_val(COMP,EigRow,J,Eig);  { Sum non-error eigenvalues. }
    Sum := Sum + Eig;
  end; {for j}
  Sum := TotVar - Sum;                          { Calculate error eigenvalues. }
  for J := Succ(NoFacs) to S do            { Pool weighting calculation loop. }
  begin
    Weight := Weight + (R - J + 1) * (C - J + 1); { Sum weight distribution. }
  end; {for J}
  REVPool := Sum / Weight;                 { Calculate pool reduced eigenvalue. }
  Sum := 0; TSq := 0;                      { Variable initialization. }
  for J := 1 to F do
  begin
    Found := Get_val( TTST,J,TestCol,Tval); { Sum the squares of the }
    Found := Get_val( TTST,J,PredCol,Pval); { differences between the test }
    Sum := Sum + Sqr((Pval - Tval));        { and predicted vectors. }
  end; {for J}
  for J := 1 to NoFacs do
  begin
    Found := Get_val( TVEC,J,1,Trval); { Sum the squares of the }
    TSq := TSq + Sqr(Trval);           { transformation vector. }
  end; {for J}
  VarT := (F * Sum)/(K1 * TSq); { Calculate vector variance. }
  VarT := VarT / ((R-NoFacs+1) * (C-NoFacs+1)); { Weight for error distrib. }
  Fvalue := VarT/REVPool;                         { Calculate F value. }
  Prob := F_prob( Fvalue,K1,K2); { Calculate area in tail of F dist. }
  SigLevel := (1-Prob) * 100;    { Calculate 1 tailed significance test. }
end; {F_test_pred_vector}

procedure Errors_in_test_vector;


var
  Ret,Rep,Aet,Spoil,Reli : real;
  SigLevel               : real;
  Found                  : boolean;
  I                      : word;
  SumOfSq,Sum            : extended;
  Test,Pred,Trnv,Re      : extended;

begin
  Found := Set_val( TTST,ErrRow,ErrCol,Vecterr);
  { Calculate the Apparent error in the test vector }
  SumOfSq := 0;
  for I := 1 to Rows do
  begin
    Found := Get_val( TTST,I,2,Pred);
    Found := Get_val( TTST,I,1,Test);
    SumOfSq := SumOfSq + Sqr((Pred - Test));
  end; {for I}
```

```
 Aet := Sqrt((SumOfSq/Rows));
 Found := Set_val( TTST,AetRow,ErrCol,Aet);
 { Calculate the Real error in the predicted vector }
 if NoFacs = Cols
 then begin
   Rep := 0;
   Exit;
 end; {if}
 SumOfSq := 0;
 for I := 1 to NoFacs do
 begin
   Found := Get_val( TVEC,I,1,Trnv);
   SumOfSq := SumOfSq + Sqr(Trnv);
 end; {for I}
 Found := Get_val( COMP,ReRow,NoFacs,Re);
 if not Found then Exit;
 Rep := Re * Sqrt(SumOfSq);
 Found := Set_val( TTST,RepRow,ErrCol,Rep);
 if Rep = 0 then Exit;
 { Calculate the Real error in the test vector }
 Sum := Sqr(Aet) - Sqr(Rep);
 if Sum < 0 then Exit;
 Ret := Sqrt(Sum);
 Found := Set_val( TTST,RetRow,ErrCol,Ret);
 { Calculate the Spoil function }
 Spoil := Ret/Rep;
 Found := Set_val( TTST,SpoilRow,ErrCol,Spoil);
 { Calculate the Reliability function }
 Reli := 100 * Sqrt((1-(Sqr(Ret) - Sqr(Vecterr))/Sqr(Aet)));
 Found := Set_val( TTST,ReliRow,ErrCol,Reli);
 F_test_pred_vector( Column,Rows,Cols,NoFacs,SigLevel);
 Found := Set_val( TTST,FRow,ErrCol,SigLevel);
end; {Errors_in_test_vector}

procedure Test_vector;

begin
  Calculate_TVEC;
  Calculate_pred_vector;
  Errors_in_test_vector;
  Refresh := true;       { Set flag to update matrix display }
end; {Test_vector}

{*************************************************************************}
{ Iterative target testing routines beyond this point }
{*************************************************************************}

const
  DifmCols = 4;          { No of columns in DIFM matrix. }
  d2Dif    = 4;          { Column of DIFM with 2nd differential }
  dOldDif : word = 3;    { Column of DIFM with last differential }
  dNewDif : word = 2;    { Column of DIFM with new differential }
                         { These values are updated with each iteration }

var
  VLen : word; { Vector length of the test vector }

procedure Copy_col_vector( SourceName : MatrixNameType;
                           SourceCol  : word;
                           DestName   : MatrixNameType;
                           DestCol    : word;
                           VecLen     : word);

var
  I    : word;      { Loop counter }
  Var1 : extended;  { Temporary variable }

begin
  for I := 1 to VecLen do
  begin
    Found := Get_val( SourceName,I,SourceCol,Var1);
    Found := Set_val( DestName,I,DestCol,Var1);
  end; {for}
end; {Copy_col_vector}

procedure Vector_differential( OrigMatx : MatrixNameType;
                               OrigCol  : word;
                               NewMatx  : MatrixNameType;
                               NewCol   : word;
                               DestMatx : MatrixNameType;
                               DestCol  : word;
                               VecLen   : word);

var
  I              : word;      { Loop control }
  Var1,Var2,dVar : extended;  { Temporary variables }
  Found          : boolean;

begin
  for I := 1 to VecLen do
  begin
    Found := Get_val( OrigMatx,I,OrigCol,Var1);
    Found := Get_Val( NewMatx,I,NewCol,Var2);
    dVar := Var2 - Var1;
    Found := Set_val( DestMatx,I,DestCol,dVar);
  end; {for I}
end; {Vector_differential}

procedure Vector_Mean_and_sd( Matx : MatrixNameType;
                              Column : word;
                              VecLen : word;
                          var Mean   : real;
                          var SD     : real);


var
  Found              : boolean;
  I                  : word;
  Value,Sum,SumSq    : extended;

begin                                             { Initialise variables. }
  Sum := 0;  SumSq := 0;
  for I := 1 to VecLen do
  begin
    Found := Get_val(Matx,I,Column,Value);        { Read entry. }
    Sum := Sum + Value;                           { Total the entries. }
  end; {for I}
  Mean := Sum / VecLen;                           { Calculate mean. }
  for I := 1 to VecLen do
  begin
    Found := Get_val(Matx,I,Column,Value);      { Read entry. }
    Value := Value - Mean;                 { Calculate the difference from the mean }
    SumSq := SumSq + Value * Value;    { Sum the squares of the differences. }
  end; {for I}
  Sd := sqrt(SumSq / (VecLen - 1));              { Calculate the Sd. }
end; {Vector_mean_and_sd}


procedure ITTInit;

var
  Col        : word;      { Columns in target test matrix. }
  Var1,Var2 : word;      { Temporary variables. }
  Iter       : extended; { Iteration no. }

begin
  Found := Matrix_dimensions( TTST,VLen,Col); { Find length of test vector. }
  Found := Matrix_dimensions( DIFM,Var1,Var2); { Check for existing matrix. }
  if not Found
  then Initialise_matrix( DIFM,VLen,DifmCols,DefHeapToUse,EXVN,'DIFMARAY.$$$');
  Found := Get_val( TTST,IterRow,ErrCol,Iter);
  if (Iter = 0) then Copy_col_vector( TTST,TestCol,DIFM,TestCol,VLen);
end; {ITTInit}

procedure ITT( var Finished : boolean);

var
  Diff,OldDiff  : extended; { Difference between the two vectors. }
  CorrCo        : extended; { Correlation coefficient }
  Iter          : extended; { Iteration number }
  Unfinished    : boolean;  { Flag for end of iteration. }

  procedure Normalize_vector;

  var                          { Sets the largest value in the vector to }
    I : word;                  { unity and scales the vector accordingly. }
    Max : extended;            { Should be passed an all positive vector. }
    Tval : extended;

  begin
    Max := 0;
    for I := 1 to VLen do
    begin
      Found := Get_val( TTST,I,TestCol,Tval);
      if (Tval > Max) then Max := Tval;        { Finds largest +ve value. }
    end; {for}
    for I := 1 to VLen do
    begin
      Found := Get_val( TTST,I,TestCol,Tval);
      Tval := Tval / Max;                      { Normalizes vector. }
      Found := Set_val( TTST,I,TestCol,Tval);
    end; {for}
  end; {Normalize_vector}

  procedure Zero_below_threshold( Threshold : real);

  var                       { Sets all values below a threshold value }
   I : word;                { to zero. }
   Tval : extended;         { Should be passed an all positive vector as }
                            { all negative values will be zeroed. }
  begin
    for I := 1 to VLen do
    begin
      Found := Get_val( TTST,I,TestCol,Tval);
      if (Tval <= Threshold) then Tval := 0;  { Set value to zero if below }
      Found := Set_val( TTST,I,TestCol,Tval); { the threshold. }
    end; {for}
  end; {Zero_below_threshold}

  procedure Zero_negative_values;

  var
    I : word;
    Tval : extended;

  begin
    for I := 1 to VLen do
    begin
      Found := Get_val( TTST,I,TestCol,Tval);
      if (Tval < 0) then Tval := 0;            { Remove negatives. }
      Found := Set_val( TTST,I,TestCol,Tval);
    end; {for I}
  end; {Zero_negative_values}

  procedure Calc_abs_diff;

  var
    I          : word;      { Loop counter }
    Pval,Tval : extended; { Temporary storage of vector values. }

  begin
    Diff := 0;
    for I := 1 to VLen do      { Sum the differences between the test and }
    begin                                        { predicted vectors. }
      Found := Get_val( TTST,I,PredCol,Pval);
      Found := Get_val( TTST,I,TestCol,Tval);
      if (Pval >= 0) then Diff:= Diff + abs(Pval - Tval);
    end; {for I}
  end; {Calc_abs_diff}

  procedure Calc_correlation;

  var
    I           : word;      { Loop control variable }
    Pval,Tval   : extended; { Temporary storage of vector values. }
    SumX,SumY   : extended; { Sums of each vector }
    SumXY       : extended; { Sum product of vectors }
    SumX2,SumY2 : extended; { Sums of squares of each vector }
    Numer,Denom : extended; { Temporary variables }

  begin
    SumX := 0; SumY := 0; SumXY := 0;
    SumX2 := 0; SumY2 := 0;
    for I := 1 to VLen do
    begin
      Found := Get_val( TTST,I,PredCol,Pval);
      Found := Get_val( TTST,I,TestCol,Tval);
    SumX := SumX + Tval;
    SumY := SumY + Pval;
    SumXY := SumXY + (Tval * Pval);
    SumX2 := SumX2 + Sqr(Tval);
    SumY2 := SumY2 + Sqr(Pval);
  end; {for I}
  Numer := ((VLen * SumXY) - (SumX * SumY));
  Denom := Sqrt(((Vlen * SumX2)-Sqr(SumX))*((Vlen * SumY2)-(Sqr(SumY))));
  if (Denom <> 0)
    then CorrCo := Numer/Denom
    else CorrCo := 0;
end; {Calc_correlation}

procedure Update_error_vector;

begin
  Found := Get_val( TTST,IterRow,ErrCol,Iter);
  Iter := Iter + 1;
  Found := Set_val( TTST,IterRow,ErrCol,Iter);
  Found := Set_val( TTST,DiffRow,ErrCol,Diff);
  Found := Set_val( TTST,CorrRow,ErrCol,CorrCo);
end; {Update_error_vector}

procedure Slope_prediction;

var
  Found           : boolean;
  Mean,SD         : real;
  I               : word;      { Loop control }
  Var1,Var2,Var3  : extended;
  Magn,Thresh     : extended;

begin
  if (Iter > 1) then
  begin
    Vector_differential(DIFM,dOldDif,DIFM,dNewDif,DIFM,d2Dif,VLen);
    Vector_mean_and_sd( DIFM,d2Dif,VLen,Mean,SD);
    Thresh := SD * 10;                  { Significance threshold }
    for I := 1 to VLen do
    begin
      Found := Get_val( DIFM,I,d2Dif,Var1);
      Magn := abs(Var1-Mean);              { Absolute magnitude of element }
      if (Magn > Thresh) then     { Test for significance }
      begin
        Found := Get_val( DIFM,I,dNewDif,Var2);
        Found := Get_val( TTST,I,TestCol,Var3);
        Var3 := Var3 + (2*Var2);                { Slope prediction addon }
        Found := Set_val( TTST,I,TestCol,Var3);
      end; {if Var1}
    end; {for I}
  end; {if Iter}
  case dOldDif = 2 of
  true  : begin
            dOldDif := 3;
            dNewDif := 2;
          end;
  false : begin
            dOldDif := 2;
            dNewDif := 3;
          end;
  end; {case}
end; {Slope_prediction}

procedure NSig_removal;

var
  Found           : boolean;
  Mean,SD         : real;      { Predicted vector values }
  dMean,dSD       : real;      { 1st differential values }
  I               : word;      { Loop control }
  Var1,Var2       : extended;  { Temporary storage }
  Magn,Thresh     : extended;  { Predicted vector values }
  dMagn,dThresh   : extended;  { 1st differential values }
  Count,dCount    : word;      { Counters }


begin
  Vector_mean_and_sd( DIFM,dNewDif,VLen,dMean,dSD);
  Vector_mean_and_sd( TTST,TestCol,VLen,Mean,SD);
  Found := Set_val( DIFM,1,dOldDif,dMean);
  Found := Set_val( DIFM,2,dOldDif,dSD);
  Found := Set_val( TTST,11,ErrCol,Mean);
  Found := Set_val( TTST,12,ErrCol,SD);
  dCount := 0; Count := 0;        { Counter initialization }
  dThresh := dSD * 0.1;           { Significance threshold - differential }
  Thresh := SD * 0.05;             { Significance threshold - pred vector }
  for I := 1 to VLen do
  begin
    Found := Get_val( DIFM,I,dNewDif,Var1);
    dMagn := abs(Var1 - dMean);            { Absolute magnitude of element }
    if (dMagn < dThresh) then
    begin
      Inc(dCount);
      Found := Get_val( TTST,I,TestCol,Var2);
      Magn := abs(Var2 - Mean);            { Absolute magnitude of element }
      if (Magn < Thresh) then
      begin
        Inc(Count);
        Found := Set_val( TTST,I,TestCol,0);
      end; {if}
    end; {if}
  end; {for I}
  Var1 := (dCount / VLen) * 100;     { Reuse of Var1 and Var2 for % calcs }
  Var2 := (Count / VLen) * 100;
  Found := Set_val( TTST,13,ErrCol,Var1);
  Found := Set_val( TTST,14,ErrCol,Var2);
end; {NSig_removal}

procedure NSig2_removal;

var
  Found           : boolean;
  Mean,SD         : real;      { Predicted vector values }
  I               : word;      { Loop control }
  Var2            : extended;  { Temporary storage }
  Magn,Thresh     : extended;  { Predicted vector values }
  Count           : word;      { Counter }
  begin
    Vector_mean_and_sd( TTST,TestCol,VLen,Mean,SD);
    Found := Set_val( TTST,11,ErrCol,Mean);
    Found := Set_val( TTST,12,ErrCol,SD);
    Count := 0;         { Counter initialization }
    Thresh := SD * 1;              { Significance threshold - pred vector }
    for I := 1 to VLen do
    begin
      Found := Get_val( TTST,I,TestCol,Var2);
      Magn := abs(Var2 - Mean);              { Absolute magnitude of element }
      if (Magn < Thresh) then
      begin
        Inc(Count);
        Found := Set_val( TTST,I,TestCol,0);
      end; {if}
    end; {for I}
    Var2 := (Count / VLen) * 100;
    Found := Set_val( TTST,14,ErrCol,Var2);
  end; {NSig2_removal}

procedure Reinforce_vector( SourceName : MatrixNameType;
                            SourceCol  : word;
                            DestName   : MatrixNameType;
                            DestCol    : word;
                            VecLen     : word);

var
  I    : word;      { Loop counter }
  Var1 : extended; { Temporary variable }

begin
  for I := 1 to VecLen do
  begin
    Found := Get_val( SourceName,I,SourceCol,Var1);
    if (Var1 > 0) then
      Found := Set_val( DestName,I,DestCol,Var1);
  end; {for}
end; {Reinforce_vector}



begin {ITT}
  OldDiff := 0;
  Calculate_TVEC;         { Perform }
  Calculate_pred_vector;  { the target test. }
  Calc_abs_diff;
  Calc_correlation;
  Vector_differential(TTST,PredCol,TTST,TestCol,DIFM,dNewDif,VLen);
  Copy_col_vector( DIFM,TestCol,TTST,TestCol,VLen);
  Errors_in_test_vector;
  Update_error_vector;
  if (abs(Diff - OldDiff) > 1e-16) then Unfinished := true;
  if Unfinished then
  begin
    OldDiff := Diff;
    Copy_col_vector( TTST,PredCol,TTST,TestCol,Vlen);
{    NSig_removal;}
{    Slope_prediction;}
{    Zero_negative_values;}
{      Normalize_vector;}
    Zero_below_threshold(ResNulTest);
    Reinforce_vector( DIFM,TestCol,TTST,TestCol,VLen);
  end; {if}
  Finished := not Unfinished;
 end; {ITT}

end. {TargTest}
```

3.10. Unit LOTUSFIL

```
unit LotusFil;

{The code contained in this unit is Copyright by T.G.Brockwell, 1989-92. All rights reserved.}


{$O+,F+}

interface

uses DOS,PFAGlobs,EMSDat;

procedure Lotus_import (Filename : PathStr;
                        Name     : MatrixNameType;
                    var Err      : boolean;
                    var ErrMsg   : Str_80);

procedure Lotus_export (Filename : PathStr;
                        Name     : MatrixNameType;
                    var Err      : boolean;
                    var ErrMsg   : Str_80);

implementation

const
  BofOC = 0;         { OpCode for beginning of file record }
  IntOC = 13;        { OpCode for integer record }
  NumOC = 14;        { OpCode for number record }
  EofOC = 1;         { OpCode for end of file record }
  DimOC = 6;         { OpCode for Dimensions record }
  Win1OC = 7;        { OpCode for Window1 record }
  ColW1OC = 8;       { OpCode for Col width Win1 record }
  HidCol1OC = 100;   { OpCode for Win1 hidden cols record }


type
  RecordHeadType = record                    { Record header construct }
                     OpCode,
                     RecLen : integer;
                   end;

  NumBodyType = record                       { Number body record }
                  Format : byte;
                  Column,
                  Row    : integer;
                  Value  : double;
                end;

  IntBodyType = record                       { Integer body record }
                  Format : byte;
                  Column,
                  Row,
                  Value  : integer;
                end;

  DimenBodyType = record                     { Dimensions body record }
                    SCol,
                    SRow,
                    ECol,
                    ERow : integer;
                  end;

  Win1BodyType32 = record                     { Window1 body record }
                     CurPosCol,               { 32 byte length for }
                     CurPosRow : integer;     { 123/2 file format }
                     Format,
                     Unused1 : byte;
                     DefColWidth,
                     ColsOnScreen,
                     RowsOnScreen,
                     LeftCol,
                     TopRow,
                     TitleCols,
                     TitleRows,
                     LTitleCol,
                     TTitleRow,
                     TLCol,
                     TLRow,
                     ColsInWin,
                     Unused2 : integer;
                   end;

  Win1BodyType31 = record                     { Window1 body record }
                     CurPosCol,               { 31 byte length for }
                     CurPosRow : integer;     { 123/1a file format }
                     Format,
                     Unused1 : byte;
                     DefColWidth,
                     ColsOnScreen,
                     RowsOnScreen,
                     LeftCol,
                     TopRow,
                     TitleCols,
                     TitleRows,
                     LTitleCol,
                     TTitleRow,
                     TLCol,
                     TLRow,
                     ColsInWin : integer;
                     Unused1more : byte;
                   end;

  ColW1BodyType = record                     { Win1 Column width body record }
                    ColID : integer;
                    Width : byte;
                  end;

  HidCol1BodyType = record                   { Win1 hidden column body record }
                      HidRec : array [1..32] of byte;
                    end;

var
  InFile      : file;            { Input file }
  OutFile     : file;            { Output file }
  Found        : boolean;          { Success flag for data structure writes }
  FirstRec     : boolean;          { Flag for first record in file }
  LastRec      : boolean;          { Flag for last record in file }
  ReadDim      : boolean;          { Flag for dimensions read }
  RecordHead   : RecordHeadType;   { Record header }
  NumBody      : NumBodyType;      { Body of number record }
  IntBody      : IntBodyType;      { Body of integer record }
  DimenBody    : DimenBodyType;    { Body of Dimensions record }
  Win1Body32   : Win1BodyType32;   { Body of Window1 record 32 byte length }
  Win1Body31   : Win1BodyType31;   { Body of Window1 record 31 byte length }
  ColW1Body    : ColW1BodyType;    { Body of Win1 column width record }
  HidCol1Body  : HidCol1BodyType;  { Body of Win1 hidden column record }
  ByteBuff     : byte;             { Buffer for discarded records }
  I            : longint;          { Loop control for record discards }

procedure Lotus_import (Filename : PathStr;
                        Name     : MatrixNameType;
                    var Err      : boolean;
                    var ErrMsg   : Str_80);
                                           { Imports numeric data from a }
                                           { Lotus 123 file into the data }
                                           { matrix }



  procedure Error (ErrNum : byte);
                                 { Passes appropriate error messages }
                                 { to Display_error_message. }
  type
    MsgType = array [1..6] of Str_80; { Array to contain messages }

  const                                                     { Message list }
    ErrorMsg : MsgType = ('Invalid version number in file',
                          'BOF record appears in the wrong place',
                          'Data appears after EOF record - Invalid file',
                          'Non zero entry for EOF record body',
                          'Unable to write to data matrix',
                          'No Dimensions record in file - unable to read');

  begin
    Err := true;
    ErrMsg := ErrorMsg[ErrNum];
    Close(InFile);
    exit;
  end; {Error}

  procedure Read_BOF;
                     { Reads the version number of the file }
                     { and checks its validity }
  const
    Ver1 = 1028; { Version code for 123/1,123/1A }
    Symp = 1029; { Version code for Symphony/1.0 }
    Ver2 = 1030; { Version code for 123/2 Symphony/1.1 }

  var
    Version : integer; { Version number from file }

  begin
    BlockRead(Infile,Version,SizeOf(Version));            { Read version no }
    if (Version <> Ver1) and                          { Compare with valid nos }
       (Version <> Symp) and
       (Version <> Ver2) then Error(1);
    if not FirstRec then Error(2);                    { Check position in file }
  end; {Read_BOF}

  procedure Read_dimensions;

  var
    Rows,Cols : word;

  begin
    BlockRead( InFile,DimenBody,SizeOf(DimenBody)); { Read body from file }
    with DimenBody do
    begin
      Initialise_matrix( Name,Succ(ERow),Succ(ECol),DefHeapToUse,EXVN,'DATARRAY.$$$');
    end; {with}
    ReadDim := true;
  end; {Read_dimensions}

  procedure Read_window1;

  var
    Row,Col : word;

  begin
  case (RecordHead.RecLen = 32) of
    true : begin
             BlockRead( Infile,Win1Body32,SizeOf(Win1Body32));
             with Win1Body32 do
             begin
               CursorPos.Column := word(CurPosCol);
               CursorPos.Row := word(CurPosRow);
               ColWidth := byte(DefColWidth);
               Origin.Column := word(LeftCol);
               Origin.Row := word(TopRow);
             end; {with}
           end; {true}
    false : begin
              BlockRead( Infile,Win1Body31,SizeOf(Win1Body31));
              with Win1Body31 do
              begin
                CursorPos.Column := word(CurPosCol);
                CursorPos.Row := word(CurPosRow);
                ColWidth := byte(DefColWidth);
                Origin.Column := word(LeftCol);
                Origin.Row := word(TopRow);
              end; {with}
            end; {false}
    end; {case}
  end; {Read_window1}

  procedure Read_integer;
                           { Reads an integer record from the file }
                           { and places it in the data structure }
  var
    Number : extended; { Allows number conversion for data structure }
  begin
    BlockRead(Infile,IntBody,SizeOf(IntBody));  { Read the body from the file }
    with IntBody do
    begin
      Number := Value;                           { Convert integer to extended }
      Found := Set_val(Name,Succ(Row),Succ(Column),Number);
      if not Found then Error(5);              { Write value in data structure }
    end; {with}
  end; {Read_integer}

  procedure Read_number;
                         { Reads a number record from the file }
                         { and places it in the data structure }
  var
    Number : extended; { Allows number conversion for the data structure }

  begin
    BlockRead(Infile,NumBody,SizeOf(NumBody));  { Read the body from the file }
    with NumBody do
    begin
      Number := Value; { Convert double to extended }
      Found := Set_val(Name,Succ(Row),Succ(Column),Number);
      if not Found then Error(5);              { Write value in data structure }
    end; {with}
  end; {Read_number}

begin {Lotus_import}
  FirstRec := true;
  LastRec  := false;
  ReadDim  := false;
  Assign(Infile,Filename);
  Reset(Infile,1);
  while not Eof(Infile) do
  begin
    BlockRead(Infile,RecordHead,SizeOf(RecordHead));
    with RecordHead do
    begin
      if (OpCode in [IntOC,NumOC]) and (not ReadDim) then Error(6);
      case OpCode of
        BofOC   : Read_BOF;
        DimOC   : Read_dimensions;
        Win1OC  : Read_window1;
        IntOC   : Read_integer;
        NumOC   : Read_number;
        EofOC   : LastRec := true;
        else
          for I := 1 to RecLen do
            BlockRead(Infile,ByteBuff,SizeOf(ByteBuff));
      end; {case}
      FirstRec := false;
      if LastRec and not Eof(InFile) then Error(3);
    end; {with}
  end; {while}
  Close(Infile);
end; {Lotus_import}

procedure Lotus_export (Filename : PathStr;
                        Name     : MatrixNameType;
                    var Err      : boolean;
                    var ErrMsg   : Str_80);


var
  Rows,Cols : word;

  procedure Error (ErrNum : byte);
                                   { Passes appropriate error messages }
                                   { to Display_error_message. }
  type
    MsgType = array [1..3] of Str_80; { Array to contain messages }

  const                                                        { Message list }
    ErrorMsg : MsgType = ('Matrix not found in data structure',
                          'Matrix to large for spreadsheet file',
                          'Data point not found in data structure');


  begin
    Err := true;
    ErrMsg := ErrorMsg[ErrNum];
    Close(OutFile);
    Erase(OutFile);
    exit;
  end; {Error}

  procedure Write_BOF;

  const
    VersionNo : integer = 1030;

  begin
    with RecordHead do
    begin
      OpCode := BofOc;
      RecLen := 2;
    end; {with}
    BlockWrite(OutFile,RecordHead,SizeOf(RecordHead));
    BlockWrite(OutFile,VersionNo,SizeOf(VersionNo));
  end; {Write_BOF}

  procedure Write_Dimensions;

  begin
    with RecordHead do
    begin
      OpCode := DimOC;
      RecLen := 8;
    end; {with}
    with DimenBody do
    begin
      Scol := 1;
      Srow := 1;
      ECol := integer(Pred(Cols));
      ERow := integer(Pred(Rows));
    end; {with}
    BlockWrite( OutFile,RecordHead,SizeOf(RecordHead));
    BlockWrite( Outfile,DimenBody,SizeOf(DimenBody));
  end; {Write_dimensions}

  procedure Write_win1_descriptor;
```

```
    procedure Write_window1;

    begin
      with RecordHead do
      begin
        OpCode := Win1OC;
        RecLen := 32;
      end;
      with Win1Body32 do
      begin
        CurPosCol := Pred(integer(CursorPos.Column));
        CurPosRow := Pred(integer(CursorPos.Row));
        Format    := 241;
        Unused1   := 0;
        DefColWidth := 9;
        ColsOnScreen := 8;
        RowsOnScreen := 20;
        LeftCol      := Pred(integer(Origin.Column));
        TopRow       := Pred(integer(Origin.Row));
        TitleCols    := 0;
        TitleRows    := 0;
        TLCol        := 4;
        TLRow        := 4;
        ColsInWin    := 72;
        Unused2      := 0;
      end; {with}
      BlockWrite( OutFile,RecordHead,SizeOf(RecordHead));
      BlockWrite( OutFile,Win1Body32,SizeOf(Win1Body32));
    end; {Write_Window1}

    procedure Write_HidCol1;

    var
      I : byte;

    begin
      with RecordHead do
      begin
        OpCode := HidCol1OC;
        RecLen := 32;
      end; {with}
      with HidCol1Body do
      begin
        for I := 1 to 32 do
        begin
          HidRec[I] := 0;
        end; {for}
      end; {with}
      BlockWrite( OutFile,RecordHead,SizeOf(Recordhead));
      BlockWrite( OutFile,HidCol1Body,SizeOf(HidCol1Body));
    end; {Write_HidCol1}

  begin {Write_win1_descriptor}
    Write_window1;
    Write_HidCol1;
  end; {Write_win1_descriptor}

  procedure Write_data (Name : MatrixNameType);

  var
    I,J       : integer;
    Number    : extended;
    Found     : boolean;

  begin
    with RecordHead do
    begin
      OpCode := 14;
      RecLen := 13;
    end; {with}
    with NumBody do
    begin
      Format := 255;
      for I := 1 to Rows do
      begin
        Row := Pred(I);
        for J := 1 to Cols do
        begin
          Found := Get_val(Name,I,J,Number);
          if not found then Error(3);
          Value := Number;
          Column := Pred(J);
          BlockWrite(OutFile,RecordHead,SizeOf(RecordHead));
          BlockWrite(OutFile,NumBody,SizeOf(NumBody));
        end; {for J}
      end; {for I}
    end; {with}
  end; {Write_data}

  procedure Write_EOF;

  begin
    with RecordHead do
    begin
      OpCode := 1;
      RecLen := 0;
    end; {with}
    BlockWrite(OutFile,RecordHead,SizeOf(RecordHead));
  end; {Write_EOF}

begin {Lotus_export}
  Found := Matrix_dimensions(Name,Rows,Cols);
  if not Found then Error(1);
  if (Rows > MaxInt) or (Cols > MaxInt) then Error(2);
  Assign(OutFile,Filename);
  Rewrite(OutFile,1);
  Write_BOF;
  Write_dimensions;
  Write_win1_descriptor;
  Write_data(Name);
  Write_EOF;
  Close(OutFile);
end; {Lotus_export}

end. {unit LotusFil}
```

3.11. Unit IO_UNIT

```
unit IO_Unit;

{The code contained in this unit is Copyright by T.G.Brockwell, 1989-92. All rights reserved.}

{$O+,F+}
             { Contains function and procedures involved in input }
             { and output to any of the standard DOS ports, including }
             { screen, keyboard and disk. }

{******************************************************************}

interface

uses Overlay,DOS,PFAGlobs,EPAimprt,EMSDat,PFAVid,PFAUtils,Lotusfil;

procedure Write_data_file (FileName : PathStr;
                           Name     : MatrixNameType);

procedure Import_file (Filetype : TypeOfFile);

procedure Export_file (Filetype : TypeOfFile);

{******************************************************************}

implementation

type
  DataPoint = record
                Row,Col : word;
                Value   : extended;
              end;

{******************************************************************}

procedure Write_data_file (FileName : PathStr;
                           Name     : MatrixNameType);

                           { Writes all the data contained in a named array }
                           { to a disk file in record form, the fields      }
                           { being Row & Col co-ordinates and the data      }
                           { value. }

{-----------------------}

begin
  Save_file( Name,FileName);
end; {Write_data_file}

{******************************************************************}

procedure Read_data_file
          (FileName : PathStr;
           Name     : MatrixNameType);

{-----------------------}

begin
  Read_file( Name,DefHeapToUse,EXVN,Filename);
end; {Read_data_file}

{-----------------------}

procedure EPA_import (Filename : PathStr;
                      Name     : MatrixNameType;
                  var Err      : boolean;
                  var ErrMsg   : Str_80);

begin
  EPAimprt.EPA_import( Filename,Name,Err,ErrMsg);
end; {EPA_import}

procedure Import_file (Filetype : TypeOfFile);

var
  EscPress : boolean;
  Path     : PathStr;
  Err      : boolean;
  Ok       : boolean;
  ErrMsg   : Str_80;
  Msg      : Str_80;

begin
  Err := false;
  Get_load_filename(Path,EscPress);
  if EscPress then exit;
  Clr_menu;
  Msg := 'Select the name of the matrix the file is to be loaded as';
  Ok := false;
  while not Ok do
  begin
    Change_disp_matrix(Msg,EscPress);              { Select matrix to load as }
    if EscPress then exit;
    if not (CurrentMatrix in [NONE,RESI])           { If wrong selection then }
      then Ok := true;
    if not Ok then
    begin
      Status := ER;                                { Indicate error condition }
      Write_status(Status);
      Write(Bell);
      Msg := 'You cannot load a file to this matrix, please select another';
    end; {if}
  end; {while}
  Ok := Delete_matrix(CurrentMatrix);                     { Remove old matrix }
  Status := WA;
  Write_status(Status);
  case FileType of
    Native : Read_data_file(Path,CurrentMatrix);
    ASCII  : EPA_import(Path,CurrentMatrix,Err,ErrMsg);
    Lotus  : Lotus_import(Path,CurrentMatrix,Err,ErrMsg);
  end; {case}
  if Err then Display_error_message(ErrMsg);
  Change_current_matrix(CurrentMatrix);
  Status := RE;
  Write_status(Status);
  if CurrentMatrix in [DATA] then FileChanged := false;
end; {Import_file}

procedure Export_file (Filetype : TypeOfFile);

var
  EscPress : boolean;
  Path     : PathStr;
  Err      : boolean;
  Ok       : boolean;
  ErrMsg   : Str_80;
  Msg      : Str_80;

begin
  Err := false;
  Get_save_filename(Path,EscPress);
  if EscPress then exit;
  Clr_menu;
  Msg := 'Select the name of the matrix to be saved to file';
  Ok := false;
  while not Ok do
  begin
    Change_disp_matrix(Msg,EscPress);                 { Select matrix to save }
    if EscPress then exit;
    if not (CurrentMatrix in [NONE{,RESI}])           { If wrong selection then }
      then Ok := true;
    if not Ok then
    begin
      Status := ER;                                  { Indicate error condition }
      Write_status(Status);
      Write(Bell);
      Msg := 'You cannot save this matrix to file, please select another';
    end; {if}
  end; {while}
  Status := WA;
  Write_status(Status);
  case FileType of
    Native : Write_data_file(Path,CurrentMatrix);
    Lotus  : Lotus_export(Path,CurrentMatrix,Err,ErrMsg);
  end; {case}
  if Err then Display_error_message(ErrMsg);
  Change_current_matrix(CurrentMatrix);
  Status := RE;
  Write_status(Status);
  if CurrentMatrix in [DATA] then FileChanged := false;
end; {Export_file}

end. {IO_UNIT}
```

3.12. Unit PFAVID

```
unit PFAVid;

{The code contained in this unit is Copyright by T.G.Brockwell, 1989-92. All rights reserved.}


{$O+,F+}

interface

uses Overlay,Crt,DOS,PFAGlobs,Utils,Vid_Util,PFAUtils;

procedure Wave_by_by;

procedure Draw_screen;

procedure Write_number_window;

procedure Clr_menu;

procedure String_input (Prompt      : Str_80;
                    var InpStr    : Str_255;
                        FieldSize : byte;
                    var EscPress  : boolean);

procedure Change_disp_matrix (Msg : Str_80;
                              var EscPress : boolean);

procedure Get_load_filename (var Path : PathStr;
                             var EscPress : boolean);

procedure Get_save_filename (var Path : PathStr;
                             var EscPress : boolean);

procedure Display_error_message(ErrMsg : Str_80);

procedure Write_text_window;

procedure Close_text_window;

procedure Write_msg_in_window(Msg : Str_80);

procedure Write_factor_number(Number : word);

procedure Update_iteration_number(IterationNo : word);

procedure Finished_factor( var PVar,PCumVar : extended;
                           var SigLevel : real);

implementation

procedure Wave_by_by;

begin
  TextMode(CO80);
  GotoXY(28,12);
  Write('Andy is waving goodbye');
  Delay(2000);
  GotoXY(35,13);
  Write('Goodbye');
  Delay(2000);
  GotoXY(35,14);
  Write('Goodbye');
  GotoXY(1,25);
end; {Wave_by_by}

procedure Draw_screen;          { Draws the non number related part of screen }

  procedure Draw_top_window;            { Draws the top 3 lines of the screen }

  begin
    Colour_box(1,1,80,3,LightGray,Black,False);           { Set window colour }
    Fill_screen(1,1,80,3,' ');                                  { Blank window }
    Write_status(Status);
    Write_location;
  end; {Draw_top_window}

  procedure Draw_bottom_line;           { Draws the bottom line of the screen }

  begin {Draw_bottom_line}
    Colour_box(1,25,30,25,LightGray,Black,False);         { Set window colour }
    Update_time;                                            { Write date/time }
  end; {Draw_bottom_line}

begin {Draw_screen}
  Refresh := true;                              { Flag number area to redraw }
  Draw_top_window;
  Draw_bottom_line;
  Fill_screen(1,4,80,24,' ');                             { Blank number area }
  RedrawScreen := false;                                { Reset drawing flag }
end; {Draw_screen}

procedure Draw_headings;                      { Write column and row headings }

  procedure Write_col_headings;   { Writes column heading in centre of column }

  var
    CodeString,ColLabel : Str_255;    { Contains column coded in letters }
    ColStart,Offset     : byte;       { Cursor location variables }
    I,J                 : byte;       { Loop control variables }

  begin
    for I := 0 to Pred(NoOfCols) do                       { For each column }
    begin
      ColStart := (I * ColWidth) + 6;       { Calculate starting coordinate }
      FillChar(ColLabel,Succ(ColWidth),' ');                 { Blank string }
      ColLabel[0] := Char(ColWidth);
      if Headings then                        { Test for numbers or letters }
        begin
          Convert_to_base_26(Origin.Column + I,CodeString);   { Find letters }
          Strip(CodeString,[' ']);           { Remove spaces from CodeString }
        end {if then}
      else
        Str((Origin.column + I),CodeString);       { Turn number into string }
```

```
     Offset := (ColWidth-Length(CodeString)) div 2; { Centralise CodeString }
     Insert(CodeString,ColLabel,Succ(Offset));       { Place label centrally }
     GotoXY(ColStart,4);
     TextAttr := Get_attribute(ColStart,4);
     Write(ColLabel);                                          { Write label }
   end; {for I}
 end; {Write_col_headings}

 procedure Write_row_numbers;                            { Writes row numbers }

 var
   NumStr : Str_5;      { Contains row number as a string }
   I      : byte;       { Local loop control }

 begin
   for I := 5 to (NoOfRows + 4) do                      { For each screen row }
   begin
     NumStr := '     ';                                 { Initialize string }
     Str((Origin.Row + I - 5), NumStr);       { Convert row number to string }
     NumStr[0] := Chr(5);                { Set string length to column width }
     GotoXY(1,I);
     TextAttr := Get_attribute(1,I);
     Write(NumStr);                                           { Write string }
   end; {for}
 end; {Write_row_numbers}

 procedure Write_matrix_name;          { Writes current matrix name at bottom }
                                                                  { of screen }
 const
   FrgCol = White;  { Constant for foreground colour }
   BkgCol = Blue;   { Constant for background colour }

 var
   OldAttr : byte;  { Saves current text attribute }

 begin
   Colour_box(32,25,62,25,LightGray,Black,False);     { Write block of colour }
   Fill_screen(32,25,62,25,' ');                        { Clear existing text }
   GotoXY(32,25);                                           { Position cursor }
   OldAttr := TextAttr;                           { Save current text attribute }
   TextAttr := FrgCol+ BkgCol*16;                         { Set new attribute }
   Write(MatrixNameStr[CurrentMatrix]);            { Write current matrix name }
   TextAttr := OldAttr;                               { Restore text attribute }
 end; {Write_matrix_name}

begin {Draw_headings}
  Colour_box(1,4,80,4,White,Blue,False);                { Colour row headings }
  Colour_box(1,5,5,24,White,Blue,False);             { Colour column headings }
  Colour_box(6,5,80,24,White,LightGray,False);        { Colour number window }
  NoOfCols := (75 div ColWidth);    { Number of columns possible on the screen }
  NoOfRows := 20;
  case (CurrentMatrix = None) of                      { Set dimensions of screen }
    true  : begin
              NoOfCols := 1;
              NoOfRows := 1;
            end; {true}
    false : begin                                   { Exception handling for small }
              if MatrixDim.Column < NoOfCols                              { matrices }
                then NoOfCols := MatrixDim.Column;
              if MatrixDim.Row < NoOfRows
                then NoOfRows := MatrixDim.Row;
            end {false}
  end; {case}
  Write_col_headings;
  Write_row_numbers;
  Write_matrix_name;
end; {Draw_headings}

procedure Write_number_window;

var
  CellValue : Str_255;    { String containing shortened number }
  Row,Col   : word;       { Cell coordinates }
  ColStart  : byte;       { Coord of column start }
  I,J       : byte;       { Local loop control }

begin
  Draw_headings;                                { Draw row and column headings }
  for I := 5 to (NoOfRows + 4) do                      { For each screen row }
  begin                                                     { For each column }
    for J := 0 to Pred(NoOfCols) do
    begin
      Row := Origin.Row + I - 5;                                    { Calculate }
      Col := Origin.Column + J;                              { cell coordinates }
      Value_to_string(ColWidth,Row,Col,CellValue); { Get cell value as string }
      ColStart := (J * ColWidth) + 6;             { Calculate print position }
      GotoXY(ColStart,I);
      TextAttr := Get_attribute(ColStart,I);
      Write(CellValue);                                      { Write cell value }
    end; {for J}
  end; {for I}
  Draw_cursor;
  Refresh := false;
end; {Write_number_window}

procedure Clr_menu;   { Clears menu and message lines and restores TextAttr }

begin
  TextAttr := Black*16 + LightGray;
  GotoXY(1,2);
  ClrEol;
  GotoXY(1,3);
  ClrEol;
  Write_status(Status);
end; {Clr_menu}

procedure String_input (Prompt     : Str_80;  { Takes string input in a field }
                    var InpStr     : Str_255;              { and allows editing }
                        FieldSize  : byte;
                    var EscPress   : boolean);

var
  Inchr,Inctl   : char;     { Key input from keyboard }
  Default       : Str_255;  { Original value for string }
  FieldStart    : byte;     { Points to start of input field }
  CursorPos     : byte;     { Points to current cursor position in field }
  CursorSize    : CurType;  { Holds current size of cursor }
  StrPos        : byte;     { Points to current position in input string }
```

```
FieldOffset  : byte;    { Offset of field start from start of string }


procedure Edit_string;      { Called if default not accepted to edit string }

var
  Ins : boolean;       { Flag to signal insertion not overwriting }
  InitLoop : boolean; { Flag to force handling of existing keypress }
  Loop : integer;      { Idle time counter }

  procedure Write_field;       { Displays string in field at current offset }

  begin
    GotoXY(FieldStart,WhereY);                     { Move cursor to field start }
    Write(Copy(InpStr,Succ(FieldOffset),FieldSize));            { Write string }
    GotoXY((FieldStart + Pred(CursorPos)),WhereY);          { Position cursor }
    StrPos := FieldOffset + CursorPos;                     { Calculate StrPos }
  end; {Write_field}

  procedure Add_char;              { Adds, Inserts or overwrites a character }

  begin
    case Ins of
      true  :  Insert(Inchr,InpStr,StrPos);          { Insert a new character }
      false :  if StrPos = Length(InpStr)
               then Insert(Inchr,InpStr,StrPos) { Adds new character at end }
               else  InpStr[StrPos] := Inchr; { Overwrites old character }
    end; {case}
    case Length(InpStr) > FieldSize of         { If string longer than field }
      true  :   FieldOffset := Succ(FieldOffset);           { Increase offset }
      false :  CursorPos := Succ(CursorPos);      { else move cursor in field }
    end; {case}
  end; {Add_char}

  procedure Move_left;                            { Handles left arrow key input }

  begin
    if CursorPos > 1                                  { If not at left of field }
    then CursorPos := Pred(CursorPos)                       { move cursor left }
    else if FieldOffset > 0                       { or if not at left of string }
         then FieldOffset := Pred(FieldOffset);             { decrease offset }
  end; {Move_left}

  procedure Move_right;                          { Handles right arrow key input }

  begin
    if StrPos < Length(InpStr) then      { If cursor is not at end of string }
    begin
      case CursorPos < FieldSize of       { If cursor is not at end of field }
        true  :  CursorPos := Succ(CursorPos);             { move cursor right }
        false :  FieldOffset := Succ(FieldOffset);     { or increment offset }
      end; {case}
    end; {if}
  end; {Move_right}

  procedure Move_home;                                  { Handles Home key input }

  begin
    CursorPos := 1;                { Set cursor position to start of string }
    FieldOffset := 0;                         { Set offset to start of string }
  end; {Move_home}

  procedure Move_end;                                    { Handles End key input }

  begin
    case FieldSize < Length(InpStr) of      { If string is longer than field }
      true  :  begin
                 CursorPos := FieldSize;{ then place cursor at end of field }
                 FieldOffset := Length(InpStr) - FieldSize; { set offset to }
               end; {true}                             { display end of string }
      false :  CursorPos := Length(InpStr);  { else cursor at end of string }
    end; {case}
  end; {Move_end}

  procedure Delete_char;                                  { Handle Del key input }

  begin
    Delete(InpStr,StrPos,1); { Delete a char at the current position }
    if (Length(InpStr) = 0) then InpStr := ' '; { Add a space if deleted }
  end; {Delete_char}                              { last char in string }

  procedure Backspace;                          { Handles Backspace key input }
   begin
    Move_left;                                      { Move left one character }
    Write_field;                          { Rewrite field to clear character }
    Delete_char;                                                { and delete it }
  end; {Backspace}

  procedure Toggle_insert;

  begin
    Ins := not Ins;
    case Ins of
      true  : CursorSize := Big;
      false : CursorSize := Small;
    end; {case}
    BIOSCursor(CursorSize);
  end; {Toggle_insert}

begin {Edit_string}
  InitLoop := true;                      { Flag for keypress already received }
  Ins := false;                                   { Default to overwrite mode }
  InpStr := InpStr + ' ';               { Add a space to the string for input }
  Move_end;                                   { Move cursor to end of string }
  BIOSCursor(CursorSize);                                          { Show cursor }
  repeat
    Write_field;                                                  { Write string }
    if not InitLoop then
    begin
      Loop := 0;
      repeat
        Inc(Loop);
        if Loop = 10000
        then begin
          BIOSCursor(Off);
          Update_lights;
          Update_time;
          BIOSCursor(CursorSize);
```

```
            Loop := 0;
          end; {if}
        until KeyPressed;
        Read_Kbd(Inchr,Inctl);                                { Read keystroke }
      end; {if}
      case Inctl of                                         { Act on key input }
        NullChr,
        Slash     : Add_char;
        Cur_left  : Move_left;
        Cur_right : Move_right;
        Home_key  : Move_home;
        End_key   : Move_end;
        Del_key   : Delete_char;
        BkSpc     : Backspace;
        Ins_key   : Toggle_insert;                    { Toggle insert mode }
        CR        : InpStr[0] := Pred(InpStr[0]);        { Lose space at end }
        Esc       : InpStr := Default;
      else Write(Bell);
      end; {case}
      InitLoop := false;          { Flag to force handling of first key press }
    until Inctl in [Esc,CR];
    BIOSCursor(Off);                                  { Turn cursor off again }
  end; {Edit_string}

begin {String_input}
  if not(Status = ER) then Write_status(ED);   { Show edit status if no error }
  Default := InpStr;                                   { Save original value }
  FieldOffset := 0;                                      { Initialize offset }
  CursorPos := 1;                                    { and cursor position }
  CursorSize := Small;                            { Set cursor size to Small }
  Write(Prompt);                                          { Write the prompt }
  ClrEol;
  FieldStart := WhereX;         { Save the position of the start of the field }
  Write(Copy(InpStr,1,FieldSize));                   { Write the current value }
  repeat
    Update_lights;
    Update_time;
  until KeyPressed;
  Read_Kbd(Inchr,Inctl);                                        { Read a key }
  if not(Inctl in [Esc,CR]) then               { If key isn't enter or escape }
  begin
    if Inctl in [NullChr,Slash] then              { If the key is a character }
    begin
      FillChar(InpStr,Succ(FieldSize),' ');           { then clear the field }
      InpStr[0] := Chr(FieldSize);
      GotoXY(FieldStart,WhereY);
      Write(InpStr);
      InpStr := '';                                  { and clear the string }
    end; {if}
    Edit_string;                              { else edit the existing string }
  end; {if}
  if Inctl = Esc
    then EscPress := true
    else EscPress := false;
  Write_status(Status);                         { Redisplay the default status }
end; {String_input}

procedure Get_load_filename (var Path : PathStr;     { Gets and error checks a }
                            var EscPress : boolean);  { filename for loading }
var
  Ok         : boolean;
  InpStr     : Str_255;   { Filename string }
  Prompt,Msg : Str_80;    { Text prompts }
  DirInfo    : SearchRec; { File info from disk }

begin
  Clr_menu;                                        { Initialization section }
  TextAttr := Get_attribute(1,2);                  {                        }
  Ok := false;                                      {                       }
  Prompt := 'Filename = ';                           {                      }
  Msg := 'Enter the name of the file you wish to load';  {                  }
  InpStr := '';                                          {                  }
  while not Ok do
  begin
    GotoXY(1,3);                                           { Write message }
    Write(Msg);
    GotoXY(1,2);
    String_input(Prompt,InpStr,65,EscPress);      { Get string from keyboard }
    if EscPress = true then exit;               { Leave if procedure aborted }
    Path := InpStr;
    FindFirst(Path,AnyFile,DirInfo);                   { Try to find the file }
    Ok := DosError = 0;                                    { Test for success }
    if not Ok then                                         { If no file then }
    begin
      Status := ER;
      Write_status(Status);                            { Give error message }
      Write(Bell);
      Msg := 'File does not exist - Enter correct filename';      { and retry }
    end; {if}
  end; {while}
  Status := RE;                                      { Restore system status }
  Write_status(Status);
end; {Get_load_filename}

procedure Get_save_filename (var Path : PathStr;       { Gets and error checks }
                            var EscPress : boolean); { a filename for saving }

var
  Ok         : boolean;
  InpStr     : Str_255;   { Filename strings }
  Prompt,Msg : Str_80;    { Text prompts }
  DirInfo    : SearchRec; { File info from disk }

begin
  Clr_menu;                                         { Initialization section }
  TextAttr := Get_attribute(1,2);                   {                        }
  Ok := false;                                      {                        }
  Msg := 'Enter the filename to save matrix to';    {                        }
  InpStr := '';                                     {                        }
  while not Ok do
  begin
    GotoXY(1,3);                                             { Write message }
    Write(Msg);
    Prompt := 'Filename = ';
    GotoXY(1,2);
    String_input(Prompt,InpStr,65,EscPress);       { Get string from keyboard }
    if EscPress = true then exit;                { Leave if procedure aborted }
    Path := InpStr;
    FindFirst(Path,AnyFile,DirInfo);                    { Try to find the file }
    case DosError of
      2,18 : Ok := true;                               { Success if not found }
      0    : begin
               Prompt := 'Overwrite existing file y/n ? ';        { Ask for }
               Write(Bell);                             { confirmation of }
               repeat                                         { overwrite }
                 GotoXY(1,2);
                 InpStr := '';
                 String_input(Prompt,InpStr,2,EscPress);
                 if EscPress then exit;
               until UpCase(InpStr[1]) in ['Y','N'];
               if UpCase(InpStr[1]) = 'Y' then Ok := true;
             end; {0}
      else
        begin
          Status := ER;                             { Indicate error if other }
          Write_status(Status);                          { file error occurs }
          Write(Bell);
          Msg := 'File Error - Please try again
        end; {else}
    end; {case}
  end; {while}
  Status := RE;                                   { Restore system status }
  Write_status(Status);
end; {Get_save_filename}

procedure Change_disp_matrix (Msg : Str_80;         { Allows selection of the }
                              var EscPress : boolean);  { matrix displayed on }
                                                                   { screen }
var
  I,Name   : MatrixNameType; { Loop control & selected matrix }
  OldAttr  : byte;           { Storage of the current text attribute }
  NewLine  : boolean;        { Line increment flag }
  Y        : byte;           { Row coordinate }

const
  ClrMsg : Str_80 = '

  procedure Select_option (var Name : MatrixNameType;     { Gets chosen matrix }
                           var EscPress : boolean);

  var
    Inchr,Inctl : char;  { Keyboard input variables }

  const
    OptionNo : byte = 0;
    OldOpNo : byte = 0;

    procedure Write_Cursor;          { Deletes old cursor and writes a new one }

    var
      Row,Col : byte;  { Coordinates }

    begin
      Row := 4 + OldOpNo div 2;                        { Calculate row number }
      if OldOpNo mod 2 = 0                             { Calculate the column }
        then Col := 1
        else Col := 32;
      Colour_box(Col,Row,(Col+29),Row,Yellow,Blue,false);      { Delete cursor }
      Row := 4 + OptionNo div 2;                       { Calculate row number }
      if OptionNo mod 2 = 0                            { Calculate the column }
        then Col := 1
        else Col := 32;
      Colour_box(Col,Row,(Col+29),Row,Yellow,Cyan,false);  { Write new cursor }
      OldOpNo := OptionNo;
    end; {Write_cursor}

    procedure Cursor_up;                       { Moves the cursor up one option }

    begin
      if OptionNo > 1
      then OptionNo := OptionNo - 2;
    end; {Cursor_up}

    procedure Cursor_down;                   { Moves the cursor down one option }

    begin
      if OptionNo + 2 <= Ord(MaxMatrixName)
        then OptionNo := OptionNo + 2;
    end; {Cursor_down}

    procedure Cursor_left;                   { Moves the cursor left one option }

    begin
      if OptionNo > 0
        then Dec(OptionNo);
    end; {Cursor_left}

    procedure Cursor_right;                 { Moves the cursor right one option }

    begin
      if OptionNo < Ord(MaxMatrixName)
        then Inc(OptionNo);
    end; {Cursor_right}

    procedure Move_home;              { Moves the cursor to the home position }

    begin
      OptionNo := 0;
    end; {Move_home}

    procedure Move_end;                { Moves the cursor to the end position }

    begin
      OptionNo := Ord(MaxMatrixName);
    end; {Move_end}

  begin {Select_option}
    OptionNo := Ord(CurrentMatrix);         { Set current option as default }
    repeat
      Write_cursor;                                    { Write the cursor }
      repeat                                      { Loop till key pressed }
        Update_lights;
        Update_time;
      until KeyPressed;
      Read_kbd(Inchr,Inctl);
      case Inctl of
        Cur_up     : Cursor_up;                        { Handle key input }
        Cur_down  : Cursor_down;
        Cur_left  : Cursor_left;
        Cur_right : Cursor_right;
        Home_key  : Move_home;
        End_key   : Move_end;
      end; {case}
    until Inctl in [CR,Esc];
    case Inctl of
      CR  : begin
              Name := MatrixNameType(OptionNo);        { Set matrix selected }
              EscPress := false;
            end; {CR}
      Esc : EscPress := true;
    end; {case}
  end; {Select_option}

begin {Change_disp_matrix}
  NewLine := false;
  Y := 4;
  if not (Status = ER)
    then Write_status(PO);                             { Set status to point }
  Name := CurrentMatrix;                               { Set Name to default }
  GotoXY(1,3);
  Write(Msg);                                                { Write message }
  Colour_box(1,4,80,24,Yellow,Blue,false);             { Clear number screen }
  Fill_screen(1,4,80,24,' ');                          {                     }
  OldAttr := TextAttr;                         { Save current text attribute }
  TextAttr := Get_attribute(1,4);
  for I := NONE to MaxMatrixName do                        { For each matrix }
  begin                                         { Print names in two columns }
    if NewLine then
    begin
      GotoXY(32,Y);
      Write(MatrixNameStr[I]);
      if Y < 25 then Inc(Y) else Exit;
    end {if then}
    else
    begin
      GotoXY(1,Y);
      Write(MatrixNameStr[I]);
    end; {if else}
    NewLine := not NewLine;
  end; {for}
  Select_option(Name,EscPress);                            { Select a matrix }
  if EscPress then
  begin
    RedrawScreen := true;
    exit;
  end; {if}
  TextAttr := OldAttr;                              { Restore text attribute }
  Write_status(Status);                                     { Restore status }
  GotoXY(1,3);
  Write(ClrMsg);                                         { Clear the message }
  Change_current_matrix(Name);        { Change to the new matrix for display }
end; {Change_disp_matrix}

procedure Display_error_message(ErrMsg : Str_80);

begin
  Write_status(ER);
  Write(Bell);
  GotoXY(1,2);
  Write(ErrMsg);
  GotoXY(1,3);
  Write('Press any key to continue ');
  repeat
  until Keypressed;
  Halt;
end; {Display_error_message}

procedure Write_text_window;

begin
  Colour_box(1,4,80,24,yellow,green,false);
  Fill_screen(1,4,80,24,' ');
  Window(1,4,80,24);
  GotoXY(1,1);
  TextAttr := Get_attribute(1,4);
end; {Write_text_window}

procedure Close_text_window;

begin
  Window(1,1,80,25);
  RedrawScreen := true;
end; {Close_text_window}

procedure Write_msg_in_window(Msg : Str_80);

begin
  WriteLn(Msg);
end; {Write_msg_in_window}

procedure Write_factor_number(Number : word);

begin
  Write('Factor: ',Number);
  GotoXY(13,WhereY);
  Write('Iteration: ');
end; {Write_factor_number}

procedure Update_iteration_number(IterationNo : word);

begin
  GotoXY(24,WhereY);
  Write(IterationNo);
end; {Update_iteration_number}

procedure Finished_factor( var PVar,PCumVar : extended;
                           var SigLevel : real);

var
  PVarStr,CVarStr,SLStr : string;

begin
  GotoXY(29,WhereY);
  Str(PVar:7:3,PVarStr);
  Str(PCumVar:7:3,CVarStr);
  Str(SigLevel:5:1,SLStr);
  WriteLn('Variance:',PVarStr,'%  _variance:',CVarStr,'%  %_:',SLStr,'%');
end; {Finished_factor}

end. {PFAVid}
```

3.13. Unit MATHUNIT

```
unit Mathunit;

{The code contained in this unit is Copyright by T.G.Brockwell, 1989-92. All rights reserved.}

{$O+,F+}

{*****************************************************************}

interface

{*****************************************************************}

uses Overlay,DOS,PFAGlobs,EMSDat,Utils,PFAVid,IO_Unit;

procedure Pca (AbsoluteError : extended);

{*****************************************************************}

implementation

{*****************************************************************}

procedure Covariance_matrix (var Rows,Cols : word;
                             var TotVar : extended);

                                    { Calculates the covariance matrix }
                                    { and enters it into the data }
                                    { structure with a name of RESI. }

{----------------------}

var
  Found                 : boolean;
  I,J,K                 : word;
  IMax,KMax             : word;
  Element,Point1,Point2 : extended;

{----------------------}

begin
  TotVar := 0;
  IMax := Cols;                       { Set outer loop values }
  KMax := Rows;
  Element := 0;                      { Initialise data point. }
  for I := 1 to IMax do              { Matrix multiplication loops. }
  begin
    for J := 1 to I do
    begin
      for K := 1 to KMax do
      begin
        Found := Get_val(DATA,K,J,Point1);  { Read the two }
        if not Found then exit;             { data points, }
        Found := Get_val(DATA,K,I,Point2);
        if not Found then exit;
        Element := Element + (Point1 * Point2);      { multiply together and }
      end; {for K}                                   { add to data point. }
    Found := Set_val(RESI,I,J,Element);     { Write point to }
    if I <> J then                          { residual matrix. }
      Found := Set_val(RESI,J,I,Element);   { Mirror point in matrix }
    if not found then exit;
    if I = J then TotVar := TotVar + Element;
    Element := 0;                           { Re-initialise data point. }
    end; {for J}
  end; {for I}
end; {Covariance_matrix}

{*****************************************************************}

procedure Set_Z_to_residual( var Cols : word);
                               { Produces the covariance matrix which }
                               { is identical to RESI at the outset of }
                               { Pca. }

{----------------------}

var
  I         : word;
  K         : word;
  Element   : extended;

{----------------------}

begin
  for I := 1 to Cols do                  { Make every element of Z }
  begin                                       { identical to RESI. }
    for K := 1 to Cols do
    begin
      Found := Get_val(RESI,I,K,Element);
      Found := Set_val(COVA,I,K,Element);
    end; {for K}
  end; {for I}
end; {Set_Z_to_residual}

{*****************************************************************}

procedure Initialise_factor_matrix (var Cols   : word;
                                    var LoopNo : word);

                                 { Assigns the first approximation value }
                                 { for the eigenvector. }

{----------------------}

var
  Point1  : extended;
  I       : word;
  Found   : boolean;

{----------------------}

begin
  Point1 := sqrt(Cols)/Cols;   { Calculate initial approximation. }
  for I := 1 to Cols do          { Loop through the entries setting them to }
  begin                          { their appropriate values. }
    Found := Set_val(LOAD,LoopNo,I,Point1);
  end; {for I}
end; {Initialise_factor_matrix}

{****************************************************************}

procedure Calc_intermediate_matrix (var Cols   : word;
                                    var LoopNo : word);

                                   { Calculates a better approximation }
                                   { to the eigenvector but mixed with }
                                   { the eigenvalue. }

{-----------------------}

var
  I       : word;
  K       : word;
  Found   : boolean;
  Element : extended;
  Point1  : extended;
  Point2  : extended;

{-----------------------}

begin
  for I := 1 to Cols do          { For each entry of the intermediate matrix, }
  begin
    Element := 0;                { set its value to zero. }
    for K := 1 to Cols do
    begin
      Found := Get_val(RESI,I,K,Point1);
      Found := Get_val(LOAD,LoopNo,K,Point2);
      Element := Element + Point1 * Point2;        { Calculate the new point. }
    end; {for K}
    Found := Set_val(INTM,I,1,Element);
  end; {for I}
end; {Calc_intermediate_matrix}

{****************************************************************}

procedure Calc_normalisation_const (var Cols      : word;
                                    var NormConst : extended);

                                      { Extracts the eigenvalue from }
                                      { the eigenvector and returns it }
                                      { to the calling routine. }

{-----------------------}

var
  Found     : boolean;
  I         : word;
  Point1    : extended;

{-----------------------}

begin
  NormConst := 0;              { Initialise the eigenvalue. }
  for I := 1 to Cols do
  begin
    Found := Get_val(INTM,I,1,Point1);    { Read each point and }
    NormConst := NormConst + Point1 * Point1;     { calculate constant. }
  end; {for I}
  NormConst := Sqrt(NormConst);                   { Calculate eigenvalue. }
end; {Calc_normalisation_const}

{****************************************************************}

procedure Calc_eigenvector (var Cols      : word;
                            var NormConst : extended);
                                      { Uses the eigenvalue to produce }
                                      { a better approximation to the }
                                      { eigenvector. }

{-----------------------}

var
  Found   : boolean;
  I       : word;
  Element : extended;

{-----------------------}

begin
  for I := 1 to Cols do
  begin
    Found := Get_val(INTM,I,1,Element);  { For each entry, }
    Element := Element / NormConst;              { calculate eigenvector. }
    Found := Set_val(INTM,I,1,Element);
  end; {for I}
end; {Calc_eigenvector}

{****************************************************************}

function Test_for_completed_factor (var NormConst : extended;
                                    var LoopNo    : word;
                                    var Cols      : word) : boolean;

                                          { Tests to see if the best }
                                          { approximation has been }
                                          { found and returns true }
                                          { if so. }

{-----------------------}

var
  I        : word;
  Found    : boolean;
  Finished : boolean;
  Point1   : extended;
  Point2   : extended;

{-----------------------}

begin
```

```
  Test_for_completed_factor := false;          { Set function to unfinished. }
  I := 0;
  repeat
    Inc(I);
    Found := Get_val(INTM,I,1,Point1);
    Found := Get_val(LOAD,LoopNo,I,Point2);
    Finished := (Abs(Abs(Point1) - Abs(Point2)) < FacExTest); { Test for complete }
  until (not Finished) or (I = Cols);                 { extraction of factor. }
  if Finished then
  begin
    Found := Set_val(COMP,EigRow,LoopNo,NormConst);       { Store eigenvalue. }
  end; {if}
  Test_for_completed_factor := Finished;
end; {Test_for_completed_factor}

{******************************************************************}

procedure Rewrite_factor_matrix (var Cols   : word;
                                 var LoopNo : word);

                                  { Writes the latest approximation }
                                  { to the factor matrix. }

{-----------------------}

var
  I      : word;
  Found  : boolean;
  Point1 : extended;

{-----------------------}

begin
  for I := 1 to Cols do
  begin
    Found := Get_val(INTM,I,1,Point1);        { Write eigenvector }
    Found := Set_val(LOAD,LoopNo,I,Point1);   { to Factor matrix. }
  end; {for I}
end; {Rewrite_factor_matrix}

{******************************************************************}

procedure Calc_residual_matrix (var Cols      : word;
                                var LoopNo    : word;
                                var NormConst : extended);

                                       { Removes the contribution of }
                                       { the newly calculated eigenvector }
                                       { from the rest of the matrix. }

{-----------------------}

var
  Found  : boolean;
  Point1 : extended;
  Point2 : extended;
  Point3 : extended;
  I,K    : word;

{-----------------------}

begin
  for I := 1 to Cols do
  begin
    Found := Get_val(LOAD,LoopNo,I,Point2);
    for K := 1 to Cols do
    begin
      Found := Get_val(RESI,I,K,Point1);
      Found := Get_val(LOAD,LoopNo,K,Point3);
      Point1 := Point1 - (Point2 * Point3 * NormConst); { Remove factors }
      Found := Set_val(RESI,I,K,Point1);                { contribution from }
    end; {for K}                                        { Residual matrix. }
  end; {for I}
end; {Calc_residual_matrix}

{******************************************************************}

function Residual_matrix_zero_test (var LoopNo : word;
                                    var Cols   : word) : boolean;

                                          { Tests to see if all the }
                                          { factors have been removed }
                                          { from the data and returns }
                                          { true to the calling routine }
                                          { if so. }

{-----------------------}

var
  Found     : boolean;
  Extracted : boolean;
  I,K       : word;
  Point1    : extended;

{-----------------------}

begin
  Residual_matrix_zero_test := true;
  if LoopNo = Cols then exit;                   { If all possible factors are }
  I := 0;                                       { extracted then leave. }
  repeat
    Inc(I);
    K := 0;
    repeat
      Inc(K);
      Found := Get_val(RESI,I,K,Point1);      { Search every entry of the }
      Extracted := (Abs(Point1) < ResNulTest);{ Residual matrix to see if }
    until (Extracted = false) or (K = Cols);  { factors have been extracted. }
  until (Extracted = false) or (I = Cols);
  Residual_matrix_zero_test := Extracted;
end; {Residual_matrix_zero_test}

{******************************************************************}

procedure Calc_abstract_row_matrix (var Rows   : word;
                                    var Cols   : word;
                                    var LoopNo : word);
                                    { Calculates the complementary matrix }
                                    { to the factor matrix allowing the data }
                                    { to be rebuilt using any number of }
                                    { factors. }

{----------------------}

var
  Found  : boolean;
  I,J,K  : word;
  Point1 : extended;
  Point2 : extended;
  Point3 : extended;

{----------------------}

begin
  for I := 1 to Rows do
  begin
    for J := 1 to LoopNo do
    begin
      Point1 := 0;
      for K := 1 to Cols do
      begin
        Found := Get_val(DATA,I,K,Point2);
        Found := Get_val(LOAD,J,K,Point3);
        Point1 := Point1 + (Point2 * Point3);     { Calculate element. }
      end; {for K}
      Found := Set_val(SCOR,I,J,Point1);
    end; {for J}
  end; {for I}
end; {Calc_abstract_row_matrix}


{*****************************************************************}

procedure Real_and_other_errors (var LoopNo   : word;
                                 var Rows,Cols : word;
                                 var TotVar    : extended);

                                     { Calculates the Real and Imbedded }
                                     { errors and the Indicator function }
                                     { for the extracted eigenvalues. }

{----------------------}

var
  SumOfEigenvalues     : extended;
  Eigenvalue           : extended;
  RealError            : extended;
  ImbeddedError        : extended;
  IndicatorFunc        : extended;
  Found                : boolean;
  I                    : word;

{----------------------}

begin
  SumOfEigenvalues := 0;
  for I := 1 to (LoopNo - 1) do
  begin
    Found := Get_val(COMP,EigRow,I,Eigenvalue);
    SumOfEigenvalues := SumOfEigenvalues + Eigenvalue;
    RealError := sqrt((TotVar-SumOfEigenvalues) / (Rows * (Cols - I)));
    ImbeddedError := RealError * sqrt(I / Cols);
    IndicatorFunc := RealError / ((Cols - I) * (Cols - I));
    Found := Set_val(COMP,RERow,I,RealError);
    Found := Set_val(COMP,IERow,I,ImbeddedError);
    Found := Set_val(COMP,INDRow,I,Indicatorfunc);
  end; {for I}
end; {Real_and_other_errors}

{*****************************************************************}

procedure Mean_and_sd_of_data_matrix (var Rows,Cols : word;
                                      var Mean,Sd   : extended);

                                       { Calculates the mean and standard }
                                       { deviation of the data in the data }
                                       { matrix and returns those values to }
                                       { the calling procedure. }

{----------------------}

var
  Found             : boolean;
  Row,Col           : word;
  Value,Sum,SumSq   : extended;
  Number            : word;

{----------------------}

begin
  Sum := 0;  SumSq := 0;                            { Initialise variables. }
  Number := Rows * Cols;                      { Calculate number of entries. }
  for Row := 1 to Rows do
  begin
    for Col := 1 to Cols do
    begin
      Found := Get_val(DATA,Row,Col,Value);               { Read entry. }
      Sum := Sum + Value;                                 { Total the entries. }
    end; {for Col}
  end; {for Row}                                            { Calculate mean. }
  Mean := Sum / Number;
  for Row := 1 to Rows do
  begin
    for Col := 1 to Cols do
    begin
      Found := Get_val(DATA,Row,Col,Value);                { Read entry. }
      Value := Value - Mean;               { Calculate the difference from the mean }
      SumSq := SumSq + Value * Value;     { Sum the squares of the differences. }
    end; {for Col}
  end; {for Row}
  Sd := sqrt(SumSq / (Number - 1));                      { Calculate the Sd. }
end; {Mean_and_sd_of_data_matrix}

{*****************************************************************}
```

```
procedure Misfits (var Cols,Rows,LoopNo : word);
                                          { Calculates the number of data }
                                          { points which lie more than }
                                          { 3Sd's outside the data when }
                                          { reproduced with an incremented }
                                          { series of eigenvectors. }

{----------------------}

var
  Found         : boolean;
  Mean,Sd,Sd3   : extended;
  Value         : extended;
  Point1        : extended;
  Point2        : extended;
  Point3        : extended;
  NoOfMisfits   : extended;
  I,J,K,L       : word;

{----------------------}

begin
  Mean_and_sd_of_data_matrix(Rows,Cols,Mean,Sd);          { Get mean and Sd. }
  Sd3 := 3 * Sd;                                        { Calculate test limit }
  for I := 1 to LoopNo do
  begin
    NoOfMisfits := 0;
    for J := 1 to Rows do
    begin
      for K := 1 to Cols do
      begin
        Value := 0;
        for L := 1 to I do
        begin
          Found := Get_val(SCOR,J,L,Point1);
          Found := Get_val(LOAD,L,K,Point2);
          Value := Value + Point1 * Point2;
        end; {for L}
        Found := Get_val(DATA,J,K,Point3);
        if abs(Value - Point3) >= Sd3 then
            NoOfMisfits := NoOfMisfits + 1;
      end; {for K}
    end; {for J}
    Found := Set_val(COMP,MisRow,I,NoOfMisfits);
  end; {for I}
end; {Misfits}

{********************************************************************}

procedure Standard_error_in_eigenvalue (var Rows,Cols,LoopNo : word;
                                        var AbsoluteError    : extended);

                                          { Calculate the standard error }
                                          { in eigenvalue using an }
                                          { estimate of the absolute }
                                          { error. }

{----------------------}

var
  Found         : boolean;
  SigSq,SigZ    : extended;
  SqAbsErr      : extended;
  DSIGj,DSIGk   : extended;
  Cmj,Cmk,Dsq   : extended;
  I,J,K,L,M     : word;

{----------------------}

begin
  if AbsoluteError = 0 then exit;
  SqAbsErr := sqr(AbsoluteError);
  for I := 1 to Cols do
  begin
    Found := Get_val(COVA,I,I,Dsq);
    Dsq := Dsq * SqAbsErr;
    Found := Set_val(INTM,I,1,Dsq);
  end; {for I}
  for M := 1 to LoopNo do
  begin
    SigSq := 0;
    for J := 1 to Cols do
    begin
      Found := Get_val(LOAD,M,J,Cmj);
      Cmj := sqr(Cmj);                  { square of eigenvector element }
      for K := 1 to Cols do
      begin
        Found := Get_val(INTM,J,1,DSIGj);
        case (j=k) of
          true  : SigZ := 4 * DSIGj;
          false : begin
                    Found := Get_val(INTM,K,1,DSIGk);
                    SigZ := DSIGj + DSIGk;
                  end; {case : false}
        end; {case}
        Found := Get_val(LOAD,M,K,Cmk);
        SigSq := SigSq + (Cmj * sqr(Cmk) * SigZ);
      end; {for K}
    end; {for J}
    SigSq := sqrt(SigSq);
    Found := Set_val(COMP,SEERow,M,SigSq);
  end; {for M}
end; {Standard_error_in_eigenvalue}

{********************************************************************}

procedure Errors_in_decomposition (Rows,Cols,LoopNo : word;
                                   AbsoluteError,TotVar : extended);


const
  Msg : array [1..3] of Str_80 =
              ('          Real error, Imbedded error & Indicator function',
               '          Misfits',
               '          Standard error in eigenvalue');


begin
  Write_msg_in_window(Msg[1]);
  Real_and_other_errors(LoopNo,Rows,Cols,TotVar);
  Write_msg_in_window(Msg[2]);
{  Misfits(Cols,Rows,LoopNo);    commented out as a waste of time !!!!!}
  Write_msg_in_window(Msg[3]);
  Standard_error_in_eigenvalue(Rows,Cols,LoopNo,AbsoluteError);
end; {Errors_in_decomposition}

{*****************************************************************}

procedure Clear_matrices;

begin
  Found := Delete_matrix(COVA);      { Delete any existing matrices }
  Found := Delete_matrix(RESI);      { No warning is given about the }
  Found := Delete_matrix(LOAD);      { loss of any unsaved data. }
  Found := Delete_matrix(SCOR);
  Found := Delete_matrix(COMP);
  Found := Delete_matrix(INTM);
end; {Clear_matrices}

{*****************************************************************}

procedure Matrix_init( Rows,Cols : word;
                  var Found : boolean);

var
  MatrixTypeStr : string;
  HeapToUse     : longint;

begin
  HeapToUse := Cols * SizeOf(extended);
  Initialise_matrix( INTM,Cols,1,HeapToUse,REXV,'INTARRAY.$$$');
  Found := Type_of_array(INTM,MatrixTypeStr);
  if not Found then exit;
  Write_msg_in_window(('Intermediate matrix stored '+ MatrixTypeStr));
  HeapToUse := Cols * Cols * SizeOf(extended);
  Initialise_matrix( RESI,Cols,Cols,HeapToUse,REXV,'RESARRAY.$$$');
  Found := Type_of_array(RESI,MatrixTypeStr);
  if not Found then exit;
  Write_msg_in_window(('Residual matrix stored '+ MatrixTypeStr));
  Initialise_matrix( COVA,Cols,Cols,DefHeapToUse,EXVN,'COVARRAY.$$$');
  Initialise_matrix( LOAD,Cols,Cols,DefHeapToUse,EXVN,'LODARRAY.$$$');
  Initialise_matrix( SCOR,Rows,Cols,DefHeapToUse,EXVN,'SCOARRAY.$$$');
  Initialise_matrix( COMP,CompRows,Cols,DefHeapToUse,EXVN,'COMARRAY.$$$');
end; {Matrix_init}

{*****************************************************************}

procedure Calc_variance( var NormConst,TotVar,CumVar : extended;
                         var Pvar,PCumVar : extended;
                         var LoopNo : word);

var
  Found   : boolean;

begin
    PVar := (NormConst/TotVar)*100;
    CumVar := CumVar + NormConst;
    PCumVar := (CumVar/TotVar)*100;
    Found := Set_val( COMP,PVarRow,LoopNo,PVar);
    Found := Set_val( COMP,PCVarRow,LoopNo,PCumVar);
end; {Calc_variance}

{*****************************************************************}

procedure F_test_eigenvalue( Rows,Cols,LoopNo : word;
                             NormConst,TotVar,CumVar : extended;
                             var SigLevel : real);

var
  J           : word;    { Loop control variable for weight calculation. }
  S           : word;    { Minimum of Rows and Cols. }
  R,C         : integer; { Integer values of Rows and Cols for calculation. }
  K1,K2       : integer; { Degrees of Freedom for F-test. }
  Fvalue      : real;    { F ratio for F-test. }
  REV,REVPool : real;    { Reduced eigenvalue and error eigenvalue pool. }
  Sum         : real;    { Eigenvalue pool weighting value. }
  Prob        : real;    { Area in tail of F distribution. }
  Found       : boolean;

begin
  Fvalue := 0; SigLevel := 0; Sum := 0;    { Variable initialization. }
  R := Rows; C := Cols;
  case (Rows > Cols) of
    true  : S := Cols;
    false : S := Rows;
  end; {case}
  K1 := 1; K2 := (S-LoopNo);                                  { D of F. }
  if LoopNo = S then exit;           { Exit for undefined final eigenvalue. }
  for J := Succ(LoopNo) to S do          { Pool weighting calculation loop. }
    Sum := Sum + (R - J + 1) * (C - J + 1);
  REV := NormConst / ((R - LoopNo + 1) * (C - LoopNo + 1)); { Reduced eig. }
  REVPool := (TotVar - CumVar) / Sum;                       { Pool eig. }
  if (REVPool = 0) then exit; { Exit for factors < dimensions (pure data). }
  Fvalue := REV / REVPool;                                  { F ratio. }
  Prob := F_prob(Fvalue,K1,K2);    { Calculate area in tail of F dist. }
  SigLevel :=(1-Prob) * 100;       { Calculate 1 tailed significance level. }
  Found := Set_val(COMP,SLRow,LoopNo,SigLevel);
end; {F_test_eigenvalue}

{*****************************************************************}

procedure Pca (AbsoluteError : extended);
                              { This procedure controls the flow of }
                              { the routines necessary to decompose }
                              { the data. }

{-----------------------}

const
  Msg : array [1..6] of Str_80 = ('Calculating Covariance matrix...',
                                  'Completed',
                                  'Decomposing data...',
                                  'Decomposition completed',
                                  'Calculating Errors...',
                                  'Error calculation completed');

var
```

```
  LoopNo         : word;      { Current factor number of calculations. }
  IterationNo    : word;      { Current iteration of factor extraction. }
  Rows,Cols      : word;      { Data matrix dimensions }
  TotVar         : extended;  { Total variance in data }
  CumVar         : extended;  { Sum of eigenvalues found so far. }
  NormConst      : extended;  { Eigenvalue }
  Finished       : boolean;   { Flag for completed factor extraction. }
  Found          : boolean;   { General boolean operator. }
  TimeStr        : Str_80;    { Elapsed time of decomposition. }
  SigLevel       : real;      { % Significance level of current factor. }
  Pvar           : extended;  { % variance of current factor. }
  PCumVar        : extended;  { % cumulative variance accounted for. }

{-----------------------}

begin
  Elapsed_time_timer(On,TimeStr);
  Clear_matrices;                                   { Delete existing matrices. }
  EscPress := false;
  CumVar := 0;
  Found := Matrix_dimensions(DATA,Rows,Cols);         { Find matrix dimensions. }
  if not Found then exit;                             { Leave if no DataMatrix. }
  Write_text_window;
  Matrix_init( Rows,Cols,Found);                  { Initialise matrices for data. }
  if not Found then exit;
  Write_msg_in_window(Msg[1]);
  Covariance_matrix(Rows,Cols,TotVar);{ Calculate Z in RESI matrix for speed. }
  Write_msg_in_window(Msg[2]);
  Write_msg_in_window(Msg[3]);
  Set_Z_to_residual( Cols);                            { Mirror RESI to COVA. }
  LoopNo := 0;
  repeat
    Inc(LoopNo);
    Write_factor_number(LoopNo);
    Initialise_factor_matrix(Cols,LoopNo);
    IterationNo := 0;
    repeat
      Inc(IterationNo);
      Update_iteration_number(IterationNo);
      Calc_intermediate_matrix(Cols,LoopNo);
      Calc_normalisation_const(Cols,NormConst);
      Calc_eigenvector(Cols,NormConst);
      Finished := Test_for_completed_factor(NormConst,LoopNo,Cols);
      if not Finished then Rewrite_factor_matrix(Cols,LoopNo);
      Check_for_esc;
    until Finished or EscPress;
    Calc_variance(NormConst,TotVar,CumVar,PVar,PCumVar,LoopNo);
    F_test_eigenvalue( Rows,Cols,LoopNo,NormConst,TotVar,CumVar,SigLevel);
    Finished_factor( PVar,PCumVar,SigLevel);
    Calc_residual_matrix(Cols,LoopNo,NormConst);
    if LoopNo <> Cols
      then Finished := Residual_matrix_zero_test(LoopNo,Cols);
   until Finished or EscPress;
   Write_msg_in_window(Msg[4]);
   Calc_abstract_row_matrix(Rows,Cols,LoopNo);
   Write_msg_in_window(Msg[5]);
   Errors_in_decomposition(Rows,Cols,LoopNo,AbsoluteError,TotVar);
   Write_msg_in_window(Msg[6]);
   Elapsed_time_timer(Off,TimeStr);
   Write_msg_in_window(TimeStr);
   Close_text_window;
 end; {Pca}

 end.
```

3.14. Unit MENU

```
unit menu;

{The code contained in this unit is Copyright by T.G.Brockwell, 1989-92. All rights reserved.}


{$O+,F+}

interface

uses Overlay,Crt,DOS,PFAGlobs,EMSDat,Utils,Vid_util,Mathunit,
     TargTest,IO_Unit,PFAUtils,PFAVid;

procedure Menu_system;

procedure Change_column_headings;

procedure Set_column_width;

implementation

const
  NoOfMenus = 8;          { Total number of menus in system }
  HeadingWidth = 13;      { Maximum width of each entry in a menu }
  MaxNoOfOptions = 6;     { No. of options possible in 80 cols }
  OptionGap = '  ';       { Space between options in the menu }
  CsrClrBkg = Magenta;    { Colour constant for the cursor background }
  CsrClrFrg = White;      { Colour constant for the cursor text }

type
  MenuNoRecord = record               { Record for menu stack }
                   Number : byte;   { Menu number }
                   Option : byte;   { Option No. in that menu }
                 end;
  MenuRecord = record          { Menu record, one for each option entry possible }
                 Heading    : string[HeadingWidth];  { Option name }
                 Message    : Str_80;                { Option description }
                 NextMenu   : MenuNoRecord;          { Points to next menu }
                 QuitTree   : boolean;       { Flag to quit or stay in menu }
                 HelpMsgNo  : byte;          { Relevant help message }
               end;
  MenuLineType = array [1..MaxNoOfOptions] of MenuRecord; { Makes up a menu }
  MenuListType = array [1..NoOfMenus] of MenuLineType;    { Array of menus }
  MenuStackType = array [1..NoOfMenus] of MenuNoRecord;   { Pointer stack }

var
  MenuList      : MenuListType;     { Array of menus }
  MenuStack     : MenuStackType;    { Stack of route through menus }
  StackIndex    : byte;             { Current stack position }
  LaunchOption  : boolean;          { Flag to run an option }
  QuitMenu      : boolean;          { Flag to leave menu system after operation }


procedure Select_from_menu;      { Look after selection methods from the menu }

var
inchr,inctl : char;     { Contain key codes entered }
PrevOption  : byte;     { Last selected option }
ChangeMade  : boolean;  { Flag to cause update of menu }

  procedure Highlight_option (Option : byte);                 { Puts cursor on }
                                                             { selected option }
  var
    TopX,LwrX : byte;  { Start and finish coords of the cursor }

    procedure Get_coords (Option : byte);          { Calculates TopX and LwrX }

    var
      I,Offset  : byte;

    begin
      with MenuStack[StackIndex] do              { Using current Stack entry }
      begin
        Offset := 0;                                     { Initialize offset }
        for I := 1 to Pred(Option) do          { Add up chars of each option }
          Offset := Offset + Length(MenuList[Number,I].Heading)
                  + Length(OptionGap);
        TopX := Succ(Offset);
        LwrX := Offset + Length(MenuList[Number,Option].Heading);
      end; {with}                                              { Set coords }
    end; {Get_coords}

  begin {Highlight_option}
    Get_coords(PrevOption);
    Colour_box(TopX,2,LwrX,2,LightGray,Black,false);       { Erase old cursor }
    Get_coords(Option);
    Colour_box(TopX,2,LwrX,2,CsrClrFrg,CsrClrBkg,false);   { Write new cursor }
  end; {Highlight_option}

  procedure Process_control_key;                 { Acts on control key input }

  begin
    ChangeMade := true;                               { Flag to update menu }
    with MenuStack[StackIndex] do                { Using current stack entry }
    begin
      with MenuList[Number,Option] do          { and associated menu option }
      begin
        case inctl of
          Cur_left : if Option = 1 then Write(Bell)
                     else Option := Pred(Option);         { Move cursor left }
          Cur_right : if MenuList[Number,Succ(Option)].Heading = ''
                      then Write(Bell)
                      else Option := Succ(Option);       { Move cursor right }
          Esc       : begin
                        if StackIndex = 1 then     { Leave menu if top of tree }
                        begin
                          QuitMenu := true;
                          LaunchOption := true;
                        end
                        else
                        StackIndex := Pred(StackIndex);       { Or move up one }
                      end; {Esc}
          CR        : begin
                        if NextMenu.Number = 0           { If bottom of tree }
                        then LaunchOption := true        { then run procedure }
                        else                                             { or }
                        begin                             { move down a level }
                          MenuStack[Succ(StackIndex)] := NextMenu;
                          StackIndex := Succ(StackIndex);
                        end;
                      end; {CR}
          else        begin                       { If none of the above keys }
                        Write(Bell);                                   { Beep }
                        ChangeMade := false;         { Flag no change to menu }
                      end; {else}
        end; {case}
      end; {with MenuList}
    end; {with MenuStack}
  end; {Process_control_key}

  procedure Find_option_letter;               { Finds options by first letters }

  var
    I          : byte;
    Found      : boolean;

  begin
    Inchr := UpCase(Inchr);                                { Force to capitals }
    with MenuStack[StackIndex] do                  { Using current stack entry }
    begin
      I := 0;
      repeat
        I := Succ(I);                             { Search through each option }
        Found := UpCase(MenuList[Number,I].Heading[1]) = Inchr;
      until Found or (I = MaxNoOfOptions);          { Until a match is found }
      if not Found then Write(Bell)                   { If no match then Beep }
      else                                                               { or }
      begin
        ChangeMade := true;                              { Flag to update menu }
        if MenuList[Number,I].NextMenu.Number = 0           { Check for bottom }
        then                                                        { of tree }
        begin
          LaunchOption := true;           { if so then flag to run procedure }
          Option := I;                          { Update option no. on stack }
        end {if then}
        else
        begin                { if not then increment stack and enter next menu }
          MenuStack[Succ(StackIndex)] := MenuList[Number,I].NextMenu;
          StackIndex := Succ(StackIndex);
        end; {if else}
      end; {else}
    end; {with}
  end; {Find_option_letter}

begin {Select_from_menu}
  with MenuStack[StackIndex] do                    { Using current stack entry }
  begin
    with MenuList[Number,Option] do              { and the option it points to }
    begin
      repeat
        Highlight_option(Option);           { Display cursor on current option }
        repeat
          Update_time;
          Update_lights;
        until KeyPressed;
        Read_Kbd(Inchr,Inctl);                               { Get a key stroke }
        if Inctl <> NullChr then Process_control_key    { Handle control keys }
        else Find_option_letter;                                 { and letters }
      until ChangeMade;                                { until a change is made }
      QuitMenu := QuitTree;   { Check if leaving the menu system after action }
    end; {with MenuList}
  end; {with MenuStack}
end; {Select_from_menu}

procedure Write_menu (MenuNo : MenuNoRecord);     { Writes menu to second line }
                                                                 { of screen }
var
  I : byte;

begin
  GotoXY(1,2);                                              { Move to menu line }
  I := 1;
  repeat
    Write(MenuList[MenuNo.Number,I].Heading,OptionGap);   { Write each option }
    I := Succ(I);                                       { separated by a space }
  until MenuList[MenuNo.Number,I].Heading = '';         { till no more entries }
end; {Write_menu}

procedure Write_message (MenuNo : MenuNoRecord);     { Writes a message on the }
                                                    { third line of the screen }
begin
  GotoXY(1,3);                                           { Move to message line }
  with MenuNo do
  begin
    Write(MenuList[Number,Option].Message);                     { Write message }
  end; {with}
end; {Write_message}

procedure Decompose;                                 { Starts the eigenanalysis }

begin
  Write_status(WA);                                    { Change status to wait }
  PCA(AbsError);                                                   { Decompose }
  Write_status(Status);                                       { Restore status }
end; {Decompose}

procedure Error_entry( Prompt,Msg : Str_80;
                       var TmpExt : extended);
                              { Allows entry of error estimate for the data }
                              { or test vector }


const
  Min : extended = 3.4e-4932; { Range checking constants }
  Max : extended = 1.1e4932;  { for string conversion }
  Int : boolean = false;      { Flag for extended conversion }
  AllowZero : boolean = true; { Flag to allow return of 0 }

var
  InpStr       : Str_255;  { Keyboard input }
  Code         : integer;  { String to number conversion error code }
  Ok,EscPress  : boolean;
```

```
begin
  Clr_menu;
  TextAttr := Get_attribute(1,2);
  Ok := false;
  Str(TmpExt,InpStr);
  while not Ok do
  begin
    GotoXY(1,3);
    Write(Msg);
    GotoXY(1,2);
    String_input(Prompt,InpStr,25,EscPress);         { Get input from keyboard }
    Ok := String_to_number(InpStr,TmpExt,Min,Max,Code,Int,AllowZero);
    if not Ok and (Code = 0)
      then Ok := (TmpExt = 0);
    if not Ok then
      begin
        Write_status(ER);                      { Indicate an error condition }
        Write(Bell);
      end; {if else}
  end; {while}
  Write_status(Status);                              { Restore status }
end; {Error_entry}

procedure Quit;

var
  InpStr   : Str_255;
  EscPress : boolean;

begin
  if FileChanged then
  begin
    Write(Bell);
    Clr_menu;
    while not (UpCase(InpStr[1]) in ['Y','N']) do
    begin
      GotoXY(1,2);
      String_input('Loose changes Y/N ? ',InpStr,2,EscPress);
      if EscPress then exit;
    end; {while}
    if UpCase(InpStr[1]) = 'Y' then EndSession := true;
  end {if then}
  else EndSession := true;
end; {Quit}

procedure Set_column_width;   { Sets the width of columns displayed on screen }

const
  Min : longint = 10;     { Range checking constants }
  Max : longint = 23;     { for string conversion }
  Int : boolean = true;   { Flag for integer conversion }
  AllowZero : boolean = false;

var
  Ok,EscPress : boolean;
  Code        : integer; { Error code for string/integer conversion }
  InpStr      : Str_255; { Keyboard input }
  Prompt,Msg  : Str_80;  { Text messages }

begin
  Clr_menu;
  TextAttr := Get_attribute(1,2);
  Ok := false;
  Prompt := 'Column width = ';
  Msg := 'Enter a value for the width of each column. Range of values 10-23';
  Str(ColWidth,InpStr);
  while not Ok do
  begin
    GotoXY(1,3);
    Write(Msg);
    GotoXY(1,2);
    String_input(Prompt,InpStr,3,EscPress);              { Get keyboard input }
    ColWidth := longint(ColWidth);                          { String to byte }
    Ok := String_to_number(InpStr,ColWidth,Min,Max,Code,Int,AllowZero);
    ColWidth := byte(ColWidth);                        { conversion routine }
    if not Ok then
      begin
        Write_status(ER);  { Indicate error if out of range }
        Write(Bell);
      end; {if else}
  end; {while}
  Write_status(Status);                                   { Restore status }
  CursorPos.Column := 1;                              { Move cursor to home }
  OldCurPos.Column := 1;
  RedrawScreen := true;                             { Flag to redraw screen }
end; {Set_column_width}

procedure Change_column_headings;       { Swaps between letters and numbers }
                                                   { for column headings }
begin
  Headings := not Headings;                              { Toggle headings }
  Refresh := true;                                { Flag to redraw numbers }
end; {Change_column_headings}

procedure Get_facextest;

var
  Prompt,Msg : Str_80;

begin
  Prompt := 'Accuracy = ';
  Msg := 'Enter a value for the test for completion of factor extraction ';
  Error_entry( Prompt,Msg,FacExTest);
end; {Get_facextest}

procedure Get_resnultest;

var
  Prompt,Msg : Str_80;

begin
  Prompt := 'Minimum = ';
  Msg := 'Enter a value for the smallest meaningful data value ';
  Error_entry( Prompt,Msg,ResNulTest);
end; {Get_resnultest}

procedure Get_abserror;
```

```
var
  Prompt,Msg : Str_80;

begin
  Prompt := 'Absolute Error = ';
  Msg := 'Enter an estimate for the error in the data set';
  Error_entry( Prompt,Msg,AbsError);
end; {Get_abserror}

procedure Get_vecterr;

var
  Prompt,Msg : Str_80;

begin
  Prompt := 'Vector Error = ';
  Msg := 'Enter an estimate for the error in the test vector';
  Error_entry( Prompt,Msg,VectErr);
end; {Get_vecterr}

procedure Get_nofacs;

var
  Prompt,Msg : Str_80;
  TmpExt     : extended;

begin
  Prompt := 'Number of factors = ';
  Msg := 'Enter the number of factors to be used to describe the data';
  TmpExt := NoFacs;
  Error_entry( Prompt,Msg,TmpExt);
  NoFacs := Trunc(TmpExt);
end; {Get_nofacs}

procedure Calc_TKON;

begin
  Write_status(WA);
  Calculate_TKON;
  Write_status(Status);
end; {Calc_TKON}

procedure Target_test;

begin
  Write_status(WA);
  Test_vector;
  Write_status(Status);
end; {Target_test}

procedure IT_Test;

var
  Finished : boolean;
  Tmp : string;

begin
  Write_status(WA);
  GotoXY(1,2);
  ClrEol;
  Write('Iterative target testing');
  GotoXY(1,3);
  ClrEol;
  Finished := false;
  ITTInit;
  repeat
    ITT( Finished);
    GotoXY(1,3);
    ClrEol;
    Write('Press any key for next iteration or Esc to finish');
    Write_number_window;
    repeat until KeyPressed;
    Check_for_esc;
    GotoXY(1,3);
    ClrEol;
  until Finished or EscPress;
  Write_status(Status);
end; {IT_Test}

procedure Menu_system;

var
  Selector : word;                         { Code to identify procedure to run }
  Code     : word;
  S1,S2 : str_5;                                     { Precursors of Selector }

begin
  StackIndex := 1;                                     { Menu initialization }
  MenuStack[StackIndex].Number := 1;                   { Code                }
  MenuStack[StackIndex].Option := 1;                   {                     }
  LaunchOption := false;                               {                     }
  QuitMenu := false;                                   {                     }
  Status := ME;
  repeat
    repeat
      Clr_menu;
      Write_menu(MenuStack[StackIndex]);
      Write_message(MenuStack[StackIndex]);
      Select_from_menu;
    until LaunchOption;
    with MenuStack[StackIndex] do              { Using current stack entry }
    begin
      Str(Number,S1);                    { Calculate first part of Selector }
      Str(Option,S2);                   { Calculate second part of Selector }
      S1 := S1 + S2;
      Val(S1,Selector,Code);                         { Create Selector code }
    end; {with}
    case Selector of                               { Use Selector to launch }
      21 : Decompose;
      22 : begin end;{Target_testing;}
      23 : Get_abserror;
      24 : Get_facextest;
      25 : Get_resnultest;
      31 : Import_file(Native);
      32 : Export_file(Native);
      41 : Change_disp_matrix('Select the Matrix to be displayed',EscPress);
      42 : Set_column_width;
      43 : Change_column_headings;
      51 : Quit;
      61 : Import_file(Lotus);
      62 : Import_file(ASCII);
      71 : Export_file(Lotus);
      72 : begin end;{ASCII_export;}
      81 : Get_nofacs;
      82 : Get_vecterr;
      83 : Calc_TKON;
      84 : Target_test;
      85 : IT_Test;
    end; {case}
    LaunchOption := false;
  until QuitMenu;                         { Switch to quit or stay with menu }
  Status := RE;
  Clr_menu;
end; {Menu_system}

procedure Read_menu_file;                  { Reads menu information from disk }

const
  FileName = 'Menu.dat';

var
  Found : boolean;
  DirInfo : SearchRec;
  InFile : file of MenuListType;

begin
  FindFirst(FileName,AnyFile,DirInfo);            { Searches for file in DIR }
  if DosError in[2,18] then
  begin
    ClrScr;
    Write('File MENU.DAT not found in default directory',Bell);
    Halt;                                          { Terminal program error }
  end; {if}
  Assign(InFile,FileName);
  Reset(InFile);
  Read(InFile,MenuList);
  Close(InFile);
end; {Read_menu_file}

begin
  Read_menu_file;                                  { Unit initialisation }
end. {Menu}
```

3.15. Unit KEYOPS

```
unit keyops;

{The code contained in this unit is Copyright by T.G.Brockwell, 1989-92. All rights reserved.}

{$O+,F+}

interface

uses Overlay,Crt,PFAGlobs,Utils,Vid_util,EMSDat,PFAUtils,PFAVid,Menu;

procedure Handle_key;

implementation

procedure Move_cursor_up;             { Moves the cursor up within the screen }
                                             { boundaries or moves the screen }
begin
  if CursorPos.Row > 1 then                                { If inside screen }
    begin
      CursorPos.Row := Pred(CursorPos.Row);            { Move cursor position }
      Draw_cursor;                                            { Redraw cursor }
    end {if then}
  else
    begin
      if (CursorPos.Row = 1) and (Origin.Row > 1) then    { If outside screen }
        begin                                           { but inside data set }
          Origin.Row := Pred(Origin.Row);            { Move screen boundaries }
          CursorPos.Row := 1;                         { Reset cursor position }
          Refresh := true;                           { Flag to redraw numbers }
        end {if then}
      else                                   { If outside screen and data set }
        Write(Bell);                                                  { Bleep }
    end; {if else}
end; {Move_cursor_up}

procedure Move_cursor_down;         { Moves the cursor down within the screen }
                                             { boundaries or moves the screen }
begin
  if CursorPos.Row < NoOfRows then                         { If inside screen }
    begin
      CursorPos.Row := Succ(CursorPos.Row);            { Move cursor position }
      Draw_cursor;                                            { Redraw cursor }
    end {if then}
  else
    begin
      if (CursorPos.Row = NoOfRows)                   { If outside screen but }
      and ((Origin.Row+NoOfRows) <= MatrixDim.Row) then     { inside data set }
        begin
          Origin.Row := Succ(Origin.Row);            { Move screen boundaries }
          CursorPos.Row := NoOfRows;                  { Reset cursor position }
          Refresh := true;                           { Flag to redraw numbers }
        end {if then}
      else                                   { If outside screen and data set }
        Write(Bell);                                                  { Bleep }
    end; {if else}
end; {Move_cursor_down}

procedure Move_cursor_left;         { Moves the cursor left within the screen }
                                             { boundaries or moves the screen }
begin
  if CursorPos.Column > 1 then                             { If inside screen }
    begin
      CursorPos.Column := Pred(CursorPos.Column);      { Move cursor position }
      Draw_cursor;                                            { Redraw cursor }
    end {if then}
  else
    begin
      if (CursorPos.Column = 1) and (Origin.Column > 1) then     { If outside }
        begin                                      { screen but inside data }
          Origin.Column := Pred(Origin.Column);      { Move screen boundaries }
          CursorPos.Column := 1;                      { Reset cursor position }
          Refresh := true;                            { Flag to draw3 numbers }
        end {if then}
      else                               { If outside screen and outside data }
        Write(Bell);                                                  { Bleep }
    end; {if else}
end; {Move_cursor_left}

procedure Move_cursor_right;       { Moves the cursor right within the screen }
                                             { boundaries or moves the screen }
begin
  if CursorPos.Column < NoOfCols then                      { If inside screen }
    begin
      CursorPos.Column := Succ(CursorPos.Column);               { Move cursor }
      Draw_cursor;                                            { Redraw cursor }
    end {if then}
  else
    begin
      if (CursorPos.Column = NoOfCols)    { If outside screen and inside data }
      and ((Origin.Column + NoOfCols) <= MatrixDim.Column) then
        begin
          Origin.Column := Succ(Origin.Column);      { Move screen boundaries }
          CursorPos.Column := NoOfCols;               { Reset cursor position }
          Refresh := true;                           { Flag to redraw numbers }
        end {if then}
      else                               { If outside screen and outside data }
        Write(Bell);                                                  { Bleep }
    end; {if else}
end; {Move_cursor_right}

procedure Move_page_up;                         { Moves up one screen of data }

begin
  if Origin.Row >= NoOfRows then               { If room to move a whole page }
    Origin.Row := Origin.Row - NoOfRows                          { then do so }
  else
    Origin.Row := 1;                           { Else set page to top of data }
  Refresh := true;                                   { Flag to redraw numbers }
end; {Move_page_up}

procedure Move_page_down;                     { Moves down one screen of data }
```

```
begin
  if (Origin.Row + NoOfRows) < (MatrixDim.Row - Pred(NoOfRows))  { If room to }
    then Origin.Row := Origin.Row + NoOfRows   { move a whole page then do so }
  else
    Origin.Row := MatrixDim.Row - Pred(NoOfRows);   { Else set bottom of data }
  Refresh := true;                                  { Flag to redraw data }
end; {Move_page_down}

procedure Move_page_right;                      { Moves right one screen of data }

begin
  if ((Origin.Column + (2*NoOfCols) < MatrixDim.Column))      { If room to move a }
  then Origin.Column := Origin.Column + NoOfCols      { whole page then do so }
  else Origin.Column := MatrixDim.Column - Pred(NoOfCols);  { Else move right }
  Refresh := true;                             { of data and flag to redraw data }
end; {Move_page_right}

procedure Move_page_left;                        { Moves left one screen of data }

begin
  if Origin.Column > NoOfCols then             { If room to move a whole page }
    Origin.Column := Origin.Column - NoOfCols                     { then do so }
  else
    Origin.Column := 1;                           { Else move to left of data }
  Refresh := true;                                   { Flag to redraw numbers }
end; {Move_page_left}

procedure Move_to_home;                          { Moves to top left of data set }

begin
  Origin.Row := 1;                                      { Set data boundaries }
  Origin.Column := 1;
  CursorPos.Row := 1;                                 { Reset cursor position }
  CursorPos.Column := 1;
  Refresh := true;                                   { Flag to redraw numbers }
end; {Move_to_home}

procedure Move_to_end;                       { Moves to bottom right of data set }

begin
  Origin.Row := MatrixDim.Row - Pred(NoOfRows);         { Set data boundaries }
  Origin.Column := MatrixDim.Column - Pred(NoOfCols);
  CursorPos.Row := NoOfRows;                            { Set cursor position }
  CursorPos.Column := NoOfCols;
  Refresh := true;                                      { Flag to redraw data }
end; {Move_to_end}

procedure Move_to_top;                  { Moves to top of screen in same column }

begin
  Origin.Row := 1;                                        { Move data boundary }
  CursorPos.Row := 1;                                    { Set cursor position }
  Refresh := true;                                   { Flag to redraw numbers }
end; {Move_to_top}

procedure Move_to_bottom;            { Moves to bottom of screen in same column }

begin
  Origin.Row := MatrixDim.Row - Pred(NoOfRows);           { Move data boundary }
  CursorPos.Row := NoOfRows;                             { Set cursor position }
  Refresh := true;                                      { Flag to redraw data }
end; {Move_to_bottom}

procedure Move_to_far_left;             { Moves to rightmost column in same row }

begin
  Origin.Column := 1;                                     { Move data boundary }
  CursorPos.Column := 1;                                 { Set cursor position }
  Refresh := true                                    { Flag to redraw numbers }
end; {Move_to_far_left}

procedure Move_to_far_right;             { Moves to leftmost column in same row }

begin
  Origin.Column := MatrixDim.Column - Pred(NoOfCols);     { Move data boundary }
  CursorPos.Column := NoOfCols;                          { Set cursor position }
  Refresh := true;                                   { Flag to redraw numbers }
end; {Move_to_far_right}

procedure Edit_cell;

const
  Min : extended = 3.4e-4932; { Range checking constants }
  Max : extended = 1.1e4932;  { for string conversion }
  Int : boolean = false;      { Flag for extended conversion }
  AllowZero : boolean = true; { Flag to allow a 0 return }

var
  Prompt,Msg  : Str_80;    { Text messages }
  InpStr      : Str_255;   { Keyboard input }
  Code        : integer;   { String to number conversion error code }
  Ok,EscPress : boolean;
  Found       : boolean;
  Row,Col     : word;
  Value       : extended;

begin
  Clr_menu;
  TextAttr := Get_attribute(1,2);
  Ok := false;
  Prompt := 'Edit value = ';
  Msg := 'Edit an existing value or enter a new value';
  Row := Origin.Row + Pred(CursorPos.Row);
  Col := Origin.Column + Pred(CursorPos.Column);
  Found := Get_val(CurrentMatrix,Row,Col,Value);
  if not Found then exit;
  Str(Value,InpStr);
  while not Ok do
  begin
    GotoXY(1,3);
    Write(Msg);
    GotoXY(1,2);
    String_input(Prompt,InpStr,25,EscPress);        { Get input from keyboard }
    Ok := String_to_number(InpStr,Value,Min,Max,Code,Int,AllowZero);
    if not Ok and (Code = 0)
      then Ok := (AbsError = 0);
    if not Ok then
      begin
```

```
        Status := ER;                            { Indicate an error condition }
        Write_status(Status);
        Write(Bell);
      end; {if else}
  end; {while}
  Clr_menu;
  Found := Set_val(CurrentMatrix,Row,Col,Value);
  Status := RE;
  Write_status(Status);                                      { Restore status }
  Refresh := true;
  FileChanged := true;
end; {Edit_cell}

procedure Goto_cell;                   { Moves the cursor to a designated cell }

var
  CellAddress   : Str_255; { Keyboard input string }
  Prompt,Msg    : Str_80;  { Text messages }
  Ok,EscPress   : boolean; { Control flags }
  Row,Col       : word;    { Coordinates }

  procedure Set_coords;

  begin
    case (Col < Origin.Column) or (Col > Origin.Column+Pred(NoOfCols)) of
      false : CursorPos.Column := Succ(Col - Origin.Column);
      true  : begin
                CursorPos.Column := 1;
                if Col > MatrixDim.Column - NoOfCols then
                begin
                  Origin.Column := MatrixDim.Column - Pred(NoOfCols);
                  CursorPos.Column := Succ(Col - Origin.Column);
                end {if then}
                else Origin.Column := Col;
              end; {true}
    end; {case}
    case (Row < Origin.Row) or (Row > Origin.Row+Pred(NoOfRows)) of
      false : CursorPos.Row := Succ(Row - Origin.Row);
      true  : begin
                CursorPos.Row := 1;
                if Row > MatrixDim.Row - NoOfRows then
                begin
                  Origin.Row := MatrixDim.Row - Pred(NoOfRows);
                  CursorPos.Row := Succ(Row - Origin.Row);
                end {if then}
                else Origin.Row := Row;
              end; {true}
    end; {case}
  end; {Set_coords}

begin
  Ok := false;                                          { Initialization code }
  EscPress := false;                                    {                     }
  Prompt := 'Cell address = ? ';                        {                     }
  Msg := 'Enter cell address to goto';                  {                     }
  CellAddress := '            ';                        {                     }
  TextAttr := Get_attribute(1,2);                       {                     }
  while not Ok do
  begin
    GotoXY(1,3);
    Write(Msg);                                           { Write message line }
    GotoXY(1,2);
    String_input(Prompt,CellAddress,12,EscPress);                { Get address }
    case EscPress of
      false : begin
                Ok := Parse_coords(CellAddress,Row,Col);  { Split coordinates }
                if (Row > MatrixDim.Row)                    { Check dimensions }
                  or (Col > MatrixDim.Column)
                  then Ok := false;
                if not Ok then                          { If out of dimensions }
                begin
                  Status := ER;                               { Indicate error }
                  Write_status(Status);
                  Write(Bell);
                  Msg := 'Incorrect cell format or address out of range';
                end; {if}
              end; {false}
      true  : Ok := true;
    end; {case}
  end; {while}
  Clr_menu;
  if not EscPress
    then Set_coords;
  Refresh := true;
  Status := RE;
  Write_status(Status);
end; {Goto_cell}

procedure Call_change_matrix;

var
  Msg      : Str_80;
  EscPress : boolean;

begin
  Msg := 'Select the matrix to be displayed';
  Change_disp_matrix(Msg,EscPress);
end; {Call_change_matrix}

procedure Handle_key;                    { Deals with key input from ready mode }

var
  inchr,inctl : char;                { Contain key code and extended scan code }

begin
  Read_kbd(inchr,inctl);
  case inctl of
    Slash     : Menu_system;
    Cur_up    : Move_cursor_up;
    Cur_down  : Move_cursor_down;
    Cur_left  : Move_cursor_left;
    Cur_right : Move_cursor_right;
    Page_up   : Move_page_up;
    Page_down : Move_page_down;
    Tab,
    Ctrl_rght : Move_page_right;
    Shift_tab,
    Ctrl_left : Move_page_left;
    Home_key  : Move_to_home;
    End_key   : Move_to_end;
    Ctrl_pgup : Move_to_top;
    Ctrl_pgdn : Move_to_bottom;
    Ctrl_home : Move_to_far_left;
    Ctrl_end  : Move_to_far_right;
    F1_key    : begin                                { Reserved for help system }
                end;
    F2_key    : Edit_cell;
    F5_key    : Goto_cell;
    F6_key    : Change_column_headings;
    F7_key    : Set_column_width;
    F8_key    : Call_change_matrix;
  end; {case}
end; {Handle_key}

end. {keyops}
```

```
program TFA;

{$F+}

uses Overlay,Crt,DOS,OvInit,PFAGlobs,PFAVid,Vid_Util,PFAUtils,Keyops,EMSDat;

{ $O PFAGLOBS}
{ $O UTILS}
{ $O VID_UTIL}
{ $O OPDOS}
{ $O OPLARRAY}
{ $O EMSDAT}
{ $O PFAUTILS}
{ $O PFAVID}
{ $O EPAIMPRT}
{ $O IO_UNIT}
{ $O MATHUNIT}
{ $O LOTUSFIL}
{ $O MENU}
{ $O KEYOPS}

begin
  TextMode(CO80);
  Hide_Cursor;
  repeat
    if RedrawScreen then Draw_screen;
    if Refresh then Write_number_window;
    if Keypressed then Handle_key;
    Update_lights;
    Update_time;
  until EndSession;
  Free_EMS;
  Wave_by_by;
end.
```

Appendix 4 : Description of EPA file format

4.1. EPA Data format

The following is a description of the EPA file format used by VG Analytical for transfer of data to PC systems. Some of the information is gained from reverse engineering a FORTRAN program supplied by VG. Other aspects of the file format have been learnt by observation of the data files produced by the data system.

The EPAF program used to convert the data system files to EPA format accepts switches, two of which are known. The /D switch adds <CR LF> (ASCII #13,#10) after each 80 char logical record. This switch was not used but files transferred had the CR,LF pair added. This means that the conversion either defaults to /D or that Kermit file transfer does this automatically. The /A switch is referred to in relation to file headers appearing in the middle of a file. It is therefore concluded that the switch is used to append new data to the end of an existing file.

The format of the file is described line by line below. Each description gives the position in the line (characters are numbered from 1 to 80 along the line), the length of each field and the Pascal variable type contained in the field.

4.1.1. File header:

The file header is composed of four lines. Each line is 80 characters long or 82 characters if the CR,LF pair has been added.

A file header is identifiable by a scan number of 0 and has a format identical to a normal scan header.

A file header does not necessarily appear only at the beginning of the file, if the /A switch is used in the conversion then it may appear in the middle of the file.

4.1.1.1. First header line

Position from,	to	Length bytes	Type	Description
1	12	12	Char	Name of original data file
13	13	1	Char	scan number prefix (#)
14	18	5	Integer	scan number (right aligned)
19	20	2	Char	blank
21	28	8	Char	MM/DD/YY date
29	29	1	Char	blank
30	31	2	Integer	hour of run start
32	32	1	Char	time separator (:)
33	34	2	Integer	minutes of run start
35	80	46	Char	spaces

4.1.1.2. Second header line

Position from,	 to	Length bytes	Type	Description
1	64	64	Char	Sample identification
65	74	10	Char	blanks
75	80	6	Char	Instrument name

Notes: The Sample ID field appears to contain information about filename, calibration file, account no. and type of scan (eg MAG).

4.1.1.3. Third header line

Position from,	 to	Length bytes	Type	Description
1	64	64	Char	Run conditions
65	74	10	Char	blanks
75	80	6	Real	seconds per scan for this file

4.1.1.4. Fourth header line

Position from,	 to	Length bytes	Type	Description
1	6	6	Char	blanks
7	14	8	Char	Analyst
15	20	6	Char	blanks
21	28	8	Char	Submitted by
29	34	6	Char	blanks
35	42	8	Char	Account no
43	48	6	Char	blanks
49	68	20	Char	Formula
69	73	5	Char	blanks
74	76	3	Integer	Lowest scanned mass (right aligned)
77	77	1	Char	blank
78	80	3	Integer	Highest scanned mass (right aligned)

Notes: The fourth header line shows large discrepancies between the fields shown in the specification (above) and those observed. For this line the only fields which appear to be in the correct place are those defining the scan range.

4.1.2. Scan data:

The scan data is composed of a series of blocks, of variable number of lines, each representing one scan of data.

Data blocks are repeated for successive scan numbers until the end of file or another scan header is reached.

Data blocks are composed of a scan header and a mass/intensity list.

The last entry of a data block has a zero mass/intensity pair and is padded to the end of the line with ASCII character #32 (space).

Each data block may contain a maximum of 999 masses which gives a maximum number of 80 char lines of 77 plus the scan header.

4.1.2.1. Scan header

Position		Length	Type	Description
from,	to	bytes		
1	12	12	Char	Name of original data file
13	13	1	Char	scan number prefix (#)
14	18	5	Integer	scan number (right aligned)
19	20	2	Char	blank
21	28	8	Char	MM/DD/YY date
29	29	1	Char	blank
30	31	2	Integer	hour of run start
32	32	1	Char	time separator (:)
33	34	2	Integer	minutes of run start
35	39	5	Char	blanks
40	42	3	Integer	mins after start at end scan
43	43	1	Char	blank
44	45	2	Integer	seconds after start at end scan
46	51	6	Char	blank
52	55	4	Integer	Nominal mass of largest peak
56	56	1	Char	blank
57	65	9	Longint	Intensity of largest peak (before scaling)
66	70	5	Char	blanks
71	80	10	Longint	Reconstructed ion current = total intensity recorded in the scan before scaling

Notes: If the reconstructed ion current exceeds the maximum range of the instrument then the first character of the filename is overwritten with an * and the ion current field left blank. The data still follows the header but must be regarded as suspect.

4.1.2.2. Mass/Intensity list

Position from,	 to	Length bytes	Type	Description
1	3	3	Integer	Nominal mass (increasing order)
4	6	3	Integer	Intensity at above mass, scaled to intensity of base peak = 999
79	80	2	Integer	Line number of data

The fields nominal mass and intensity form a mass intensity pair six characters long. Every peak observed by the spectrometer has its mass rounded to the nearest integer and its intensity scaled to a percentage of the base peak for the scan to form the data pairs. The encoding occurs from low mass to high. When each line is filled with data pairs (13 pairs), a line number is appended to the end which is incremented for each succesive line to the maximum of 77.

Data continues in this fashion with a CR,LF pair after every 80 chars until a zero mass/intensity pair is met signifying the end of the data. The rest of the line is padded with blanks until character 78 where the two character integer gives the line number.

Because of the rounding of the mass number to the nearest integer it is possible to have two or more intensity pairs with the same mass number. It is necessary when converting the file for use to accomodate this or outliers will occur.

4.1.2.3. Sample EPA file

Listed below is the file header and the first scan record from a typical EPA format file.

```
TB11         #    0  10/30/90 12:10
   }  D:TB11     7070CAL   7070H     DQA/TS          MAG                  7070H
              #                   #=   G ?@A  #=   C   @  FERROC            6.61
LE RUN 2                          DQA/TS         }        CHNNNNYAN EDI   10 650
TB11         #  17   10/30/90 12:10      1 56          0  23887000       34827000
01403501500101600901701801808602700302899902900902900203222003900104001404100601
04200104300304300404400804500505400105500405600205700406000106700206900507000102
0710020810020820020830020840010950010970010000000                              03
```

Appendix 5 : Abstracts of papers and courses attended

The following courses were attended during the period of registration for PhD.
Lecture course on Mass Spectroscopy, given by Dr. E.Metcalfe, 1989.
6th COMETT Chemometrics School, Brugges, 1990.
The theory and practice of mass spectrometry, British Mass Spectrometry Society course, London school of Pharmacy, 1991.

The following are the abstracts of papers presented during the period of registration for PhD.

Paper presented at, Research Topics in Chemistry 1990, RSC sponsored meeting.

6.1. The use of factor analysis in the deconvolution of overlapping peaks in the UV-Vis spectra of selected transition metal ions.

One of the most commonly occurring problems in analytical chemistry is the quantification of a component in the presence of other interfering components, much work has been done to aid the analyst, both in developing methods of separating the component of interest, and also in developing numeric methods of data analysis to yield the single quantified component. Most methods of spectral deconvolution make assumptions about the peak shape in order to separate the components, which immediately compromises the precision of any results they produce. Factor analysis is a multivariate statistical technique that operates well in any system that may be described by a linear combination of factors and makes no assumptions as to the peak shape.

At low concentrations UV-Vis spectra provide an ideally suited system for factor analysis as all absorbances are produced by a linear combination of factors ($A = \varepsilon c$, at fixed path length) and absorbances are additive from the individual components. A series of spectra containing transition metal ions in differing concentrations in solution are digitized and subjected to factor analysis, the individual ions' spectra are searched for and identified and their concentrations found. The converse approach can also be taken and knowing the concentrations of the components the spectra of the components can be reproduced in isolation from the interference of other components in the mixture.

Paper presented at Analyticon 90, Olympia.

6.2. Target factor analysis for investigating chemical data.

The modern chemist is often faced with large amounts of data, due to many variables, and is asked to interpret it. Traditional methods such as plotting graphs and looking for trends are no longer adequate for such studies and powerful new methods have been developed, or adapted from other fields for use by the chemist. Factor analysis is one technique whic has been employed for hte analysis of multivariate data sets in chemistry. Having its roots in the behavioural sciences, the technique has been adapted for use with chemical data.

Target factor analysis is a combination of two mathematical procedures, principal component analysis and target transformation. Principal component analysis provides a mathematical method for describing the sources of variation in the data in decreasing order of significance, each source of variance is due to one of the underlying factors, which, when combined, make up the data. Every source of variance is accounted for such that the individual factors that make up the data are found, in order of importance to the data, and then the error inherent within the system is extracted as factors. The principal component analysis method is applicable to any data set whic has been produced by a linear combination of factors (i.e. each component of the data adds, without interaction with the others, to produce the measured result), which gives it enormous scope within the chemical field. Already use has been made of it in UV-Vis, IR, NIR, and Fluorescence spectroscopy where the resultant spectrum is simply the sum of each individual spectra, NMR, where again the spectra are simply additive, mass spectroscopy, gas chromatography, HPLC and many others.

One of the first benefits and indeed the original reason for employing factor analysis in chemistry was to discover the number of components which comprise the system, this may be achieved if the factors which are not due to error in the system can be chosen. Much work has been done in this area and awhole series of tests may be applied to the data in order to guide the chemist in that choice.

On discarding the factors due to error a useful side effect becomes apparent. As a substantial portion of the error has been removed from the data, then this technique may be used to improve the data set without knowledge of the source or type of any error. If however the error is left in the data then it is also possible to investigate the sources of error.

The end result of the principal component analysis is a series of factors which make up the data, however these factors are an abstract solution and have no basis in what chemists regard as real physical parameters such as concentration, dipole moment, boiling point, etc.

Target transformation provides the chemist with a means of converting the abstract factors from principal components analysis into a factor having real meaning to the chemist.

With the data reduced to its factors, application of target transformation may proceed. Here the chemist is able to draw upon his knowledge and experience by producint hypotheses about the fundamental nature of the system and testing them mathematically on the data. If, for example, data involving the retention times of a series of alkanes on some column packings were being investigated then it would be reasonable to assume that the boiling point of the original alkane may be a factor in determining the retention time. To test this assumption the boiling points of the alkanes are formed into a factor and then , using a mathematical transformation, it is attempted to convert the factors from the original data to match the test factor. Because of error in the data and the test values the correlation between test figures and transformed factor is never perfect but various functions may be calculated to show how closely they relate. If a good fit is obtained then the assumption is correct and boiling points are a factor in determining the retention time of the alkane.

This ability to test for a single factor in the data without knowing anything else about the system provides the chemist with a very powerful tool for investigating complex systems.

An explanation of the principles of target factor analysis will be given. A worked example of the technique used in the deconvolution of overlapping spectra of four transition metal ions with the subsequent determination of their concentrations will be shown.

Paper presented to the MoD Specialist Working Group -2 Molecular Spectroscopy, Bridgewater, 1990.

6.3. The application of Factor Analysis to Temperature Programmed Pyrolysis Mass Spectrometry Data (TPPyMS)

TPPyMS is currently used for the analysis of unknown polymer samples. The present use of complex polymer formulations however, poses problems with the interpretation of the data produced.

The intended objective of this research is to produce a method of mathematical analysis which will allow the number of components in a sample to be determined, compared against a library of known samples and if no match is found then an isolated spectrum produced to allow identification by normal methods.

The project has currently reached the point where some initial data has been run through the computer software to look at the amenability of the system to this form of analysis.

The results presented are from two runs of a single pure compound, the chlorendic anhydride derivative of ferrocene (FCA), pyrolysed under conditions made as reproducible as possible. The areas discussed include.

A short introduction giving the rationale behind Factor analysis and its ability to simplify complex data.

Reproducibility between runs and its effect on the identification of components.
The number of components present in the system as shown from the Factor analysis.
The information available from the different forms of data representation possible with Factor analysis.
The effect of background subtraction on the data.
The insights gained into the fragmentation of the sample.
The perturbations due to bulk effects in the probe
The use of target testing to identify individual components in the system.