Development of Data Acquisition and Analysis Methods for Chemical Acoustic Emission

by

David Bruce Sibbald

B. Sc. (Hons.), University of British Columbia, 1987

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF

THE REQUIREMENT FOR THE DEGREE OF

MASTER OF SCIENCE

in

THE FACULTY OF GRADUATE STUDIES

DEPARTMENT OF CHEMISTRY

We accept this thesis as conforming

to the required standard

THE UNIVERSITY OF BRITISH COLUMBIA

JULY 1990

In presenting this thesis in partial fulfilment of the requirements for an advanced degree at the University of British Columbia, I agree that the Library shall make it freely available for reference and study. I further agree that permission for extensive copying of this thesis for scholarly purposes may be granted by the head of my department or by his or her representatives. It is understood that copying or publication of this thesis for financial gain shall not be allowed without my written permission.

Department of Chemistry

The University of British Columbia
Vancouver, Canada

Date July 21, 1990

DE-6 (2/88)

Abstract

Acoustic Emission Analysis (AEA) is the study of the sonic (and ultrasonic) energy released by chemical systems in the form of transient waves, as the system attempts to (re)attain equilibrium. This area of chemistry, and chemical analysis, is ripe for fundamental studies since it has been little explored. The high potential of the technique as a non-invasive, non-destructive reaction monitoring scheme suggests that numerous applications will follow.

In this work, an apparatus and software have been constructed to monitor acoustic emission (AE) and collect and process AE data. A broad-band piezoelectric transducer was used to convert the acoustic signals to electrical waveforms which could be captured by a digital storage oscilloscope. These waveforms were then stored on an IBM-compatible computer for further analysis.

Analysis of the data was performed using pattern recognition techniques. The signals were characterized through the use of descriptors which can map each signal onto a multi-dimensional feature space. Visualization of the data structure in multi-dimensional space was accomplished using several methods. Hierarchical clustering was used to produce tree structures, known as dendrograms, which attempt to show clustering of the signals into various groups. Abstract factor analysis (AFA) - also called principal components analysis (PCA) - was used to project the data onto a two dimensional factor space to allow for direct viewing of structure in the multi-dimensional data.

Sodium hydroxide dissolution, aluminum chloride hydration and heat activation of Intumescent Flame Retardants (IFR's) were used to test the assembled hardware and to provide data to submit to the pattern recognition algorithms coded as part of this

work. The solid-solid phase transition of trimethylolethane (Trimet), and the liquid crystal phase transitions of two liquid crystals (α-ω-bis(4-n-decylaniline-benzilidene-4'-oxyhexane), and 4-n-pentyloxybenzylidene-4'-n-heptylaniline) were also monitored and the signals analyzed.

The pattern recognition software was able to extract much information from the acoustically emitting samples - information which would not have been apparent by using standard (uni- and bi-variate) methods of analysis. Chemical acoustic emission, coupled with pattern recognition analysis, will be able to provide the chemist with knowledge (qualitative, quantitative, kinetic, etc.) about chemical systems which are often difficult or impossible to monitor and analyze by other means.

Table of Contents

Legend to Tables

Legend to Figures

Figure Page

Figure Page

Figure Page

Figure Page

Acknowledgements

I wish to give my most heartfelt thanks to Dr. Adrian Wade. His cheerful and energetic attitude to this project has been most encouraging. I also wish to commend Dr. P. D. Wentzell. Pete was always willing to help or give advice; and when none was asked for, he related countless anecdotes which helped make the day a little "shorter".

Much appreciation is deserved by the people I worked with, for the jovial atmosphere that was created as well as for the assistance and camaraderie: Paul, Terrance, Oliver, Julie, Tony, Steve, Kevin, Ivan, Tim, Helen, Bruce, Patrick, and "little" Adrian.

It would be sadly remiss of me not to mention two people who really made U.B.C. a special place for me: Fred Mistry, for breaking up the day into manageable sized chunks; and Dudley Shallcross, for spraying coffee and listening to my advice on female psychology (usually at the same time).

A distant recognition is given to Ben Clifford, Gary Leong, and Sandi Clark for helping me get here in the first place. A deep appreciation is also offered to Sara Jane Biles for knowing just when and where to kick me when I needed it most.

Finally, I wish to dedicate this work to the two people without whom none of this would be possible, Jane and Bruce Sibbald. But for your love, compassion, and generosity, I would never have been able to try to reach so high.

I. Chemical Acoustic Emission

I.1 Introduction

Analytical chemistry is that branch of science that attempts to characterize chemical systems based on their behavior and properties. Physical attributes such as viscosity, density, mass, crystallinity and porosity are sometimes used to characterize a sample. Modern chemical analysis almost entirely depends on the interaction of the sample with some form of supplied energy. For example, studies of the interaction of samples with a beam of electromagnetic radiation have led to development of techniques such as atomic absorption, vibrational spectroscopy, fluorescence, phosphorescence, and refractive index measurement. Nuclear magnetic resonance, electron spin resonance, and mass spectroscopy are used to identify / characterize a sample based on its behavior in electric and/or magnetic fields.

Characterization of ongoing chemical reactions has been one of the more interesting problems of analytical chemistry. Time resolved methods such as differential scanning calorimetry (DSC), dilatometry, and electrochemical techniques have been employed as well as those techniques mentioned above.

Many chemical reactions release energy in some form as they occur. The usual forms of energy release considered are heat and light. Such processes can be monitored by techniques such as calorimetry or luminescence spectroscopy. Energy can also be released in the form of sound over a broad frequency range, presently known to be from a few Hz to over 1 MHz.

I.2 Acoustic Emission Studies

The human mind is very efficient at recognizing sound patterns. Both the cause and direction of a sound source are quickly identified by mental analysis of differences

in the sound characteristics. Acoustic parameters, formalized in terms such as pitch, tone, loudness, *etc.*, allow us to distinguish between a myriad of potential candidate sources for the sound. The ear is capable of detecting sound vibrations at frequencies in the range of about 10 Hz to 16 kHz. Vibrations at higher frequencies cannot be heard but still contain useful information. Recently, it was found that plants in need of water emit bursts of high frequency sound. When the plant is re-supplied with water, the sound ceases after the water columns in the stem are fully reestablished[1].

The earliest recording example of the use of sound in chemical analysis comes from Germany in *ca.* A.D. 1350, where in a manuscript containing instructions for the making of gun powder, the following information was presented:

> "If thou wilt try whether sulphur be good or not, take a lump of sulphur in thine hand and lift it to thine ears. If the sulphur crackle, so that thou hearest it crackle, then it is good; but if the sulphur keep silent and crackle not, then it is not good, and must be treated as thou shalt hear hereafter how it shall be prepared."[2]

Metallurgists are well aware of the value of sound in the production of materials. In 1936, Förster *et al.* observed that sound waves were generated during the formation of martensite[3]. The first laboratory investigation of the phenomenon of acoustic emission did not take place until 1953 when J. Kaiser reported his findings[4]. The term "Kaiser effect" describes the property of a material such that it will only generate transient mechanical waves (AE) when it is placed under a stress (or load) which is greater than that to which it has previously been exposed. This is illustrated in Figure 1. The upper trace shows the force, **F**, to which a sample is being subjected. The lower trace shows the acoustic energy, $\mathbf{V_{peak}}$ - in Volts, being emitted by the sample. The

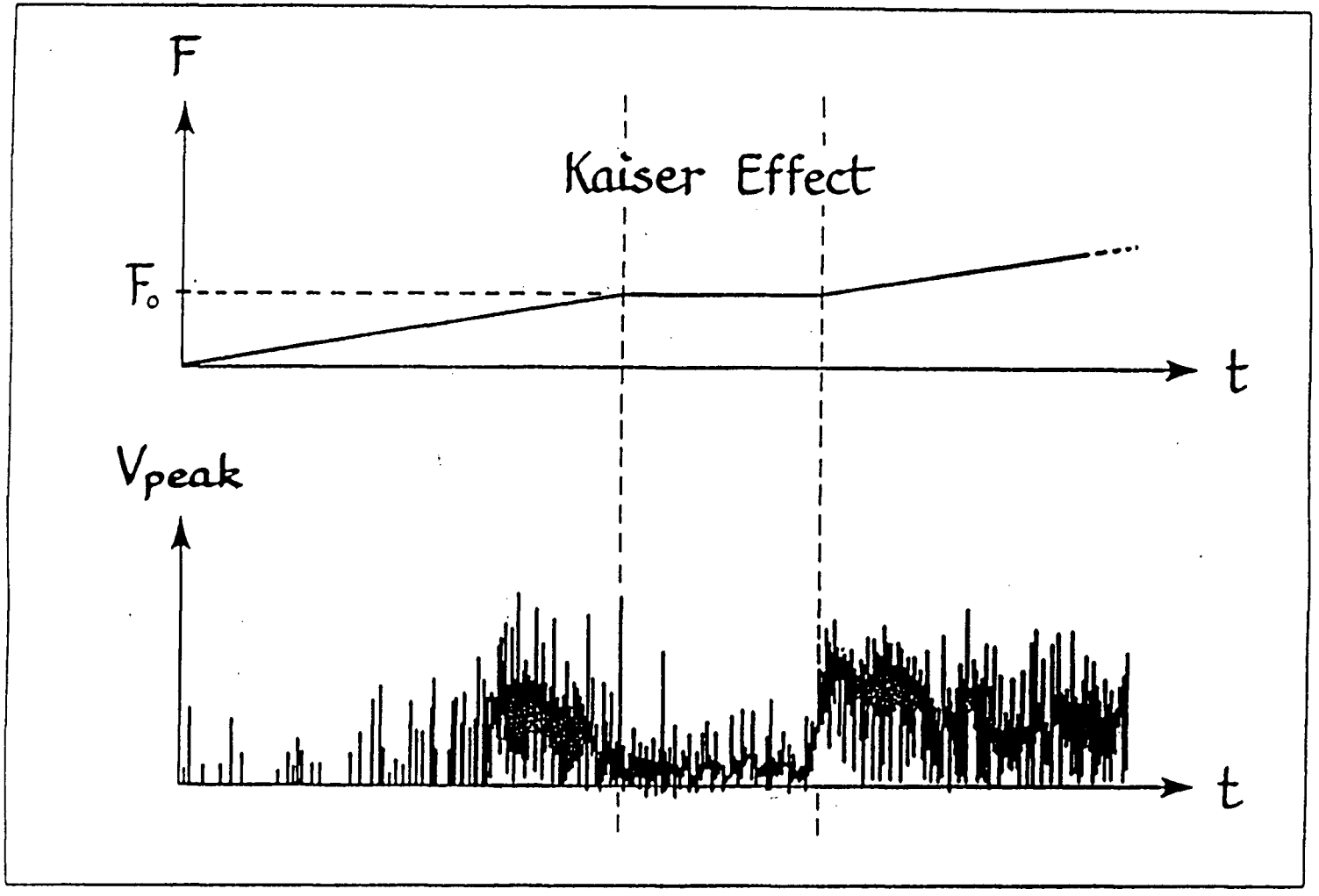


Figure 1) **The Kaiser Effect. A sample will release acoustic energy (indicated by the plot of peak Voltage, V_{PEAK}, versus time, t) only when subjected to a load, F, greater than that which it has previously experienced. Here one can see the level of AE drop off as the load is held constant at F_0.**

first load cycle involves ramping the force up to a value F_O. The load is held constant until the second load cycle at which time it is increased further. One can see that the sample was acoustically active during the first load cycle but did not emit further acoustic emissions until submitted to a greater load than had been achieved previously.

The application of acoustic emission to an engineering structure was first documented eleven years later by A. T. Green *et al.*[5]. The integrity of a Polaris Model A3 solid rocket motor case is poorly determined by conventional techniques because it is a filament wound structure (unlike steel and titanium rocket cases). In this case, high frequency piezoelectric accelerometers were attached directly to the Polaris chamber. By looking at the amount of acoustic activity at low operating pressures, it was possible to predict the burst (failure) pressure. Acoustic emission was thus introduced as a non-destructive technique for the testing of materials.

Because of its non-destructive nature, the use of acoustics for materials testing was enthusiastically examined for commercial applications[6-8]. The equipment for monitoring acoustic emission was already available. A large background of knowledge concerning acoustics and sound existed (the Journal of the Acoustical Society of America was founded in 1928) and the mathematics of the propagation of sound waves had been well studied. The need for a better understanding and detection in SONAR has resulted in a great theoretical background in the area of sound waves in water[9,10].

Many of the early acoustic methods relied not on the acoustic energy emitted (as for Kaiser's and Green's experiments) but on the attenuation of a supplied acoustic wave to which the sample was exposed. For example, the ultrasonic C-scan is performed by passing an acoustic pulse through the sample[11]. The pulse is reflected or scattered by defects or interfaces with separate regions of different acoustical

impedance. The signal will normally reflect from the back of the sample and return after the same time and with the same amplitude across the entire sample. It's failure to do so is caused by a defect, and thus a defect can be located[12].

Rayleigh waves have also been used for evaluation purposes. The surface acoustic wave (SAW) technique measures the attenuation of an acoustic signal travelling along the surface of a sample. Reflections from a crack in the surface or a flaw which lies just beneath the surface can be observed[13].

The last two decades have seen much progress in the use of Acoustic Emission (AE) for materials testing. The Journal of Acoustic Emission[14] was started in 1981 as a medium for the users of AE to publish their work.

Applications of AE have included the study of the intergranular cracking of steel. Nozue and Kishi counted the number of AE signals as well as their energy from a piece of tempered steel[15]. They discovered that the number of AE signals was proportional to the total number of micro-cracks and that the energy of the signals was nearly proportional to the crack area.

The human hearing system involves the use of two ears which allow one to determine the direction from which a sound is coming. Similarly, the location of faults in gas pipelines has been found by correlating the arrival of acoustic signals at multiple transducers placed at different locations on the pipeline[16]. This is the same technique that seismologists use to locate the epicenter of an earthquake. This type of destructive testing is widely used by the materials industry for quality control. For example, Hoa and Li report the use of AE in the testing of reinforced fibre composites[17].

A piezoelectric crystal will respond to an impulse with a particular resonant frequency. The resonant frequency depends on the mass of the crystal. A film deposited on the surface of the crystal acts to change the mass of the resonator and therefore changes the resonant frequency. This phenomenon has been used to provide a gas phase microgravimetric sensor[18]. The mass of a deposited film can be determined by measuring the change in resonant frequency of a piezoelectric crystal. This application has been extended to use in liquids. When a suitable chemically selective binding agent is incorporated into a film which is deposited on the surface of the crystal, the mass sensitive sensor becomes a chemically selective detector[19]. The extension of this technique to that of an highly selective immunosensor for antigens is discussed in an excellent review article[20].

Thermal analysis has been used to test a material for purity and integrity. The occurrence of sound during these experiments is also well documented. The existence of such emissions may be confirmed by a visit to any analytical laboratory where fire assay is performed, since as samples cool they emit profusely. Smith describes an apparatus to monitor the decrepitation of minerals[21]. This early technique is known today as thermosonimetry (TS)[22]. In this experiment, which is usually run simultaneously with differential scanning calorimetry (DSC), the acoustic activity of a sample is recorded as a function of temperature while the sample is heated and/or cooled. In this way, Clark[23] was able to characterize the phase transition of potassium dichromate that occurs at about 520 K. Lonvik[24] was able to show that the TS activity of a sample of brucite, $Mg(OH)_2$, was dependent on the site from which the sample had been mined. Also, it was shown that the frequency content of the acoustic signals differed between the samples.

Many chemical reactions also emit audible sound. Schoolboys are familiar with the nature of crystals obtained from a mixture of NH_4OH solution and I_2 such that it makes for a good prank when sprinkled on the floor in front of the chalkboard. In 1957, Ranke Madsen used his ear to determine the endpoint of a titration as acid was added to a carbonate solution[25]. In 1963, Belyaev *et al.* [26] reported acoustic activity during the crystalloluminescence of $Ba(ClO_3)_2$, LiF, and $K_3Na(SO_4)_2$. It was suggested that the luminescence was the result of the crystals cracking as they grew. However, when in 1978, van Ooijen *et al.* [27] noted acoustic activity during the precipitation of dichloro(pyrazine)zinc (II), the phenomenon was still largely a curiosity to chemists.

The observation of van Ooijen *et al.*[27] that the intensities of the signals were proportional to the concentrations of the reagents again suggested that AE could be applicable as a form of chemical analysis. In the late 70's, Betteridge *et al.* decided to investigate the phenomenon of acoustic emission to determine if AE was indeed widely observable in chemistry[28-30]. Forty-three chemical systems were investigated. These systems included the addition of $CuSO_4$ to a solution of sodium bicarbonate, the reduction of unsaturated hydrocarbons by $KMnO_4$, reactions of luminol with peroxide in the presence of copper ions, recrystallization of KCl, and the addition of Mg to a solution of ferric chloride. The results were very conclusive. Thirty-two of the systems studied were found to emit a detectable level of acoustic energy within the frequency range studied. Of the systems that were "silent", nine involved merely the mixing of two organic solvents such as CCl_4 and hexane. The setting of an epoxy resin and the addition of ferric chloride to phenol were the other systems that didn't emit detectable levels of acoustic energy. The results of the experiments were compared and pattern recognition techniques were used to group the systems into classes, showing that acoustic emission was able to be used for such a purpose. The individual signals from three of the systems were also compared and were sufficiently different to allow one to

distinguish between the different reactions. This key study proved conclusively that acoustic emission from chemical systems was not only a common occurrence, but also had analytical potential.

In 1984, Sawada *et al.* investigated AE from the gelation of sodium carbonate and calcium chloride[31]. They saw different acoustic behaviors as a function of time (see for example Figure 1) and attributed them to different mechanisms of the reaction. The acoustic emissions from the precipitation and dissolution of sodium thiosulfate, and the phase transitions of *p*-cresol, methyoxybenzilidene-4-*n*-butylaniline (MBBA), and water were also observed[32]. The results suggested that the occurrence of AE was linked to volume changes during the phase transition.

In 1986, Belchamber *et al.* [33] investigated one particular chemical system in detail with the intent of discovering whether AE was a useful quantitative tool for chemical analysis. The hydration of silica gel was chosen as this process emits very loud audible sounds, is rapid, and has analogues in industrial catalysis. Catalyst hydrolysis is difficult to monitor in-situ by other techniques. The level of AE activity was found to be dependent on factors such as particle size, amount of sample, initial water content, and temperature. For the first time, individual AE signals from a chemical reaction were captured, recorded, and analyzed in detail. Using pattern recognition techniques, and descriptive statistical factors (descriptors) which summarized aspects of the signals, frequency content, amplitude, and time-domain behaviour, they were able to classify the individual signals as being caused by particle fracture or by gas evolution. Through non-linear calibration curves, this work showed that AE could be quantitatively applied to chemical analysis and might be suited to industrial process monitoring.

Chemical Acoustic Emission is one of the primary research interests of Dr. Adrian Wade at the University of British Columbia. Graduate students have investigated acoustic emission from chemical processes such as the gelation of PVAA, PVAc and PAA, the dry-blending of PVC, the 81°C solid-solid phase transition of trimethylolethane, phase transitions of the liquid crystal α-ω-bis(4-n-decylaniline-benzilidene-4'-oxyhexane), the heat activation of intumescent fire retardants, and the 43.6°C polymorphic transition of hexachloroethane[34]. Current work in the group includes the use of acoustic emission generated by a forced extension and bending of plastics and composite materials on two "stresser" systems. This is an extension of the work in reference 30. Also, a video system is being used to correlate acoustic activity with visual changes in the sample. Another study seeks to use AE to improve the efficiency of drop weight testing procedures. Chemometric[A] techniques such as hierarchical clustering and factor analysis figure largely in the analysis of the signals. These tools were implemented as part of this present thesis work.

I.3 Apparatus for Monitoring Chemical Acoustic Emission

The apparatus used for AE experimentation has had to evolve as researchers have sought to obtain more information from each experiment, and have been provided with or have developed chemometric routines capable of automatically analyzing the larger data sets. The early AE systems were simply a microphone attached to the sample[21] (Figure 2). This enabled one to tell if and when AE was occurring as well as giving some indication as to the rate of the process being studied. The data were collected in real time and different processes could be noticed if they had different intensities in the time domain. The utility of this apparatus is limited by its poor quantitative ability and lack of frequency information. Interpretation in the absence of

A. Chemometrics is defined as "the chemical discipline that uses mathematical, statistical and other methods employing formal logic to: (a) design or select optimal measurement procedures and experiments; and (b) to provide maximum chemical information by analyzing chemical data."[35]

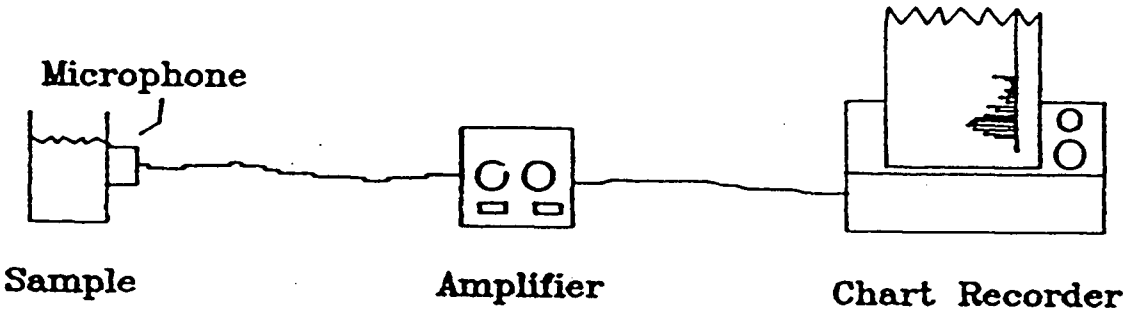


Figure 2) A simple apparatus for acoustic emission monitoring. The acoustic activity is converted into an electrical impulse by the microphone which can be amplified and used to drive a chart recorder.

these is somewhat subject to operator bias[36]. Detection of the presence of simultaneous processes is not generally possible.

With the addition of an integrator (or computer to fulfil this function - Figure 3), the AE apparatus has the same abilities but can provide more information[37]. The integration trace which quantified the total AE energy output allowed for the determination of the driving force behind the process (*ie.* whether it was first order, linear, *etc.*)[38]. The data could be fitted to a rate equation, and so give some idea about the mechanism involved. Still no information about the frequency content of the emissions was obtained, although lower frequencies could be selectively excluded.

By using multiple frequency bands and multiple integrators (Figure 4), the real-time data shows the time dependence of acoustic frequencies[39,40]. This is the principle of the multi-channel spectrum analyzer. Simultaneous processes may be distinguished if they differ in their frequencies. Noise may be eliminated by the selection of appropriate filters. This system is more expensive however, and the short time-constant integrating nature of the system makes the study of slowly emitting samples difficult. The choice of frequency bands for the filters is important as their number is limited to the number of channels. The frequencies of interest must be found experimentally. The choice of filter bands may also be predetermined by what is available from manufacturers as tunable filters are much more expensive than fixed. Even with this apparatus one still cannot visualize individual signals.

By capturing the individual signals (Figures 5 & 6), not only are they easily viewed but their frequency spectra are also available[30,33,41] Human or automated pattern recognition abilities can then be used to recognize the different types of signals which occur within each period of AE activity. The signals from stressed polymers were

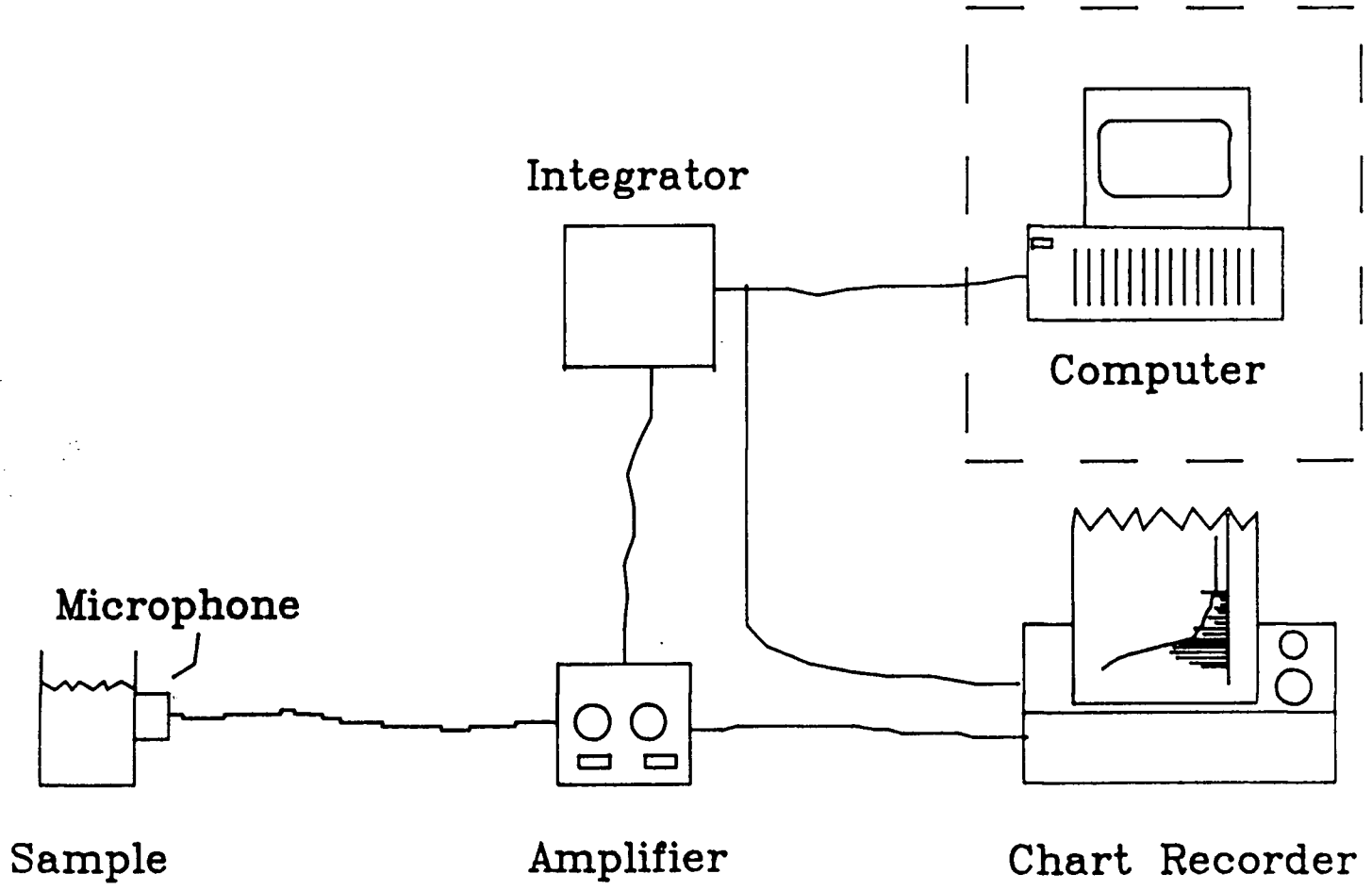


Figure 3) An integrating AE monitoring apparatus. Integration of the acoustic energy output can be plotted directly on a chart recorder by an electonic integrator or can be stored on comppter through the use of an analog to digital converter.

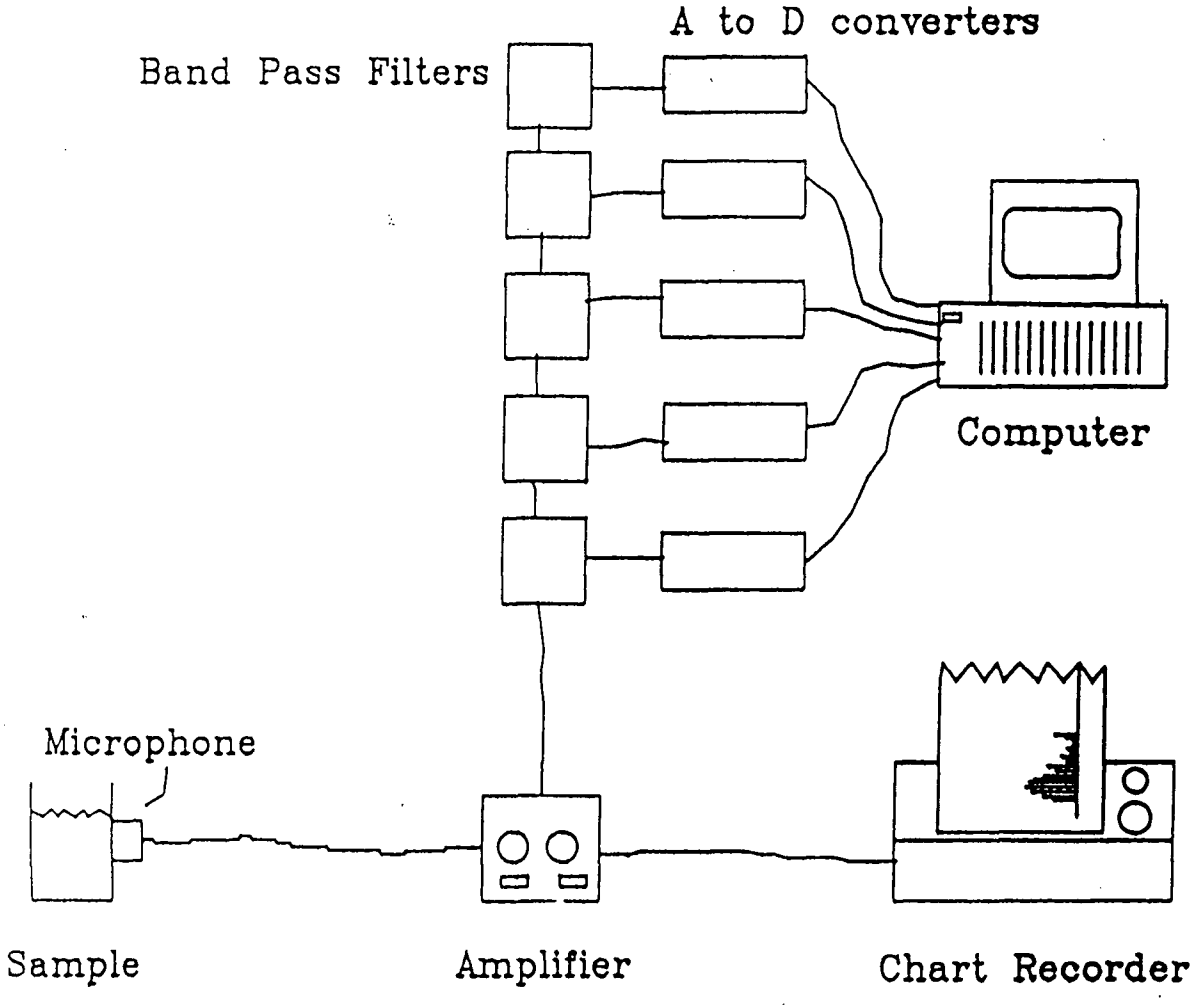


Figure 4) **AE apparatus using multiple band-pass to filters and integrators to allow for the analysis of individual frequency bands.**

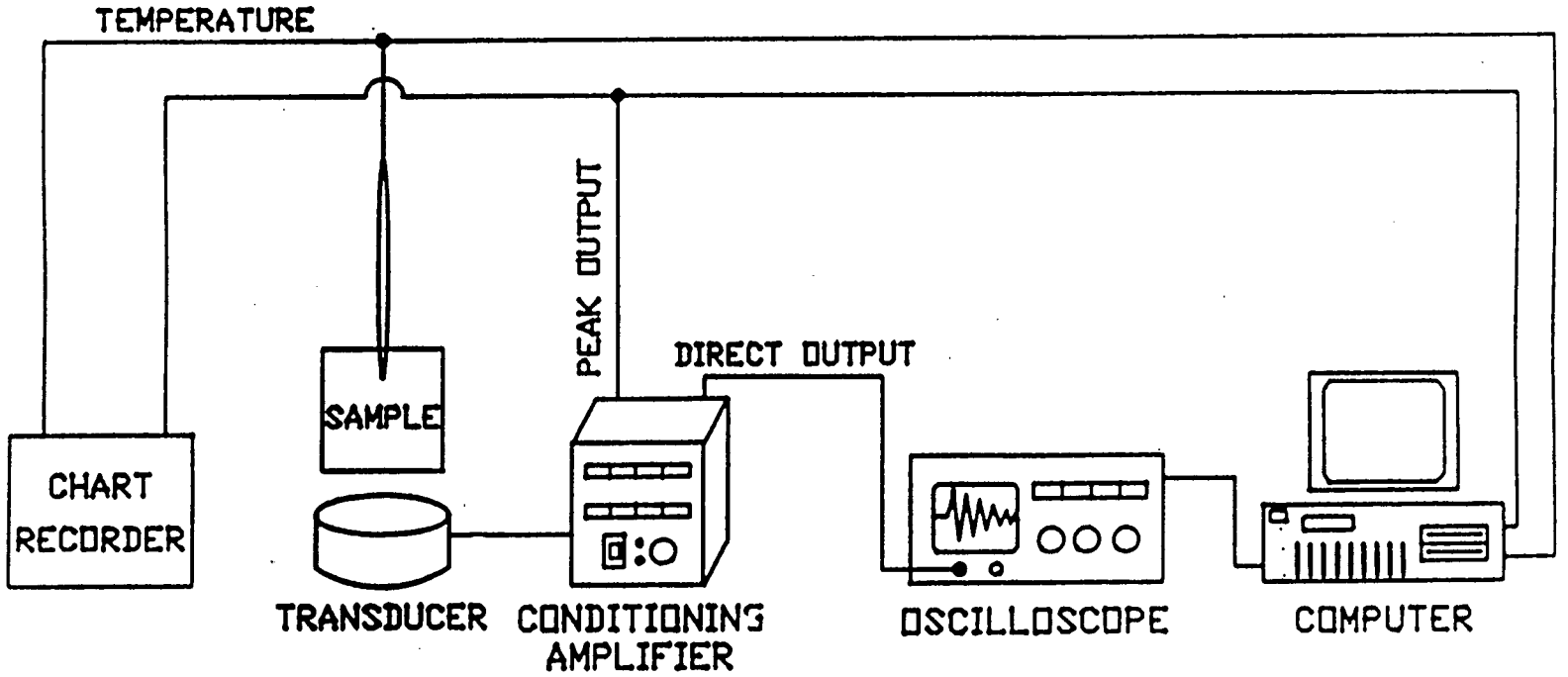


Figure 5) **AE Apparatus including digitizing oscilloscope to capture individual signals.**

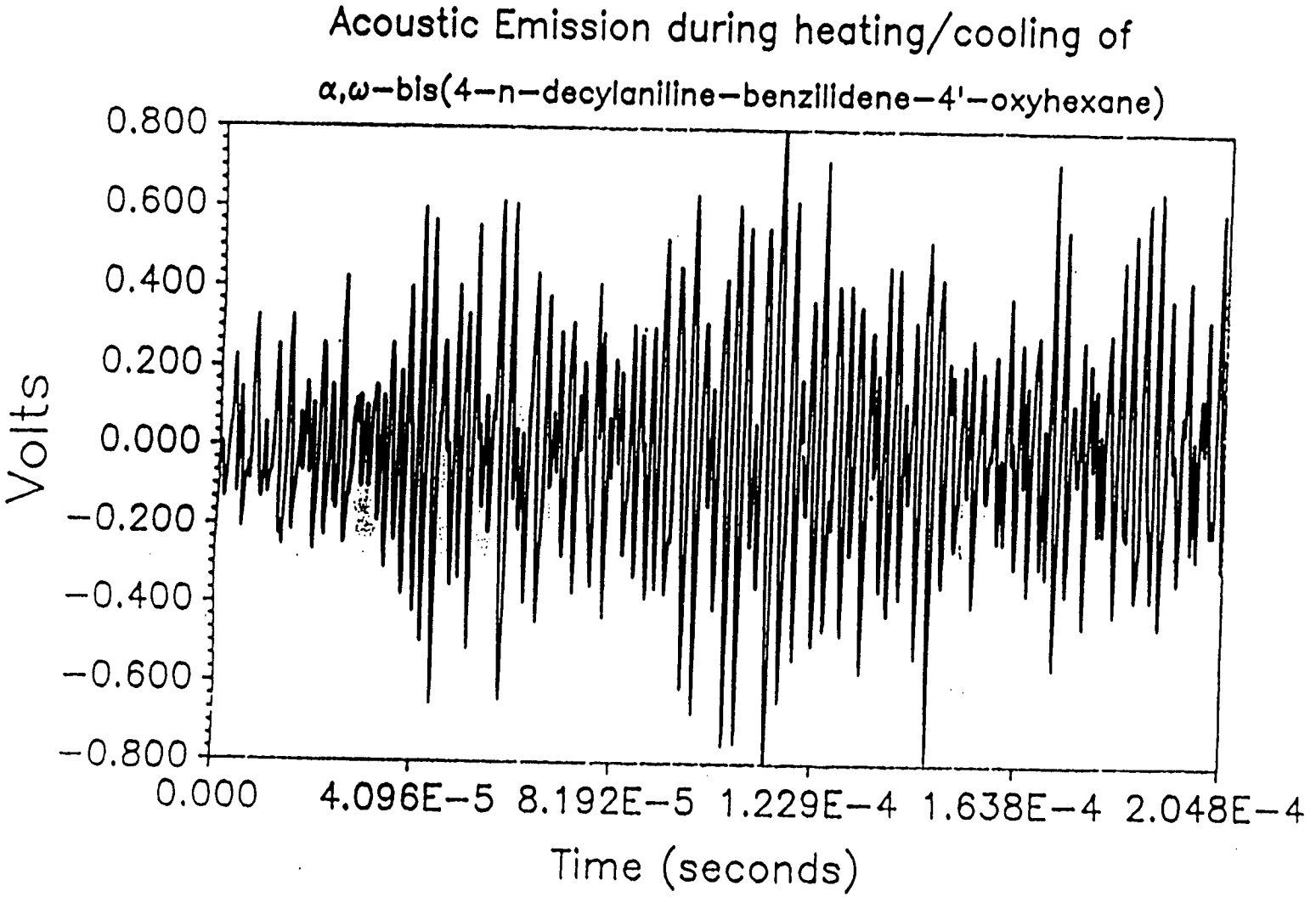


Figure 6) **A typical acoustic emission signal. (Intensity of signal - in Volts - plotted versus time - in seconds).**

able to be identified by automated pattern recognition techniques used by Belchamber *et al.* [30,42-46]. The availability of different avenues of analysis of the data means that (hopefully) more analytical insights can be revealed. Because the amount of data now collected can now easily be several megabytes per experiment, real time analysis is presently not possible and enormous storage capacities are required. The collection of every signal is not possible for a fast emitter - one where the emission rate exceeds the rate at which signals may be acquired and stored[41]. Because of this, AE energy quantitation is presently less reliable on signal capture systems[47].

A tremendous amount of information potentially exists in the individual signals as well as in how they occur and change over time. Pattern recognition techniques have previously been applied to acoustic emission data with much success. These factors, plus the present absence of a solid theoretic background regarding the origins of chemical acoustic emission, presently make empirical pattern recognition the best choice for analyzing AE data.

II. Chemometrics

The availability of less expensive electronics and computer components, and the need for even greater sensitivity and chemical selectivity, have led to development of hyphenated analytical techniques[48]. Chemists have begun to use instrumentation that produces very large amounts of multidimensional data. For example, tandem mass spectroscopy (MS/MS) uses two mass analyzers to produce two-dimensional mass spectra[49]. In hyphenated techniques, the output of one system becomes the input for another. Also common now is the use of multiple detectors to simultaneously collect multidimensional information. Examples of this are diode array detection in UV/Vis spectrophotometry[50], and Whole Column Detection (WCD) chromatography[51]. More information about a sample normally means that it should be easier to draw chemical inferences. However, when the amount of information becomes too large for interpretation within an acceptable time frame, the benefits are lost and much data remains uninterpreted. Such is the case in satellite surveillance, where it has been estimated that less than 5% of data collected is ever interpreted[52]. Indeed, it has been said that "we live in the age of the sorcerer's apprentice, in that we create data faster than we can sweep it away"[53].

The mind can easily appreciate the significance of a two dimensional display of data. Any scientific journal is full of two dimensional plots. Two dimensional representations of three dimensional surfaces are also largely recognizable - as a contour map or an evening with the family slides will confirm. When the information takes on a higher dimensionality than those we are accustomed to dealing with, namely two or three, it becomes much more difficult to visualize the structure within a data set and therefore to evaluate its meaning. What is often done is to look at successive subsets of the data in an attempt to find a pattern, eg. several two dimensional plots "simultaneously".

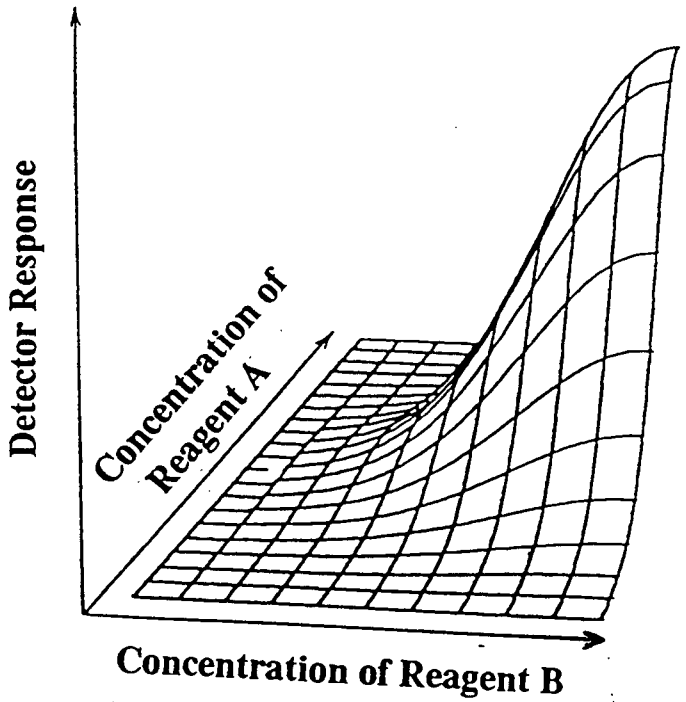


Figure 7) Three dimensional plot of detector response versus concentration for two reagents.

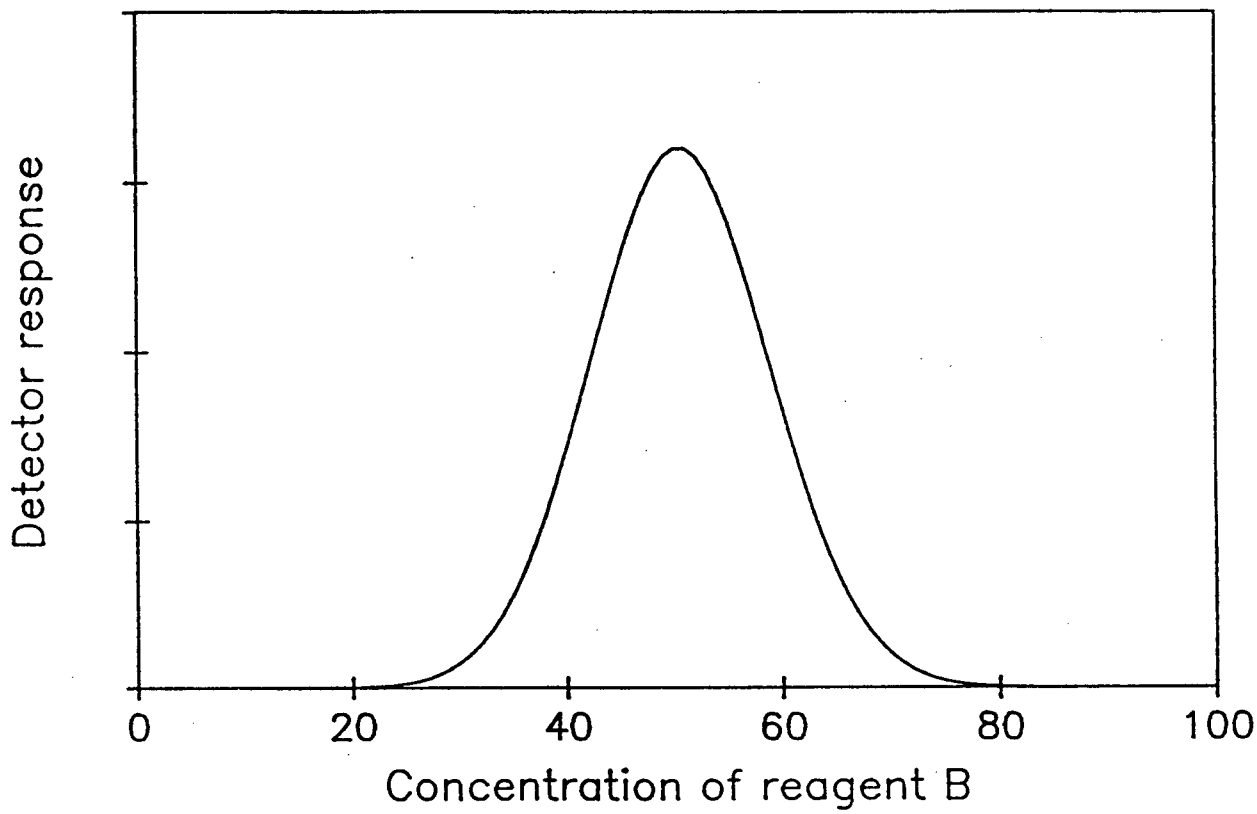


Figure 8) **Two dimensional plot of system in Figure 7. Detector response versus concentration with one reagent concentration held constant.**

Figure 7 shows a situation where a two dimensional approach may not be effective. The two horizontal axes correspond to concentrations of reagents. The vertical axis is an instrumental response (eg. absorbance). From the surface plot one can see that the best response occurs when the reagents are present in equal concentrations and that the response increases with increase of reagent. If a two dimensional view of this surface were taken (Figure 8), where concentration of reagent A is held constant and reagent B is plotted against response, the resulting sigmoidal curve would lead one to conclude that there is a maximum response attained when the varied reagent concentration, B, equals that of the reagent that is held constant, A. It would not, however, reveal the tendency of the response to increase with higher concentrations of both reagents. This example of making incorrect conclusions based on only a two dimensional view of a higher dimensioned data set shows the need for some method of displaying all the data simultaneously. The amount of information lost by showing only two dimensions of a three dimensional surface can be estimated by direct comparison. The amount of information lost by using only 2 of *m* dimensions can only be guessed at. That is, unless the mathematics of information theory[54] are used.

Many types of patterns are easily recognized by the human mind. As was previously discussed, one is able to distinguish between two different speakers simply by the sound of their voices. Indeed some workers have similarly tried to set the results of GCMS chemical analyses to music to alert operators to the presence of difficult samples[55]. One would also seek to be able to use the ability of the eye to identify trends or patterns in high dimensional data when performing chemical analysis. However, since the limit for the number of dimensions for a visual representation of data is three, a method must be used to condense the higher dimensional information into a more manageable two- or three-dimensional form. Workers have tried representing features of multivariate data as the lengths of vectors projecting from a

single point[56] or even as the size of features of a human face[57]. Keeping in mind the example given above, such graphical methods must not only be able to project *m* dimensions onto two or three dimensions, such that it is condensed into a more manageable form, but must also maintain the information content of the data. In this way, the human observer can use their own built in pattern recognition abilities.

Chemical acoustic emission may have an experiment which produces a large number of signals. Each of these signals may be represented in "feature space" by several descriptors. (This will be discussed further in chapter 5). In order to be able to adequately analyze and observe the data requires the use of methods to visualize the multi-dimensional data. Two techniques commonly used are the projection of the data onto an appropriate set of axes, and the display of the hierarchy of the data set (in the form of clustering).

II.1 Projections of Multivariate Data Sets

Figure 9 shows a two dimensional data set with 10 data points. One notices that there is a similarity among some of the data points. For example, because of its relative proximities, point 1 can be seen to be more similar to point 2 than it is to point 3. The distance between points can be thought of as a measure of their similarity. Data points that are similar will occupy the same region in what is termed a "feature space" or "descriptor space" which has the same number of dimensions as there are variables. Finding similarities in 2 dimensions is relatively simple. Data that occupy *m*-dimensions have the same property, but it are difficult to visualize. Fortunately, there is a method that allows us to project the *m*-dimensional feature space onto 2 dimensions.

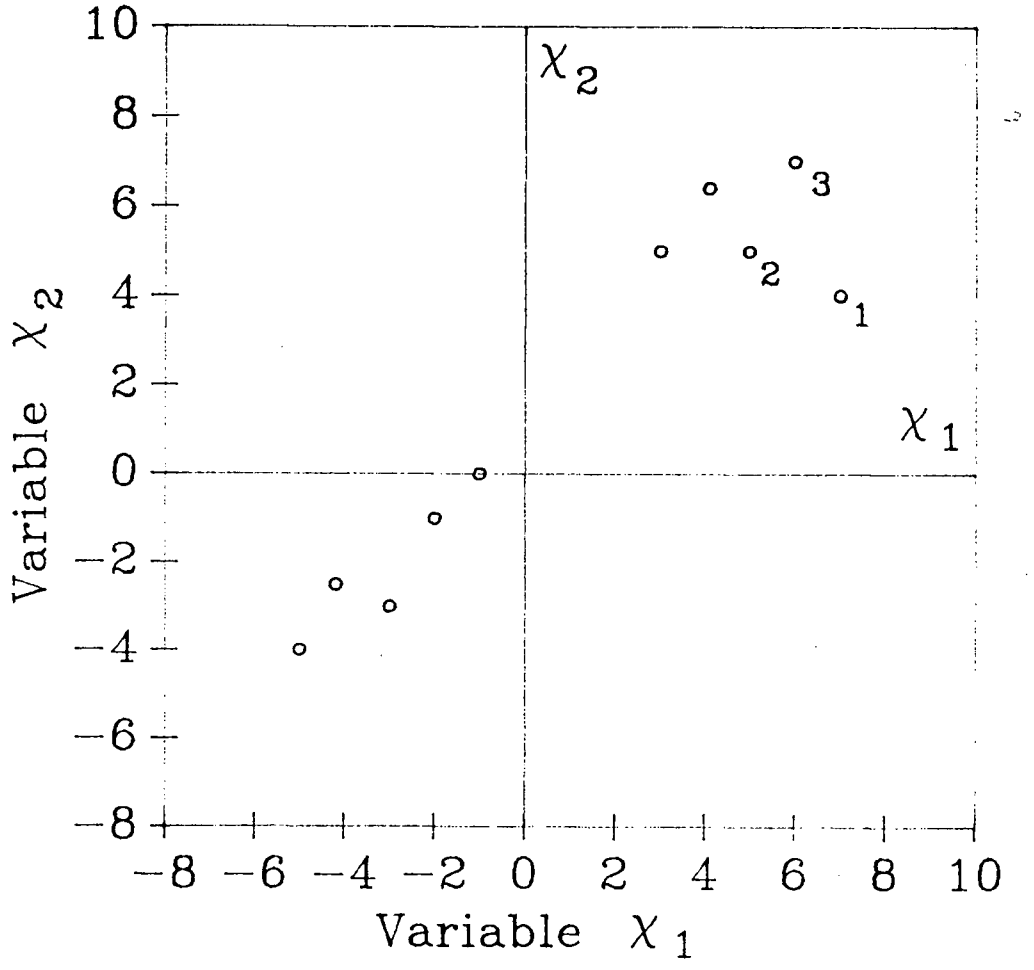


Figure 9) **Hypothetical data set containing 10 data points each of which described by two variables - X_1 and X_2. Point 1 is seen to be more similar to Point 2 than to Point 3 because of the relative distances.**

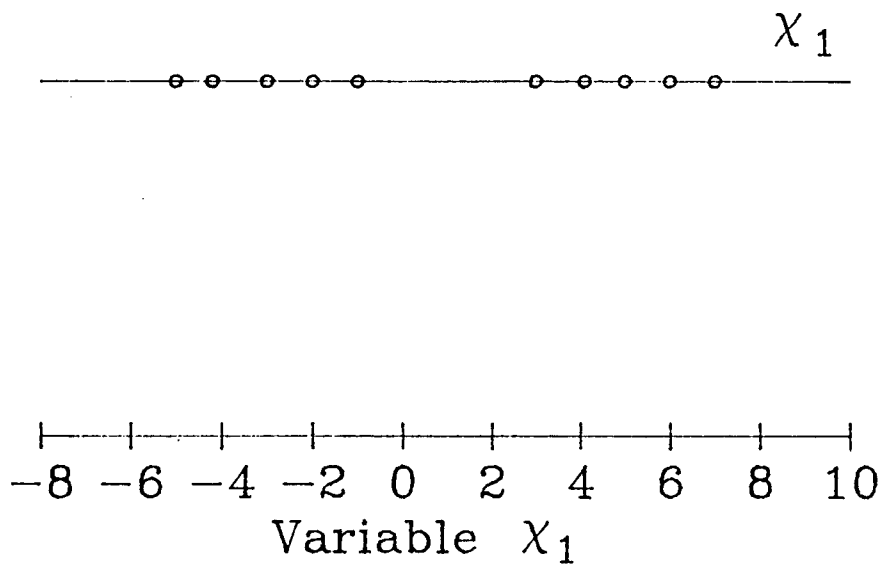


Figure 10) Projection of simulated Figure 9 data onto x_1 axis.

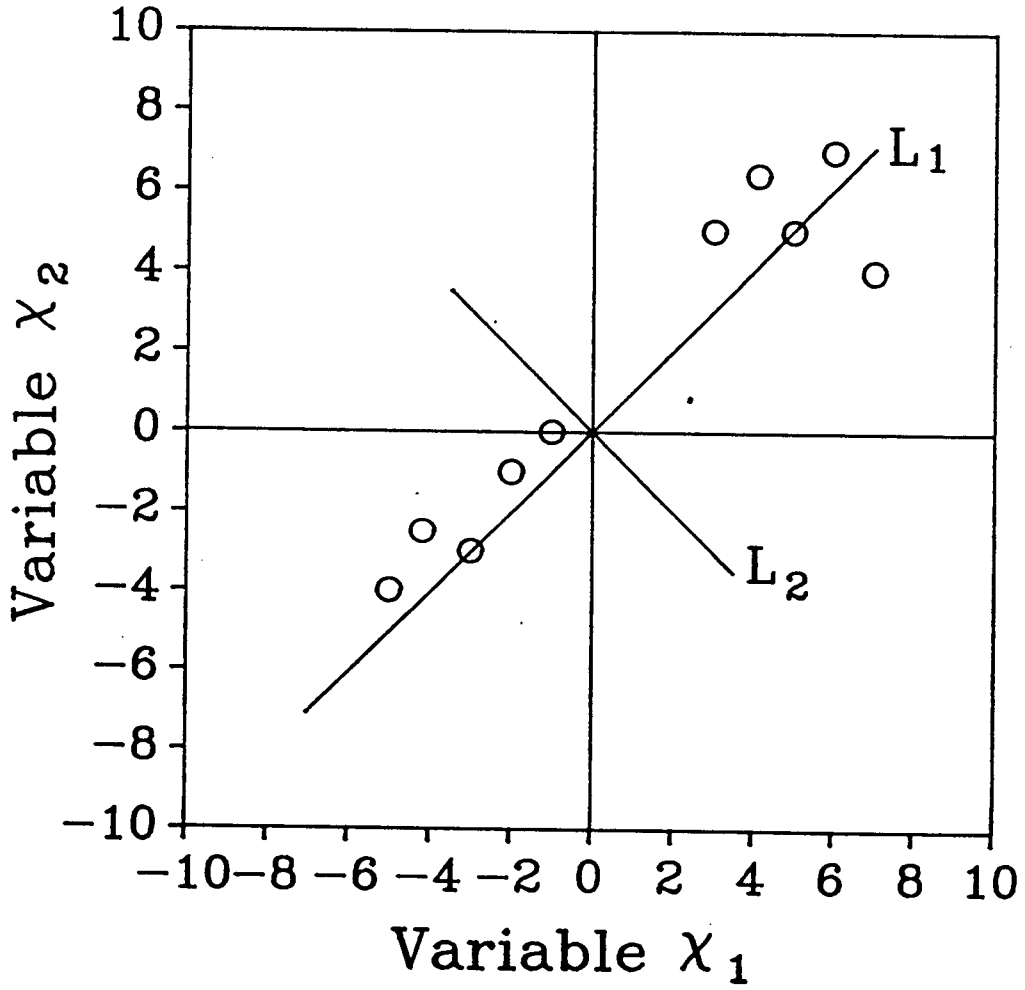


Figure 11) Data set from Figure 9. L_1 is the first principal component - which is the axis containing the most variance. L_2 is the second principal component.

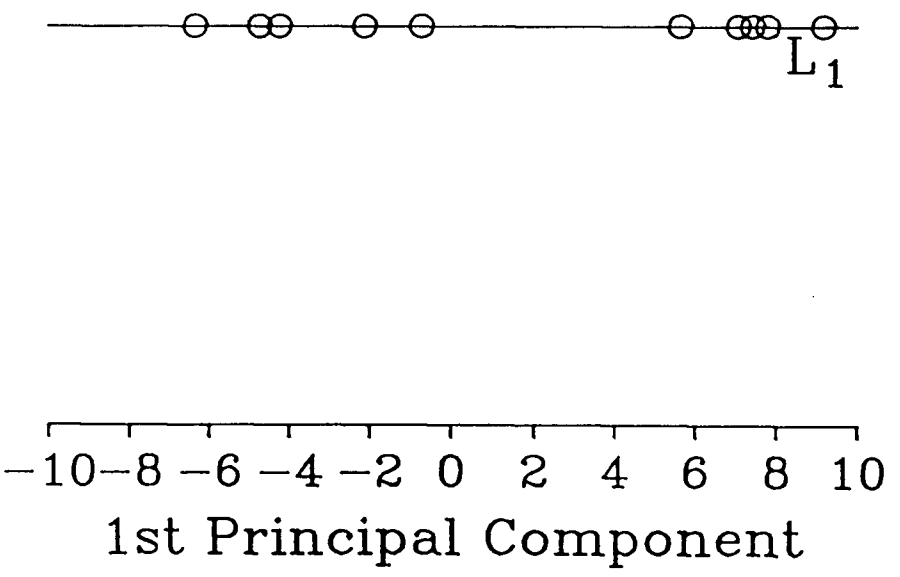


Figure 12) **Projection of Figure 9 data set onto first principal component L_1. This is the axis projection which best represents the structure of the two dimensional data set.**

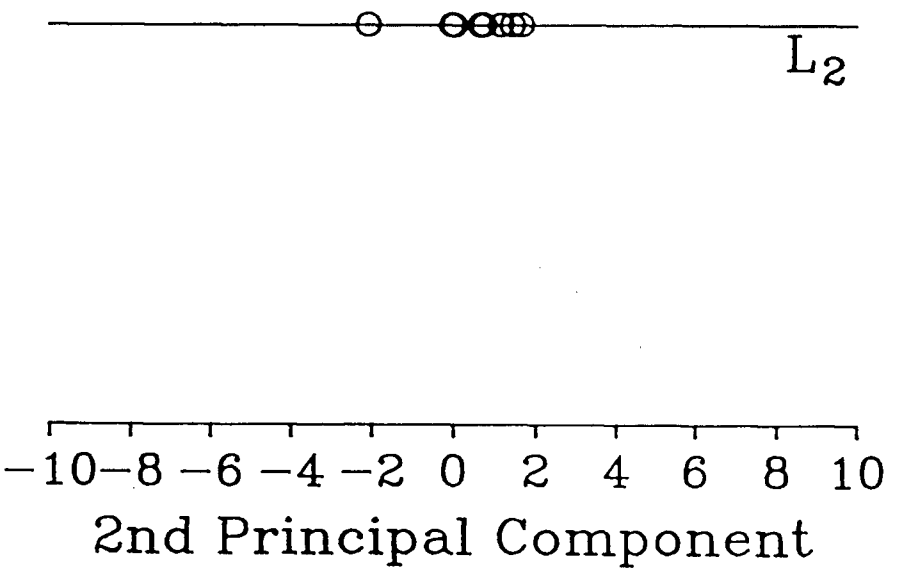


Figure 13) **Projection of Figure 9 data set onto second principal component. Structure of original data set is almost completely lost. Data structure can then be seen to be mostly contained in the first principal component.**

Taking the data in Figure 9, one can see that there are two (or three) regions in which the points tend to cluster. This clustering can still be shown if the data is plotted on only one axis. However, axes X_1 or X_2 alone may obscure this clustering (see Figure 10). Projecting the data onto line L_1 in Figure 11 results in a one dimensional plot that retains the structure of the data (Figure 12). One might arbitrarily choose line L_2 but the information contained in the data would be lost (see Figure 13). One can thus see the need for discrimination when choosing a decreased dimensionality on which to project the data. Indiscriminate use of 2 dimensional subsets of m-dimensional data sets may give meaningless pictures of the data!

In Figure 11, line L_1 is chosen to contain the maximum variation in the data. This is called a principal component. The principal component is a new variable which describes the objects. One can see that a linear combination of x_1 and x_2 is used to give the new variable l_1.

$$l_{1i} = a_1 x_{1i} + b_1 x_{2i} \qquad \text{(II.1)}$$

One can similarly reduce the data set in a similar manner for the m-dimensional case. The first principal component is chosen to maximize the variance. A second principal component is chosen orthogonal to the first and such that it accounts for as much as possible of the variation not accounted for by the first principal component. The m-dimensional space has now been projected onto a new two-dimensional space (a factor space), as defined by the two principal components.

The projection of the data set from the feature space onto the factor space is a simple mathematical transformation. The plot of the data in factor space (Figure 14) can be readily seen to be a simple rotation of the data in Figure 9. A new basis set (comprised of the factors) is chosen. These are linear combinations of the original

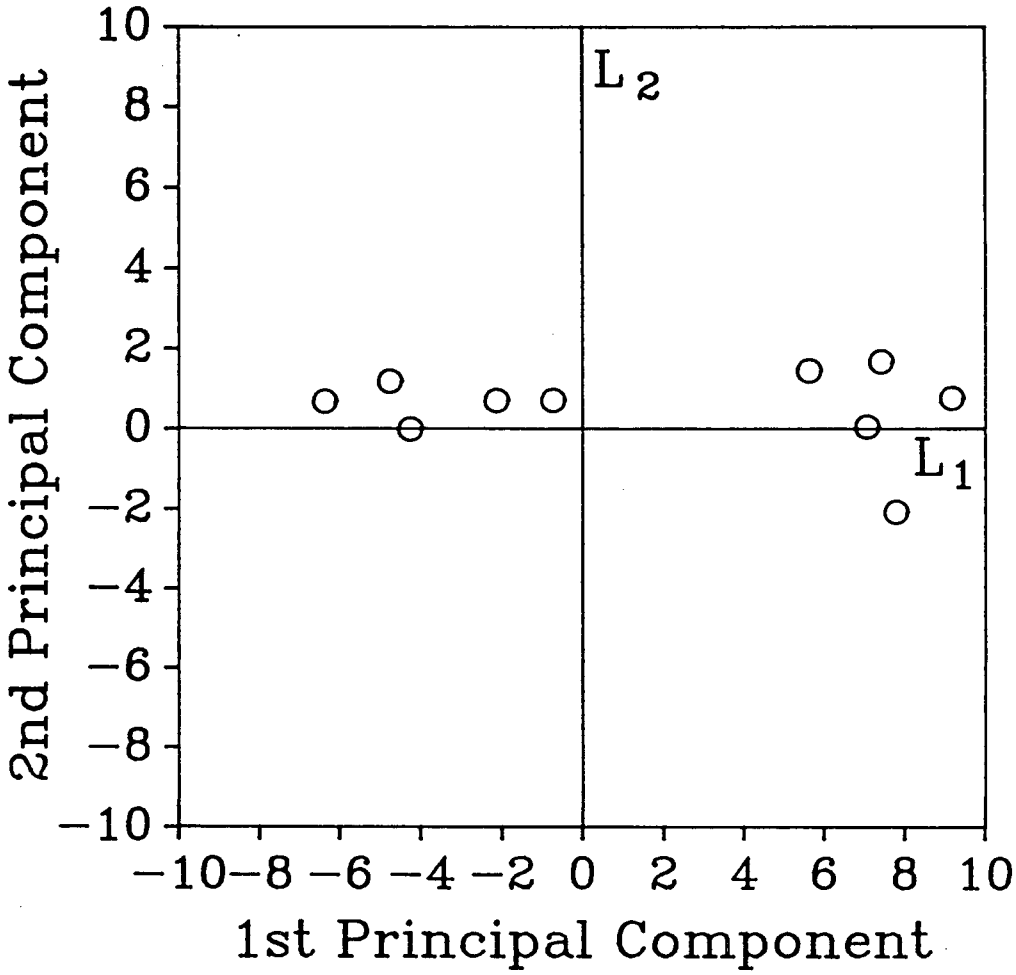


Figure 14) Data from Figure 9 projected onto factor space defined by L_1 and L_2. By comparison with Figure 9, factor analysis is a simple rotation of feature space.

experimental measurements (the features). This operation results in a set of factor scores and a set of loadings. The loadings are the coefficients of the individual features used to construct the factor vectors (eg. **a** and **b** in equation **(II.1)** above). The scores are the projections of the data points in feature space onto a particular factor (the coordinates in the new space)

To fully represent the m-dimensional data set (which resides in a feature space comprised of m dimensions) in factor space requires the use of m factors. These factors can be divided into two groups. The primary factors (or primary principal components) will contain the important information available from the data. The secondary factors will be due mostly to noise, to random errors, and to features that are mainly constant for all samples. The primary factors are therefore, those in which one will be most interested and the secondary factors can largely be ignored. This reduces the dimensionality down to only the number which contain meaningful information. More will be said later on processes used to determine if a factor is primary or secondary.

The loadings are an indication of the merit of the features used to describe the data. If a variable (for example, absorbance at a particular wavelength) has very little variation within the sampled data, it will not contribute a great deal to the primary principal components. This will result in it having a small loading. A feature which varies greatly across the data set will have a large contribution to the primary factors and therefore will have a large loading.

The scores matrix is simply the transformed data set in factor space. The score for each data point on each of the new basis vectors (factors) is the projection of the data point from feature space onto that factor. The scores matrix defining the data

points in factor space can be transformed back into the original data matrix in feature space by multiplying it by the loadings matrix.

Feature reduction is the projection of the multi-dimensional data set onto a lower dimensional factor space. This is accomplished by plotting the scores of the data for two factors against each other. In this way, one can view a projection of the feature space onto the two dimensional plane defined by the two factors chosen.

Feature selection is the process whereby the features are used or ignored based on their discriminating ability, as given by the factor loading for each feature. This allows one to ignore certain variable measurements and thus in experimental design, eliminate future experiments if a feature is not helpful in describing the structure of the data. For example, if the pH of a set of solutions is roughly the same, then pH is not a good discriminator and perhaps the measurement need not be taken for future analyses of the same type from the same system.

II.2 Display of Hierarchical Clustering

The basis of pattern recognition is the characterization of items that are similar as opposed to those that are dissimilar. Earlier, it was noted that objects 1 and 2 of Figure 9 were similar because of their closeness in feature space. Distance was used as a measure of similarity. This is intuitive for two dimensions. Objects that are similar will have similar values and therefore will occupy similar positions when plotted. For the multi-dimensional case, the same can also be said. A distance is easily measured in two dimensions - the length of the straight line between two points. For higher dimensions, where the concept of a straight line may be confusing, there are several methods of calculating the similarity of data points in feature space. The mathematics of some will be given later.

If one can measure a similarity between data points then one should be able to construct a suitable "tree structure" that would represent the pattern of the data in much the same way that an evolutionary tree represents the similarities between different classes of life.

The similarity of experimental data points can be estimated mathematically and displayed similarly. Figure 15 shows a layout of five imaginary cities. Suppose a network of train lines is to be built that connects these cities. It is desired that the length of track used be minimal and that there is a maximum of one route between any two cities (ie. no circuits). The planning of such a task can be accomplished easily. The two closest cities are joined together first. This would be cities A and B. Then a track is placed between the next two closest cities that have no connecting route. This process continues until all cities are served by at least one train track and the network of tracks serves to connect all the cities (Figure 16). This is one type of "tree" used to show the relationship between the cities. The tree in which the sum of the links is a minimum is called the "minimal spanning tree". If we wished to divide the cities into two groups (say for electoral boundaries) then it could be done by drawing the line across the longest link. This would cluster the cities into two groups. Fortunately for political engineers, cities lie on a two dimensional map. Unfortunately for scientists, data often have higher dimensionality.

The dendrogram is a different sort of tree. It is similar to the minimal spanning tree in that it can be used to divide a set of objects into groups or clusters. It differs in that it can be used to represent objects that have <u>any</u> dimensionality. One might think of the dendrogram as a transformation of the minimal spanning tree for a multi-dimensional data set onto two dimensions. The method for calculating the dendrogram is very similar to that for the minimal spanning tree.

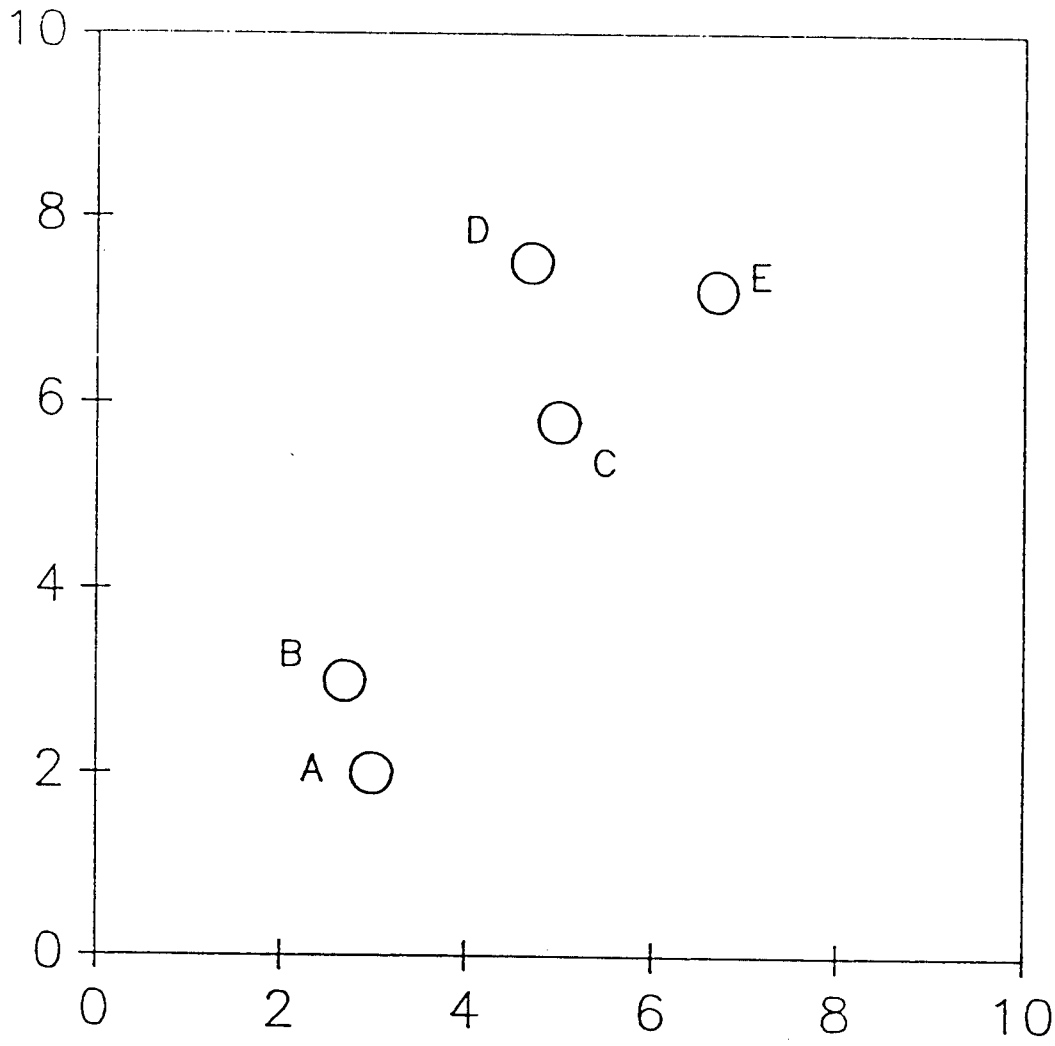


Figure 15) Layout of five imaginary cities - A-E.

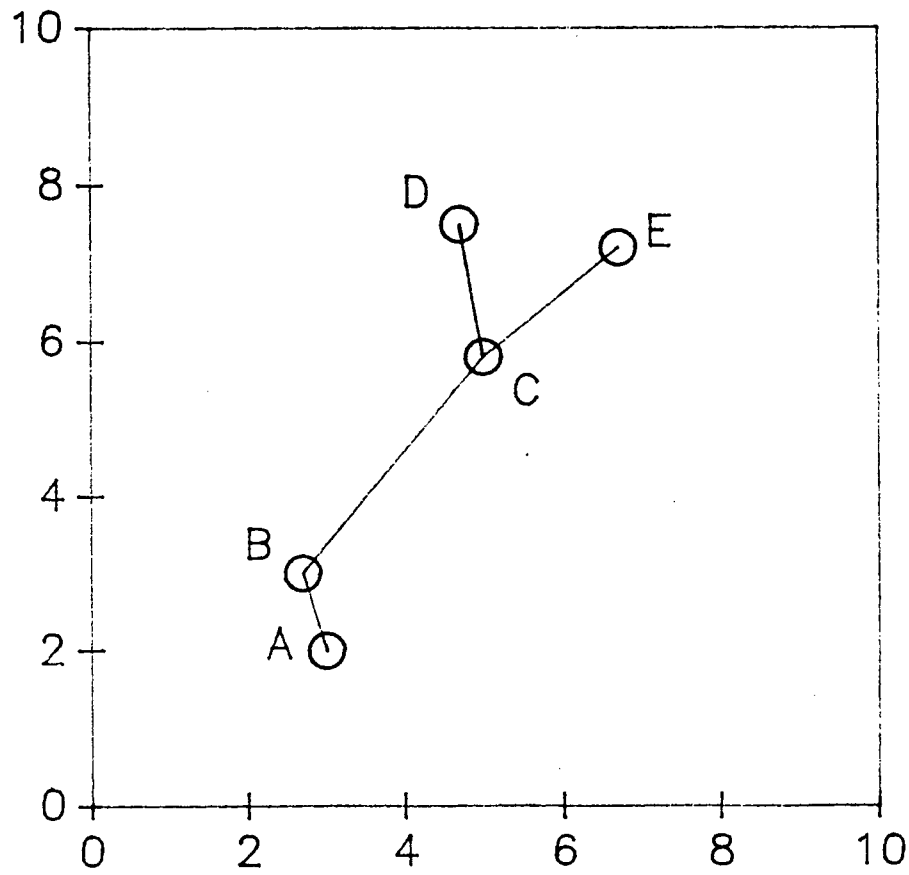


Figure 16) **The five cities joined with minimum length of train tracking. This is a minimal spanning tree for the five cities. The "boundary" along the longest section (between cities B and C) divides the cities into two groups.**

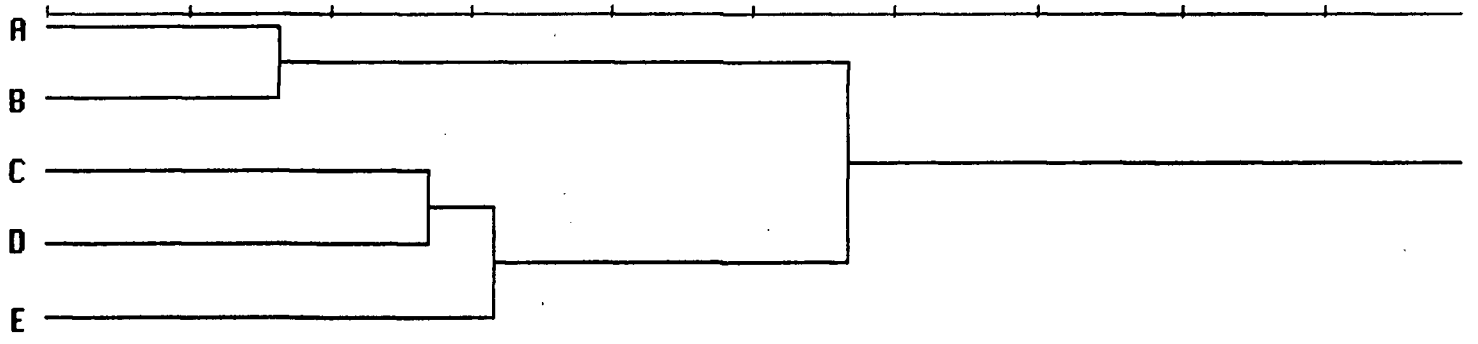


Figure 17) A hierarchical dendrogram for cities in Figure 15 showing two groups or classes.

Figure 17 is the dendrogram for the cities in Figure 15. Each city is represented by a point along the vertical axis. The distance of the link between two cities is shown by the horizontal scale. Cities that are close together will be connected by lines that don't extend very far up the vertical axis. One can see that there are indeed two clusters of cities. This is the main use of the dendrogram - to see if there is any clustering of data into specific groups.

This technique of displaying data as a dendrogram is known as agglomerative hierarchical clustering. By fusing the objects together, a stepwise set of clusters is formed. If the investigator has *a priori* knowledge of the data (that is, the data have an expected grouping), then hierarchical clustering will provide a list of which data points are members of each of the known number of clusters. If the structure of the data is not known, then the dendrogram display can be used to determine the number of clusters (subsets) present in the data (if any).

These two methods, cluster analysis[58] and principle components analysis[59], are widely used in chemistry. Chemometrics, as Massart's definition[35] suggests, includes techniques for analyzing data and choosing experiments. Chemometrics has allowed scientists to use mathematics to achieve analytical measurements and insight into highly multidimensional chemical systems which once may have seemed impossible.

The uses and methods which can be described by the term "chemometrics" are numerous. These include non-linear calibration curves, resolution of chromatogram peaks for components with almost equal elution times, experimental optimization[60], and evaluation of experimental methods. For an excellent survey of these, the reader is directed to any of several reviews on chemometrics[61-63]. There are also a number of

text books which give great detail and instruction on particular aspects of chemometrics[64-67].

II.3 Chemometrics in Acoustic Emission

The use of chemometrics for the analysis of acoustic emission data has already revealed much new information. The analysis of acoustic emission from stressed polymers using pattern recognition techniques led to an "automated" method of identifying the samples[30,68]. Using pattern recognition techniques, Chan and coworkers were able to determine abnormal conditions during the welding of carbon steel[69]. Other workers have used pattern recognition on acoustic emission data from carbon reinforced fibre composites[70]. Other approaches to the analysis of acoustic emission data have included time series analysis[71], and a two-stage clustering procedure developed by Majeed and Murthy[72]. These successful applications of chemometrics to acoustic emission data from materials lead one to assume that chemical acoustic emission analysis would benefit from the use of these and other chemometric routines. Initial attempts have confirmed this to be so[31,33].

III. Experimental

The experimental aspect of this thesis work has been:

1) the development and coding of chemometric techniques suitable for analysis of highly multivariate data, such as is obtained from chemical acoustic emission,
2) the development of a computer controlled apparatus and data acquisition software to collect chemical acoustic emission signals,
3) the application of the hardware to a number of chemical experimental systems,
4) the analysis of the chemical acoustic emission data collected using the chemometric techniques developed and coded,
5) the critical evaluation of the chemometric techniques and their utility in chemical acoustic emission studies.

III.1 Hardware.

The next section details the hardware used and the experimental chemistries studied. The experiments were carried out using the apparatus outlined in Figure 18.

III.1.1 Transducer

The sample container is placed in contact with a broadband transducer (Bruel and Kjaer, Naerum, Denmark, Model number 8312) which incorporates a 40 dB preamplifier. The transducer has a known frequency response and can access from 100 Hz to 1 MHz. Some signals were observed above 1 MHz but no calibration of the response in this region was available from the manufacturer. The transducers have an operating temperature range between -10 °C and 55 °C (although Bruel and Kjaer claim that the temperature could probably approach 100 °C for short periods without any damage[73] as has been confirmed in this laboratory). Vibrations from the work surface were effectively eliminated both by the high pass filter and by a 3/4" layer of foam padding laid underneath the transducer.

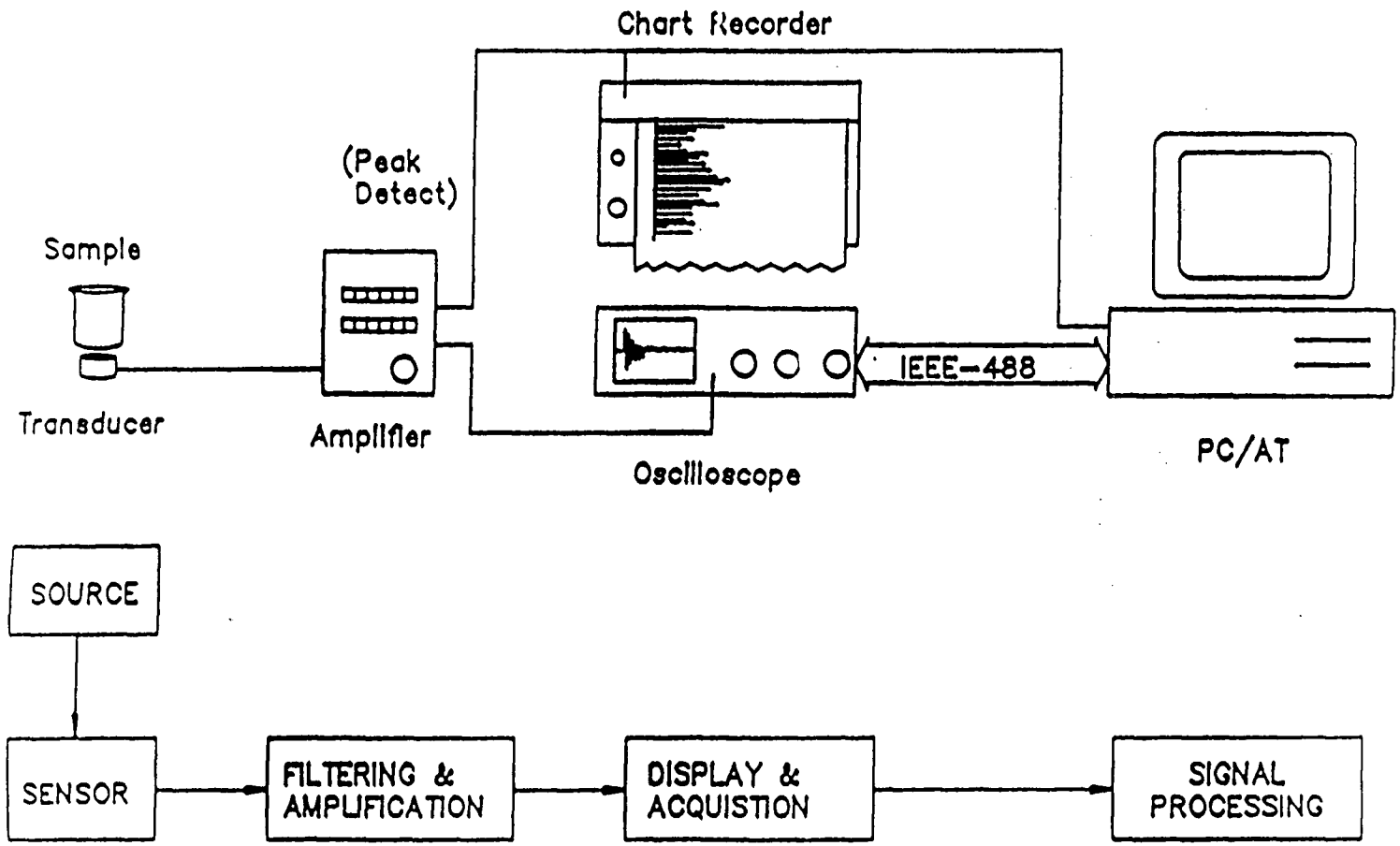


Figure 18) Chemical acoustic emission monitoring apparatus. i) Bruel and Kjaer broadband transducer - model 8312. ii) Bruel and Kjaer conditioning amplifier - model 2638. iii) Tektronix 100 MHz digital storage oscilloscope - model T2230. iv) Goerz Metrawatt chart recorder - model SE 120. v) PC/AT computer with IEEE-488 interface.

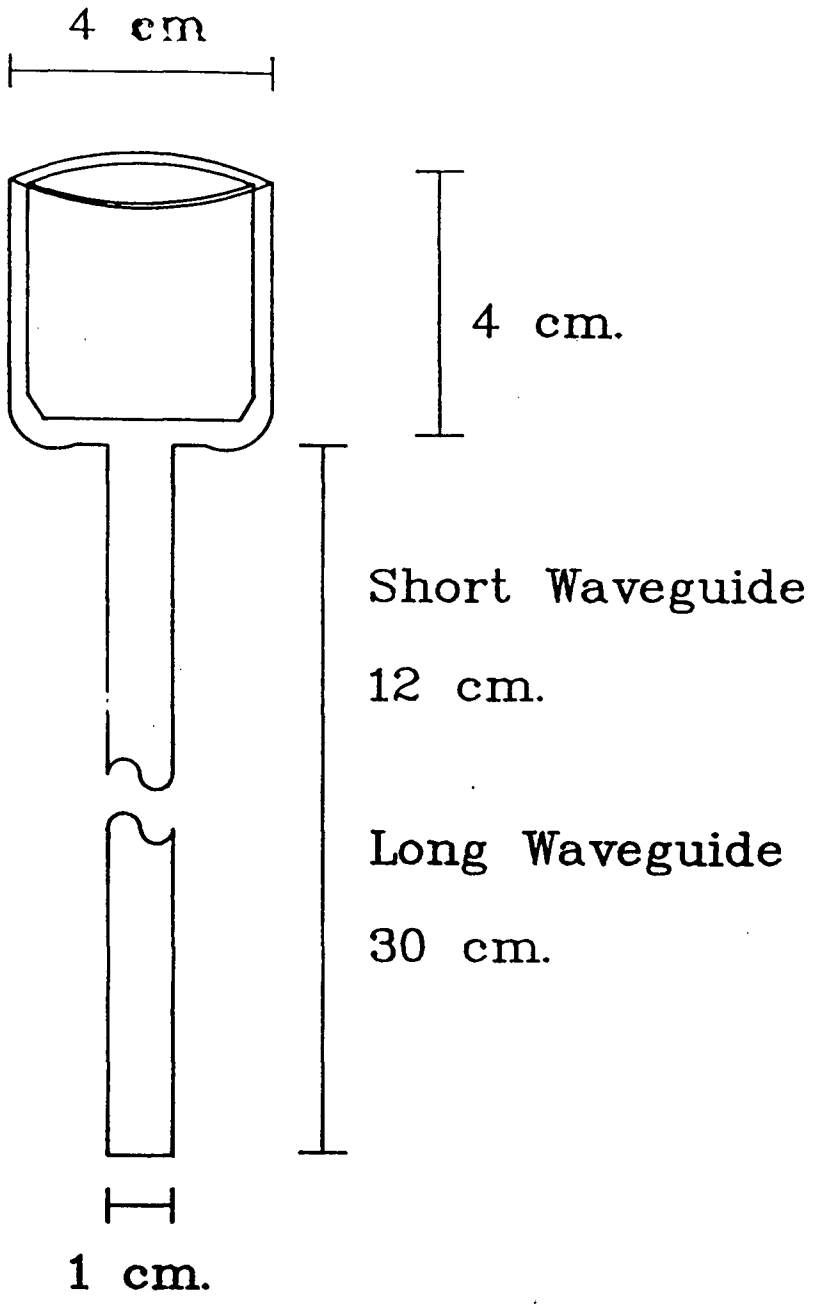


Figure 19) **Glass wave guide used to transmit acoustic signals from the sample to the transducer when the sample environment is unfavorable for the normal operation of the transducer (eg. due to high temperatures).**

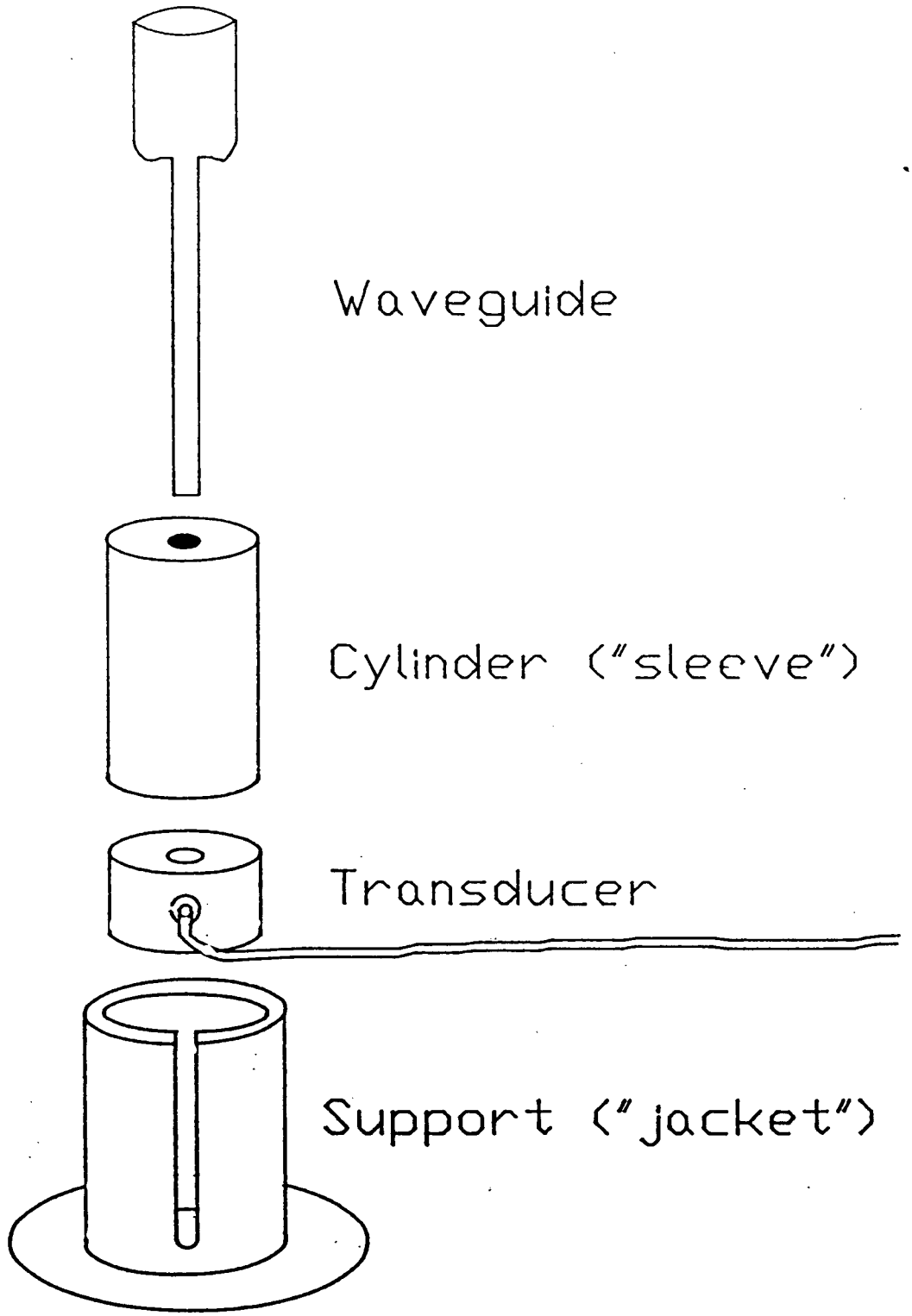


Figure 20) Jacket and sleeve used to hold waveguide and transducer.

III.1.2 Waveguide

For experiments where the maximum comfortable operating temperature (ca. 80 °C) is exceeded, a waveguide was used to transmit the acoustic signals but not the heat from the sample to the transducer. The waveguides (Figure 19) were all of glass construction and manufactured in this department. These are similar to those described by Clark[74]. The "wine glass" shape waveguides allowed the sample to be placed in the cup at the top which was situated in an oven (CENCO, Central Scientific Company of Canada). Temperatures to 250 °C could be reached although heating rate and temperature control were known to be a problem with this oven. The bottom of the waveguide extended out of a hole cut in the bottom of an oven where it is held by a metal "jacket and sleeve" (Figure 20) which was designed to fit the model 8312 transducers such that only the weight of the waveguide and sample rests on the active area of the transducer. The support was manufactured in the department. Two waveguides were designed and differed only in length. One waveguide had a stem of length 30 cm and the other 12 cm.

III.1.3 Conditioning Amplifier

The cable from the transducer was connected to a conditioning amplifier (Bruel and Kjaer, Naerum, Denmark, Model 2638) which provides band pass filter capabilities (linear, 0.1 Hz - 10 kHz, 50 kHz - 2 MHz, 100 - 2 MHz, 200 kHz - 2 MHz) as well as amplification (0 to 60 dB). The amplifier has two outputs. The direct output carries the amplified and filtered AC (alternating current) signal from the transducer. The second output from the amplifier is of the "sample and hold" variety which yields a DC (direct current) damped peak voltage with a time constant of about 200ms[75]. This is connected to a chart recorder (Goerz Metrawatt, Vienna, Austria, Model SE 120) to obtain traces such as seen in Figure 21, and is the output used by integrating AE monitoring systems[47].

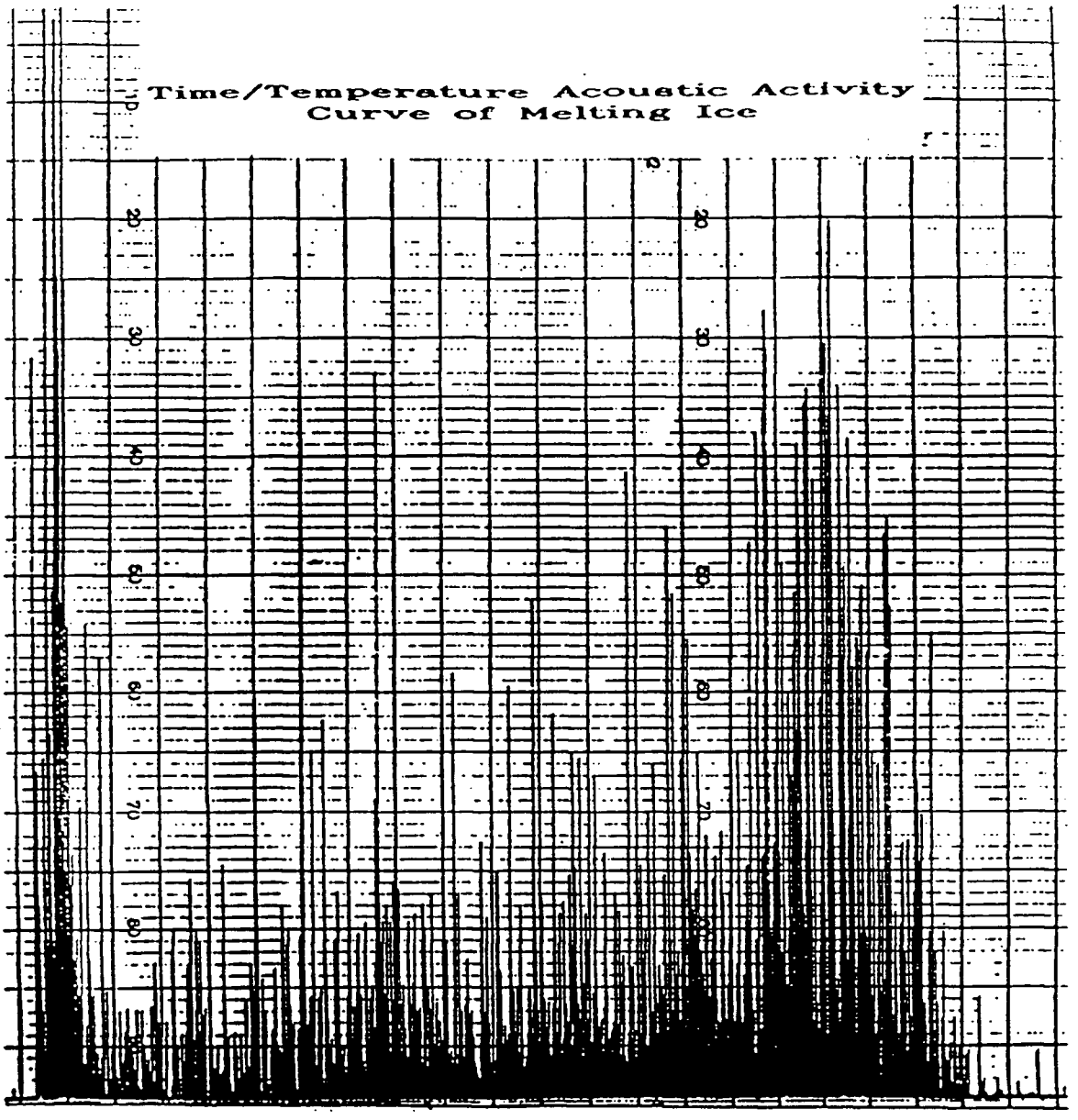


Figure 21) Chart recorder trace of acoustic emission from melting ice. Vertical axis measures intensity of acoustic emission (in Volts) versus time on horizontal axis. (Data provided by Dr. P. D. Wentzell).

III.1.4 Digital Storage Oscilloscope

The direct (a.c.) output is attached to a 100 MHz digital storage oscilloscope (Tektronix, either model T2230 or T2340A). The oscilloscope digitizes the incoming signal into a record of 1024 (or 4096) 8-bit integers according to its operating mode. Many operating parameters may be controlled by a microcomputer across an IEEE-488 parallel interface[76]. Details of the control language used are given elsewhere[77,78]. The Volts-per-division setting, VDIV, defines the Voltage range that is visible on the oscilloscope's screen. This is also the range into which the signal is digitized into values from 0 to 255. The maximum Voltage in the range is digitized as 255 - ie. the largest number that may be represented by eight bits. The minimum Voltage, which will be negative, is digitized as zero. A zero Volt signal will be digitized as 127 (Figure 22). The signal is then stored in the scope's memory as an array of numbers which represent the Voltage at each time interval. The actual length of the time interval for each signal acquisition is determined by the time-per-division setting, TDIV. The oscilloscope samples and digitizes at 100 MHz and then averages the results to give the correct TDIV. Records of 1024 points were used throughout this work due to the larger storage space and longer analysis time 4096 point signals would require.

III.1.5 Computer and Instrumentation Interface

The signals were then downloaded over an IEEE-488 parellel interface to a PC/AT IBM-compatible computer for storage on a hard-disk. Our original work was done with 12 MHz IBM PC/AT compatible microcomputers (NORA Systems, Vancouver) and was later continued using a 20 MHz Intel 80386-based portable microcomputer (model T5200 Toshiba). Both were equipped with an Intel 80287 or 80387 math coprocessor chip and a standard GPIB interface card. The interface to the

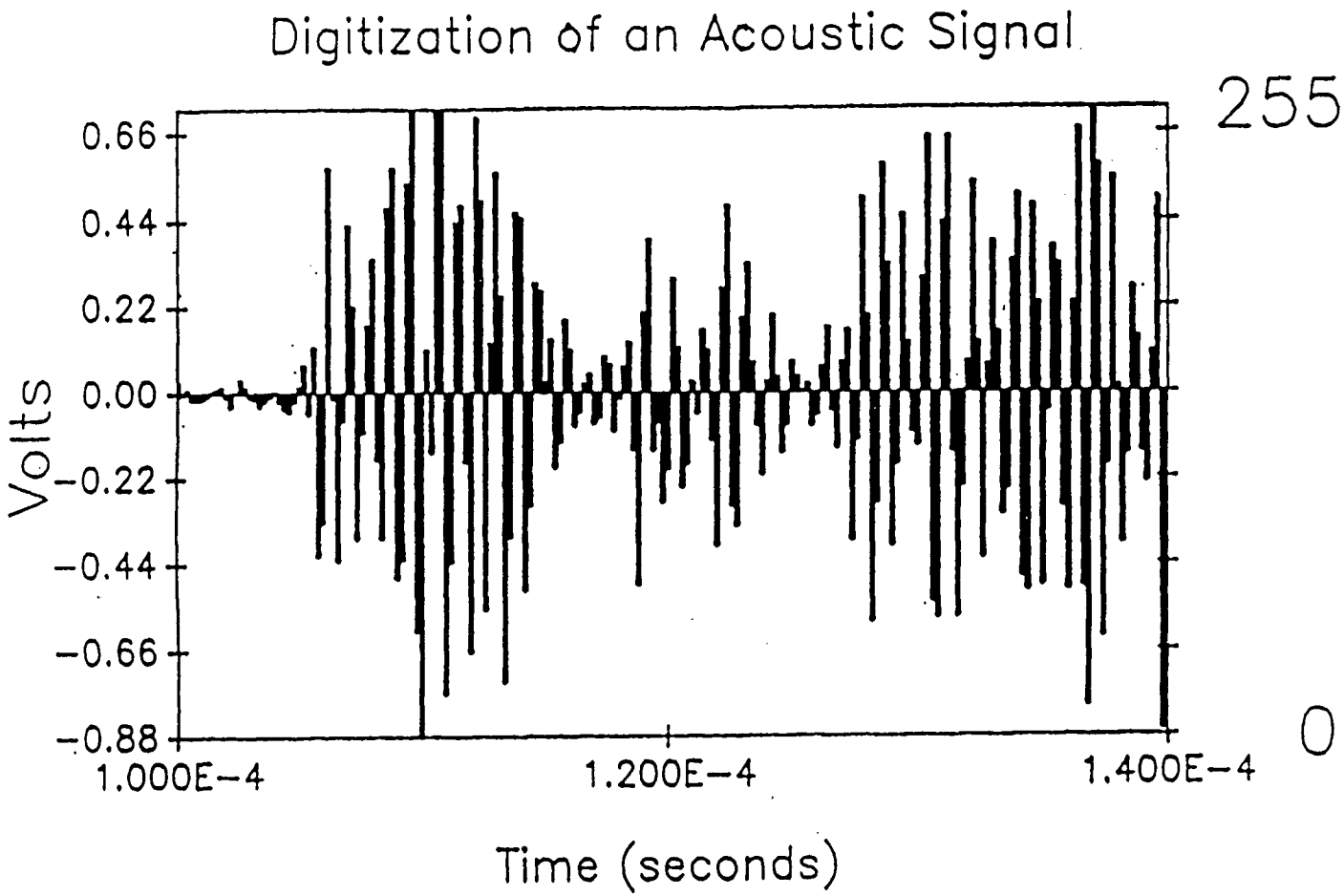


Figure 22) Acoustic emission signal with voltage scale on left vertical axis and corresponding digitization level between 0 and 255 on right vertical axis.

oscilloscope was provided by drivers written in assembly language and available from National Instruments (National Instruments Corp., Austin, Texas)[79].

III.2 Software

With the exception of the GPIB-PC interface drivers, all software used was coded in this laboratory. All programming is currently done in Microsoft QuickBASIC (version 4.00B, MicroSoft Inc., Mississauga, Ontario), although earlier programming used QuickBASIC 3.0, 4.0, as well as GWBASIC. The programs for data analysis and display require an Enhanced Graphics Adapter (EGA) card and EGA monitor. The programs include SIGVIEW, DENDGRAM, ABSCAT, and AEMUNCH. The data acquisition program, QAQ, and one of the analysis programs, VARI-TRAPS, do not require the EGA card. The operation of these programs will be outlined in Chapter Five.

IV. Chemical Systems Studied

The work by Betteridge *et al.*[28] showed that many types of chemical systems are acoustically active. Several different systems were investigated in this work to characterize their acoustic behavior. The pattern recognition methods coded were applied to the data collected. The chemical reactions explored include the dissolution of NaOH, the solid-solid phase change of trimethylolethane (TRIMET), the activation of intumescent flame retardants, the hydration of $AlCl_3$, and liquid crystal phase changes.

IV.1 Dissolution of Sodium Hydroxide

The dissolution of sodium hydroxide is exothermic and occurs fairly rapidly. This process has been studied using acoustic emission in this laboratory and has shown to provide useful information[38,41]. Pellets of sodium hydroxide from BDH (Toronto) were used to provide acoustic signals to test the equipment.

IV.2 Trimethylolethane (Trimet)

Trimethylolethane (Figure 23) is a polyol with a neopentyl structure and is a white, solid powder at room temperature. The compound has very favorable properties for industrial application (as evidenced by its use in plastics, paint resin, lubricants, and as coatings for pigments), due to its hardness, resistance to water, heat and light degradation and oxidation[80].

Trimet has a melting point of 190°C without decomposition and has a plastic crystal transformation at 80°C. The plastic crystal transformation allows Trimet to absorb a fair amount of heat energy without changing temperature (ΔH = 46.1 cal/g cf. ΔH_{fusion} = 10.7 cal/g)[80]. Trimet's thermal properties have made it useful as an

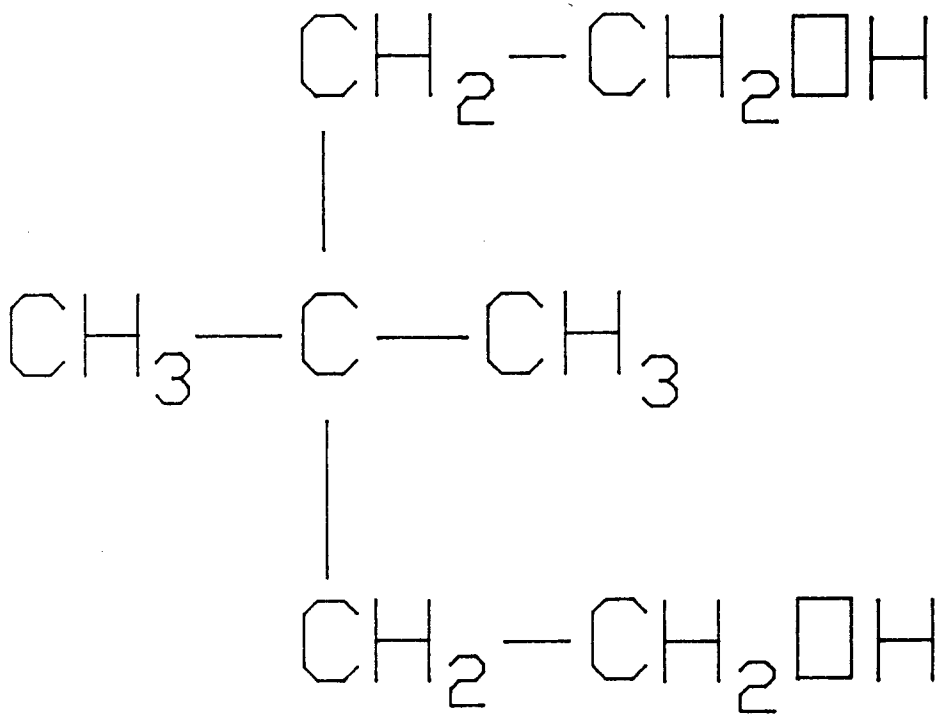


Figure 23) Trimethylolethane (2,2 dimethyl-1,3-dipropanol).

insulating material and as a heat storage medium for solar energy panels. The sample of solid powder TRIMET was graciously provided by Dr. P. Eckler of International Minerals and Chemical Corporation (IMC), Terre Haute, Indiana.

IV.3 Intumescent Fire Retardants (IFR)

Intumescent fire retardants (IFR's) are designed to be used as coatings for electrical wiring that will prevent the wire from being damaged by heat, and smoke during a fire[81-86]. IFR's are typically composed of i) a material which yields acid at temperatures between 100°C and 250°C (such as an inorganic acid); ii) a polyhydric, carbon-rich material; iii) an organic amide, amine, or azo compound; iv) a material which can serve as a blowing agent by decomposing and releasing gases (such as a halogenated compound); and v) a hydrated salt. The protective nature of an IFR is due to its behavior upon heating. The upper layer of the plastic body softens. The blowing agent then releases gases which cause the plastic to foam. The acid released then causes cross-linking and the polymer sets. Five samples of polypropylene samples mixed with ammonium polyphosphate and pentaerythritol were sent by Dr. L. M. Shorr of IMI (TAMI), Haifa, Israel.

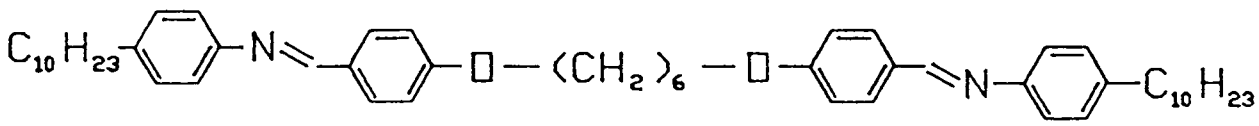
IV.4 Hydration of Aluminum Chloride

Aluminum chloride, $AlCl_3$, reacts violently with water to produce aluminum hydroxide and hydrochloric acid gas.

$$AlCl_3 + 3H_2O \text{---} > Al(OH)_3 + 3HCl$$

This reaction is performed industrially to analyze for $AlCl_3$ by checking the pH of the water to which the $AlCl_3$ is added. This technique is suspect, considering that large amounts of HCl are released as vapor, rather than fully absorbed into solution. It was thought that acoustic emission monitoring of this process might provide a better alternative.

Liquid Crystal



α,w-bis(4-n-decylaniline-benzilidene-4'-oxyhexane)

Figure 24) α-ω-bis(4-n-decylaniline-benzilidene-4'-oxyhexane)

IV.5 Liquid Crystals

IV.5.1 α-ω-bis(4-n-decylaniline-benzilidene-4'-oxyhexane)

The liquid crystal α-ω-bis(4-n-decylaniline-benzilidene-4'-oxyhexane) (Figure 24) has the unusual property of being rigid at the ends and loose in the middle. A small sample was provided by Professor G. Luckhurst (University of Southampton, U.K.) for preliminary investigation by acoustic emission. With a report of liquid crystal transformations being acoustically active[32], it was hoped that the phase transitions of this compound (which occur between 100°C and 180°C would also be acoustically active.

IV.6.2 4-n-pentyloxybenzylidend-4'-n-heptylaniline

A sample of a second liquid crystal was also provided by Professor G. Luckhurst. The phase transitions of this compound fall within the allowed operating temperature of the transducer.

$$C \overset{30.0\,^\circ C}{\text{---------}} S_G \overset{41.0\,^\circ C}{\text{---------}} S_B \overset{51.5\,^\circ C}{\text{---------}} S_C \overset{53.1\,^\circ C}{\text{---------}} S_A \overset{63.2\,^\circ C}{\text{---------}} N \overset{78.0\,^\circ C}{\text{---------}} I$$

This experiment was performed by Dr. P. Y. T. Chow and used a temperature probe in direct contact with the sample.

V. Development of Data Acquisition and Analysis Procedures

This project was begun in October 1987. There were no previous students of this research group working in this area, since this group was founded at around the same time. Careful design of both equipment and software was critical if meaningful chemical acoustic emission data were to be collected, and the system developed was to be widely applicable. The criteria used were that the apparatus should:

i. be relatively inexpensive;

ii. allow broad range of acoustic frequencies to be monitored;

iii. be controlled by one or more personal computers of the IBM PC family, and not be dependent on UBC site mainframes;

iv. provide integral mass storage for raw data and processed files;

v. provide adequate signal capture rates (> 2 MHz burst mode) and resolution (minimum of 8-bit i.e. 0.4%)

vi. use software capable of supporting the highly graphical user interface needed for signal processing applications;

vii. be useful to both the university laboratory, and to industry.

viii. where possible, use commercially available hardware but in-house developed software (the reasons for which are discussed below).

xi. be expandable and modular to allow future needs to be met (e.g. use of two sensors or faster acquisition rates).

x. where possible exceed both the hardware and software specifications of the equipment developed by Prof. D. Betteridge's research group at University of Wales in Swansea, and at British Petroleum Research Centre, Sunbury-on-Thames, UK.

Hardware Design: An appropriate starting point for this was the experimental apparatus which was published by Belchamber *et al.* in 1983[30], and modified for subsequent use in 1986[33]. A full comparison of these systems with those designed as part of this work is given in Table 1.

Table 1. Comparison of Acoustic Emission Apparatus Types

Type	Equipment	Type of Data	Analysis	Limitations
A	- microphone - chart recorder	- qualitative data - "if and when" AE is occurring	- detect different processes if different intensities in time domain	- no frequency info. - no quantified data
B	- integrator - chart recorder - storage device (eg. PC)	- quantized measure of AE energy	- look at nature of driving force. - rate kinetics from integration of AE energy	- no frequency data - can't view individual signals
C	- multiple band pass filters - integrators - storage device	- quantized measure of acoustic frequency components	- elimination of noise components - can see different processes if different frequency components	- more expensive - can't see individual signals
D	- high speed digitizer - storage device	- digitization of individual signals	- pattern recognition of individual signal types - different classes of signals visible if different signal shapes	- not real-time data - energy quantization less reliable - most expensive

These earlier systems were based around Digital Equipment Corporation (DEC) MINC 11/23 processors and peripherals. The far less expensive IBM PC/XT and PC/AT range looked to be the most suitable replacement for these. Apple Macintosh systems were the only other real contender, but at higher expense, without an open architecture, and with far fewer commercially available data acquisition cards. Fast 12 MHz PC/AT's were readily available in the fall of 1987. Newly introduced Intel 80386-based systems, while still expensive, promised to provide an upgrade route compatible with the hardware and software to be initially developed. Indeed, this has since allowed us to achieve faster performance with minimal additional development. A further perceived advantage was that the maintenance of DEC hardware is generally the domain of DEC specialists, and that the cost of hardware and software maintenance contracts from DEC was 10-15% of the initial cost per annum, a recurring annual charge which we could well do without. Maintenance of IBM PC's, where necessary, could be handled by local suppliers or by the skilled staff of our electronics workshop.

We chose to keep the transducer and conditioning amplifier specified by Belchamber, and previously used by Betteridge[33], since at the time no better products were known of, and the bandwidth of these units was considered acceptable. Funding was not available for apparatus capable of handling multiple sensors at this time.

Belchamber[30] originally used a Tektronix transient digitizer to capture individual signals. More recent work had used rack-based analog-to-digital convertors (made specifically by DEC for MINC systems). The functionality of both could be achieved by any of a range of more recent digital storage oscilloscopes produced by Tektronix, some of which were available at a substantial discount to the university. These were capable of communicating with IBM PC class computers via a (slow) serial

interface or a much faster IEEE-488 parallel instrumentation interface. The latter interface option was chosen. A Rayonics fast transient capture system was also actively pursued, since this provided extended record lengths, however, the Tektronix 2230 oscilloscope eventually purchased was half the price and adequate additional funding was not available at that time.

Software: Data acquisition and processing programs developed by Betteridge's group and used by Belchamber[30,33] were written in DEC FORTRAN. These contained many useful routines, but provided essentially character-based output. In part this was due to the difficulty of writing quality graphical interfaces in standard FORTRAN 77. It was decided to abandon FORTRAN in favour of a language more able to make full use of the PC environment, including colour, pixel-based graphics, and pointing devices such as a mouse or light-pen. The alternative languages initially considered were Pascal, BASIC, C, and FORTH. Although other workers in this department had made wide use of Pascal (Turbo Pascal, Borland, Scotts Valley, CA.), it was not ideal because of its inflexible syntax (compulsory declaration of variables, use of semicolons and BEGIN/END statements, etc.) and its decreasing use in academic environments. The lower level languages, C and FORTH, while very powerful and fast, were known to result in code which was difficult to maintain by students who were, first and foremost, chemists by training. FORTH had the obvious advantage of true multitasking, which would have simplified some aspects of instrument control, but is not very readable. A fast, compiled version of BASIC, which allowed structured programming and modularity like Pascal, the mathematical capabilities of FORTRAN or C, and easy control of instruments (like FORTH) through drivers, was the obvious choice. MicroSoft QuickBASIC was selected. Borland's TurboBASIC could also have been used, but has since not been supported so widely by hardware manufacturers.

Since we were not going to be using the same hardware as Betteridge, purchase of their software was not sought. Rather, the mathematical principles behind key routines they used and several others were researched, and encoded. In-house development of fresh data acquisition and analysis software had several valuable advantages.

i) the ability to add to and otherwise customize the code to exactly meet the needs of this research project, and its sponsors;

ii) an intimate understanding of the algorithms employed and the opportunity to use the most appropriate algorithm to solve each problem;

iii) the ability to design and use proprietary binary file formats ideally suited to chemical acoustic emission data, and so save valuable storage space which would have been required by less efficient encoding (e.g. as larger flat ASCII or binary-format database files which may carry a large overhead);

iv) the training of a research student in multivariate methods of data analysis and chemometrics;

v) long term maintenance of the software routines encoded by future students in this group, and easy incorporation of these routines into their own programs (thus saving them considerable effort).

An equivalent commercial software package (ICEPAK, TekTrend International, Montreal, PQ) was available for one small part of the task - signal classification. However, its cost was US$ 9999 per copy for a compiled version without source code. The utility of this code for analyzing our data was evaluated by staff at the distributing company in cooperation with our postdoctoral researcher, Peter Wentzell. It was considered useful, but too restrictive for general usage. Even if commercial software

had been available at a more reasonable price, the lack of source code would have severely restricted its utility in the research environment.

The major task of this thesis was design and development of this software. The task was achieved largely by the author, with help from Peter Wentzell and Ph. D. student Oliver Lee. A discussion of this code, and more recent additions by other members of this research group, will be submitted for publication[87].

V.1 Data acquisition - the QAQ.BAS program

The program QAQ.BAS (Quick AcQuisition), co-written with D. A. (Tony) Boyd, and subsequently modified by O. Lee and K. A. Soulsbury, is a driver program for the oscilloscope (Figure 25) which allows the user to enter the experimental parameters such as experiment title, amplifier setting, etc. and then collects the signals. (See Appendix). The experimental information is stored along with the signals in an .AEA file (Acoustic Emission Analysis). This facilitates consistent comparisons with other AE experiments. While using the program the oscilloscope is constantly monitored by the microcomputer. The oscilloscope is continuously digitizing the electrical input from the transducer and amplifier. A trigger threshold is set slightly above the level of background noise. When the sample produces an acoustic signal of magnitude greater than the preset threshold, the oscilloscope is triggered and stores the waveform. The computer, upon being notified that a signal has been acquired, downloads the signal from the oscilloscope, stores it on the hard disk, and then signals the oscilloscope that it is ready to receive another signal.

The .AEA file is a raw data file. One is created for each experiment. This contains a header, which has all the operating parameters of the experiment and some

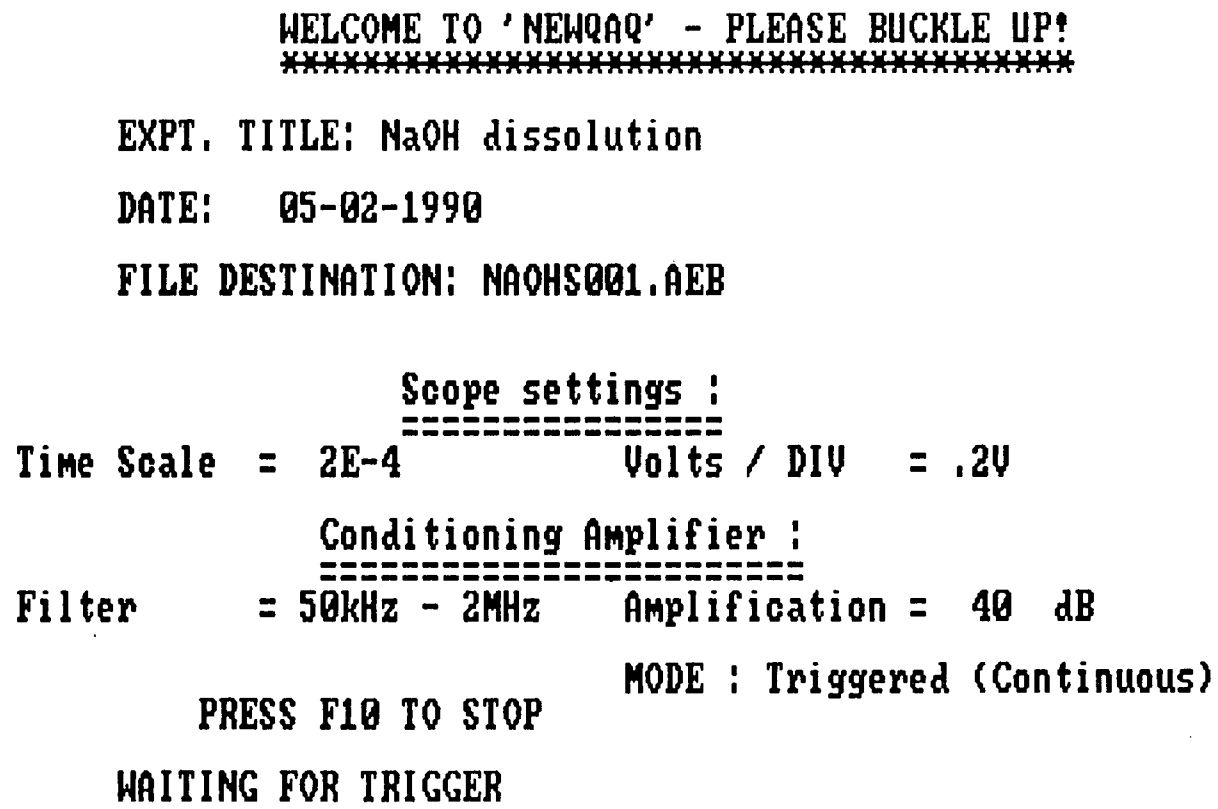


Figure 25) QAQ - Program display during acoustic emission experiment.

comments, followed by the signals produced during the experiment (as linear arrays of integers from 0 to 255) and the time of acquisition of each signal (recorded using the computer's internal clock). The signals are identified by number (stored as a four character string) and are each assigned a four character classification (which initially is the same by default for all signals from any particular experiment). The data files are in such a format that allows inclusion of parameters such as pH, temperature, etc. (see Appendix 1).

The use of 8-bit integers (which have a numerical range of 0 to 255) to store the signal is the most efficient means in terms of disk storage. The array of integers can be converted into the signal's voltage profile by using the voltage per division setting of the oscilloscope which is read and recorded by the computer. For example, to convert the amplitude-time profile of the signal, t - where t_i is the integer value for the signal amplitude at time interval i, to the voltage profile s, the following equation is used:

$$s_i = (t_i - t_A) \cdot VDIV \cdot 8 / 1024 \qquad (V.1)$$

where t_A is the average value of t_i. There are 8 divisions in which the 0 to 255 range is divided.

V.2 Viewing Individual Signals - SIGVIEW.BAS

The signals themselves can be recalled and viewed at any time after the experiment is terminated. Dr. P. D. Wentzell has written a program, named SIGVIEW[88], (Figure 26) which can display each individual waveform graphically (Figure 27). With this it is possible to scan backwards and forwards through the .AEA file. It is also possible to search through the file and select individual signals for viewing

based on their time of acquisition, numeric identifier, or their assigned class. A Fast Fourier transform (FFT) of the signals can be calculated at any point and viewed (Figure 28). An option for windowed FFT's is also included. Hardcopy output is possible by selecting the option to save a signal in ASCII[B] format for later use by commercially available plotting programs such as SIGMAPLOT (version 3.0, Jandel Scientific, Sausolito, California) and LOTUS 1-2-3 (release 2.2, LOTUS, Cambridge, Maryland).

V.4 Signal Classification and Editing

SIGVIEW allows one to change the classification of a signal. The four character field which contains the class for each signal can be changed to something meaningful. Experimental experience allows some signals to be immediately recognized as noise. Electrical noise (Figure 29) has a very distinct waveform when compared to a transient from a chemical reaction (eg. Figure 27). Also easily identified are signals originating elsewhere in the mains power supply, possibly due to a relay switching machinery on or off (Figure 30).

It is also possible to acquire signals which have their maximum and/or minimum amplitude outside the range that the oscilloscope was set to digitize (Figure 31). This type of signal may not be useful for frequency based data analysis as it will have a frequency spectrum which contains artefacts due to the artificial restriction of the signal to a maximum/minimum value. Grossly over range signals should perhaps be largely ignored for data analysis purposes.

B. American Standard Codes for Information Interchange. This type of file is one in which the data is stored in an unencrypted form. The file is therefore "readable" without the use of a program to decode the information.

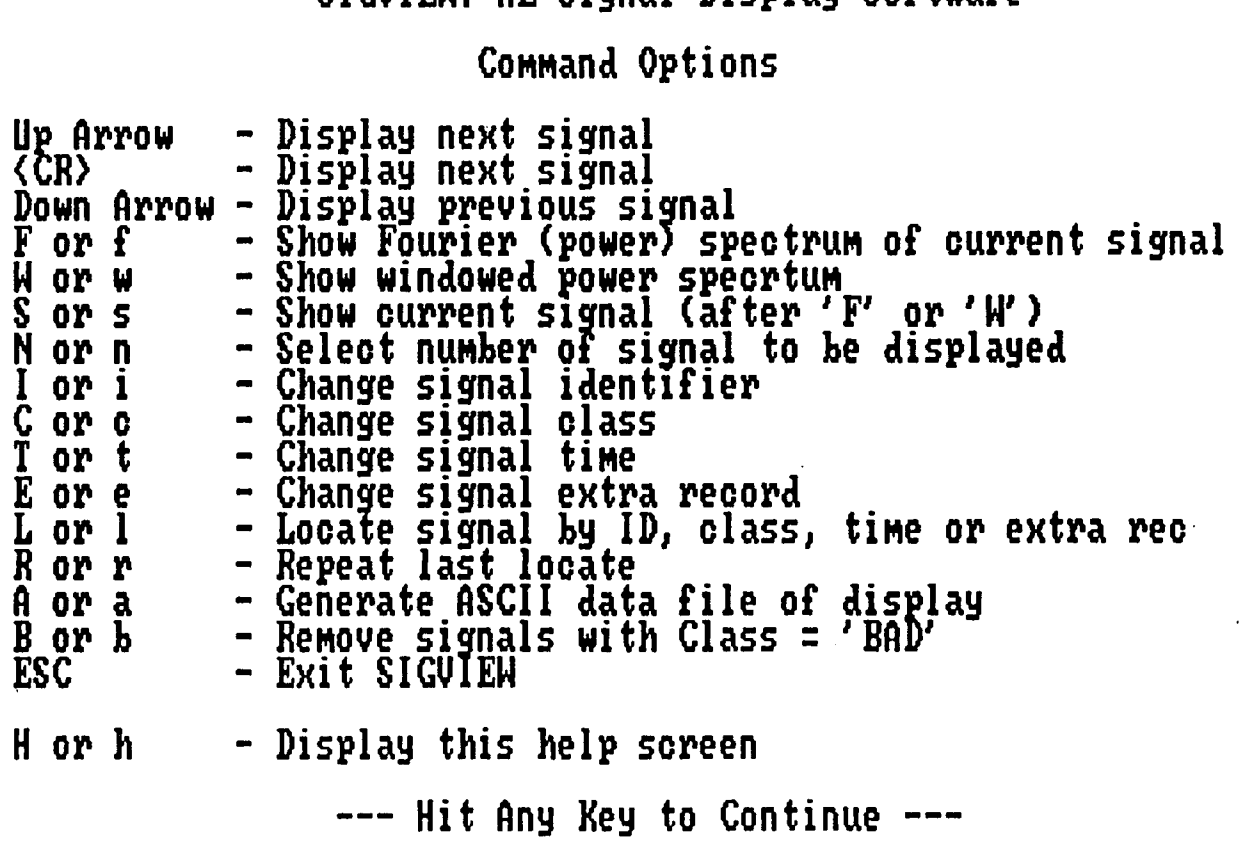


Figure 26) SIGVIEW - Program options for viewing acoustic emission experimental data from .AEA files.

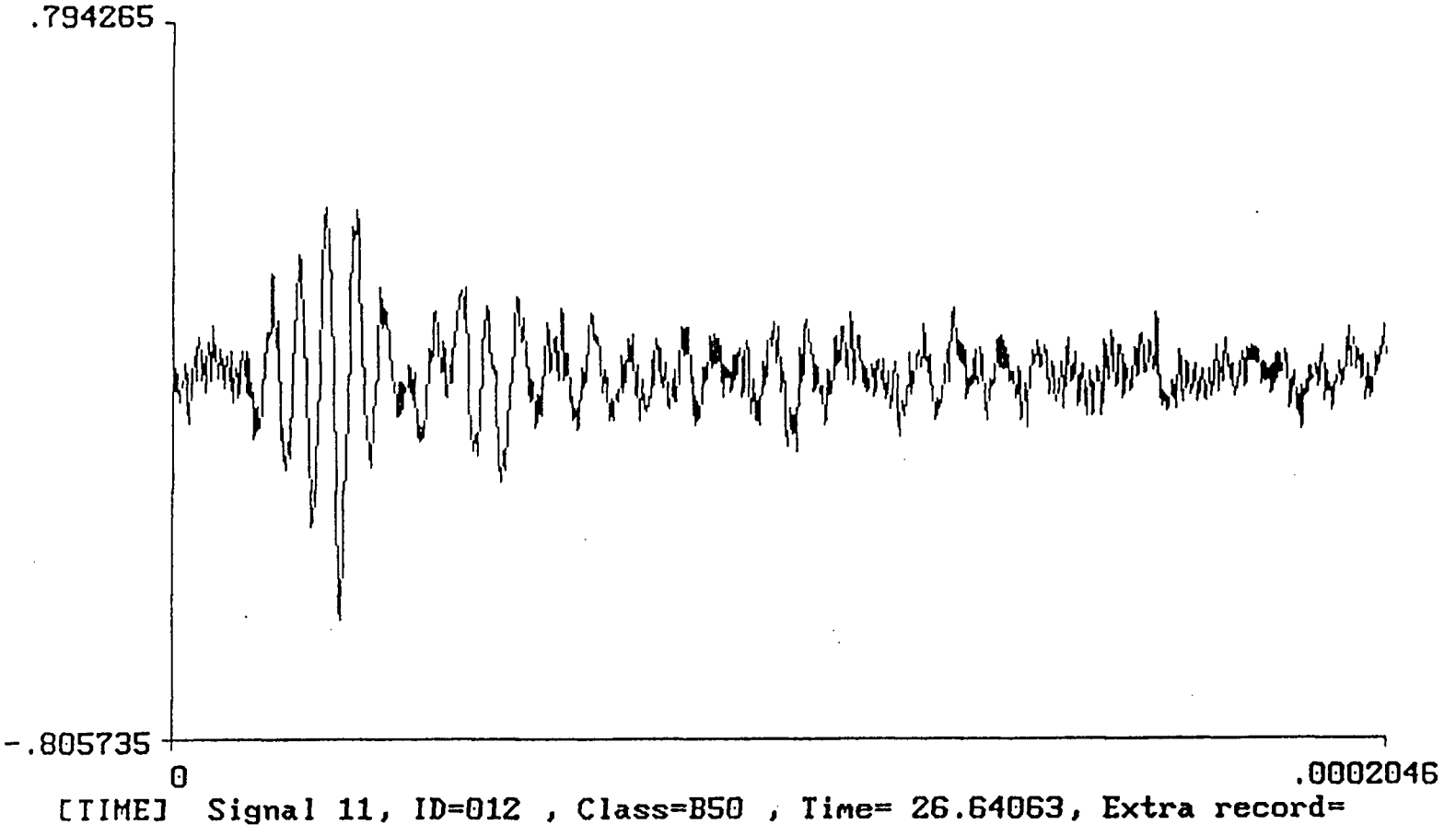


Figure 27) SIGVIEW - Program display of acoustic signal from cooling of liquid crystal - 4-n-pentyloxybenzylidene-4'-n-heptyaniline. Vertical axis is measured in Volts and Horizontal axis is measured in seconds.

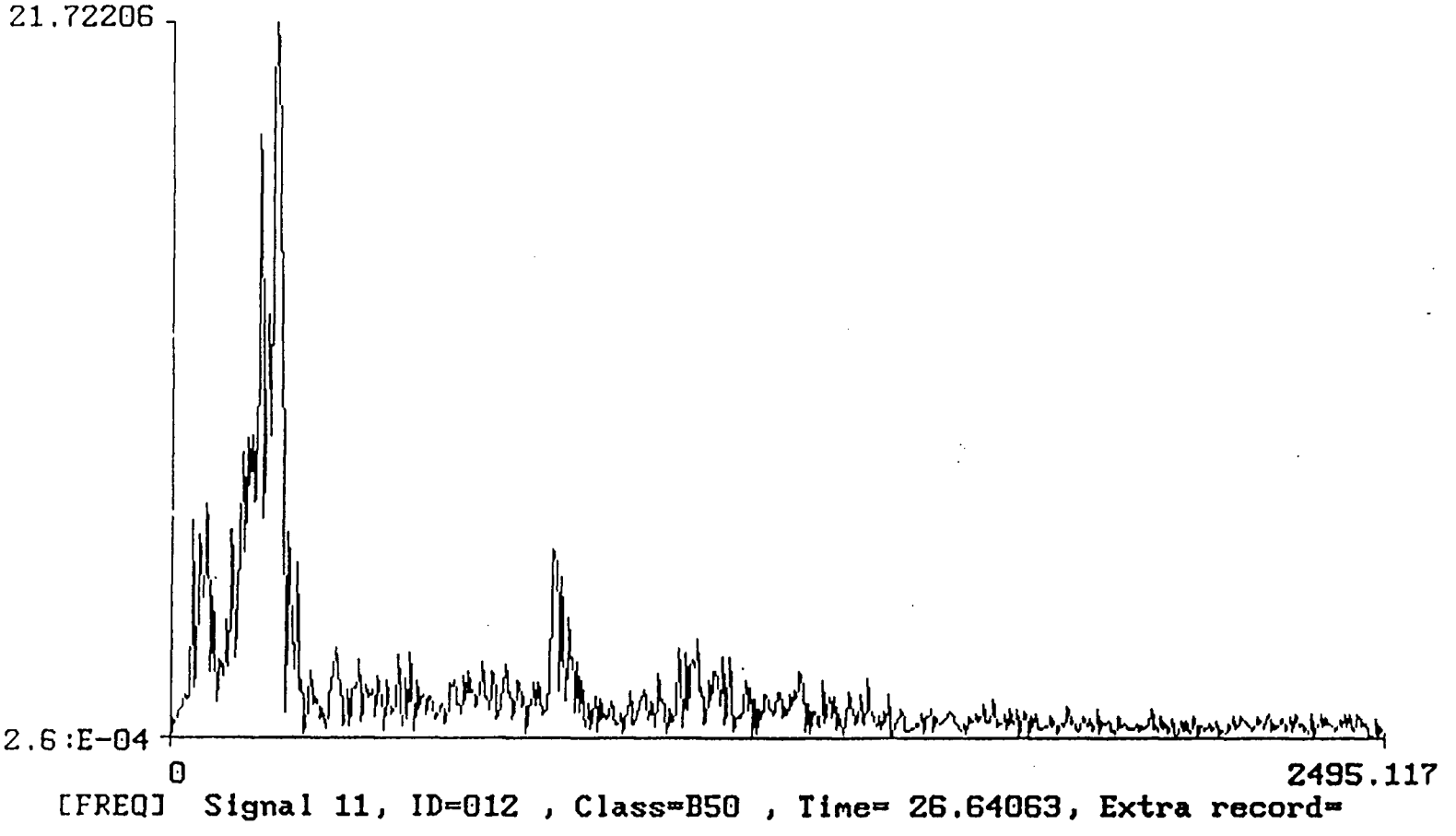


Figure 28) SIGVIEW - Program display of calculated power spectrum of signal in Figure 27. Horizontal axis is measured in kHz. Vertical axis corresponds to arbitrary units.

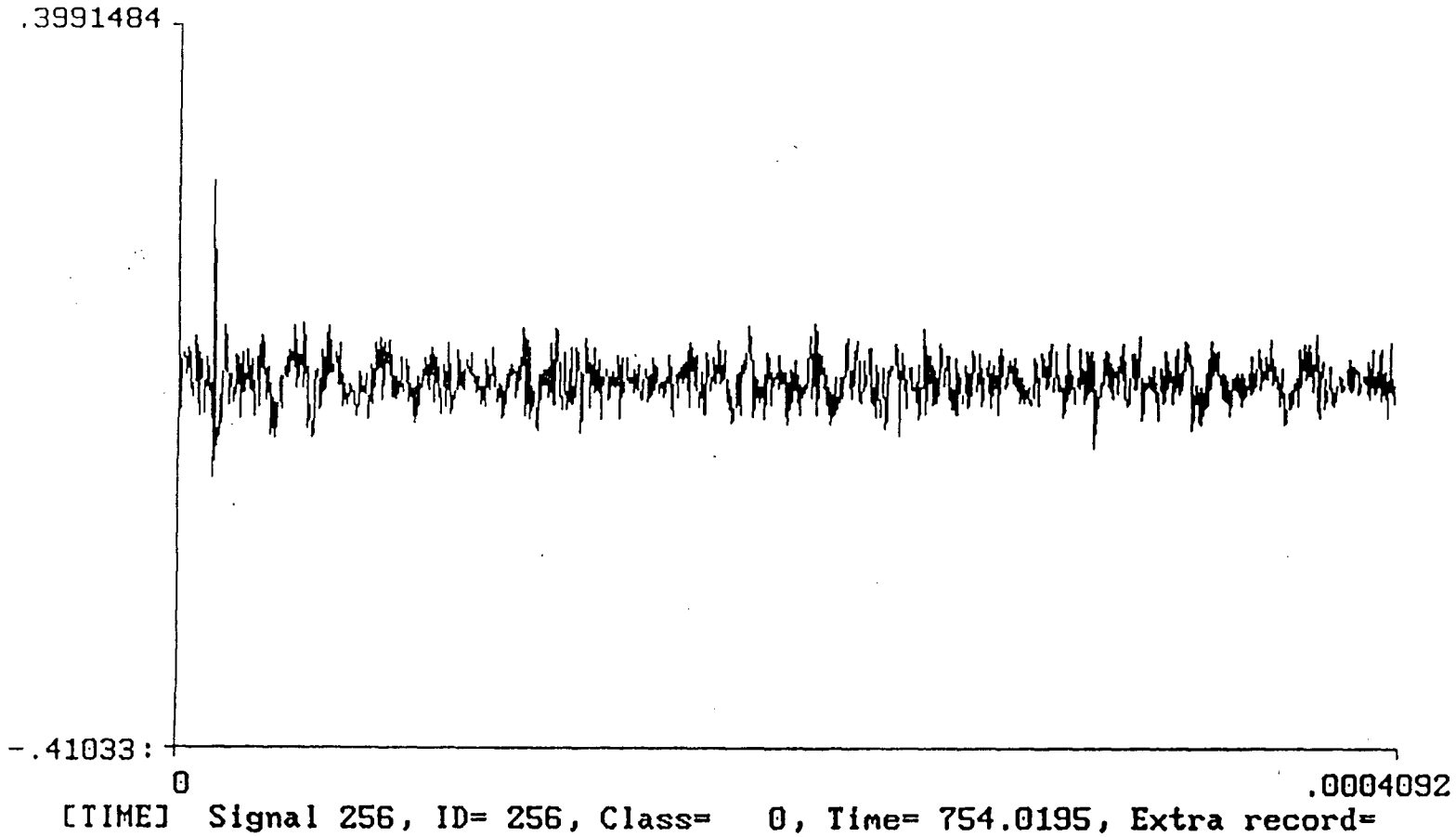


Figure 29) SIGVIEW - Program display of electrical noise signal - background noise with one spike. Vertical axis is measured in Volts and Horizontal axis is measured in seconds.

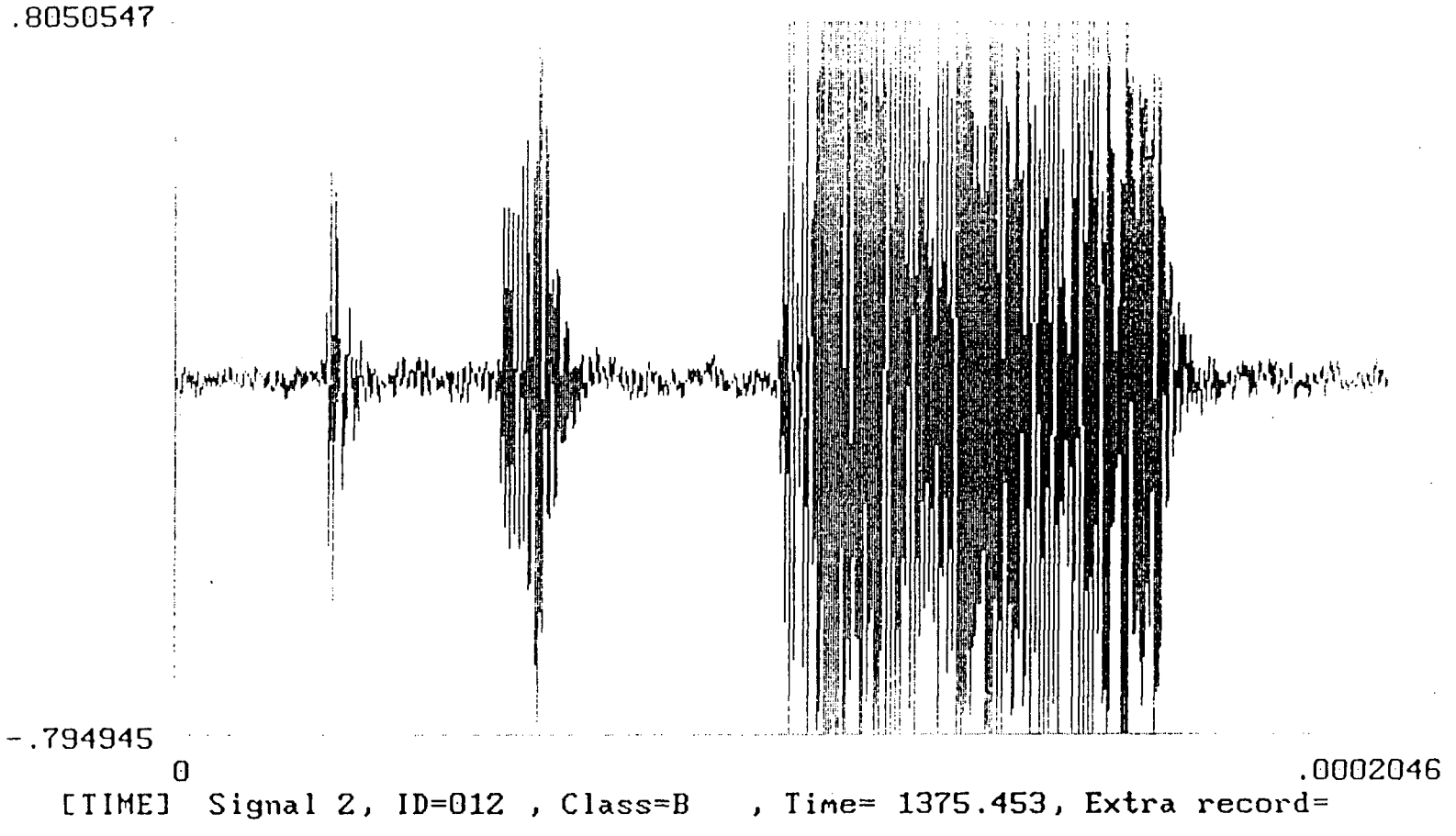


Figure 30) SIGVIEW - Program display of electrical signal (noise) repeatedly found on power mains. Vertical axis is measured in Volts and Horizontal axis is measured in seconds.

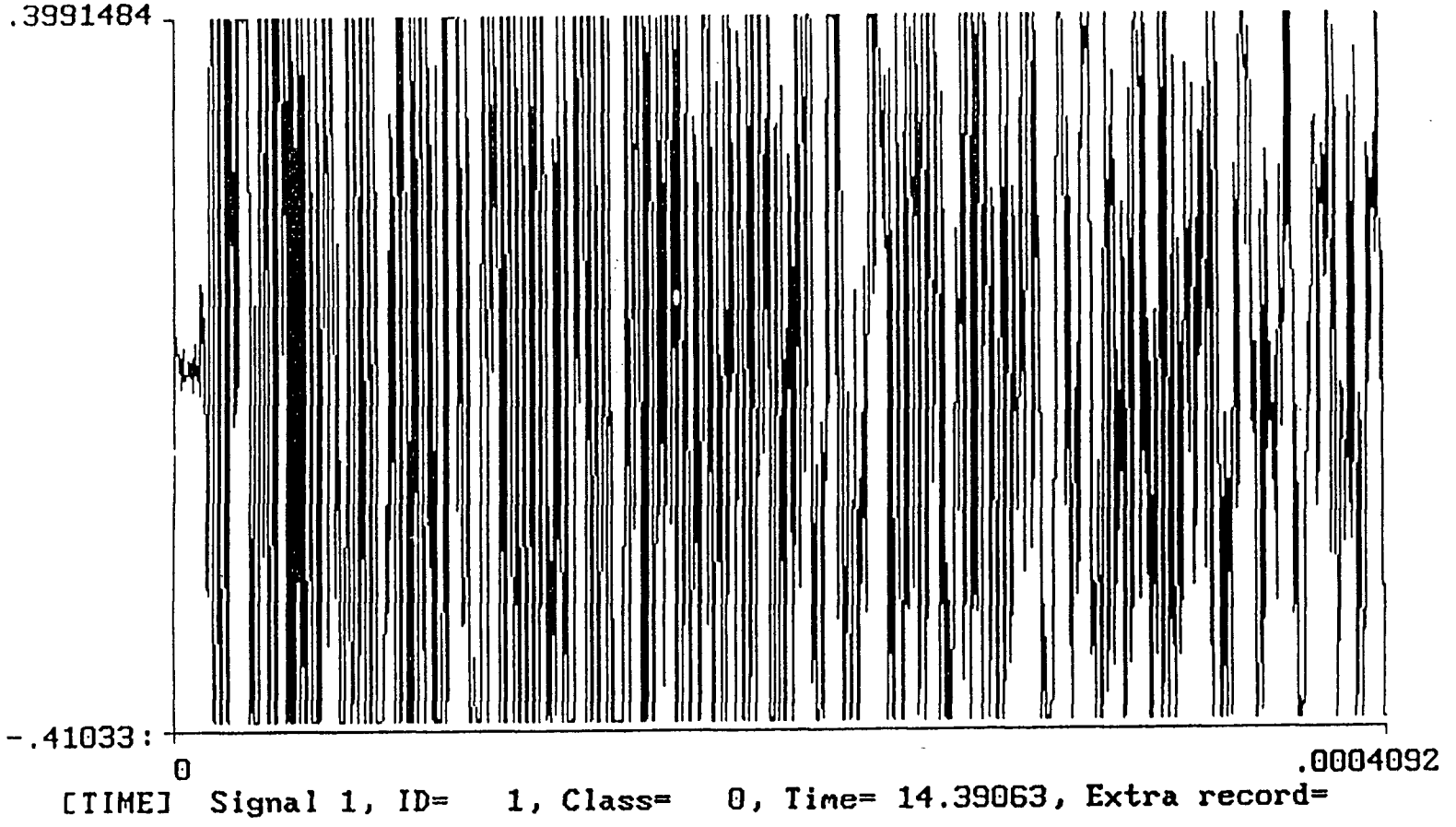


Figure 31) SIGVIEW - Program display of intense acoustic signal having amplitude beyone the Voltage range of the oscilloscope settings. Vertical axis is measured in Volts and Horizontal axis is measured in seconds.

In the case of such "improper signals", the class field can be changed to "BAD_". An option exists in SIGVIEW to delete the signals which have the class of "BAD_". This results in a smaller data file which will be more representative of the experiment. However, if unwisely used it can eliminate useful data and so lead to disaster. The electrical noise signals are easily identified, have essentially the frequency content of the background and show up as sharp spikes on the chart recorder. For these reasons they may indeed be useful as they give some indication of the background frequency components and are helpful in that they can be used to calibrate the time axis on the chart recorder output where there are no other signals.

An experiment may produce numerous AE signals. As discovered later, some samples emit continuously for several hours. At an acquisition rate of one signal about every other second for the Tektronix T2230 scope, perhaps 7500 signals could be recorded for a single experiment! Not only does this require ca. 7.5 Mbytes of storage but to look through these by hand even with the ease of utility of the SIGVIEW program would take too long and would be unable to provide much meaningful information. Thus other more automated methods of data analysis are necessary.

V.5 Frequency Content Analysis Methods - VTRAPS.BAS

To get a better understanding of the way the frequency content of the acoustic signals changes over the course of the experiment a three dimensional view is convenient. These can be given by time resolved average power spectra (TRAPS) and time resolved total power spectra. Commercial software such as SURFER (v. 3.0, Golden Software, Golden, Colorado) is available to plot three dimensional data sets. A program was needed to extract the time domain information from the data file, calculate the Fourier transform, obtain the power spectrum, and store this data as a

three dimensional data file in ASCII format for use by the plotting program. The power spectrum of a signal shows the energy of each frequency component and is calculated from the FFT by summing the squares of the amplitude of real and imaginary components for each frequency interval and taking the square-root of the result. This project was begun by J. A. Horner and evolved into VARI-TRAPS - a program written largely by O. Lee of this research group.

The duration of the experiment is divided into time windows. The power spectra of the signals that fall into each particular window are averaged together. The result is a data file (with the extension .TRA) which contains three dimensional data - time, frequency, and intensity. This can then be used to generate a three-dimensional plot of how the frequency content of the acoustic emissions of an experiment (or of experiments) changes over time (Figure 32). It is also possible to average several experiments together in this way, or compute difference maps.

VARI-TRAPS will also calculate the variance spectrum (Figure 33) of an experiment. The frequency spectrum is an array of 512 points - each corresponding to a narrow frequency range. The variance of each frequency component within each time window is calculated for the duration of the experiment. If two different types of signals are generated by a process, and the frequency content of these signals is different, then there will be high variance for frequencies that are present in one of the signal types and not present in the other. Conversely, there will be a low variance for frequency components which are either present or absent from both.

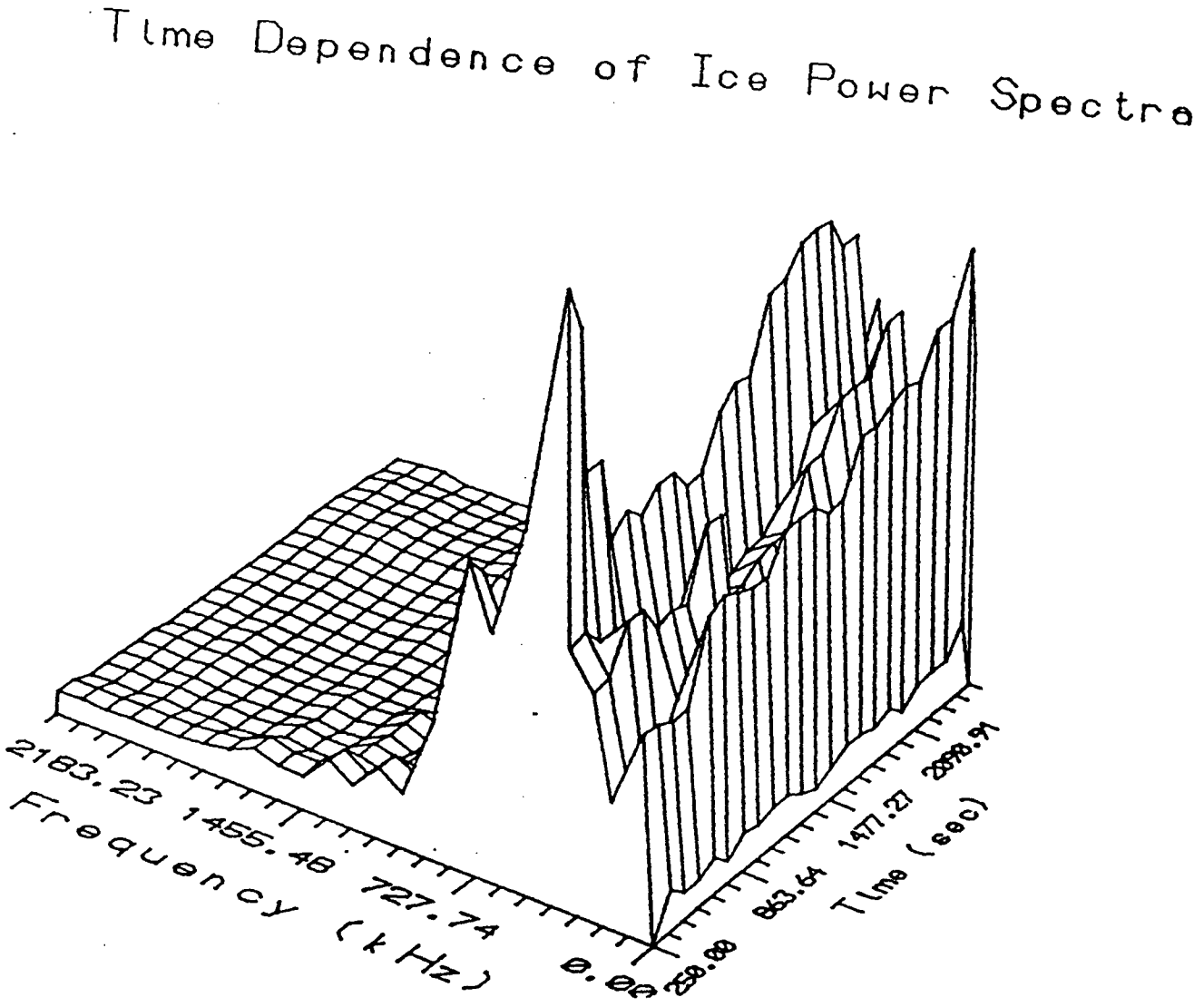


Figure 32) VTRAPS - Time resolved average power spectrum for melting ice. (Experiments and data analysis performed by Dr. P. D. Wentzell.)

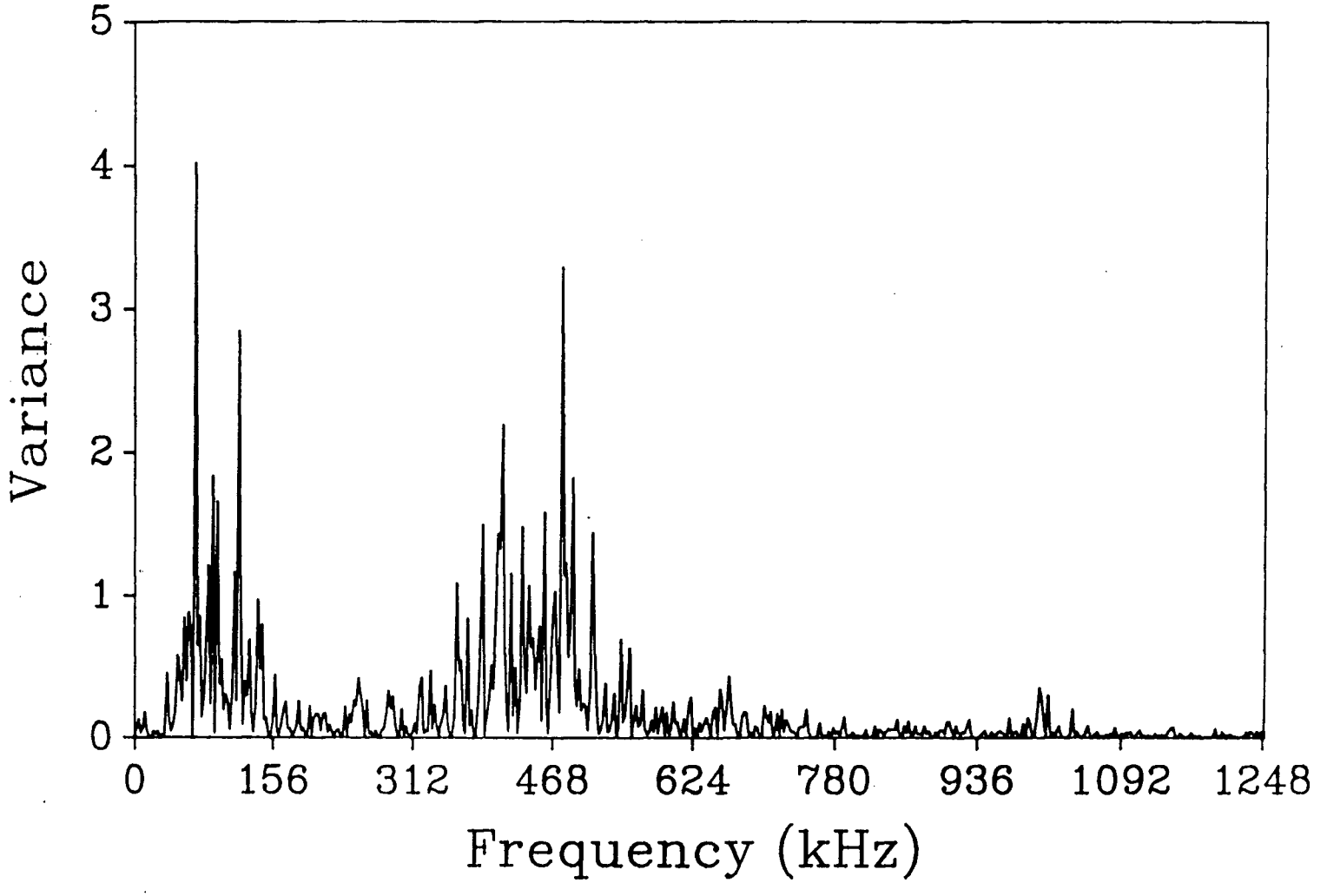


Figure 33) **VTRAPS - Frequency variance spectrum for the liquid crystal 4-n-pentyloxybenzylidene-4'-n-heptylaniline. Two regions of high variance suggest that more than one process is occurring.**

V.5 Pattern Recognition

Pattern recognition techniques allow one to see more of the structure present in a multi-dimensional data set. The signals may belong to one or more classes. One would wish to see this and assign classes to the signals.

The acoustic emission data is not directly suited to pattern recognition in the form it is collected. Although the data file of time domain signals can be thought of as a multi-dimensional data set (amplitude of the signal at each sample time for each signal), it would not produce as meaningful results for a number of reasons. The signal is a digitized electrical signal resulting from a vibration at the transducer. While, signals may be very similar in terms of energy level, decay rate, and frequency content, the amplitudes and time domain behavior of the waveforms may differ due to small variations in the frequency spectrum and the many different locations within the sample that an acoustic event can occur. An acoustic signal is a vibration which must travel to the transducer. The path the signal takes will affect the final detected waveform due to dispersion and echoes.

To allow for pattern recognition of the acoustic emission data, the signals are not used themselves but rather each signal is characterized by a number of "descriptors". A descriptor is a numerical value which describes a statistical property of the signal. Some properties which may be described in this way are root-mean square voltage (RMS), kurtosis, area, median and modal frequency (terms defined in section VI.1). This transforms the raw signal data set into a smaller multi-dimensional data set that can be effectively subjected to pattern recognition techniques.

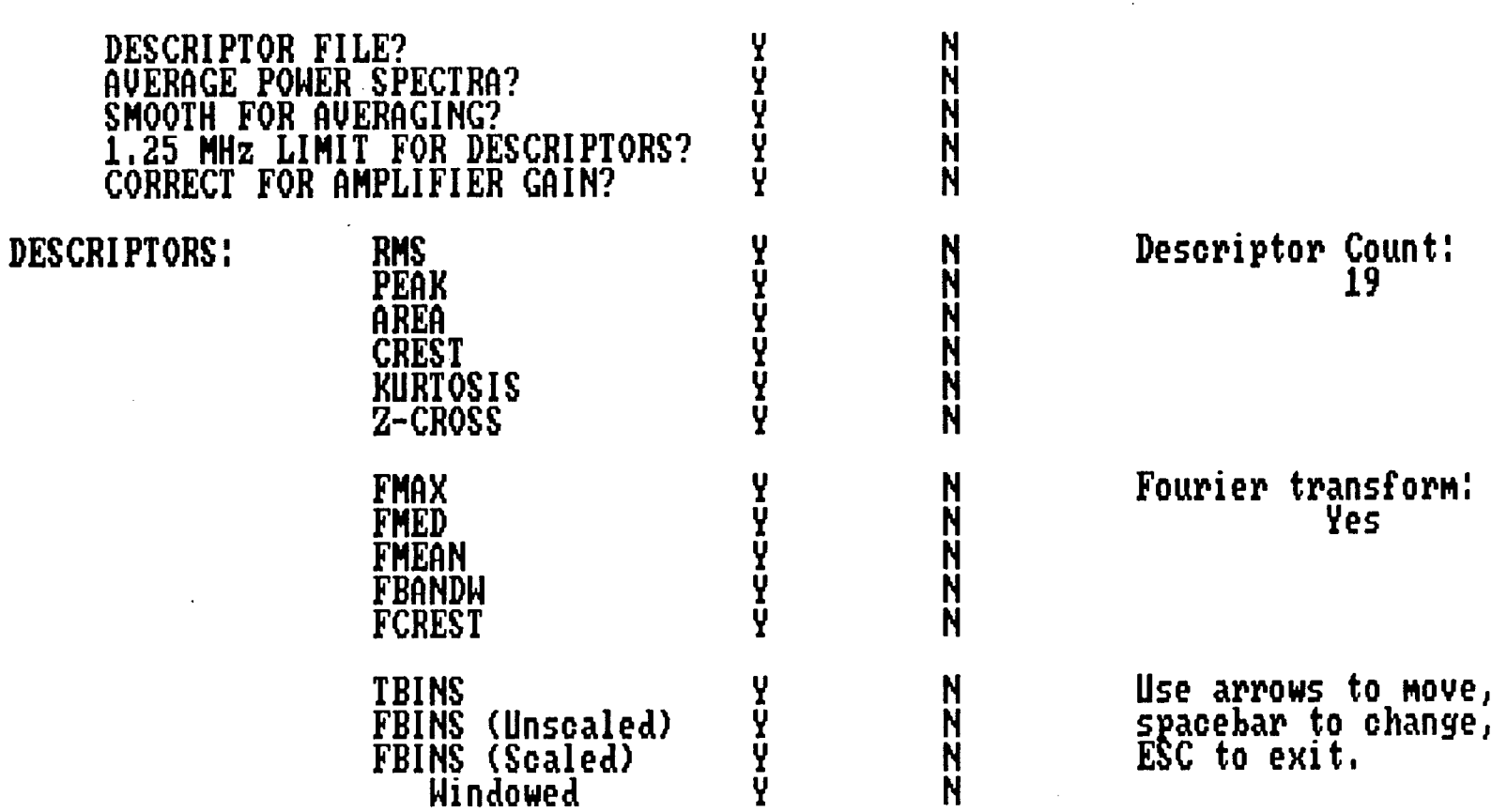


Figure 34) AEMUNCH - Menu giving choice of descriptors to generate for descriptor file (.DS1).

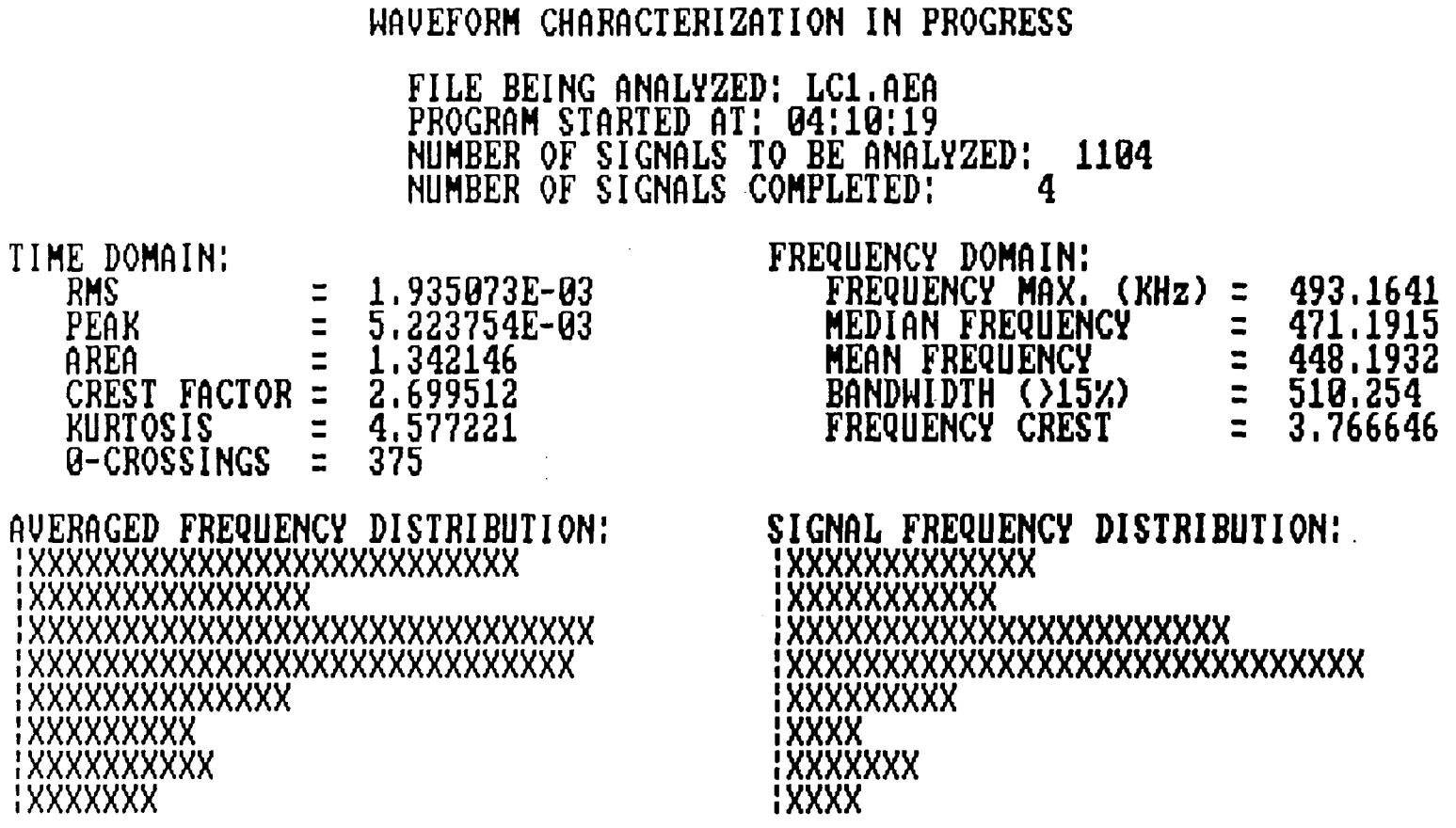


Figure 35) AEMUNCH - Display during calculation of descriptors.

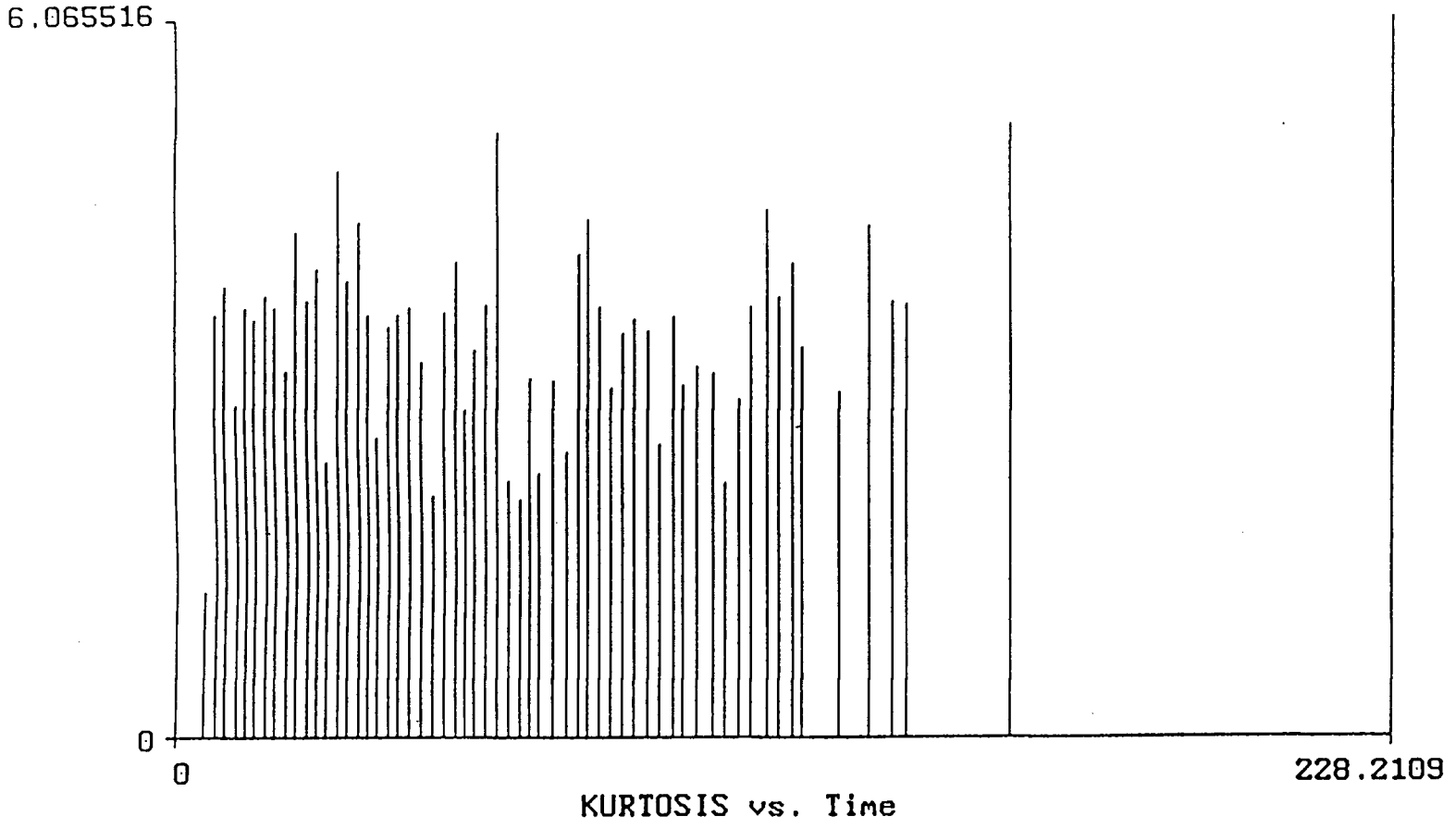


Figure 36) **AEMUNCH - Program display of kurtosis plotted versus time (in seconds) for NaOH experiment using long waveguide. (3 units have been added to the kurtosis for each signal for plotting).**

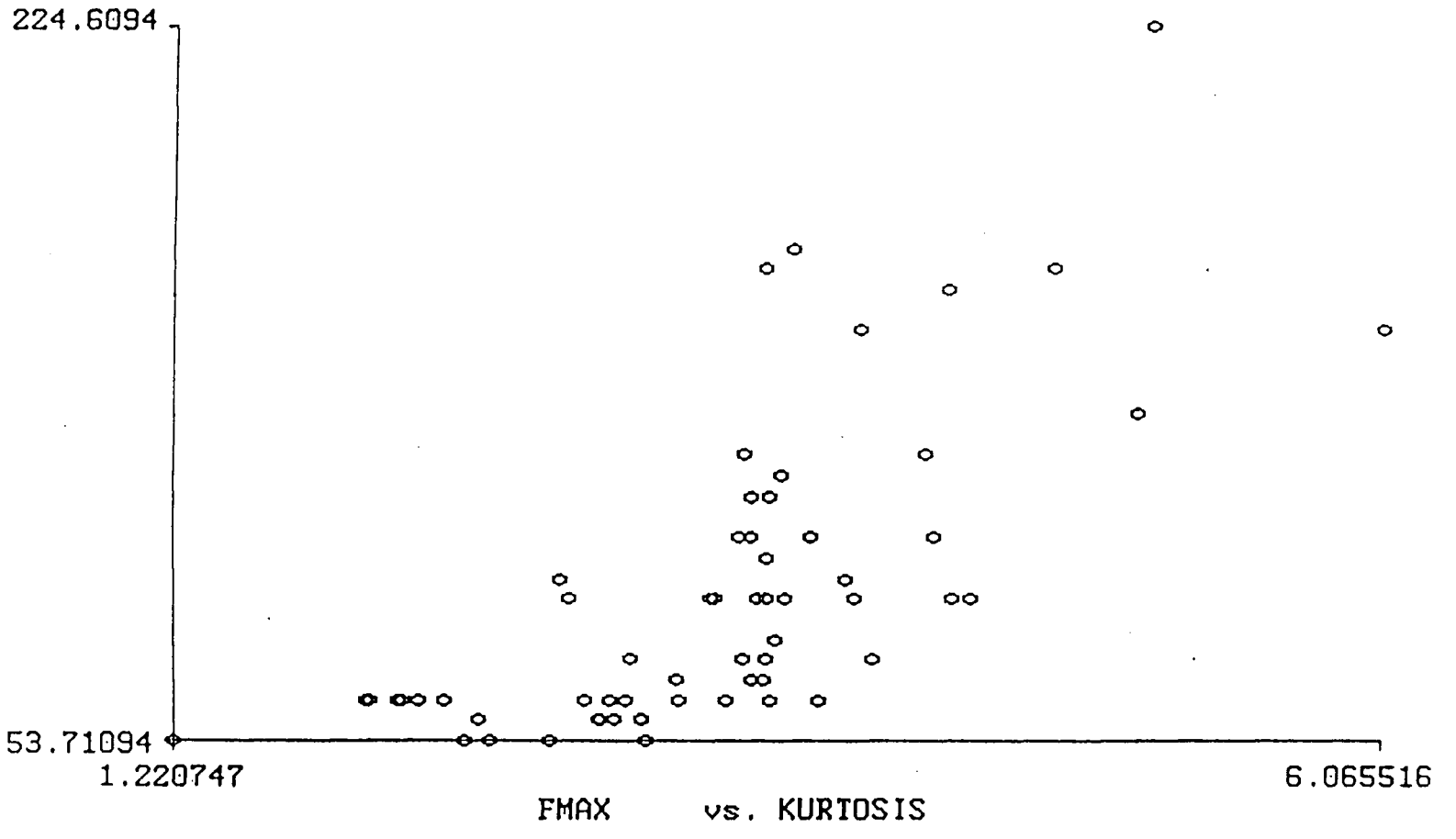


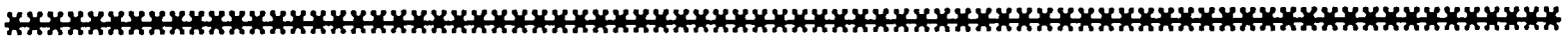
Figure 37) AEMUNCH - Program display for scatterplot of frequency of maximum intensity (FMAX) versus kurtosis (plus three) for NaOH experimental data.

V.6 Descriptor Generation - the Program AEMUNCH.BAS

The program AEMUNCH.BAS, written by Dr. P. D. Wentzell (with assistance from this author and D. A. Boyd), takes an experimental data file (.AEA) and for each signal calculates a desired selection of descriptors (Figure 34). If frequency domain descriptors are requested then the Fourier transform is also calculated when the descriptors are generated (Figure 35). The resulting multi-dimensional data set is saved in a file with the extension .DS1, which contains the descriptors for each signal in an experiment. The descriptor values can be plotted against time (Figure 36), or against each other (Figure 37) using AEMUNCH.BAS. But this only allows for visualization of two dimensional subsets of the data which very likely won't reveal all of the interesting information present in the multi-dimensional data.

V.7 Hierarchical Cluster Analysis - DENDGRAM.BAS

Cluster analysis allows for the visualization of the clustering of the data in feature space. For this, the program DENDGRAM.BAS was written by D. B. Sibbald. This will first load in a descriptor file. Then the operator has a choice of options for scaling the data (Figure 38 - a discussion of scaling techniques follows). Any of several methods for calculating the dendrogram may be chosen (Figure 39) and the calculation follows. Visual display of the dendrograms on the screen allows one to view in close-up detail, and to change the method of display used (Figures 40 - 42). This work has been reported in the literature[89] and is included as an appendix. It is possible that a dendrogram may have been calculated for a set of data with a sufficiently large number of signals such that the resolution of the graphics display is unable to display the signal identifier for each individual signal. (In this case, a page of the signals can be selected (Figure 43) and the identifiers for this page viewed.) The dendrogram is stored in a file with the extension .DEN. This contains information on the original data file, the



SCALING

1) Auto-scaling

2) Range scale (0 - +1)

3) Perform no scaling

4) Special functions.

Use arrows to choose. Press RETURN to select.

Figure 38) DENDGRAM - Menu offerring choice of scaling options.

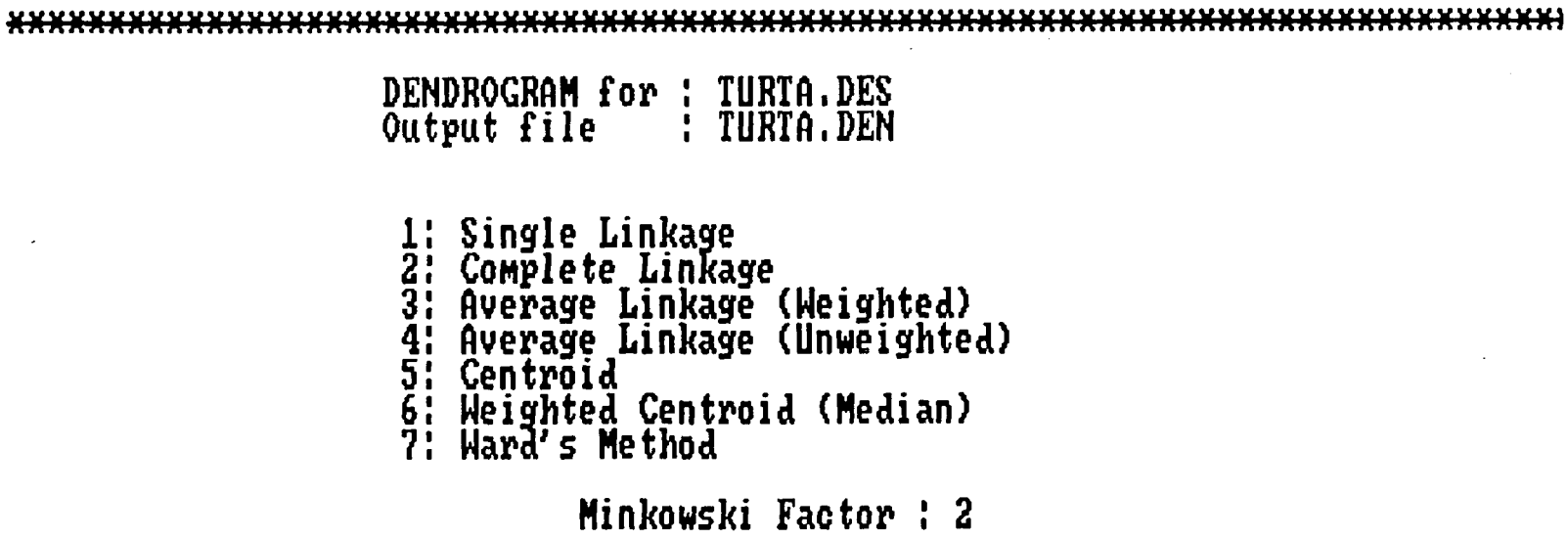


Figure 39) DENDGRAM - Choice of different methods for calculating dendrograms.

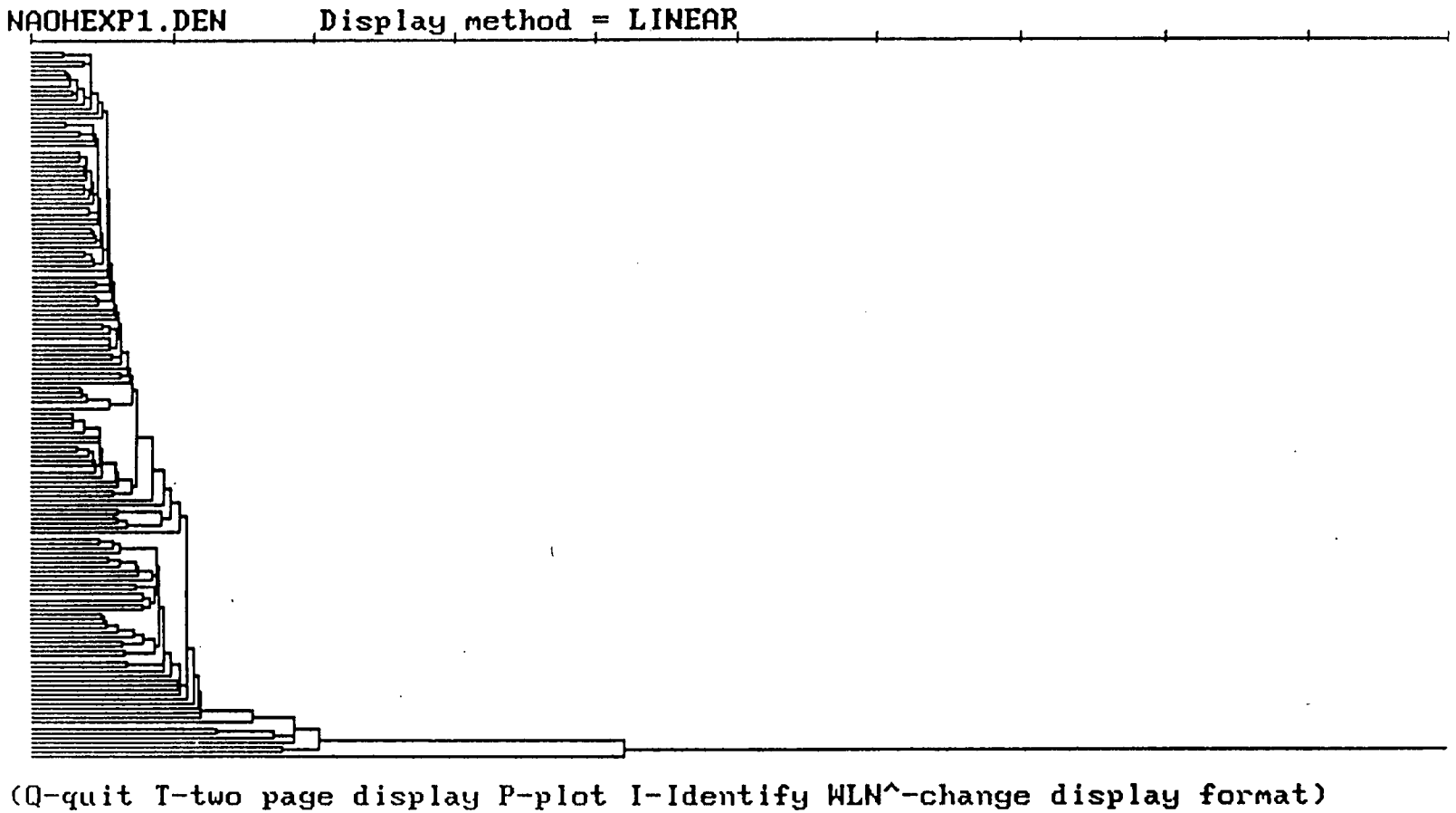


Figure 40) DENDGRAM - Dendrogram display. Signal labels are given on left. Horizontal axis is (dis)similarity.

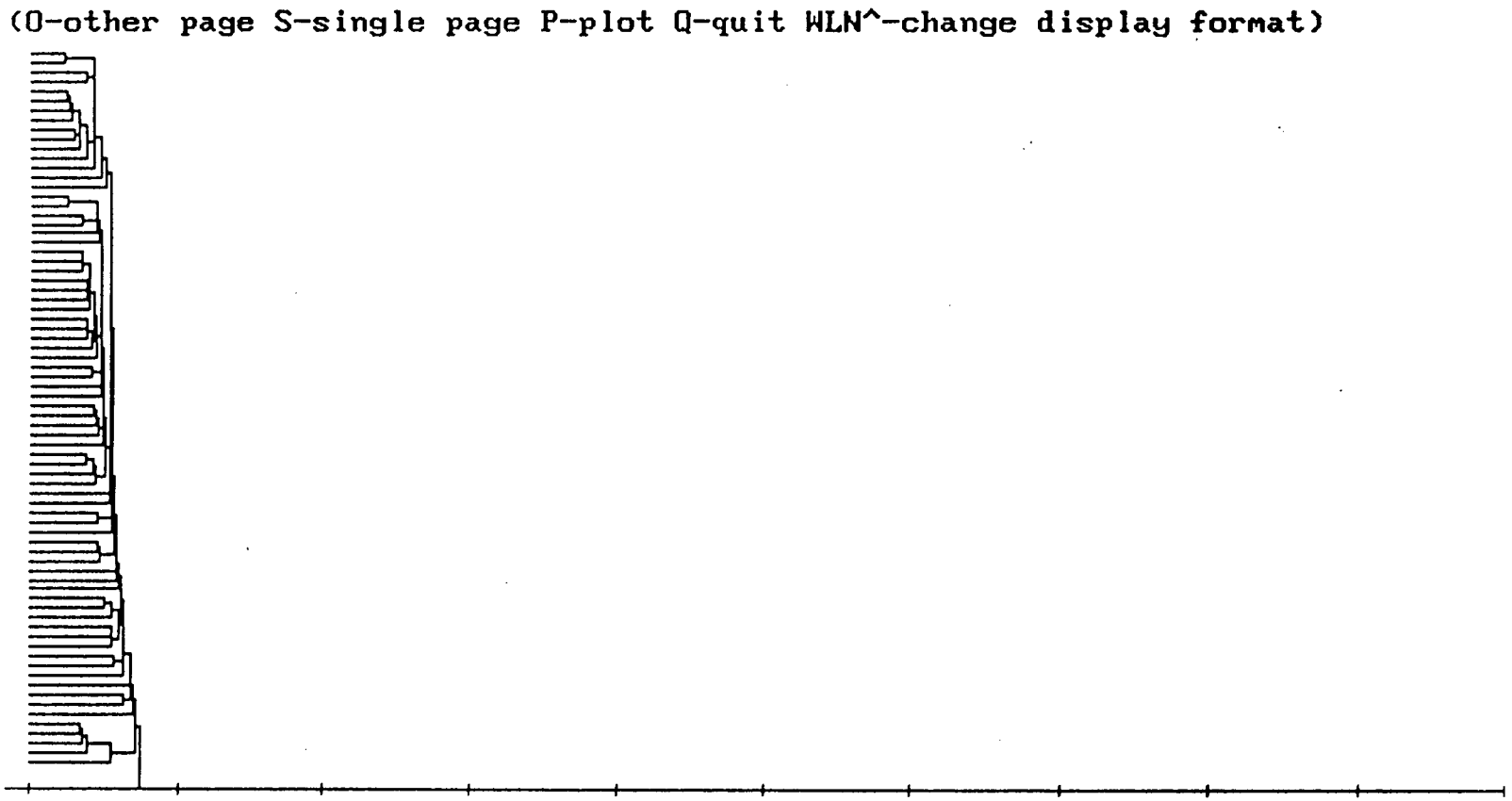


Figure 41) DENDGRAM - A "page" can be selected for a detailed listing of the samples producing the dendrogram. This shows one half of the dendrogram in Figure 40.

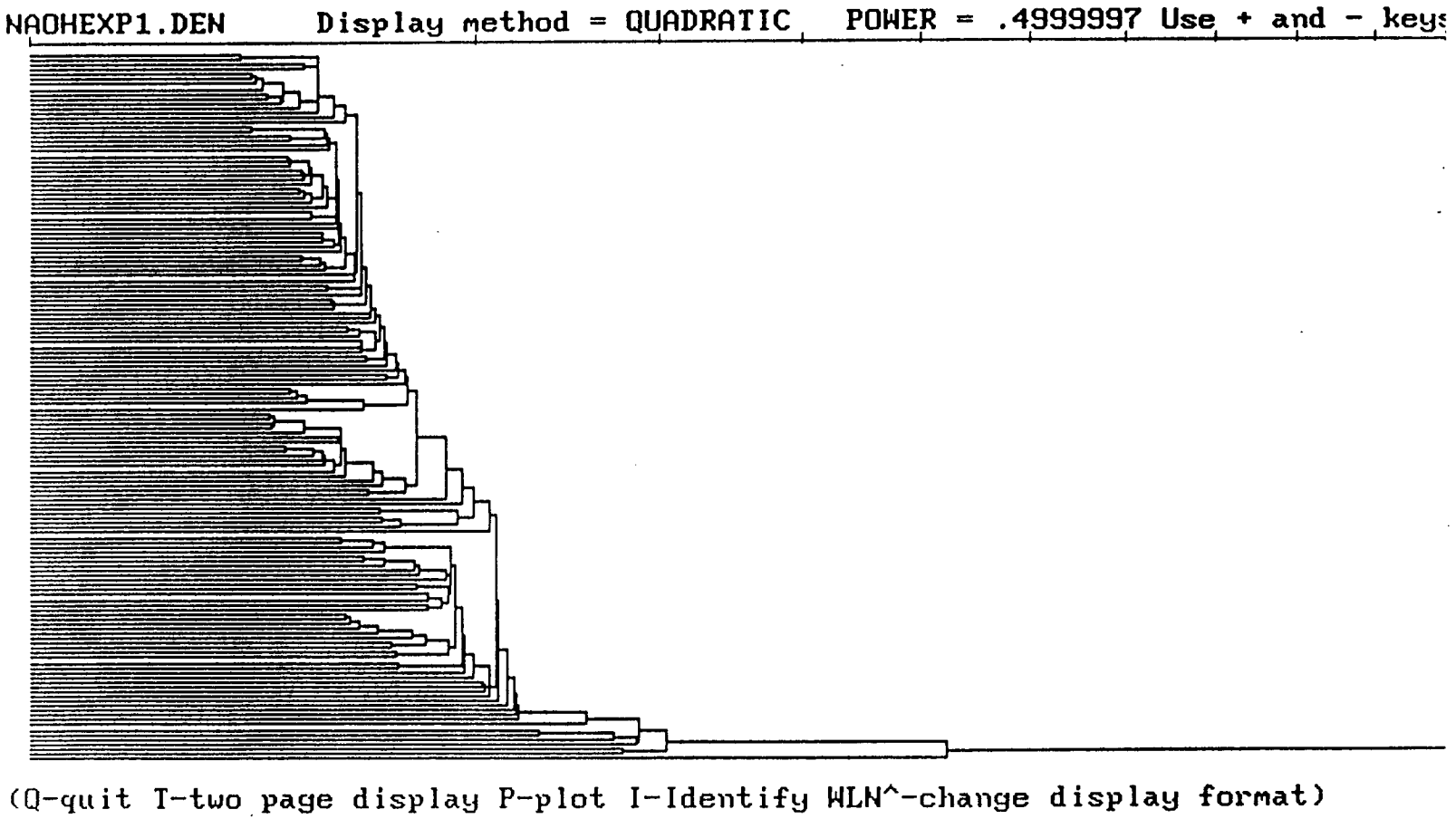


Figure 42) DENDGRAM - Use of a square-rooted similarity axis.

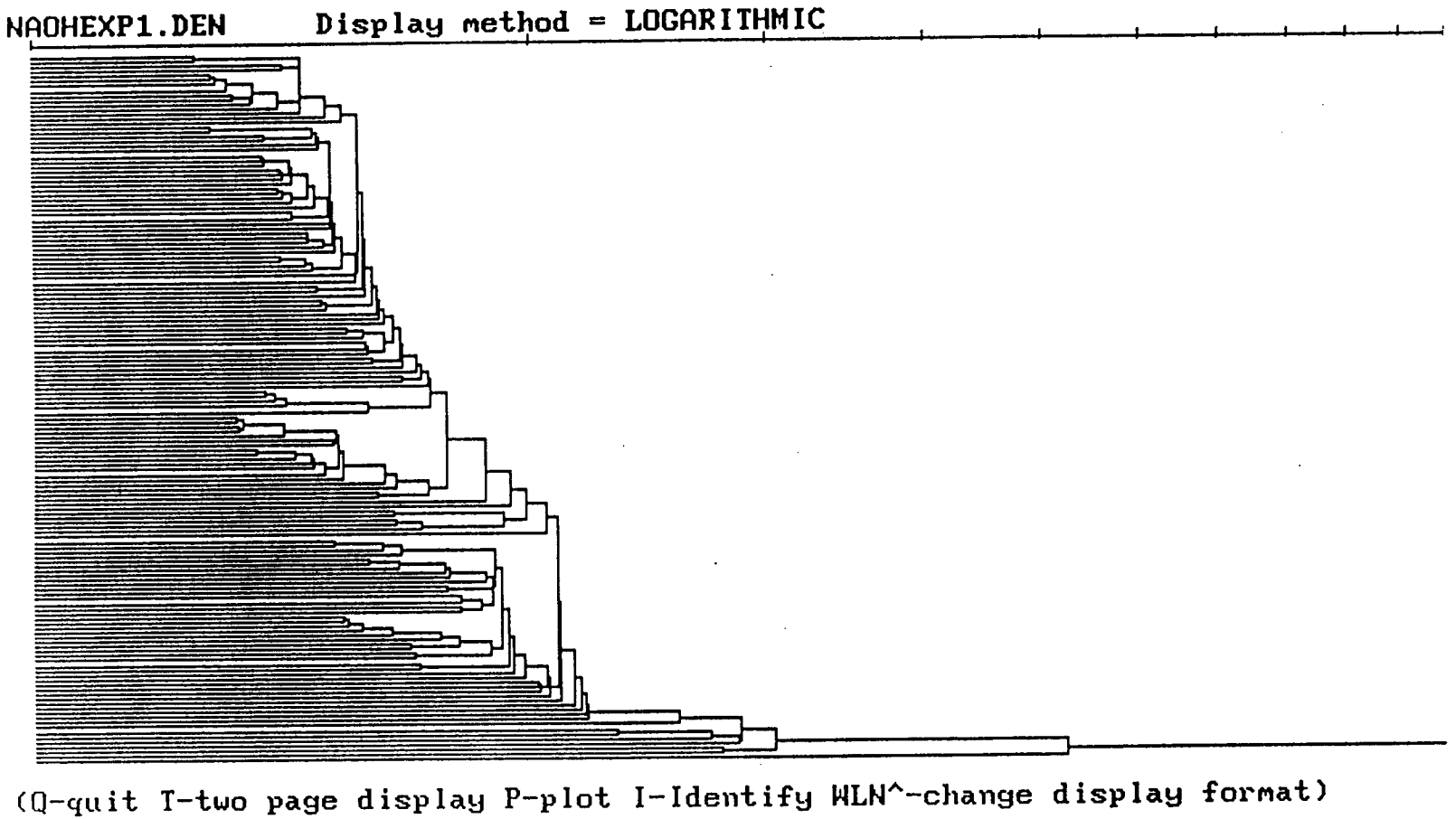


Figure 43) DENDGRAM - Use of exponential similarity axis for display of dendrogram in Figure 40.

method of calculating the dendrogram, and the data scaling option that was performed.

V.8 Factor Analysis - ABSCAT.BAS

While giving an idea of the clustering of the data points, a dendrogram doesn't actually give a picture of the data itself. Abstract factor analysis (AFA)[C], also called principal components analysis (PCA) facilitates projection of a multi-dimensional data set onto two dimensions for viewing while retaining most of the data's structure.

The program ABSCAT.BAS, written by this author[90], will take the data from a descriptor file and allow several viewing and processing options (Figure 44). As with AEMUNCH.BAS, the data can be displayed as X-Y plots. This will show relations between two of the descriptors or give a time profile of one of the descriptors. The program calculates the set of loadings and scores for the data set (Figure 45). Stepwise parameters that are useful in determining whether a factor is a primary or secondary factor are printed into data file with the extension .RES. The AFA results, the factor loadings and scores matrix, are stored in another data file (.AF2). The loadings of the variables on the factors can be viewed to give the relative importance of each feature (Figure 46). From the .AF2 file, the factors can be plotted against each other to view projections of the data set onto any two dimensional plane defined by two of the factors.

One useful feature of ABSCAT.BAS is that signals with different values in their class fields are plotted in different colors. This greatly assists the operator in evaluation of the data. When viewing a plot, it is desirable to be able to identify a particular signal

---

C. Abstract factor analysis produces linear combinations of the descriptors that contain the best distribution of variance for viewing the data but which don't correspond to any actual property of the data - thus the term **abstract** factor analysis. Target factor analysis, by comparison, attempts to project the data onto factors that correspond to real properties (such as a UV-Vis absorption spectrum).

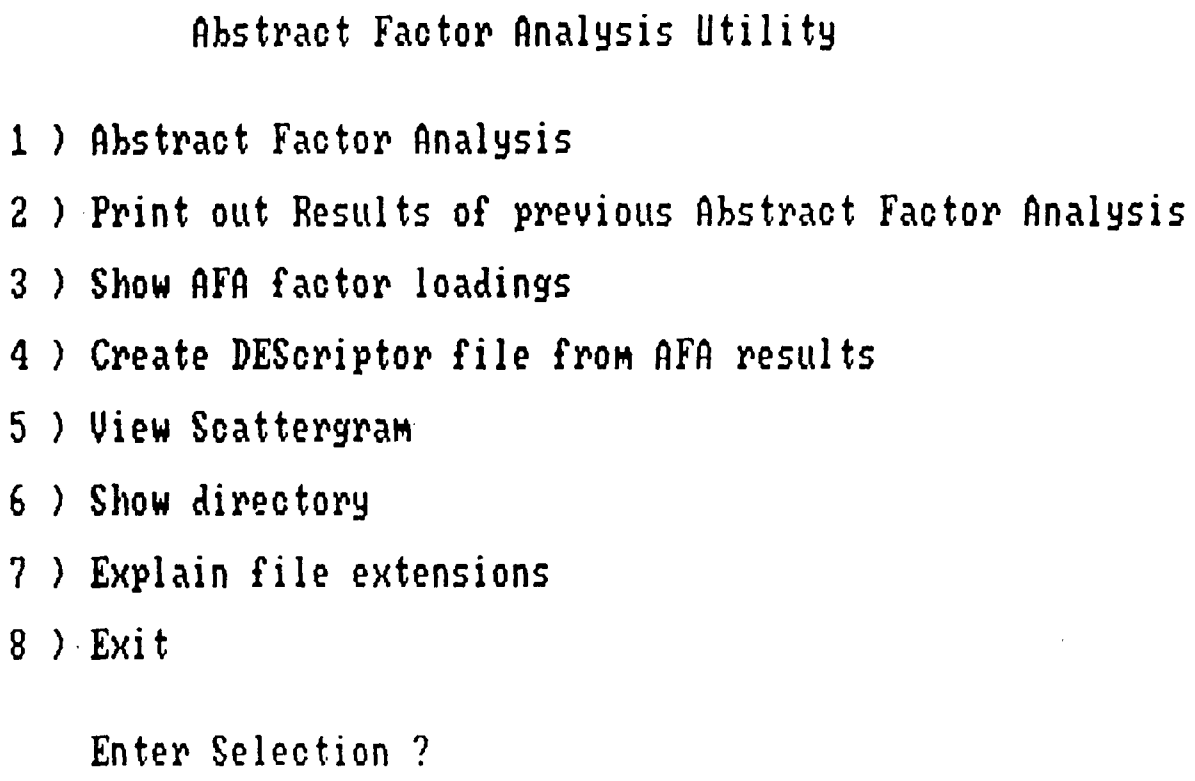


Figure 44) ABSCAT - Program's main option menu.

```
Now Performing Abstract Factor Analysis

 Calculating covariance matrix :  11 / 11
  Now calculating vector #  11 of 11
 Calculating scores matrix. ( 113/ 297 )
```

Figure 45) ABSCAT - Performing factor analysis. The covariance matrix is first calculated and is then used to determine the principal components ("vectors"). The factors are then used as a new basis set to project the original data into the "scores" matrix.

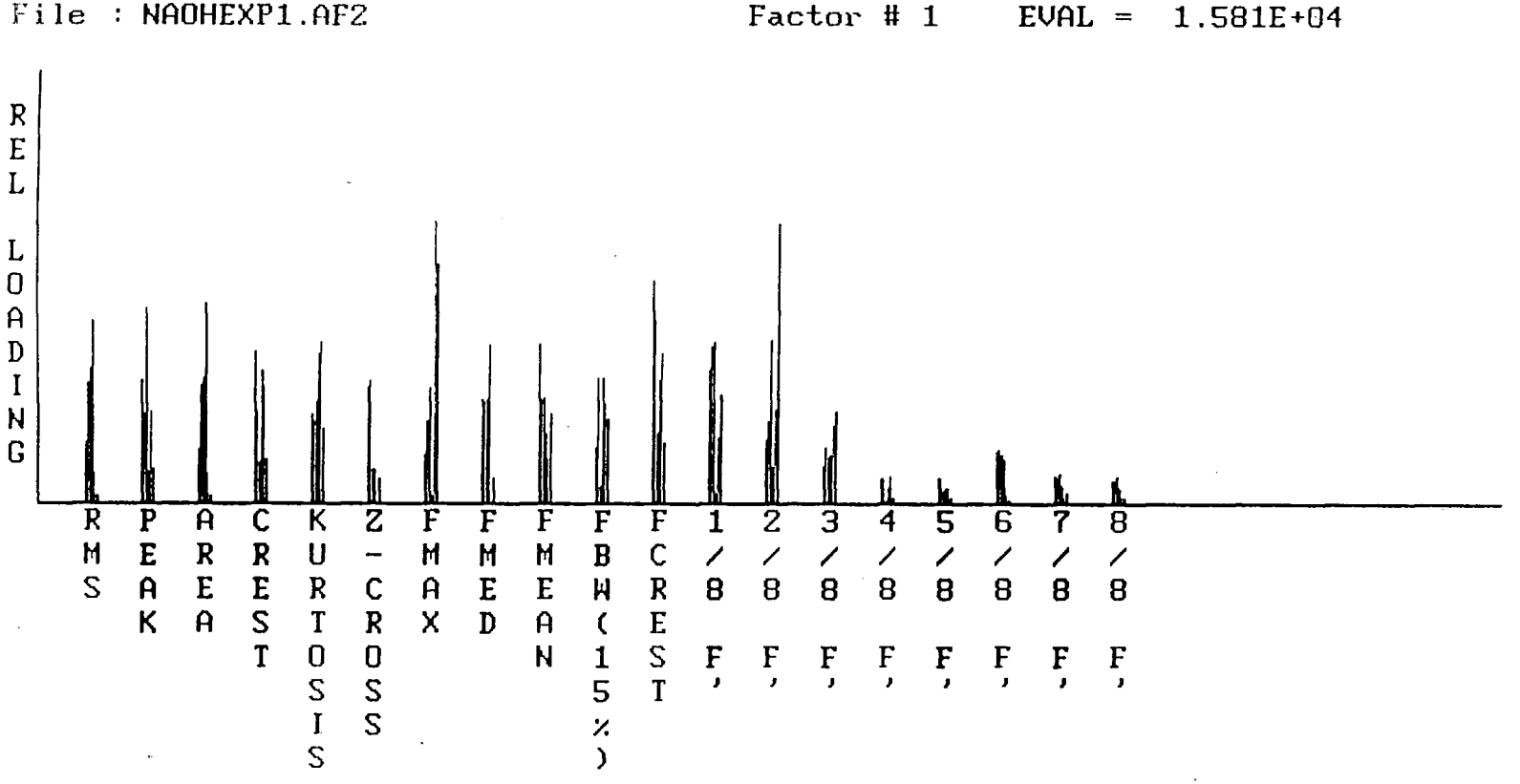


Figure 46) ABSCAT - Factor loading display of first six factors. Each vertical bar corresponds to the loadings of a particular feature (descriptor) on the first six principal components starting with the one listed at the upper right (in this cas factor 1). Each "group" is comprised of six individual vertical lines corresponding to each of the first six factors.

which is represented by a dot on the screen. This is possible within ABSCAT.BAS. A cursor is placed on the screen and can be positioned by the operator (Figure 47). When on or near the point(s) of interest, a bubble is opened up. This can be expanded or contracted to enclose all the data points of interest (Figure 48). Information on all the data points which are within the bubble can be listed including the signal's ID, class, and the values for the two variables / factors being plotted (Figure 49). If a group of data points are selected, the class designation of all such points can be changed. This allows one to view the points as separate colors on the display as well as on different displays using the same data. In this way, the structure of the data that is projected onto one plane for viewing can be compared with the display obtained when it is projected onto another plane. If desired, the new class designations can be stored in the descriptor file for future reference.

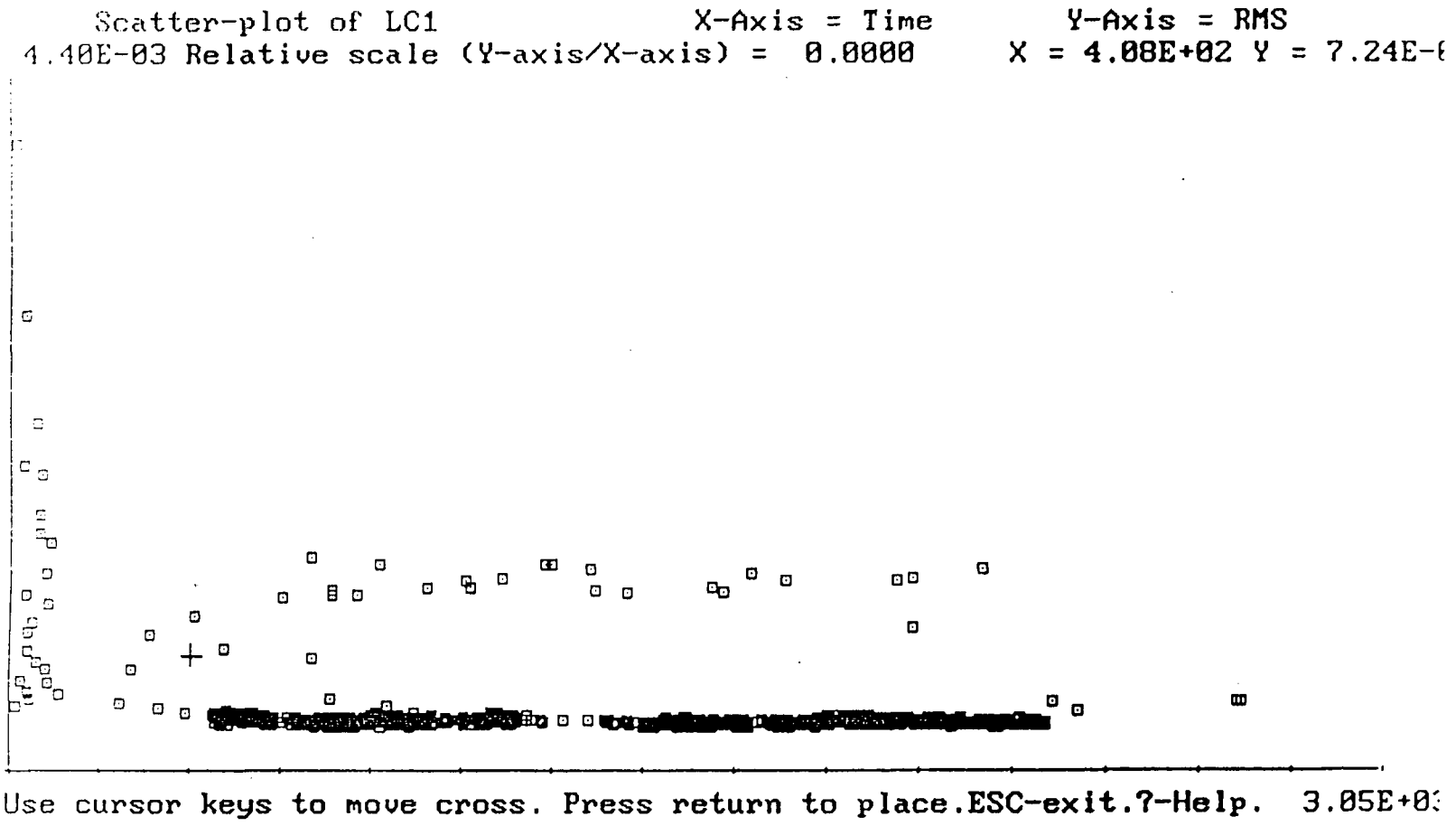


Figure 47) ABSCAT - Scatterplot of RMS Voltage versus time (in seconds) for liquid crystal - 4-n-pentyloxybenzylidene-4'-n-heptyaniline. The "cross-hair" is a cursor which allows the user to help identify the points which correspond to each signal.

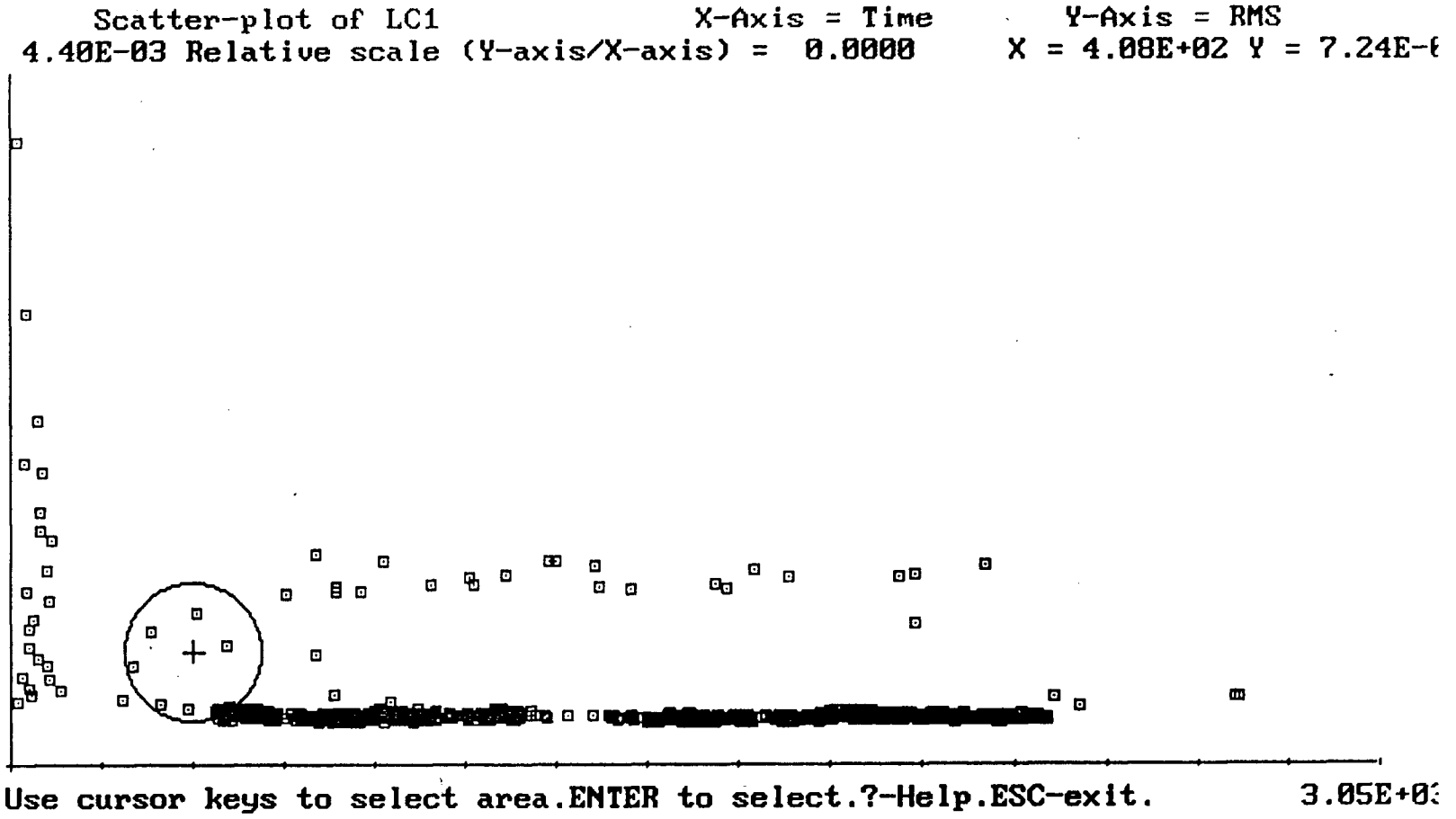


Figure 48) ABSCAT - "Bubble" which opens up around cross-hairs to enclose points of interest for identification.

Identification of points centered about Time = 4.0829E+02 ± 1.531E+02
RMS = 7.2421E-04 ± 4.434E-04

ID	CLASS	X-value	Y-value	Distance from crosshairs	
24	E-N	2.7573E+02	6.3124E-04	ΔX= 1.326E+02	ΔY= 9.297E-05
25	E-N	3.1643E+02	8.6398E-04	ΔX= 9.186E+01	ΔY= 1.398E-04
26	0	3.3933E+02	3.7999E-04	ΔX= 6.896E+01	ΔY= 3.442E-04
27	0	3.9821E+02	3.5991E-04	ΔX= 1.008E+01	ΔY= 3.643E-04
28	E-N	4.1919E+02	9.7760E-04	ΔX= 1.090E+01	ΔY= 2.534E-04
29	0	4.6061E+02	3.4373E-04	ΔX= 5.232E+01	ΔY= 3.805E-04
30	0	4.6160E+02	3.3898E-04	ΔX= 5.331E+01	ΔY= 3.852E-04
31	0	4.6253E+02	3.2259E-04	ΔX= 5.424E+01	ΔY= 4.016E-04
32	0	4.6346E+02	3.1792E-04	ΔX= 5.517E+01	ΔY= 4.063E-04
33	0	4.6445E+02	3.5190E-04	ΔX= 5.616E+01	ΔY= 3.723E-04
34	0	4.6539E+02	3.2676E-04	ΔX= 5.710E+01	ΔY= 3.975E-04
36	0	4.6797E+02	3.3196E-04	ΔX= 5.968E+01	ΔY= 3.922E-04
37	0	4.6890E+02	3.2263E-04	ΔX= 6.061E+01	ΔY= 4.016E-04
38	0	4.7016E+02	3.2737E-04	ΔX= 6.187E+01	ΔY= 3.968E-04
47	E-N	4.8274E+02	7.7276E-04	ΔX= 7.445E+01	ΔY= 4.855E-05
49	0	4.8483E+02	3.5537E-04	ΔX= 7.654E+01	ΔY= 3.688E-04
55	0	4.9148E+02	3.6661E-04	ΔX= 8.319E+01	ΔY= 3.576E-04
58	0	4.9466E+02	3.6011E-04	ΔX= 8.637E+01	ΔY= 3.641E-04

Space bar to see next page, B for previous page, P to see plot,
F to create list file, C to change classification, ESC to exit.

Figure 49) ABSCAT - Identification of signals which have data points within the cursor bubble. In this example, the signals with the class designation "E-N" are due to electrical noise and have higher RMS values than the signals due to the liquid crystal.

VI. Chemometric Methods Used in this Work

VI.1 Descriptors

For analysis, each of the many individual acoustic signals (eg. Figure 6) was described by an array of descriptors. These descriptors evolved over the course of the work and include :

1) Root Mean Square Voltage - RMS

$$RMS = \left[\sum_{i=1}^{n} (s_i^2) / n \right]^{1/2} \qquad \text{(VI.1)}$$

Here, s_i is the voltage of the signal at each time interval t_i and **n** is the number of time intervals ie. the length of the signal in points. This gives a measure of the energy of the signal. More intense signals and those which decay slowly will have higher RMS values.

2) Maximum Peak - MAXPK

$$MAXPK = \max | s_i | \qquad \text{(VI.2)}$$

The value of the signal's highest peak.

3) Area

$$AREA = \Sigma | s_i | \qquad \text{(VI.3)}$$

The summed absolute area under the signal

4) Crest

$$CREST = MAXPK / RMS \qquad \text{(VI.4)}$$

This is a measure of how "spiky" the signal is.

5) Number of zero crossings - **Z-CROSS**

This is the number of times that the signal crosses the zero point.

6) **Kurtosis**

This is the fourth statistical moment and describes the spread of the data about its mean value. A Gaussian distribution will have a kurtosis of zero.

$$\mathbf{KUR} = \quad 1 / n \; \Sigma \; [\; (s_i - \bar{s}) / \sigma \;]^4 - 3 \qquad \text{(VI.5)}$$

$\bar{s}$ is the average value of the signal - normally zero volts and σ is the population standard deviation of all the signal values, s_i, about zero.

$$\sigma = [\Sigma \; (s_i)^2 / \; (n - 1) \;]^{1/2} \qquad \text{(VI.6)}$$

7-14) RMS time octiles

The RMS of each eighth of the time signal is calculated.

The next descriptors[D] are based on the power spectrum (Figure 50) which is calculated for each signal. Each point f_i corresponds to the intensity at frequency interval number **i**. The length of the power spectrum, **len(f)**, is one half the length of the signal from which it is calculated (ie. **len(f) = n / 2**). Fr_i, the frequency at interval **i**, can be found using the sampling rate **TDIV**. Since there are 100 points per division on the oscilloscope, the sampling rate is **100 / TDIV** and thus

$$Fr_i = (1 / 2) \; (100 \; / \; TDIV) \; (i - 1) \; / \; len(f) \qquad \text{(VI.7)}$$

D. These are calculated using the entire frequency spectrum up to the Nyquist frequency. They will therefore be affected by the setting of the acquisition rate of the scope.

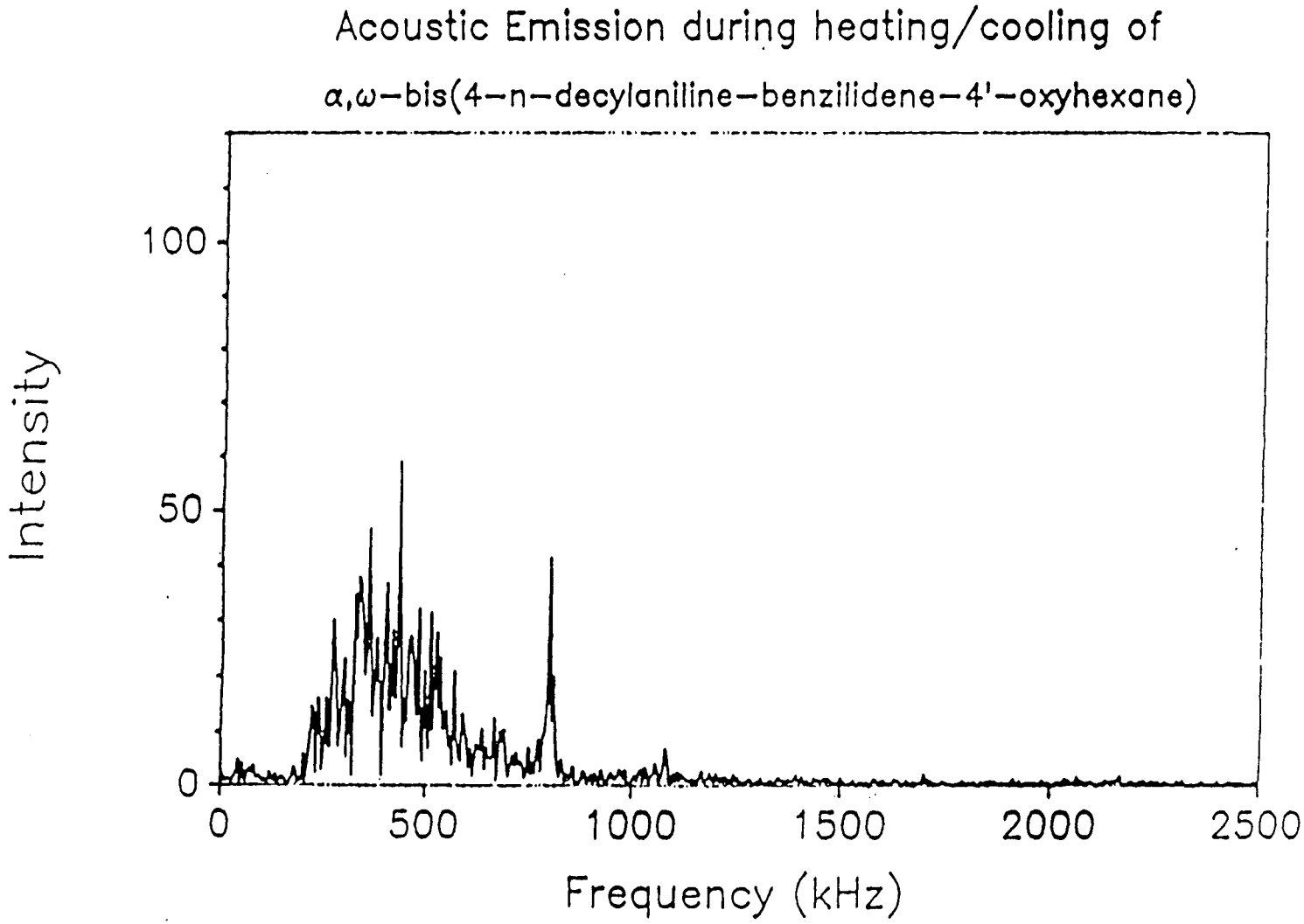


Figure 50) Frequency power spectrum of acoustic signal in Figure 6. Vertical axis measures intensity in arbitrary units.

15) **FMAX** - Frequency with maximum intensity

The frequency component which is most prevalent.

$$\mathbf{FMAX} = \mathbf{Fr_i} \text{ such that } \mathbf{f_i} = \mathbf{max} \mid \mathbf{f} \mid \qquad \textbf{(VI.8)}$$

16) **FMED** - Median frequency

The median point of the entire power spectrum. **FMED** is chosen such that:

$$\mathbf{Fr_j}: \quad \sum_{i=1}^{j} \mathbf{f_i} = \sum_{i=j+1}^{len(f)} \mathbf{f_i} \qquad \textbf{(VI.9)}$$

17) **FMEAN** - Mean frequency

The weighted average of all frequency components.

$$\mathbf{FMEAN} = \Sigma \ (\mathbf{f_i} \cdot \mathbf{Fr_i}) \ / \ \mathbf{len(f)} \qquad \textbf{(VI.9)}$$

18) **FSD** - Frequency standard deviation about the mean

$$\mathbf{FSD} = \sum_{i}^{len(f)} (\mathbf{Fr_i} - \mathbf{FMEAN})^2 \ / \ \mathbf{len(f)} \qquad \textbf{(VI.11)}$$

A signal with one predominant frequency component will have a low **FSD**. Conversely, a signal containing many (or broad) frequency components will have a relatively large **FSD**.

19) **FCREST** - Frequency crest

This is similar to the time domain crest. It is calculated by dividing the value of the largest peak, ie. the intensity at **FMAX**, by the total **RMS** power of the power spectrum.

20) **BW15%** - 15% Band width

The frequency range about the **FMAX** which is above 15% of the intensity at **FMAX.**

21-28) RMS Octiles of the power spectrum

These values are calculated from each eighth of the power spectrum.

Only first order descriptors are used here, ie. the descriptor values computed for each signal are dependent solely on the properties of that signal. Second order descriptors would include inter-signal dependent features such as (eg.) time between consecutive signals.

The work by Belchamber *et al.* used just 5 descriptors (amplitude, variance, half life, median frequency, band width)[30,91]. Further descriptors are presently under evaluation by other workers in this group and will be reported in future publications[87,103].

Through the use of these descriptors, each individual acoustic signal can be represented by a point in a descriptor space which has one dimension for each descriptor. This space is also termed a feature space. Two techniques that transform this feature space into a form that can be viewed in two dimensions are Principle Components Analysis (PCA) and hierarchical clustering. The assumption behind these methods is that similar data points will occupy the same region of descriptor space and will thus appear "near" to one another. Therefore, one can classify pairs of data points as being similar or dissimilar based on the distance between them.

VI.2 Scaling

Before calculating the distance between points based on several descriptors, the units of each descriptor must be considered. Different descriptors will have different units with different ranges and / or magnitudes which will make comparisons difficult unless compensation is made for this. The method of scaling used in pattern recognition depends on the nature of the data. For example, in gas chromatography (GC), data are commonly scaled such that the area under the chromatogram is unity - thus eliminating the effects of sample size on the peak areas.

Before discussing the techniques of scaling implemented in this work it will be helpful to define some terms. The data matrix can be represented by **X** where x_{ij} represents the value of the j^{th} descriptor for the i^{th} signal. Thus:

$$X = \begin{bmatrix} x_{11}, & x_{12}, & \cdots, & x_{1p} \\ x_{21}, & x_{22}, & \cdots, & x_{2p} \\ & & & \\ x_{q1}, & x_{q2}, & \cdots, & x_{qp} \end{bmatrix}$$

where **p** is the number of descriptors and **q** is the number of signals. The scaled data set will be represented by **Y**.

Descriptors with similar units but different base values may be mean centered. For example, if for a set of solutions, the concentrations of two components are measured, the average concentration of one component may be 100 ppm while that of the other component may be 10 ppm. Mean centering is accomplished by replacing the original data set **X** by a new set **Y** calculated as follows:

$$y_{ij} = x_{ij} - x_{Aj} \qquad \text{(VI.12)}$$

and $$x_{Aj} = (1/q) \sum_{i=1}^{q} (x_{ij}) \qquad \text{(VI.13)}$$

ie. x_{Aj} is the mean value for descriptor **j**.

It can easily be demonstrated that mean centering may not be sufficient. In the above example, a change in concentration of 50% in the 10 ppm concentration can be completely obscured by a 20% change in the 100 ppm concentration. This problem can be avoided by the use of range scaling. Range scaling sets the maximum value of the descriptor to one and the minimum to zero.

$$y_{ij} = (x_{ij} - x_{MINj}) / (x_{MAXj} - x_{MINj}) \qquad \text{(VI.14)}$$

Range scaling eliminates most of the problems associated with having descriptors with different units. The resulting descriptor columns all have values between zero and one and there is no dependence on the units of the descriptors. However, if in a set of measurements there is an outlier - a point which lies far from the mean of the measurements - then range scaling will place too much weight on that point and have the affect of making that descriptor meaningless for pattern recognition purposes, as demonstrated in Figure 51.

To account for different units amongst the descriptors as well as allowing for the possibility of outliers, other scaling algorithms have been proposed, and implemented here. Normalization, such as in the gas chromatography case, is often used. The data set is scaled such that the sums of the values for the scaled descriptors for a sample equal a convenient constant, K (normally one).

$$\text{ie.} \quad \sum_j (y_{ij}) = K \qquad \text{(VI.15)}$$

Thus the transformation performed is

$$y_{ij} = K \cdot x_{ij} \; / \; \sum_j (x_{ij}) \qquad \text{(VI.16)}$$

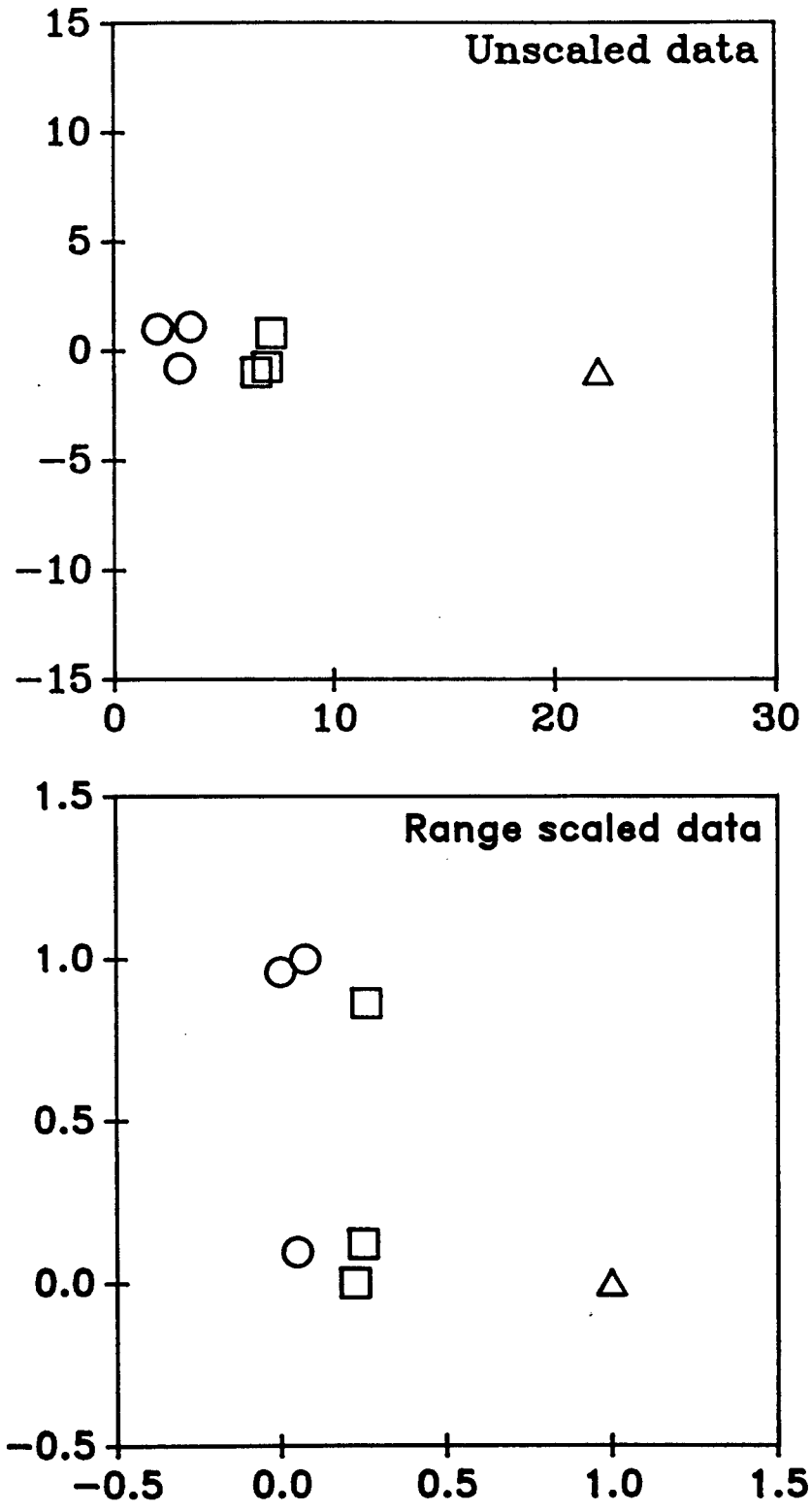


Figure 51) a) Simulated two dimensional data set with outlier in one dimension. b) Result of range scaling data between 0 and 1. Data has been compressed in one dimension, changing apparent structure.

More generally, normalization involves setting the sum of the squares of the descriptors for each data point to a constant.

$$y_{ij} = K \left[x_{ij} \Big/ \left(\sum_{j} x_{ij}^2 \right)^{1/2} \right] \qquad \text{(VI.17)}$$

Normalization is a process where the vector describing the individual data points is scaled independently of any other data vector. It should be taken into consideration that normalizing a data vector (a row in the above matrix) forces the data to lie on a certain hyperplane within the feature space. For example, Figure 52 shows a data set which appears to have two well separated clusters. When the data set is normalized, the data are forced to lie on a circle defined by the normalization equation. This also happens in higher dimensions.

With these considerations, methods which scale by descriptor (scaling by column - such as range scaling) rather than by each data vector (scaling by row - as in normalization) were chosen. One method, auto-scaling, has been previously used for pattern recognition for AE data[30,33]. Auto-scaling is also known as the Z-transform and involves scaling the data matrix **X** such that each of the variable vectors (columns) have zero mean and unit variance. This is accomplished by the following.

$$y_{ij} = (x_{ij} - x_{Aj}) \Big/ \sigma_{i} \qquad \text{(VI.18)}$$

$$\text{where } x_{Aj} = (1/q) \sum_{i} (x_{ij}) \qquad \text{(VI.19)}$$

x_{Aj} is the average value for descriptor **j** and σ_i is the standard deviation of descriptor **j** about the mean.

$$\sigma_i = \left[(1/q) \sum_{i} (x_{ij} - x_{Aj})^2 \right]^{1/2} \qquad \text{(VI.20)}$$

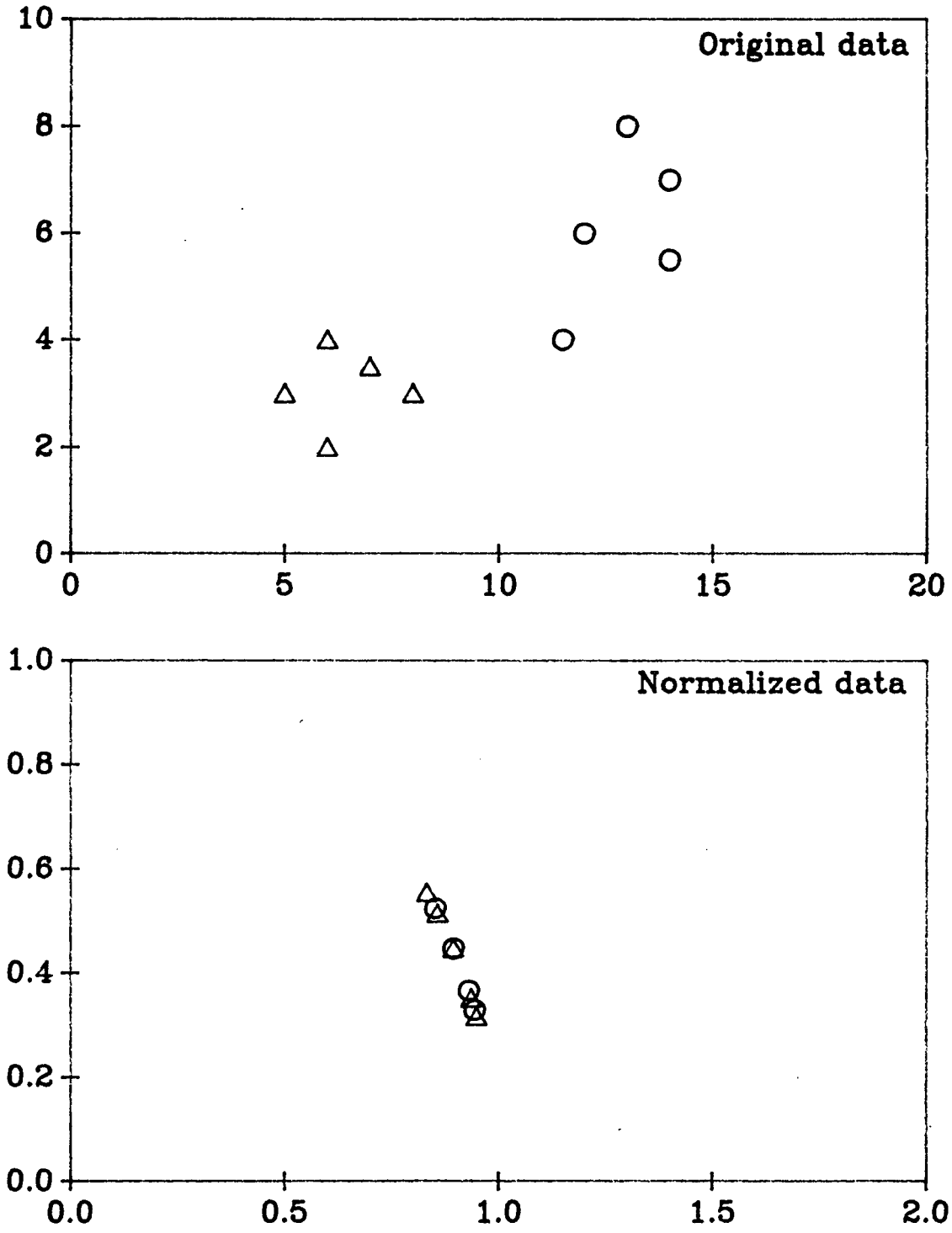


Figure 52) a) A two dimensional data set with two clusters. b) Data set normalized. The grouping of the data has been lost due to imposition of normalization criteria.

By auto-scaling each descriptor to have unit variance, we give equal weight to each descriptor. Thus, the effects of the descriptors' units and absolute magnitudes are eliminated. The problem of outliers skewing the scaled data is also minimal. Even auto-scaling has its drawback, and that is that if a variable is essentially non-variant or has variation due only to random noise, then its importance becomes greatly exaggerated. The minor fluctuations suddenly have as much meaning as a critical descriptor. This has been addressed in this work by grouping similar descriptors such that they have a common scaling, as will be described below.

In our pattern recognition studies, we include the power present in each octile of the frequency spectrum (Figure 53). If the sampling rate is set such that the Nyquist frequency falls well above the maximum response frequency at which the transducer has a significant response, then there will be sections of the frequency spectrum that describe nothing other than noise. In most of the experiments reported here, the Nyquist frequency was 2.5 MHz. With the transducer sensitive only to about 1.2 MHz, this can leave perhaps 3 octiles that contain little analytical information. Of course, one may sometimes observe where very intense emissions which occur above the somewhat arbitrary 1.2 MHz boundary. Scaling of the less variant octiles in such a way as to have as much importance as the more highly variant octiles would clearly be wrong for processes which have all their emission (eg.) below 700 kHz. The method of scaling thus used for this work is a modification conceived during this work. LINK scaling involves auto-scaling the data with the exception of the frequency octiles, which were scaled such that

$$y_{ij} = (x_{ij} - x_{Aj}) / \sigma_F \qquad \text{(VI.21)}$$

where σ_F is the standard deviation of the values in first octile (Figure 54). The first octile normally has the most variance of all the octiles. In this way, the relative importance of the less variant octiles is maintained in relation to the first octile. Of

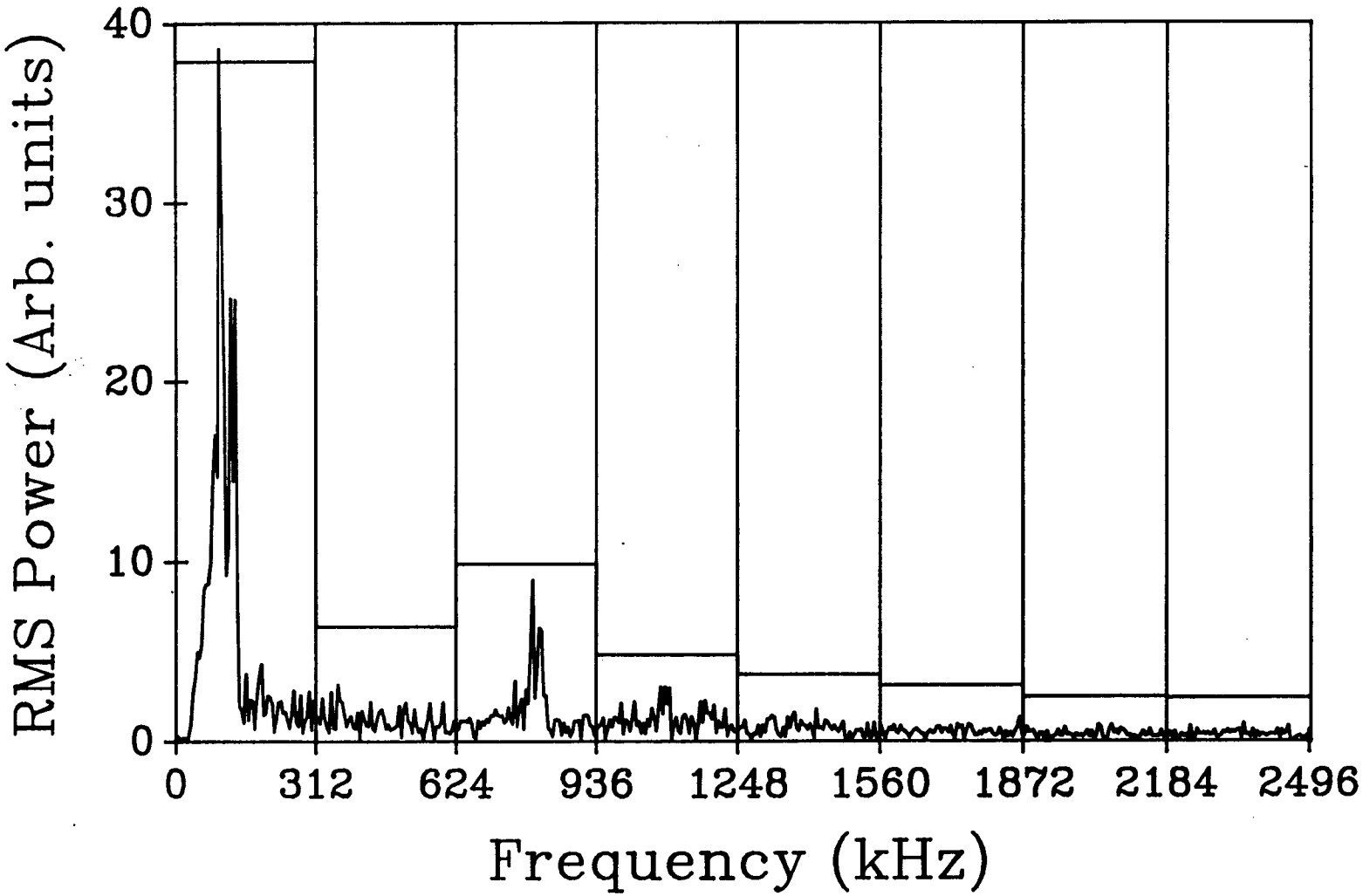


Figure 53) Frequency spectrum of signal from NaOH hydrolosis divided into 8 octiles. Maximum frequency is determined from the sampling rate by the Nyquist theorem. The horizontal bars correspond to the RMS power of each octile. (RMS powers are multiplied by a factor of 10 for plotting).

course, if it is known in advance that any descriptor is not fulfilling a useful purpose but only describing noise or remaining largely constant, then it should be removed from the analysis.

VI.3 Similarity and Distance

With the feature space now defined by unitless descriptors, a measure of the signals' similarities can be calculated. The distance between the position of pairs of data points in the feature space is taken as the measure of dissimilarity. As with scaling, there are different techniques for calculating the distance between points in the multi-dimensional feature space.

The most commonly used measure is the Euclidean distance. This is simply an extension of Pythagorean principle to more than two dimensions.

$$d_{ij} = [\sum_k (y_{ik} - y_{jk})^2]^{1/2} \qquad \text{(VI.22)}$$

where d_{ij} is the distance between the data points representing signals **i** and **j**. The Manhattan or city block distance is calculated as the sum of the vertical and horizontal distances along each axis (descriptor).

$$d_{ij} = \sum_k | y_{ik} - y_{jk} | \qquad \text{(VI.23)}$$

The difference between this measure and the Euclidean distance is shown in Figure 55. These two metrics are actually special cases of the Minkowski formula[66].

$$d_{ij} = [\sum_k | y_{ik} - y_{jk} |^r]^{1/r} \qquad \text{(VI.24)}$$

One can see that when **r** equals two, this equals the Euclidean distance and when **r** equals one, the Manhattan distance. Other values of **r** are possible, but these are not common.

SCALING : TMEC\TME1.DS1

Variable	Variance	Average	Maximum	Minimum	Mode
1	8.700E-04	6.3122E-02	1.8886E-01	2.4011E-02	Auto scale
2	1.162E-02	2.5542E-01	8.5010E-01	1.2730E-01	Auto scale
3	5.546E+02	4.9588E+01	1.5922E+02	1.8122E+01	Auto scale
4	3.322E+00	4.4548E+00	1.2145E+01	2.4294E+00	Auto scale
5	3.446E+01	5.5428E+00	4.8528E+01	2.2335E+00	Auto scale
6	5.521E+04	7.0448E+02	1.2549E+03	3.9063E+01	Auto scale
7	2.446E+04	7.3374E+02	1.2402E+03	3.3691E+02	Auto scale
8	1.150E+04	8.4362E+02	1.2319E+03	5.1641E+02	Auto scale
9	1.297E+03	5.6600E+02	6.5407E+02	4.4776E+02	Auto scale
10	1.208E+01	9.8468E+00	1.7801E+01	3.2785E+00	Auto scale
11	2.378E-01	9.5110E-01	2.5005E+00	2.3165E-01	Auto scale
12	3.443E-01	1.1818E+00	2.7127E+00	2.3230E-01	Link scale
13	3.219E-01	1.9347E+00	2.7559E+00	6.0903E-01	Link scale
14	1.592E-01	6.9538E-01	2.0488E+00	2.1303E-01	Link scale
15	6.366E-02	3.8914E-01	1.8186E+00	1.2322E-01	Link scale
16	1.243E-02	3.2112E-01	8.2313E-01	8.5206E-02	Link scale
17	5.970E-03	1.9480E-01	5.2122E-01	7.4046E-02	Link scale
18	3.154E-03	1.8741E-01	5.0983E-01	7.1246E-02	Link scale

3/8 F'
Column 13 , Mode = Auto Scale mean to : 0.0000,Use variable 12 variance
Press L to load in saved Format file, P - Print, ESC - Execute

Figure 54) DENDGRAM - Link scaling of frequency RMS octiles avoids amplification of noise in eight octiles (variables 12-19) if the higher variance of the first octile (variable 12) is used.

The interpoint distances are then calculated. This results in a similarity matrix **D** containing the distances between all pairs of data points.

$$\mathbf{D} = \begin{bmatrix} d_{11}, & d_{12}, & \ldots, & d_{1q} \\ d_{21}, & d_{22}, & \ldots, & d_{2q} \\ . & & & \\ . & & & \\ d_{q1}, & d_{q2}, & \ldots, & d_{qq} \end{bmatrix}$$

d_{ij} is the distance between points **i** and **j**. This matrix is symmetrical and the diagonal elements, d_{jj}, are zero.

VI.4 Hierarchical Cluster Analysis

The procedure for hierarchical cluster analysis is outlined below. The two most similar points, (ie. those with the shortest distance between them) are fused together into one cluster. A new distance matrix is then calculated. The procedure is repeated until only one cluster is present. Different methods for calculating a dendrogram exist and these differ mainly in the method of calculating a new distance matrix at each step[58]. The methods available in the DENDGRAM program written by D. B. Sibbald[87 89] can be divided into two groups: those that rely solely on the distance matrix and those that refer to the actual descriptor values to calculate the new distance matrix.

Table 2 contains a sample two dimensional data set. From this we can calculate the distance matrix in Table 3 (using the Euclidean distance).

Table 2. Sample two dimensional data set.

	x	y
1)	0.0	3.6
2)	1.07	1.45
3)	0.0	0.0
4)	6.68	4.02
5)	3.0	2.8

Table 3. Distance matrix for data in Table 2.

	1	2	3	4	5
1)	0				
2)	2.4,	0			
3)	3.6,	1.8,	0		
4)	6.7,	6.2,	7.8,	0	
5)	3.1,	2.3,	4.1,	3.8,	0

Using the distance matrix, the first fusion made is between points 2 and 3 (at a distance of 1.8). This leaves the new distance matrix in Table 4.

Table 4. Distance matrix after first fusion.

	1	2-3	4	5
1)	0			
23)	[ ],	0		
4)	6.7,	[ ],	0	
5)	3.1,	[ ],	3.8,	0

The spaces marked [ ] denote the distances between the remaining points and the newly formed cluster. These will be calculated and filled in according to the dendrogram method being used.

Single linkage takes the distance between a point and a cluster as the shortest distance between the point and any of the points in the cluster. The new distance matrix using the single linkage method is shown in Table 5.

Table 5. Distance matrix using Single Linkage.

	1	2-3	4	5
1)	0			
23)	2.4,	0		
4)	6.7,	6.2,	0	
5)	3.1,	2.3,	3.8,	0

Complete linkage on the other hand, takes the distance as the largest distance between the point and any of the points in the cluster, and gives the new distance matrix shown in Table 6. A graphical representation of the difference between single linkage and complete linkage is shown in Figure 56.

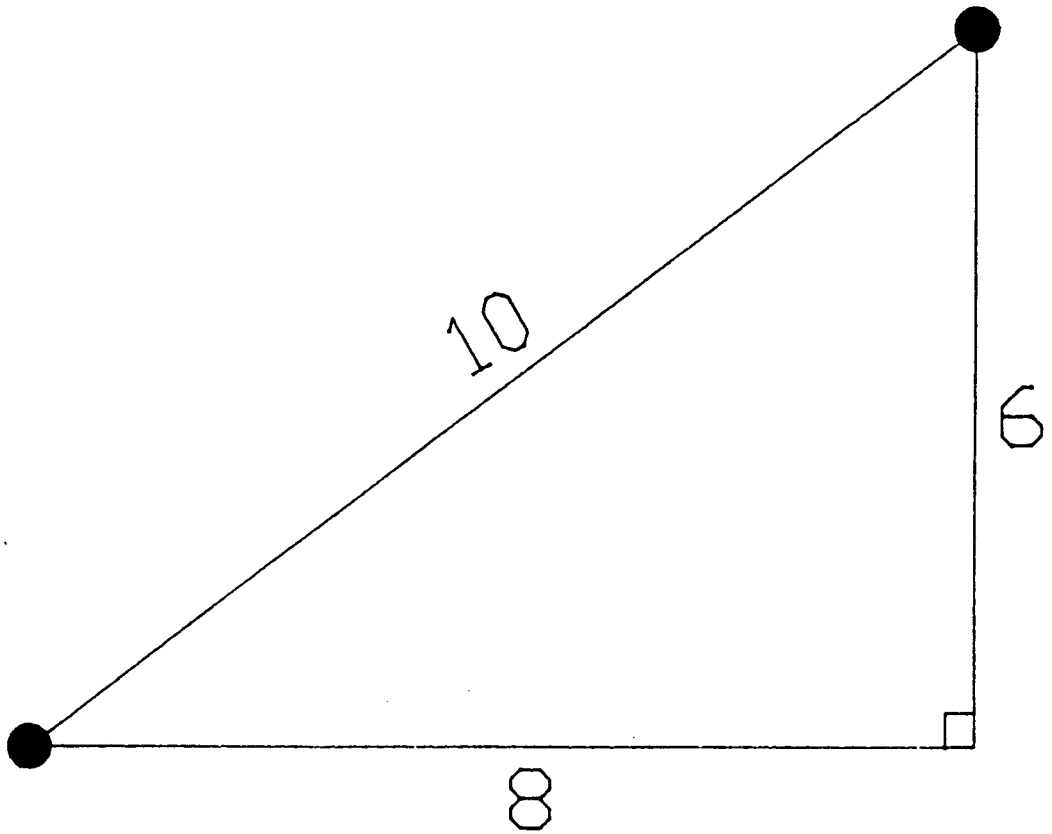


Figure 55) Difference between Euclid vs Manhattan distances for two dimensional example. Euclidean distance between points is 10 while the Manhattan "city block" distance is 14.

Table 6. Distance matrix using Complete Linkage.

	1	2-3	4	5
1)	0			
23)	3.6,	0		
4)	6.7,	7.8,	0	
5)	3.1,	4.1,	3.8,	0

Average linkage, as the name suggests, takes the distance between the remaining points and the new cluster as the average distance between the point and the points in the cluster. The new distance matrix for average linkage is given in Table 7. To continue with the algorithm, the shortest distance is taken as the next fusion. One notices that in each method's algorithm, the second fusion is between two different pairs of objects!

Table 7. Distance matrix using Average Linkage.

	1	2-3	4	5
1)	0			
23)	3.0,	0		
4)	6.7,	7.0,	0	
5)	3.1,	3.4,	3.8,	0

There are two different ways of applying the average linkage algorithm. To outline the difference the next step will be examined. When point #1 is joined with cluster 2-3, the matrix is reduced further and the distances from each of points #4 and 5 to the new cluster 1-2-3 need to be calculated. The weighted average linkage method takes this distance as the average of the distances between the object in question and the distances to the previous two components of the new cluster. For the case of point #4 this distance is

$$d_{123\text{-}4} = (d_{1\text{-}4} + d_{23\text{-}4}) / 2 = (6.7 + 7.0) / 2 = 6.85. \qquad \text{(VI.25)}$$

The unweighted average linkage method however, calculates this distance as the average distance between the object in question and all points in the new cluster. This is calculated for point # 4 as

$$d_{123\text{-}4} = (d_{1\text{-}4}) / 3 + (d_{2\text{-}4})/3 + (d_{3\text{-}4}) / 3 \qquad \text{(VI.26)}$$
$$= (d_{1\text{-}4}) / 3 + (d_{23\text{-}4}) \times 2 / 3$$
$$= (6.7)/3 + (7.0)\ 2 / 3 = 6.9$$

The methods involving direct use of the descriptor values also progress in the same way. Two points / clusters are fused to form a new object and a new distance matrix is calculated. The difference is that for each calculation of a new distance matrix element, these algorithms rely on the original (scaled) descriptor values (matrix Y) rather than manipulating the distance matrix D (as the methods discussed above). Therefore, these algorithms are more complicated and require more computing time.

In the centroid methods, when a cluster is formed, the position in feature space representing the center of the cluster - the centroid - is calculated. It is the centroid to which the distances are calculated for the elements of the new distance matrix. Like the average linkage method, the centroid method can be performed weighted or unweighted. The unweighted centroid method (or Grower's method) uses the mass center of the new cluster as the centroid. The weighted centroid method uses the mid-point of the two previous centroids / positions as the centroid. Figure 57 illustrates this difference.

Ward's method differs markedly from the other methods. Instead of calculating the distances between points and clusters, the amount of information lost with each possible fusion is calculated. At each step, each possible fusion is considered. The new

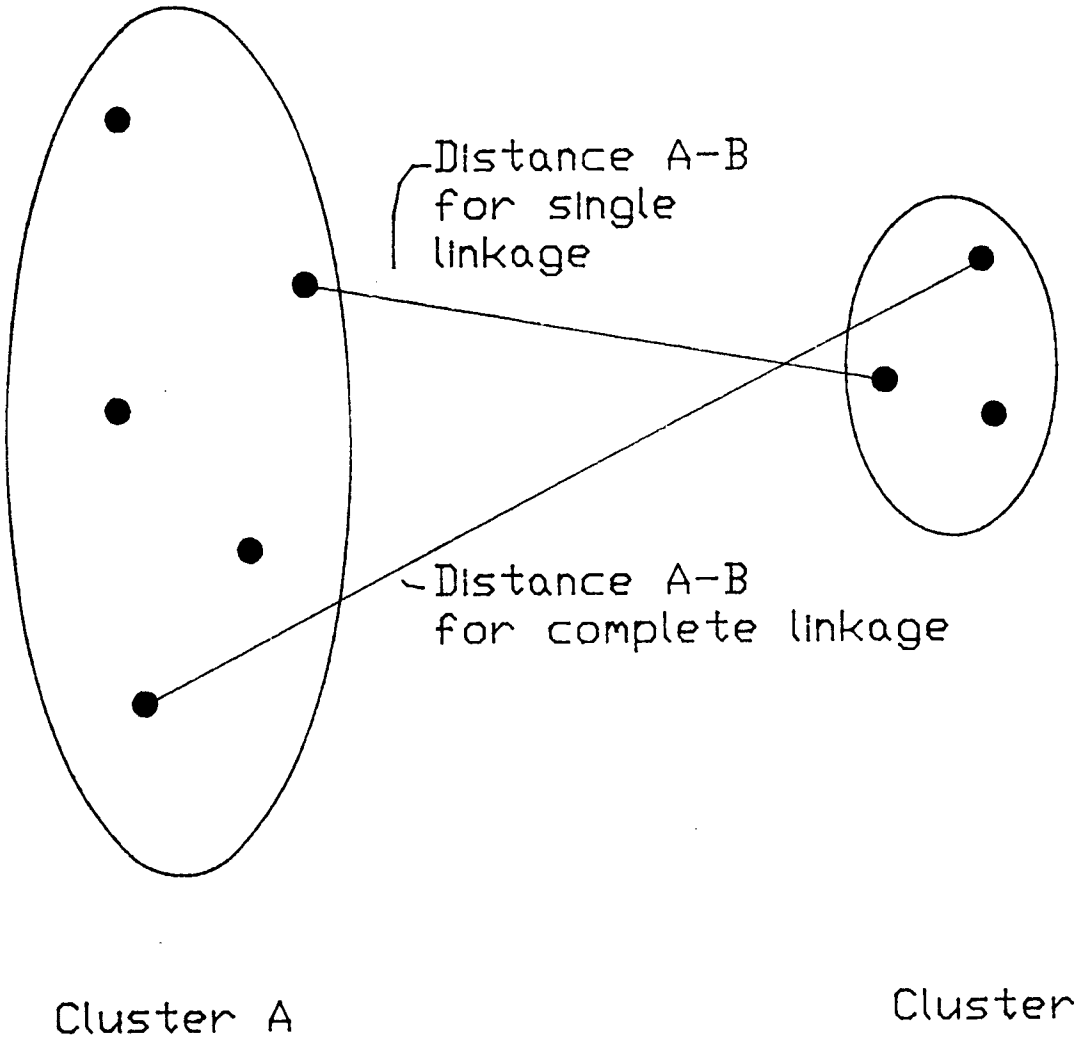


Figure 56) Graphic representation of difference between single linkage and complete linkage for distance between two clusters. Single linkage uses the shortest distance between any two points in the cluster; complete linkage uses the longest.

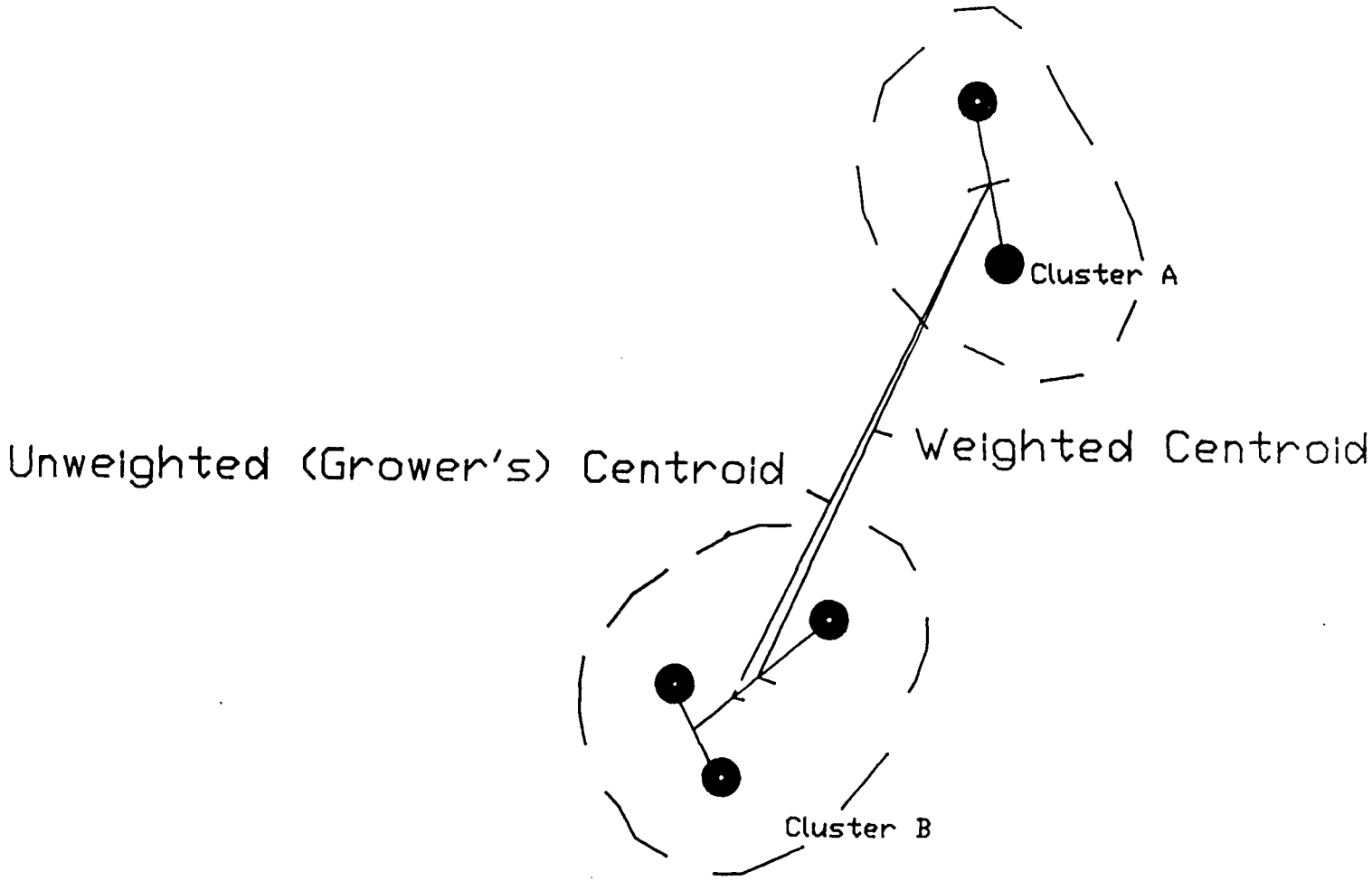


Figure 57) **Graphic representation of difference in centroid location for weighted and unweighted centroid methods. The location of the centroid of the new cluster formed will lie half way between the centroids of clusters A and B using the Weighted Centroid Method. The Unweighted Centroid Method will use the "center of mass" as the centroid of the new cluster.**

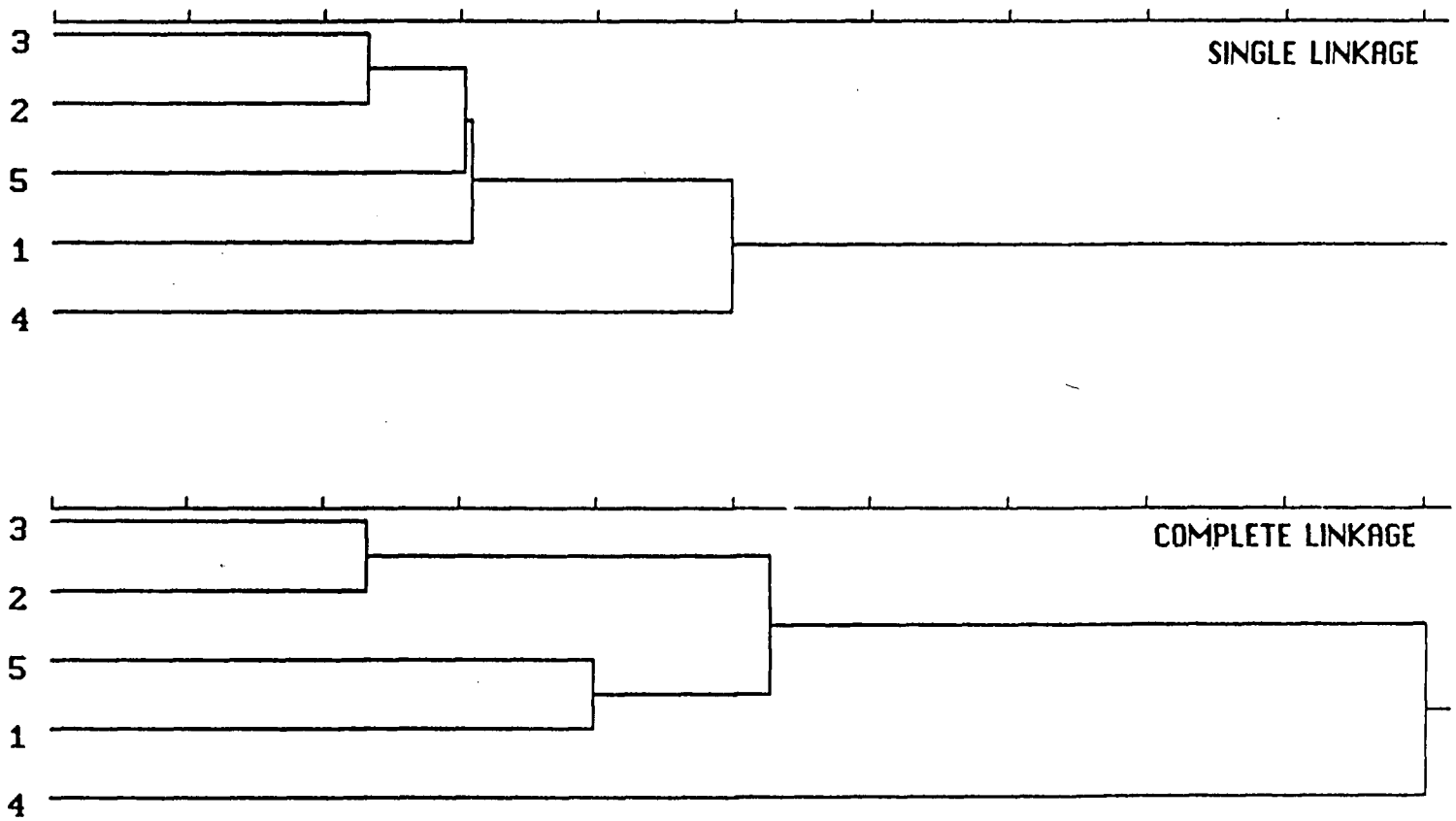


Figure 58) **Dendrograms calculated from data in Table 2 using single linkage and complete linkage.**

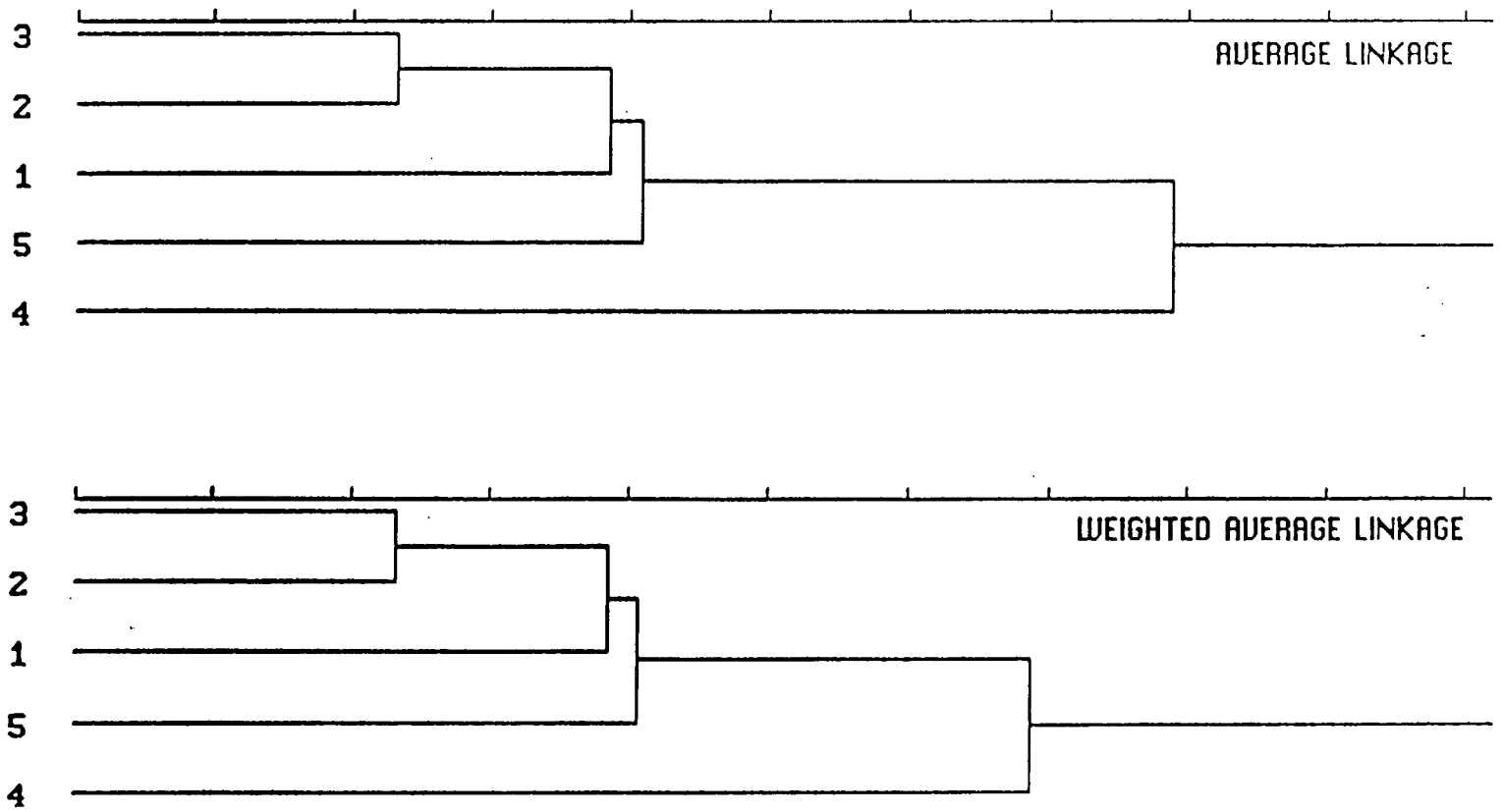


Figure 59) Dendrograms calculated for data in Table 2 using unweighted average linkage and weighted average linkage.

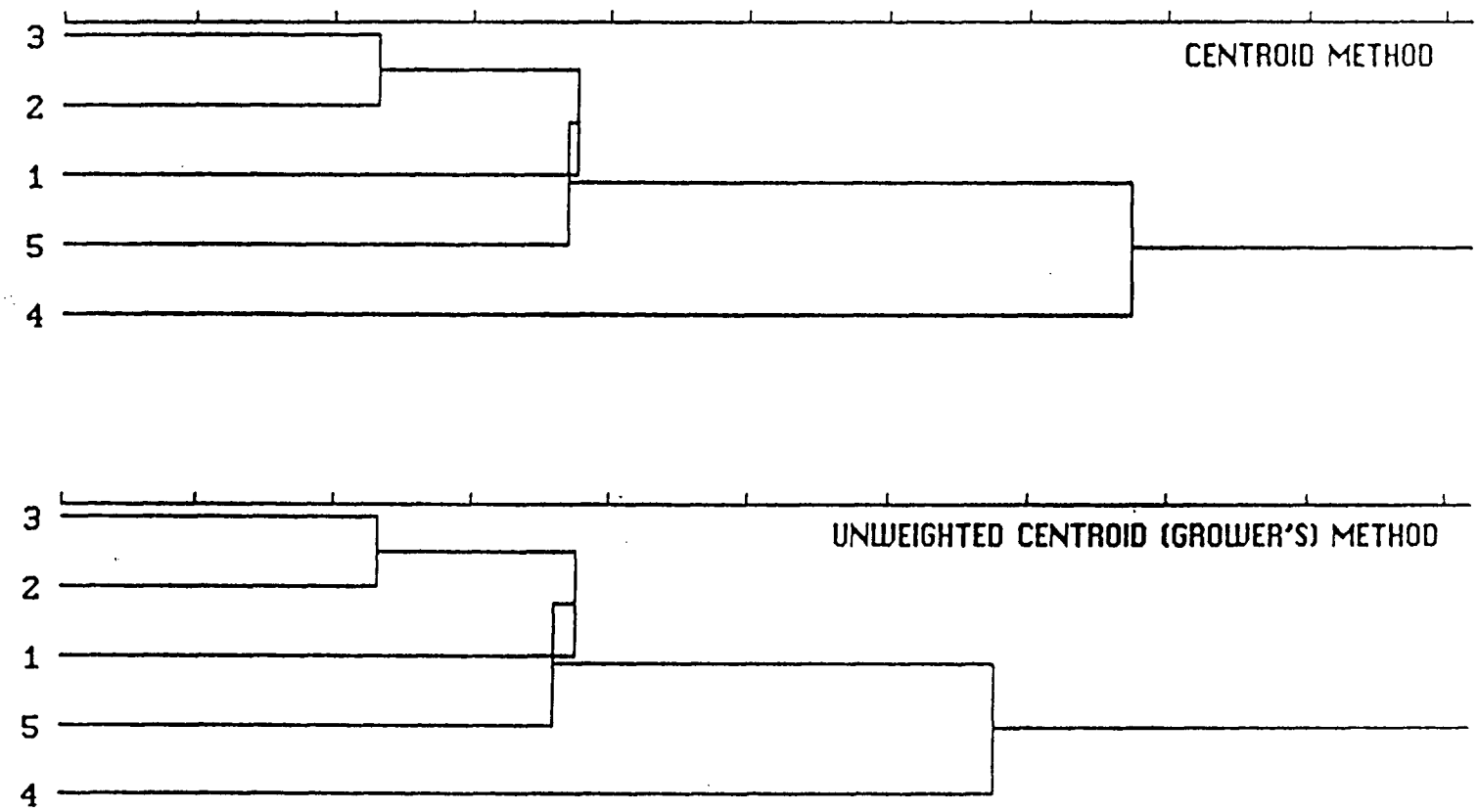


Figure 60) **Dendrograms calculated from data in Table 2 using the centroid method and the unweighted centroid method (Grower's method).**

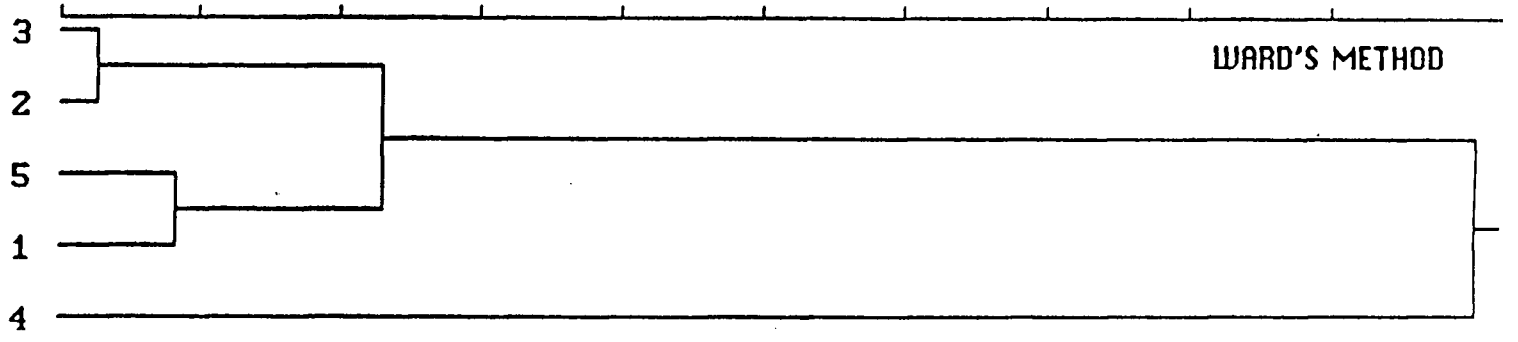


Figure 61) Dendrogram from data in Table 2 calculated using Ward's method.

mass-centered centroid is calculated for each hypothetical fusion and the Error in the Sum of Squares (ESS) is calculated.

The sum of squares for a data point is the error produced when describing the point as part of a centroid. If the coordinates of a centroid are c_j, (with the data point in question being y_i), the ESS for that point is

$$ESS_i = \sum_j (y_{ij} - c_j)^2 \qquad \text{(VI.27)}$$

The ESS_i is calculated for each point and then totaled over all points to give the ESS for each possible fusion.

$$ESS = \sum_i ESS_i \qquad \text{(VI.28)}$$

The fusion made is the one with the least ESS value - ie. the least total sums of the squares of the differences between the data point and the centroid in each descriptor.

The dendrograms for each method using the distance matrix are shown in Figures 58 - 61. It is visible that each method produces a slightly different dendrogram from the same data.

VI.5 Abstract factor analysis (AFA)

While the dendrogram gives a clustering of the objects, AFA can be used to obtain a two dimensional projection of the data such that the information loss is minimal and the structure of the data set is maintained.

The scaled data matrix represents the feature-space position of the data points. AFA transforms the data matrix such that a new set of basis vectors is chosen. These

basis vectors define the factor space. The new basis vectors are called factors or principle components. The first factor is chosen as the axis along which a maximum of the data set's variance is contained. The feature space can then be reduced in dimensionality essentially by subtracting out the variance accounted for by this factor. A second factor (orthogonal to the first) is similarly chosen to contain a maximum of variance of the reduced feature space. This process is continued until the feature space is entirely mapped into the factor space.

The program ABSCAT, written by D. B. Sibbald[90] performs abstract factor analysis on the acoustic emission descriptor data. The same scaling method is used to remove any dependence on the units of the descriptors before applying the routine. The first step is the calculation of the covariance matrix *C*.

$$c_{ij} = \sum_{k}^{p} (y_{ki} \cdot y_{kj}) \qquad \textbf{(VI.27)}$$

This will be a square matrix with the same number of dimensions as there are descriptors - **p**.

A normalized approximation of the first new basis vector to be extracted is made, L_1, by setting the l_i equal to the square root of *p*.

$$\mathbf{L_1} = (l_1, l_2, ..., l_p)$$

A new approximation is obtained by multiplying by the covariance matrix.

$$L_{1(NEW)} = L_{1(OLD)} \cdot C \qquad \textbf{(VI.29)}$$

The eigenvalue $\mathbf{E_1}$ associated with the new vector, $L_{1(NEW)}$, is calculated and the vector is normalized.

$$\mathbf{E_1} = [\Sigma\ (l_i)^2\ ]^{1/2} \qquad \textbf{(VI.30)}$$

$$l_i = l_i\ /\ \mathbf{E_1} \qquad \textbf{(VI.31)}$$

This procedure is repeated until consistency is achieved. This is tested for by checking the value of E_{NEW} against the previous value, E_{OLD}. If there is less than $1/1000^{th}$ of a percent change then the vector L_N is accepted. The residual covariance matrix C_R is then calculated.

$$c_{Rij} = c_{ij} - [E_N \cdot l_i \cdot l_j] \qquad \text{(VI.32)}$$

The second factor is extracted the same way. The same initial approximation is made for L_2 and the sequence of obtaining successive estimates is performed by multiplication with C_R. Eventually, a complete set of factors is extracted and these form the set of basis vectors for factor space: **F** - the loadings matrix.

$$\mathbf{F} = \begin{pmatrix} L_1 \\ L_2 \\ L_3 \\ \cdot \\ L_p \end{pmatrix}$$

The scaled original data set is then projected from feature space onto factor space to produce the scores matrix represented by **Z**. This is easily accomplished by multiplying the scaled data matrix **Y** by **F'**, the transpose of **F**.

$$\mathbf{Z} = \mathbf{Y}\,\mathbf{F}' \qquad \text{(VI.33)}$$

The scores matrix, **Z**, is a representation of the data matrix **Y**. The rows correspond to the coordinates of the data point in the factor space in the same way that the rows in the data matrix corresponded to each individual signal's descriptors.

A two dimensional plot can now be made of the data in matrix **Y** which contains as much of the structural information possible. Plotting the scores of the first two factors against each other is achieved by plotting the first two columns of **Z**. This is a hyperplane in the multi-dimensional feature space that contains the most variance in the data. Hopefully, any structure of the data in feature space that was hidden by multi-dimensional nature will be visible in factor space.

VII. Results and Discussion

VII.1 Detailed Characterization of the Hardware and Software Developed

VII.1.1 Effect of Transducer on Observed Signals

The piezoelectric transducer served the function of converting the ultrasonic signal into an electrical voltage. An ideal transducer[8] would allow for an exact transduction of the vibration into a representative electrical signal and would have the following characteristics: a) good sensitivity, b) broad band response, c) flat response over the range of sensitivity, and d) low cost. The sensitivity to a broad range of frequencies allows the observation of signals with different frequency components and adequate quantization and characterization of their frequencies with a single transducer. That the response be constant to all frequencies is also important. For calculations of signal energy (root mean square voltage), signals of equal power but different frequency should not give vastly different apparent energies.

The frequency response curve of the Bruel and Kjaer Model 8312 transducer used (#1381596) is given in Figure 62. This is the factory calibration and is given only to 1 MHz with a resolution of 50 kHz. As is shown, other transducers, even those by the same manufacturer with the same model number, have different responses[41]. The sensitivity to frequencies above 1 MHz is not reported but since some signals have been seen to contain components as high as 1.7 MHz, this suggests that the transducer still has a limited response above its reported operating range. It is unknown whether the frequency response is limited by the piezoelectric material or by the built-in pre-amplifier. One must hope that the ongoing development in transducer design is able to produce more ideal transducers so that research groups may directly compare their work with that of others. This topic is discussed in the literature[41,92]. Indeed, new transducers by Acoustic Emission Technology Corporation (Sacramento, California)[93] may suit this purpose and standards are emerging[94,95].

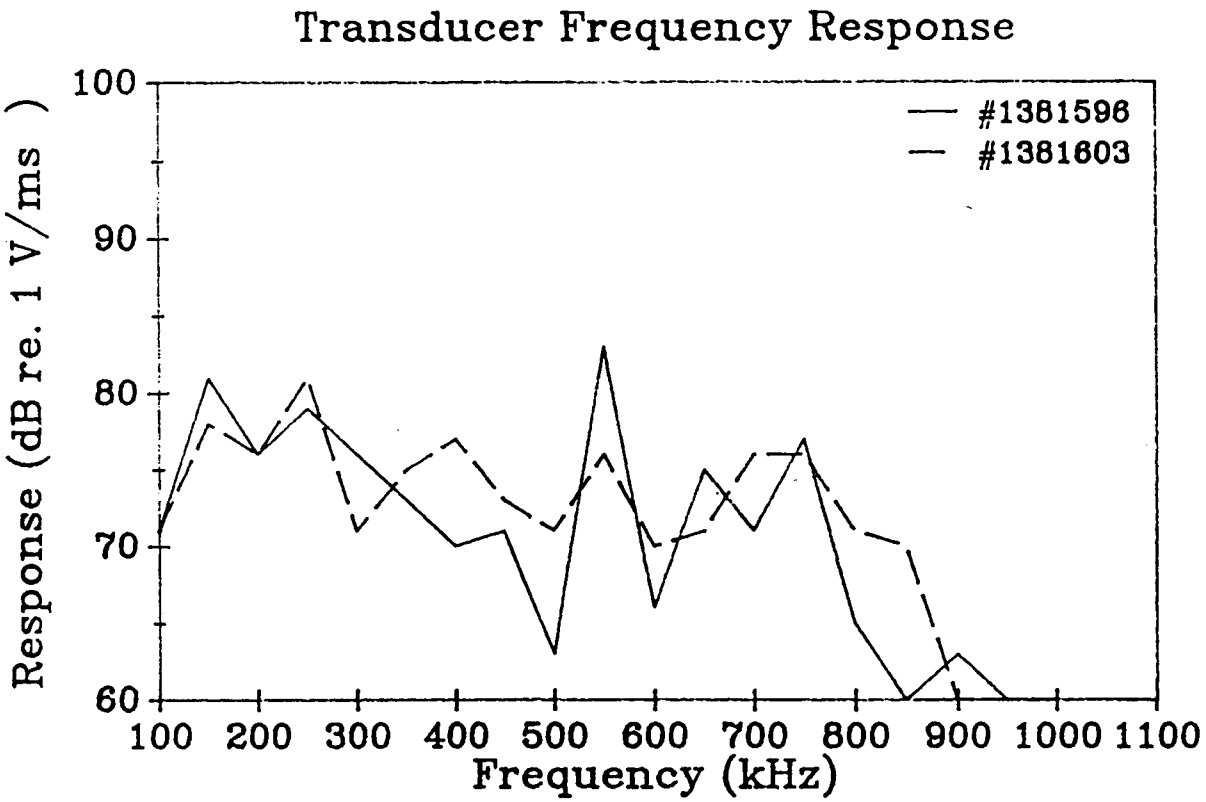


Figure 62) Frequency response of two different Bruel & Kjaer model 8312 transducers.

VII.1.2 Effect of Ambient Noise

The Bruel and Kjaer transducer was found to be sufficiently sensitive to most of the systems investigated. Use of the 50 kHz - 2 MHz filter was seen to successfully render the system immune to ambient room noise in tests with speech, clapping, and "noise" from a nearby cassette-radio.

Comparison of AE data obtained using different transducers of this type is not directly possible without very high resolution (12-bit minimum) data acquisition. Different techniques have been suggested for characterization of the frequency response of transducers[96]. It has been suggested to measure the sensitivity as a function of frequency and correct the resulting signal to yield the "true" signal[97-99] but this requires a large amount of computation for each signal. This was not acceptable for this work in part because of computational time, and in part because of the 8-bit resolution limitation of this apparatus. This is why the same transducer was used for all experiments.

VII.1.3 Effect of Signal Acquisition Rate

The ability of the scope to digitize the incoming signals and download the data to the computer was also satisfactory for pattern recognition (ie. qualitative) purposes. The model T2230 was able to transmit a 1024 byte signal every 1.5 seconds. This meant that faster emitting systems released far more acoustic signals than could be captured. However, the sampling of these was thought to be quite adequate for pattern recognition and classification purposes. Quantification of total acoustic energy was not possible using this system, as signals occurring during data transmission were missed and thus could not contribute to the energy sum. Use of the model T2430A oscilloscope, which was able to capture between ten and twenty signals a second, still

was subject to the same limitations, although using this scope, data sets of up to 7500 signals have been captured.

VII.1.4 Effect of Trigger Level

An unfortunate feature of the T2230 was that the trigger level could not be controlled by computer, and had to be adjusted manually. The knob is not marked with any degradations nor has it any detents. Moreover, the computer interface to the scope was incapable of monitoring the trigger level. Thus, maintaining consistent trigger levels between experiments for future comparison was difficult. Marks on the dial and manual measurement of the trigger mark on the scope display were used to be consistent. Experiments with too low a trigger level will have baseline noise as a large proportion of signals while those with too high a trigger level will selectively discriminate against low intensity acoustic signals (ie. they will be absent from the final data file). Fortunately, the more capable T2430A oscilloscope purchased later in this work not only has a detented knob for setting the trigger level, but the value of the trigger level is displayed (in mV) on the screen and can be set and accessed via the computer-oscilloscope interface.

VII.1.5 Effect of Waveguide on Observed Signals

A number of the experiments involved the use of a waveguide. It was important to characterize the ability of the waveguide to transmit the acoustic signals successfully to the transducer. This was investigated with NaOH pellets dissolving in water. Four experiments were performed: one using just a beaker as the sample vessel; one using each of the two waveguides; and a "blank" run which collected background noise. The average power spectra for the four experiments are given in Figure 63. Background signals were collected with a lower trigger level setting than used for the three experiments with NaOH. The waveguides were able to transmit the signals from the

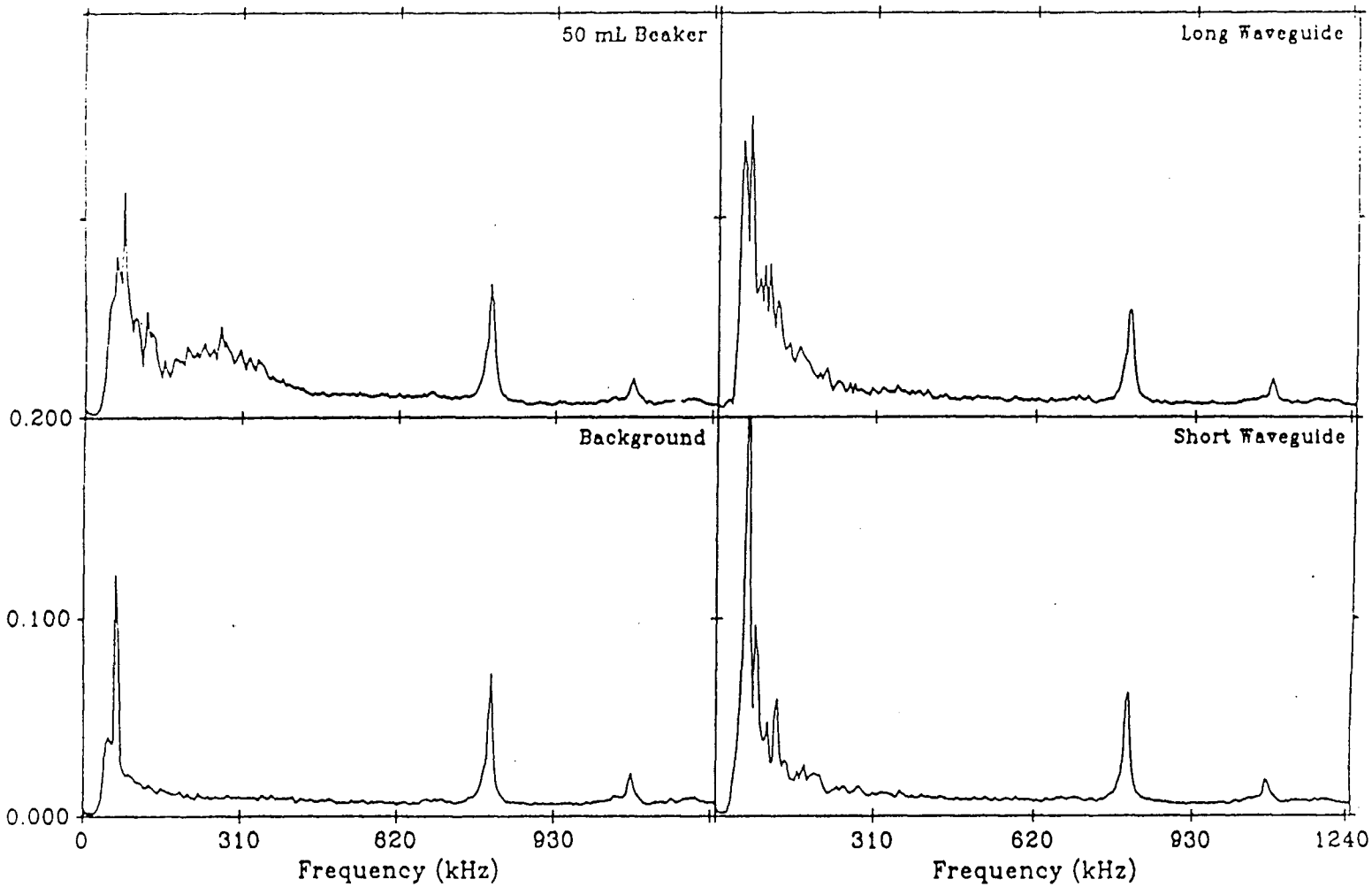


Figure 63) Four average power spectra for the NaOH hydration using different sample holders. The waveguides can be seen to attenuate the higher frequencies (above about 250 kHz) as compared to the beaker.

NaOH sample to the transducer. Some attenuation of the signals was evident. In the experiment using the beaker, the signals contained frequency components of up to about 400 kHz. A peak at about 800 kHz is a sharp resonance of the transducer, and is seen in all spectra. The short waveguide attenuates the higher frequency components of signals such that they are limited to about 250 kHz. The spectrum for the long waveguide also shows the acoustic signals attenuated to about 250 kHz but with a sharper profile at around 75 kHz compared to that of the short waveguide. The attenuation of the frequencies can be seen in Figure 64 with the lower average values of FMED and FMEAN for the experiments using the waveguides. These results are not surprising since workers in sonar have known for some time that higher frequencies are more readily attenuated as a function of distance than lower ones[100].

It can also be seen from Figure 63 that the use of the 50 kHz - 2 Mhz filter setting on the amplifier does not guarantee the exclusion of frequency components which fall outside this range. Although the frequency spectra show very little power below 50 kHz, there is still some power present. This suggests that the removal of these frequencies does not appear to be as would be expected from an ideal filter. It is thus apparent that the nature of the amplifier used becomes another factor to be considered.

This experiment indicated the need to perform replicate experiments with the same sample vessel and / or waveguide apparatus. Comparisons of experiments performed with different waveguides are complicated by the different attenuations. Sound is attenuated according to a relationship similar to Beer's Law[101]. The coupling agent used to attach the apparatus to the transducer has also been shown to affect the signals acquired and is another important factor[33]. Silicone grease was therefore used for all experiments.

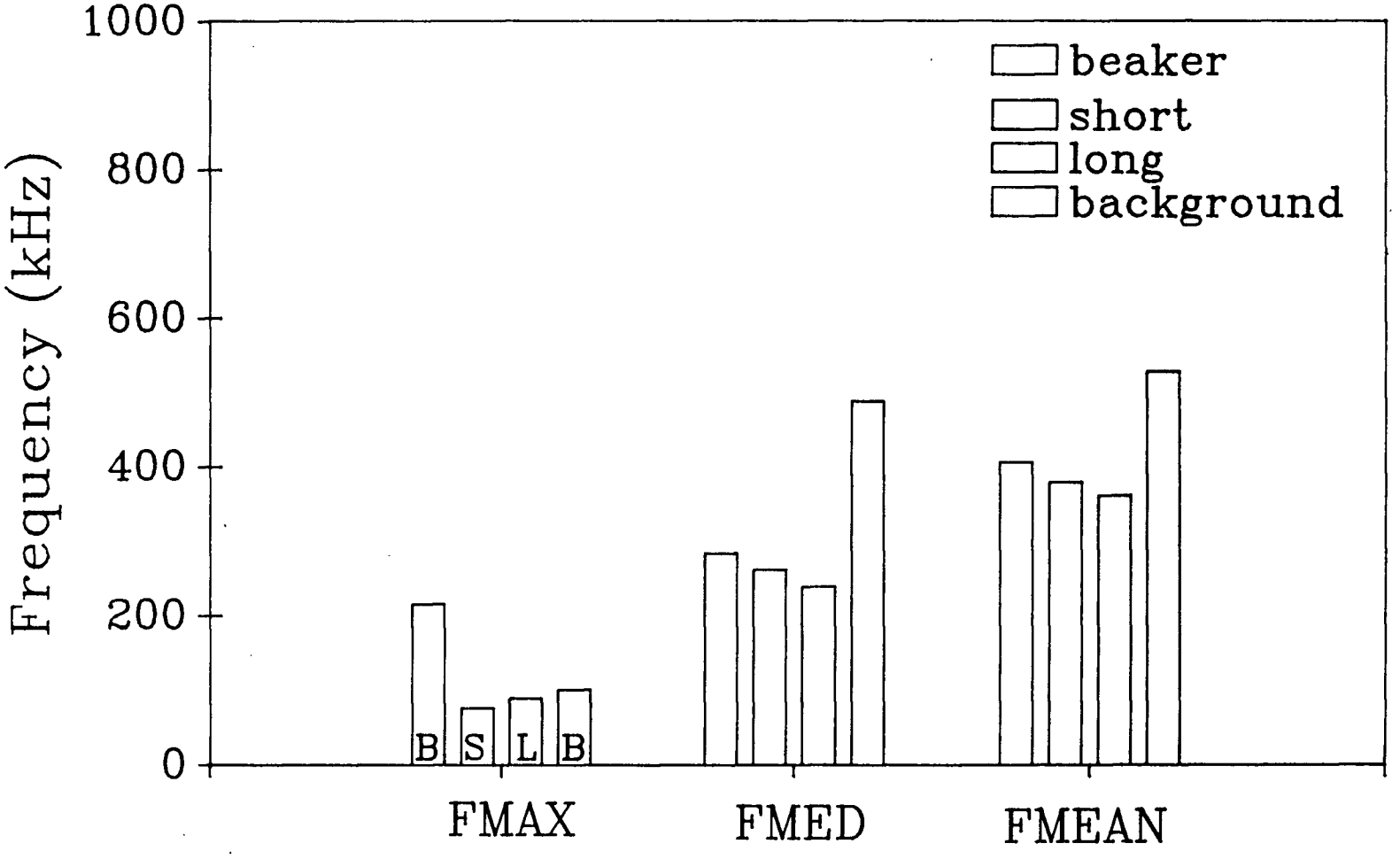


Figure 64) Median frequency (FMED), frequency of maximum intensity (FMAX), and mean frequency (FMEAN) for the four NaOH data sets. It can be seen that the waveguides attenuate the signals by the lower values of FMED, FMEAN, and FMAX compared to those acquired with the 50 mL beaker.

VII.1.6 Selection of Descriptors for Pattern Recognition

The choice of the best descriptors depends somewhat on the data being used. As this work was to lead toward largely automated data analysis, it was decided initially to use all descriptors at each step until studies could be done to determine more general conclusions about them. The higher octiles are included as the actual frequencies they refer to will depend on the acquisition rate settings on the equipment.

Some descriptors are better at discriminating between different classes of signals than others. The performance of the descriptors to describe each signal sufficiently for pattern recognition analysis depends on the choice of descriptors used. Fisher weights of the descriptors have been calculated to show their discriminating capabilities[102]. This paper concluded that frequency domain descriptors gave the best discriminating values. The time domain descriptors such as **RMS**, **PEAK**, and **AREA** tended to be affected by the distance from the source to the transducer and thus may not be as qualitatively useful for large volume samples.

Figure 65 shows the correlation between the **AREA** and **RMS** descriptors obtained for the liquid crystal, 4-n-pentyloxybenzylidene-4'-n-heptylaniline. Most points lie on a line, that is, there appears to be a high correlation of these two descriptors in this experiment. Using ABSCAT to identify points, it was found that the points lying off the line were due to signals which have amplitudes greater than the range digitized (Figure 66). The near-linear relationship between **AREA** and **RMS** would suggest that including both in pattern recognition algorithms would be redundant. While the linear relationship is useful to determine signals which may be corrupt, pattern recognition would be more consistent (and would take less computing time) by eliminating one or the other. Also, a composite descriptor, such as the ratio **AREA / RMS** could be used.

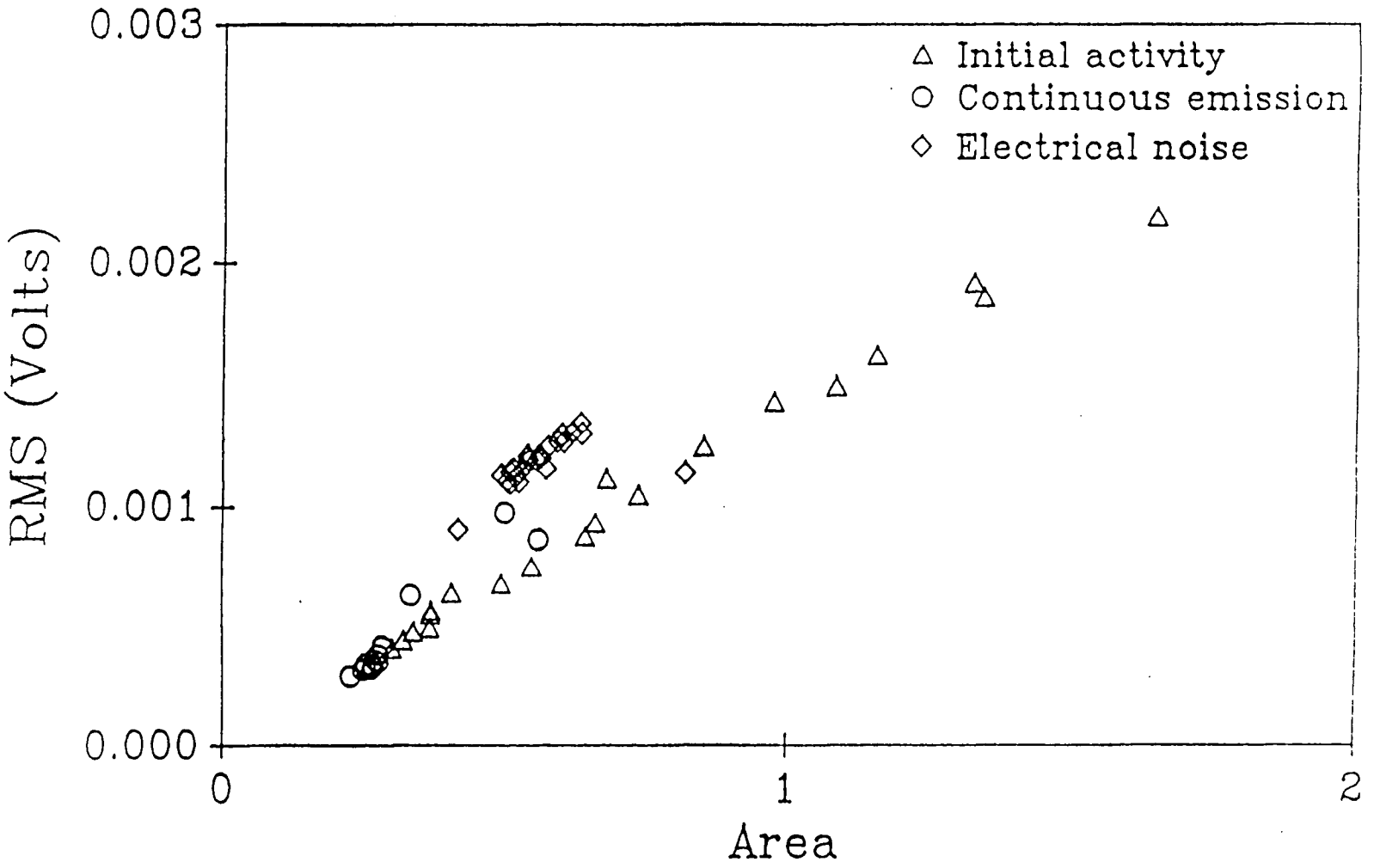


Figure 65) Plot of RMS Voltage versus signal AREA for the liquid crystal, 4-n-pentyloxybenzylidene-4'-n-heptylaniline. The data points tend to lie on a curve; except for the electrical noise signals. This type of plot can therefore be used to determine a signal's validity.

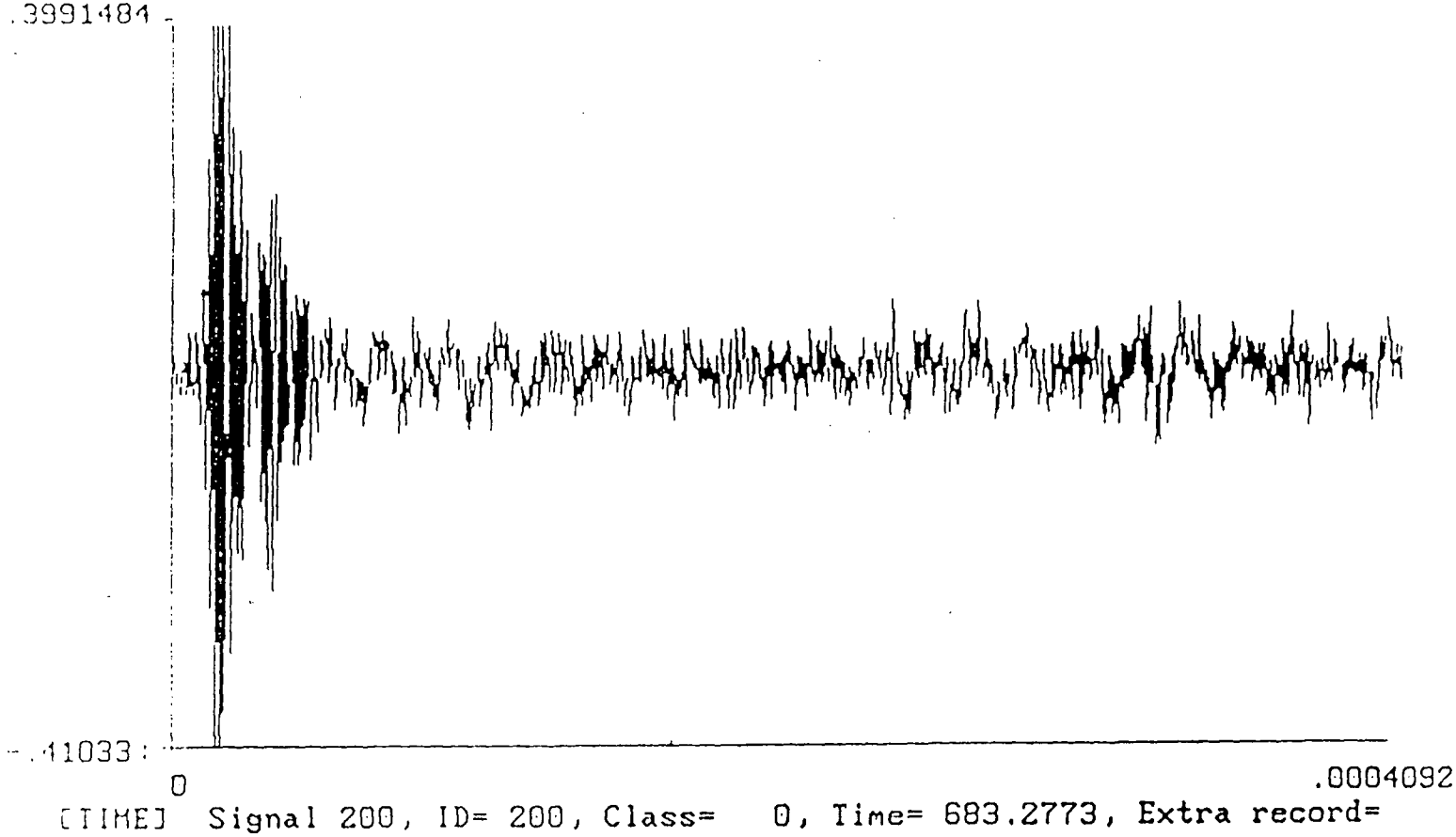


Figure 66) **SIGVIEW - Program display of acoustic signal with amplitude outside Voltage range digitized by oscilloscope. Vertical axis is measured in Volts and Horizontal axis is measured in seconds.**

Table 8

Summary of Factor analysis 11-09-1989

NAOHEXP1.DS1 => Link Scaling => NAOHEXP1.AF2

N	Eigen-Value (l_N)	l_N/l_{N+1}	l/l_{AV}	RMS	VAR	CPV
1	2100	2.37	9.71	5.96E-1	5.11E-1	51.12
2	885	2.16	4.10	4.46E-1	2.16E-1	72.69
3	410	2.12	1.90	3.55E-1	9.98E-2	82.68
4	193	1.31	8.93E-1	3.03E-1	4.70E-2	87.38
5	148	1.24	6.84E-1	2.56E-1	3.60E-2	90.98
6	119	1.74	5.52E-1	2.11E-1	2.91E-2	93.88
7	68.4	1.35	3.17E-1	1.80E-1	1.67E-2	95.55
8	50.6	1.26	2.34E-1	1.53E-1	1.23E-2	96.78
9	40.1	1.27	1.86E-1	1.28E-1	9.78E-3	97.76
10	31.5	1.38	1.46E-1	1.03E-1	7.68E-3	98.53
11	22.7	1.59	1.05E-1	8.17E-2	5.54E-3	99.08
12	14.3	1.30	6.64E-2	6.43E-2	3.50E-3	99.43
13	11.0	2.74	5.09E-2	4.67E-2	2.68E-3	99.70
14	4.02	1.35	1.86E-2	3.83E-2	9.79E-4	99.80
15	2.98	1.27	1.38E-2	3.07E-2	7.26E-4	99.87
16	2.34	1.51	1.08E-2	2.29E-2	5.69E-4	99.93
17	1.55	1.32	7.16E-3	1.59E-2	3.77E-4	99.97
18	1.17	4.72	5.43E-3	6.64E-3	2.86E-4	99.99
19	0.249	----	1.15E-3	0.00	6.06E-5	100.0

---

l_{AV} = average eigenvalue = 216

RMS = Residual Mean Squared error

VAR_N = variance accounted for by eigenvalue N

$$CPV = \text{cumulative percent variance} = \sum_{i=1}^{N} VAR_i$$

Table 8 shows the stepwise discriminating parameters for the NaOH data. The eigenvalues are used to calculate functions which give an indication as to the number of primary factors. The average value for the eigenvalues is sometimes used. Factors which have an eigenvalue less than that of the average eigenvalue are labelled as

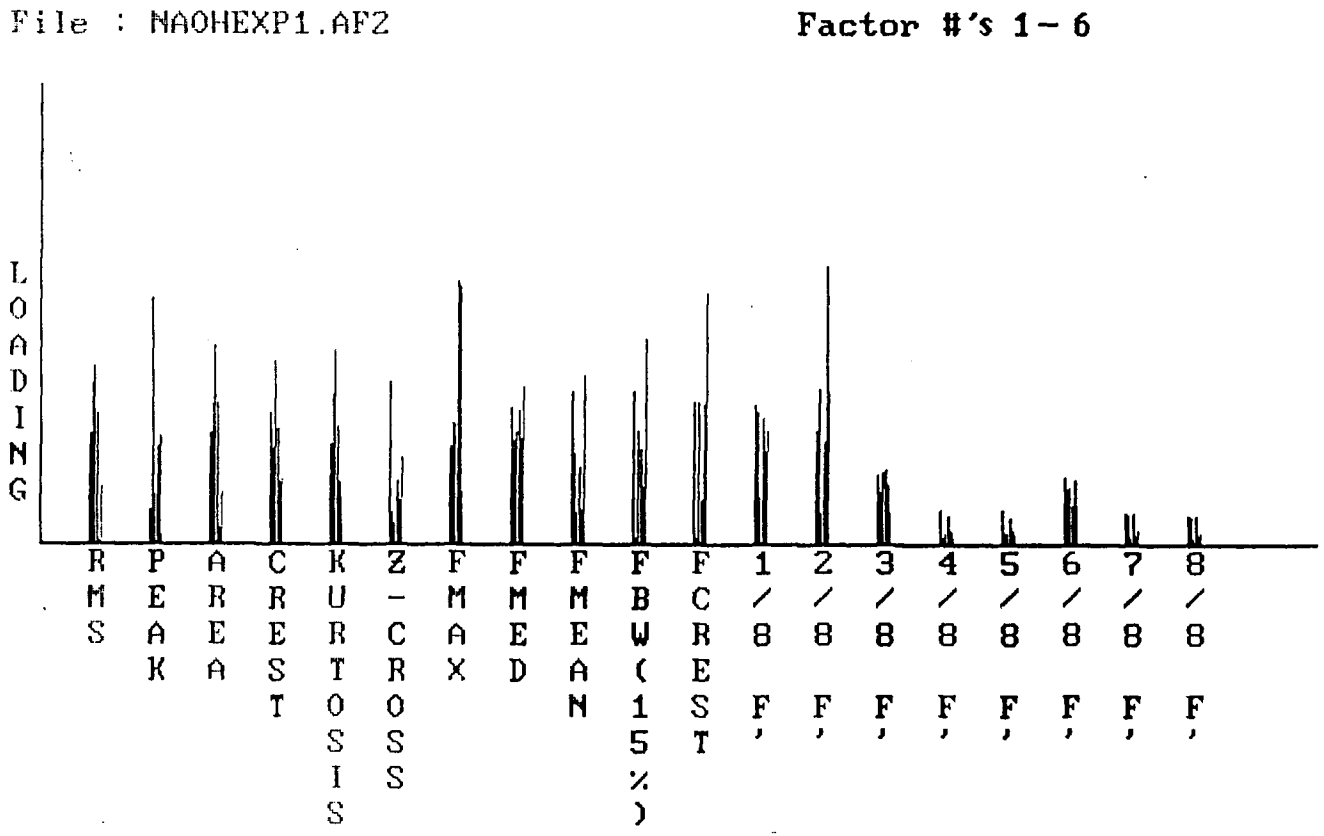


Figure 67) ABSCAT - Factor loadings for first six factors for NaOH experiment. Emphasis on primary factors is due largely to all descriptors other than higher numbered frequency octiles.

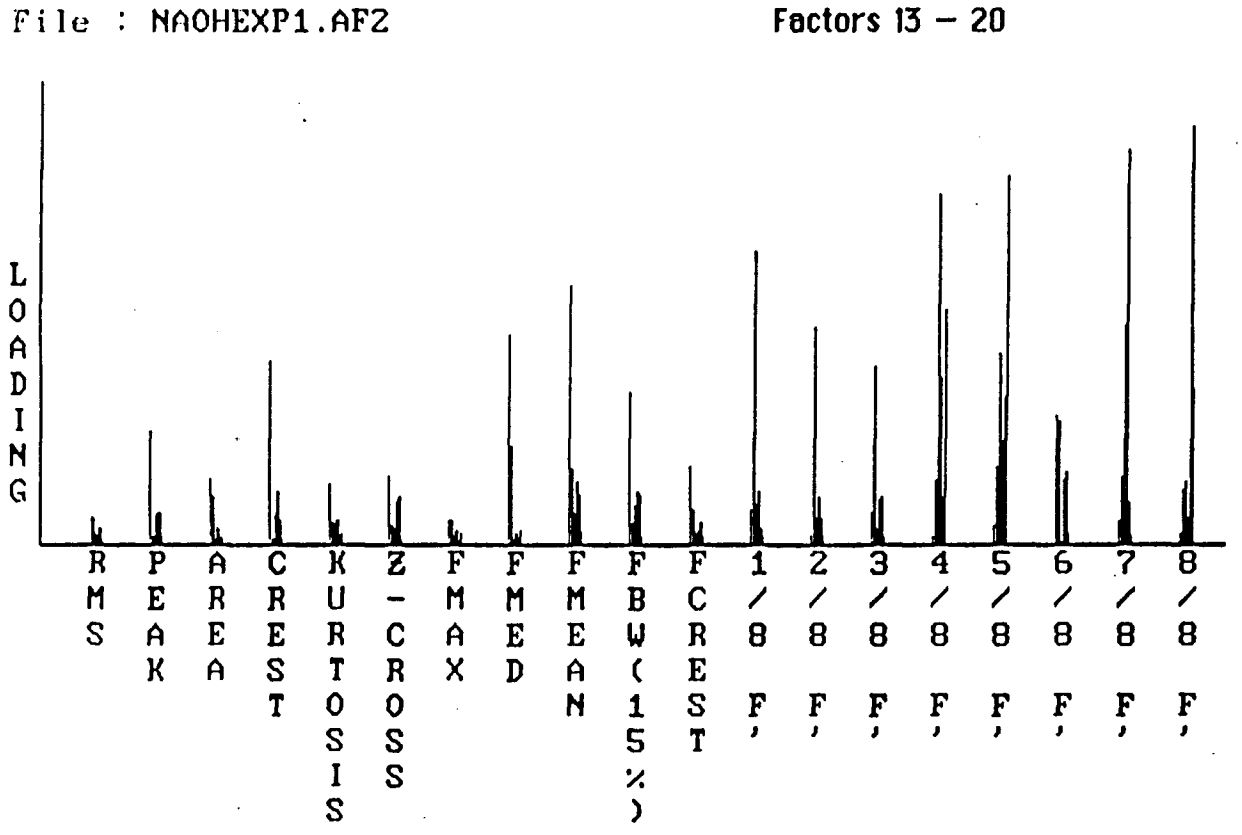


Figure 68) ABSCAT - Factor loadings for least significant factors for NaOH experiments.

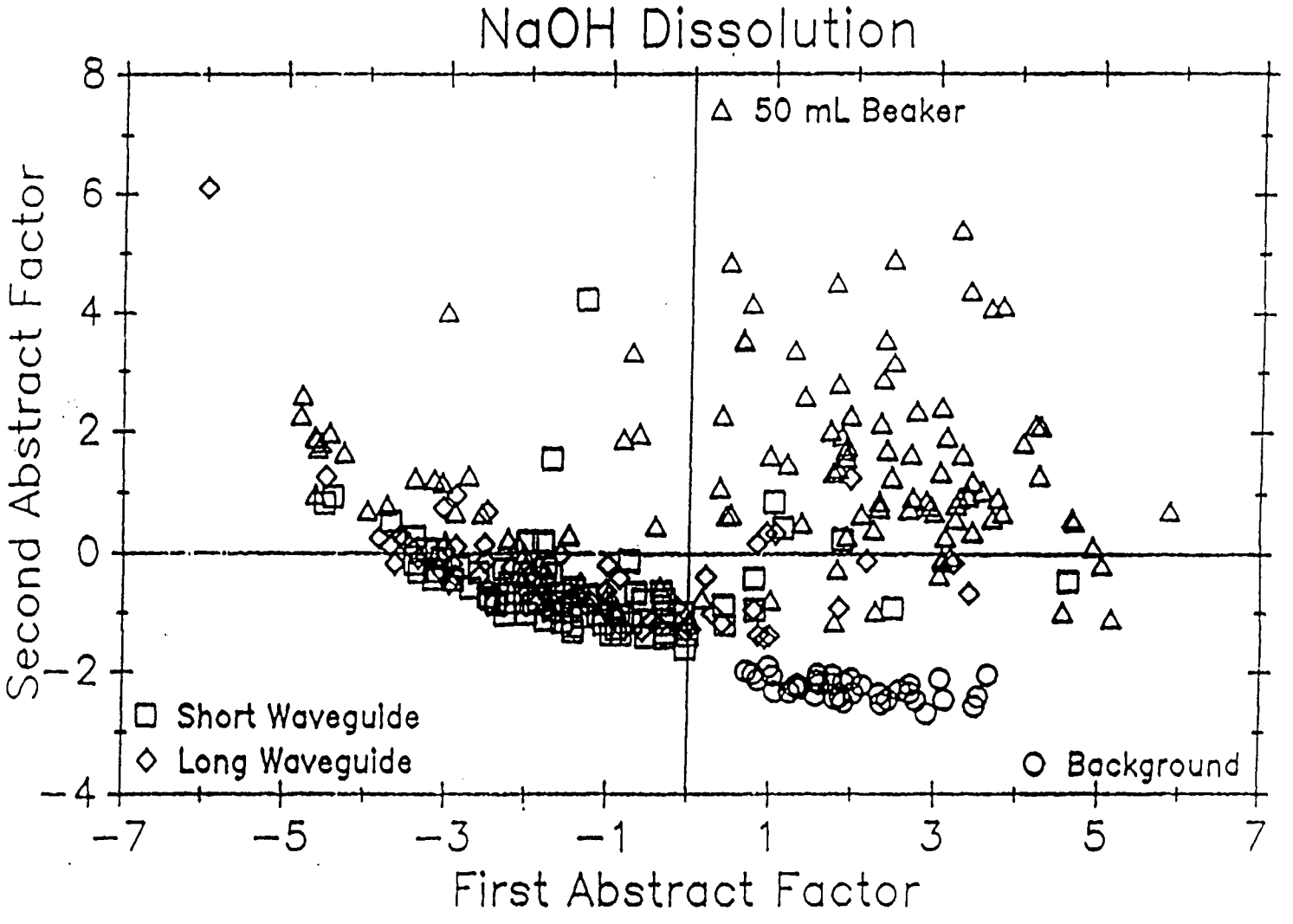


Figure 69) Plot of factor space defined by first two principal components for the NaOH experiments. The background signals, denoted by a circle, can be seen to occupy a seperate region of the factor space from the signals acquired from the dissolving NaOH.

secondary factors. Using this criteria, the NaOH data seems to have only three or four primary factors. Other criteria sometimes used also suggest that there are perhaps only four primary factors. The first four factors for the NaOH data account for 80% of the variance in the 19 dimensional feature space as shown in Table 8 by the Cumulative Percent Variance column (CPV). The factor loadings display from ABSCAT shows the loading of the descriptors on the first six factors for the combined NaOH experiments (Figure 67). All descriptors except the higher numbered frequency octiles contribute to the variance of the first six factors. This is also visible by looking at Figure 68 which shows the features which comprise the least significant factors. This picture suggests that these octiles may not be as useful as descriptors.

By looking at the space (two-dimensional plane) defined by the first two factors (Figure 69), it can be seen that the background signals are separated from the NaOH signals by factor analysis. Also, the spread of the signals acquired from samples in waveguides is less than that of the signals acquired from the sample in a beaker - further evidence of the attenuation of the signal by the acoustic path from the sample to the transducer.

VII.1.7 Choice of Scaling Technique for Descriptors

As discussed above, the use of auto-scaling alone was found to be inadequate for the automated analysis. The frequency octiles alone contained differing variances. (The actual frequency ranges of the octiles will depend on the settings of the digitizing oscilloscope, but an automated method of analysis should account for this automatically). Link scaling was used to prevent the descriptors corresponding to the less variant RMS frequency octiles from being given greater importance than they are due.

VII.1.8 Visualization of Signal Classes Using Dendrograms

The program DENDGRAM has seven different methods for calculating a dendrogram for a set of data. In addition, several different measures of similarity are possible. The Minkowski exponent allows the use of the squared Euclidean, the Manhattan City Block, or any distance measure which can be calculated using the Minkowski formula (given in Chapter 6). This means that a large number of different dendrograms can be generated from the same set of data. The dendrograms in Figures 58 - 61 are all calculated from the same data in Table 2. These make it clear why comparisons of the structure of different data sets by looking at dendrograms calculated by different methods (and that includes the distance metric used), is not recommended. It is advisable to use a method which is familiar to the operator and to do all comparisons using this method. The choice of which to use is not a simple matter. The literature is full of advice[58,61,88] but the bottom line is that there is no "best" method. The choice of algorithm for the calculation of the dendrogram can be a matter of personal preference rather than one of formal guidelines.

The squared Euclidean distance measure is almost exclusively used in this work (and in this laboratory, and largely in the field of pattern recognition in general). This is not because of any particular benefit of this measure over others but because of the familiar geometry involved.

Our use of the dendrogram to show the number of classes present in a set of data has led us to try different methods in a search for the one best method. The outliers of a set of data, which usually correspond to signals due to background or even electrical noise, could be easily identified by a dendrogram regardless of the method used. No one method appeared to be the best. Each had its own merits and pitfalls. Single linkage has the method most commonly used in this laboratory, but perhaps this

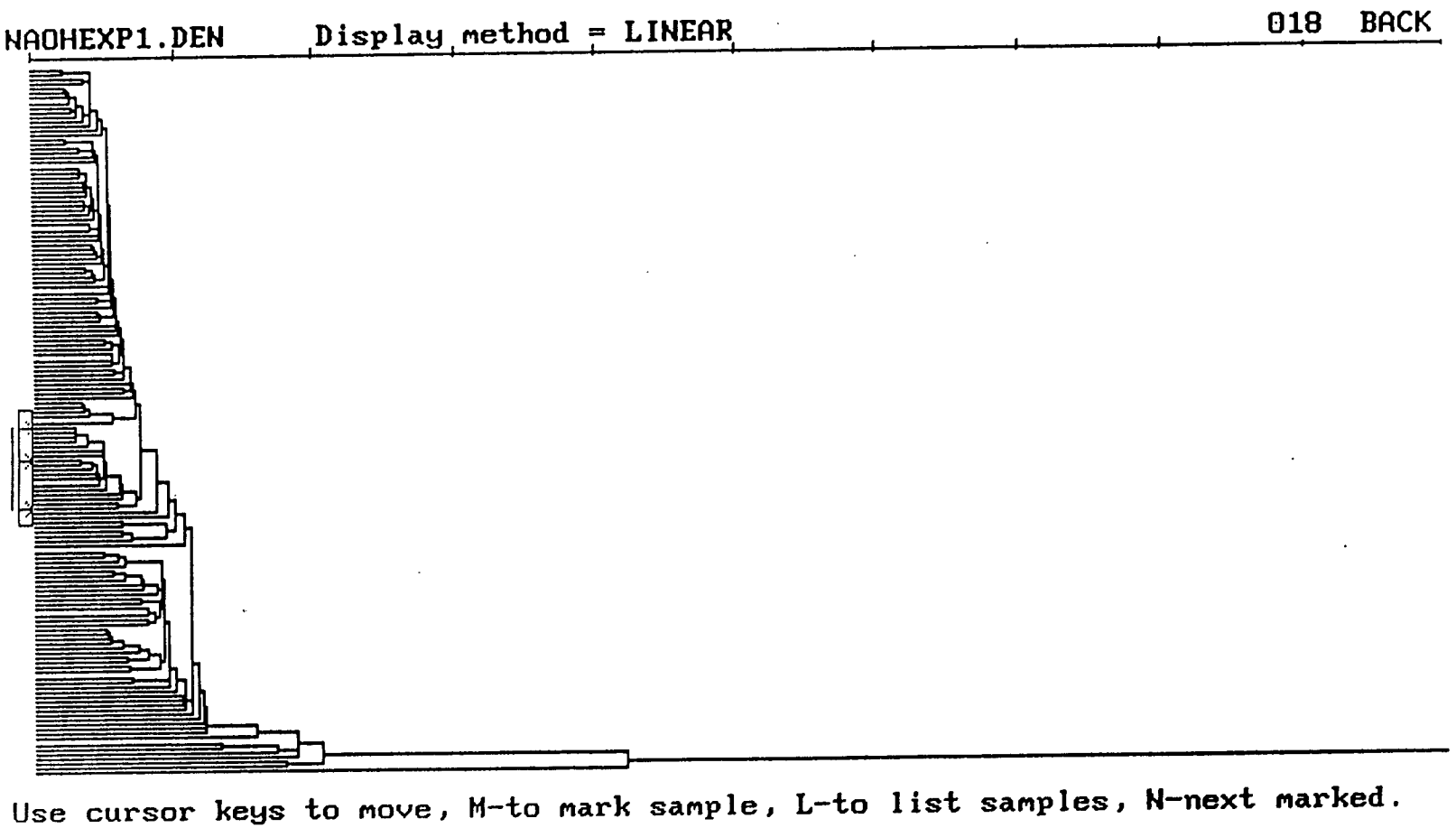


Figure 70) Dendrogram for NaOH experiments. The windowed area includes all the signals from the collection of background showing them to be separated from the other signals.

is simply because single linkage is the default which appears first in the list of methods offered by DENDGRAM (Figure 39).

The use of a dendrogram to determining the number of possible cluster-types of signals is somewhat subjective. By the use of appropriate display methods, it may be easier for the operator to make judgments. For example, the axis of similarity need not be linear[88]. DENDGRAM has options which allow for exponential display (Figure 41) and for various powers of the similarity axis (Figure 42). Early in our work, we found the use of the centroid dendrogram methods prone to cross-over. This complicated analysis, and the use of a "relative" similarity axis was developed[89]. In this way, the occurrence of cross-over does not create problems for interpretation. These different display methods help to "stretch" the dendrogram so that its structure and therefore the relationships among the elements of the data set can be better visualized. None of the display methods actually change the dendrogram but only serve to enhance the information. (This is discussed in reference 89).

Figure 70 shows the dendrogram calculated for the NaOH experiments. The background signals are all found together as a single class of signals (denoted by the window on the figure). This shows the ability of the dendrogram to allow for some simple classification of the types of signals. It is apparent however, that the dendrogram is not able to be used to make such conclusions by itself. It is though, a useful means of visualization of the data structure.

VII.2 Chemical Systems

VII.2.1 Sodium Hydroxide dissolution

Solid NaOH has been used in this laboratory and has been shown to be a good system to study using acoustic emission[47]. Sodium hydroxide dissolution was used as an acoustic source to test the effects of the pathway from the sample to the transducer.

The dissolution of sodium hydroxide occurs spontaneously. The pellets of NaOH dissolved completely in between 3 to 5 minutes. The reaction releases acoustic energy while the pellet is dissolving and ceases once the NaOH is completely dissolved. The source of the acoustic emission may be due to the fracture of the pellet structure and localized boiling. Bubbles are released from the pellet, and this may also be the source of acoustic energy. The bubbles may be due to absorbed air being emitted or localized boiling occurring.

VII.2.2 Trimethylolethane (Trimet)

Trimet was shown to be very acoustically active, as the chart recorder peak voltage output trace shows (Figure 71). The reported temperature[80] of the solid-solid phase transition is 81°C. When the temperature of the sample was raised through that temperature, no acoustic emissions were detected by the apparatus. The oven was then allowed to cool to room temperature. No detectable emissions were released by the sample as the temperature of the oven dropped through the transition temperature. Indeed, no appreciable emissions were observed from the sample until the oven reached a temperature of 40°C. At this point, it was thought that no acoustic activity would be seen, as the oven had effectively cooled to within ten degrees of the temperature of the room. The 15 gram sample then started emitting acoustic signals, and continued to release the acoustic energy for four hours. At first glance this suggests an extremely large energy storage capacity - "move over cold fusion". The proposed use

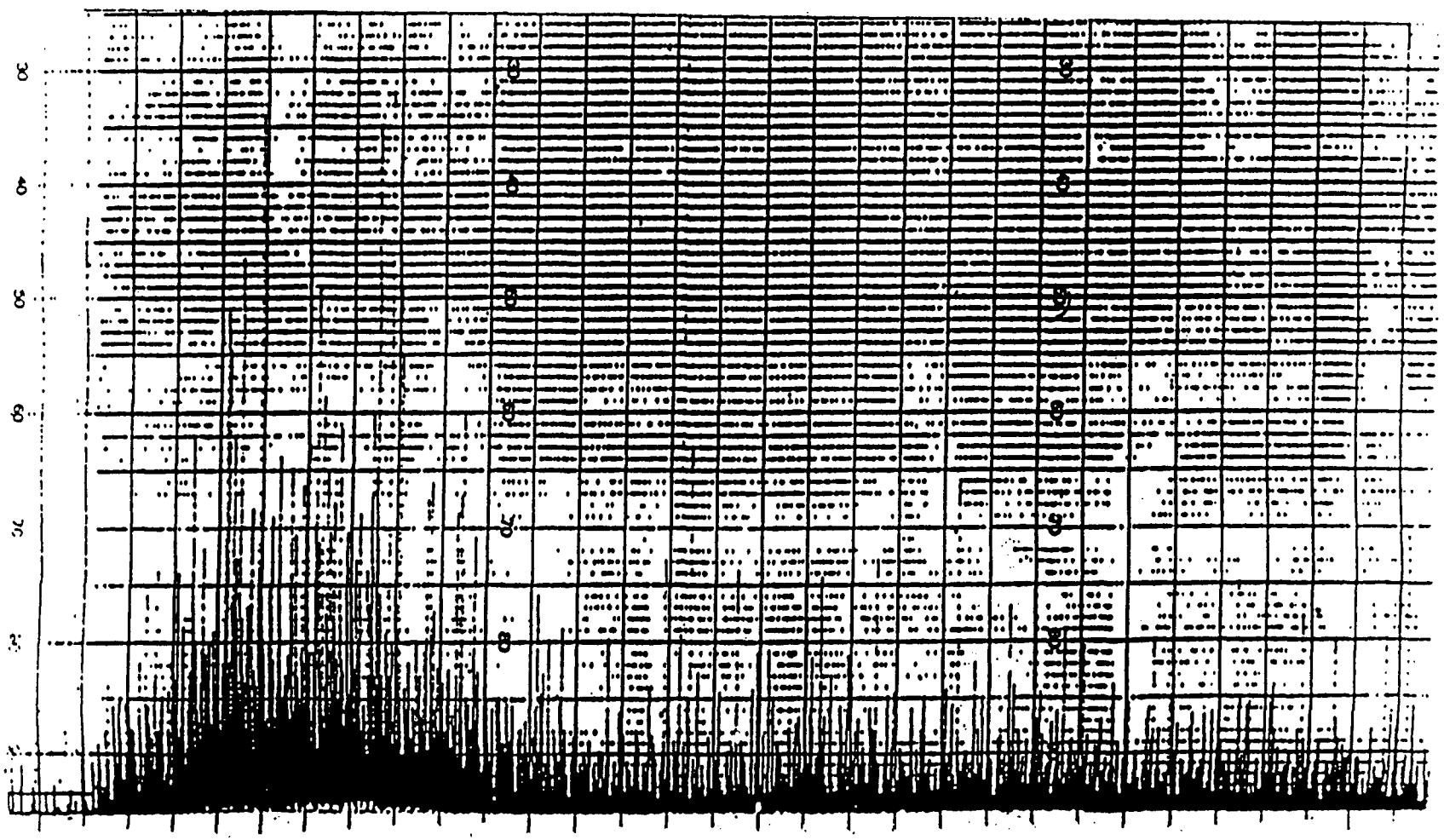


Figure 71) Acoustic activity during cooling of 10 grams Trimet. Intensity plotted versus time. Each horizontal division equals two minutes.

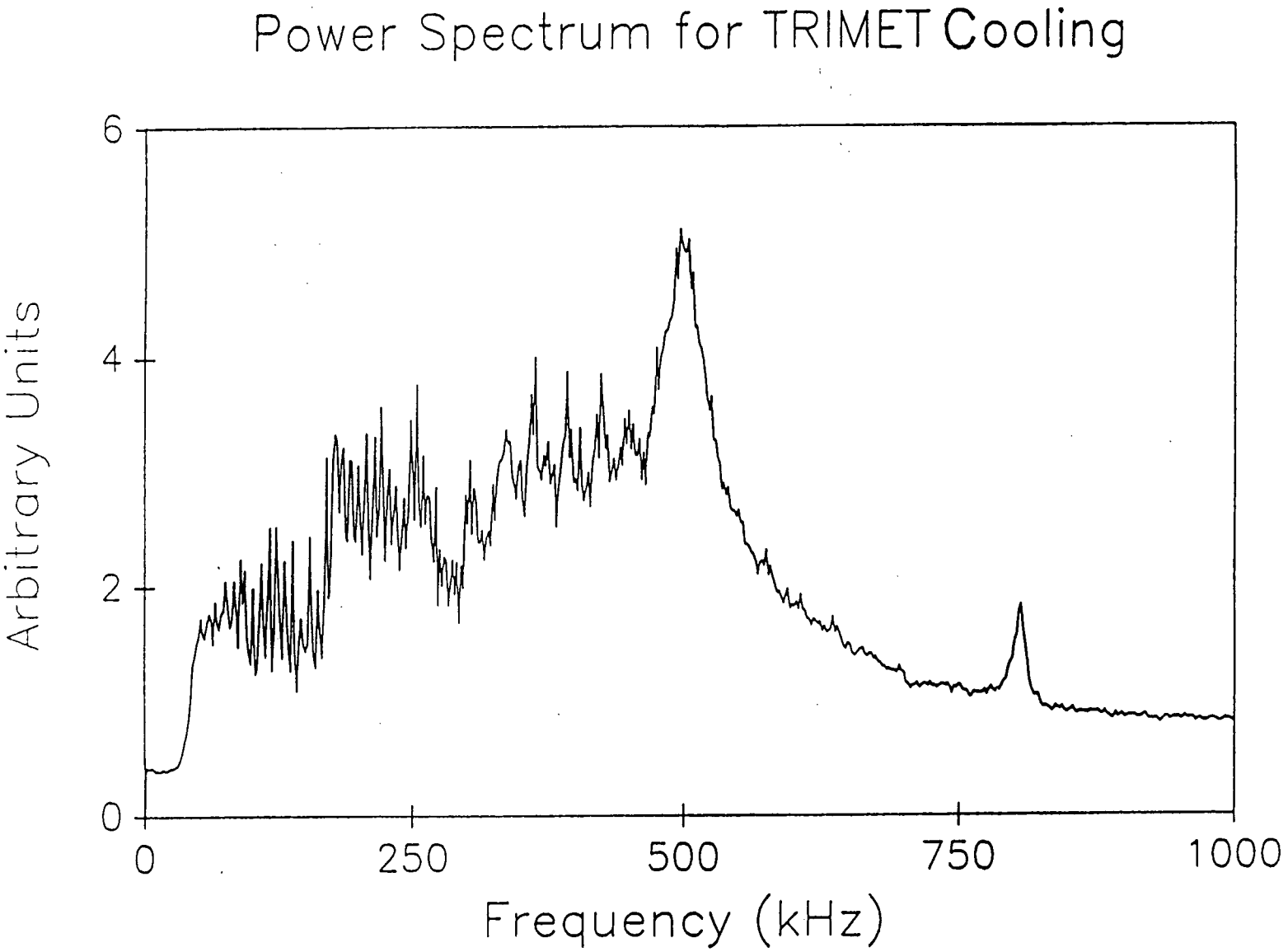


Figure 72) Frequency power spectum for acoustic activity during cooling of Trimet.

of this material as an energy storage medium for solar energy panels seems to be full of promise. This is even more apparent when the duration of the acoustic activity is taken into account. The release of energy in the form of acoustic vibrations is believed to accompany the release of other forms of energy - such as heat. The sample appeared to have some success at maintaining the heat of the 81°C transition against an ambient temperature of 40°C.

The power spectrum of the cooling Trimet is shown in Figure 72 and is quite obviously different from the NaOH spectrum (using the long waveguide in Figure 61). This suggests that a different process (or processes) is occurring in these systems.

VII.2.3 Intumescent Fire Retardants (IFR)

The intumescent fire retardants proved to be tricky samples on which to perform acoustic emission experiments with the current apparatus. The IFR samples were fine, white powders - physical mixtures of polypropylene, ammonium polyphosphate, and pentaerythritol. The minimum activation temperature of the samples was about 200°C and temperatures to 400°C would be necessary to ensure complete carbonification of the plastic substrate. The first experiment was performed in the oven. The oven had a maximum operating temperature only slightly above 200°C and thus the sample was slowly heated to this temperature.

The physical dynamics of the IFR mechanism are such that slow heating doesn't approximate well to the conditions of a fire. To circumvent this in further experiments, a Bunsen burner was used as the heat source with the sample in the short wave guide. Even with this approach, the rate of heating was not consistent between experiments.

The samples of IFR were found to be very active acoustically once a rapid rate of heating could be achieved. It was not clear whether the signals originated from the release of the gas (blowing agent) or from the setting of the polymer. One experiment in which the gases released from the sample actually caught fire produced a large number of signals while the fire burned. This perhaps would be the best technique to monitor this process by AE as it is similar to the actual operating conditions of the IFR and provides a more consistent heating. It is difficult to make judgements about the differences in the samples based on data that is collected under non-standard conditions.

VII.2.4 Hydration of Aluminum Chloride

The aluminum chloride experiments were performed using the integrator/analyzer acoustic emission apparatus developed in this laboratory[47]. There is no frequency data in this approach and as such no pattern recognition can be performed. One benefit of the use of this apparatus is the ability to regenerate the AE intensity plot normally only provided as a chart recorder tracing. The 200 ms time constant provided allows the output signal to drive a chart recorder and a computer analog to digital converter.

The samples of $AlCl_3$ all produced the same shape of acoustic activity plot. Figure 73 shows this plot for one of the samples. By comparison of the initial and final baselines, some air hydration of the $AlCl_3$ was taking place before the addition of water. The initial addition of water produced a very large spike - due to the instantaneous emission of acoustic energy as the first cloud of HCl was released. This reaction produced very intense acoustic emissions - pointing to the vigorous nature of the reaction.

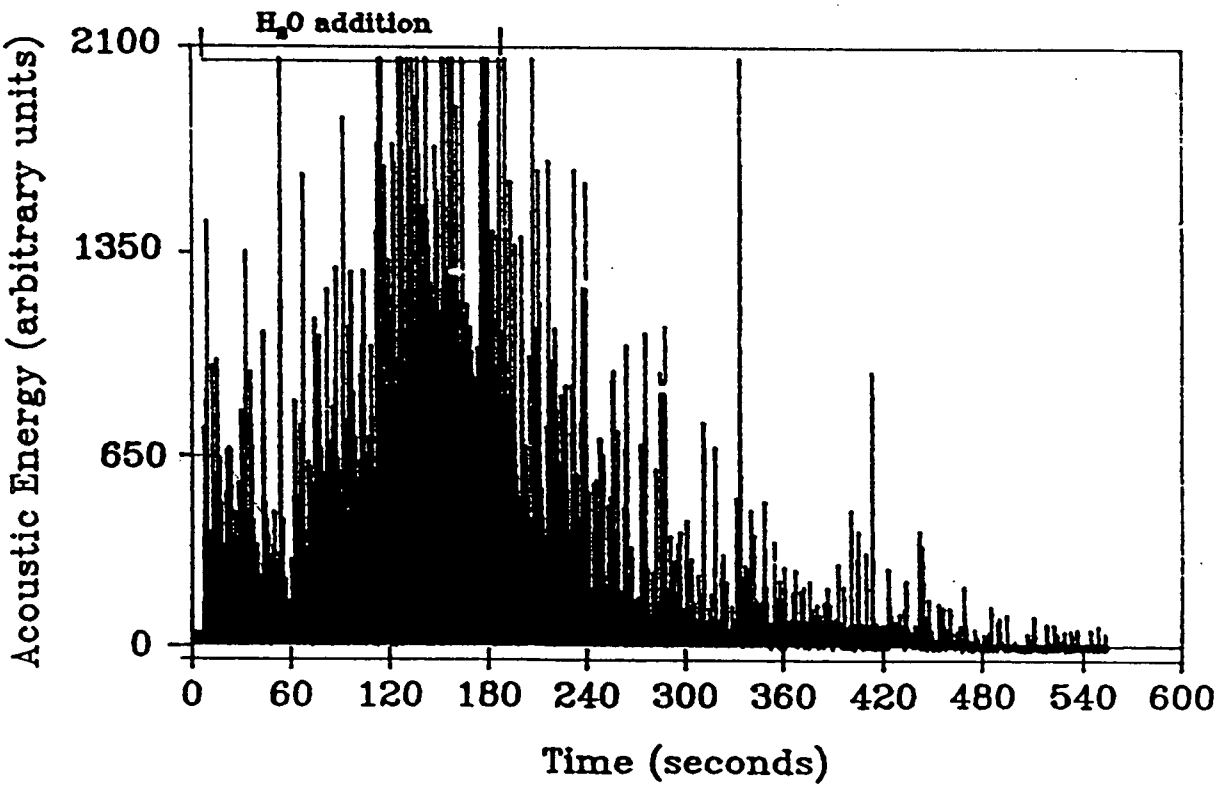


Figure 73) Acoustic activity from 5 mL of H_2O added to 0.5 grams AlCl3.

The AE activity returned to the baseline at the completion of the reaction. AE therefore, follows the progress of the reaction as emissions are only detected during the reaction. The source of the acoustic activity may be attributed to HCl evolution, crystal fracture, and possibly localized boiling.

As this is a heterogeneous reaction, it is a difficult one to monitor in situ and characterize by other means. The AE plot (Figure 73) appears to show a two-phase reaction, but the initial phase may be simply a mixing anomaly. A better method of monitoring this system may be to dissolve the $AlCl_3$ in dry methanol and merge the MeOH with a H_2O stream.

Figure 74 shows the pH of the final solutions as a function of weight of $AlCl_3$. One notices that there is a correlation. Because of the logarithmic nature of the pH scale, accurate interpolations are somewhat difficult. In the hopes that AE may provide a more consistent correlation with mass, the total integration of the peak level output of the conditioning amplifier is plotted against mass (Figure 75). The lack of simple correlation is obvious. Furthermore, one would expect an increase in detected AE with an increase in sample weight. It is known that the point at 1.5 grams $AlCl_3$ in Figure 75 may not be valid as the sample vessel was not centered on the active element of the transducer. While the removal of this data point still doesn't leave a simple correlation, its presence raises the issue of acoustic coupling between the sample and the transducer. How much a slight variation in coupling affects the transmission of the AE is not well understood, but is well known that there is an effect[8,41,96]. The further question is raised of the transmission properties of the different 50 mL vessels used for each sample. Obviously, a consistent approach is demanded.

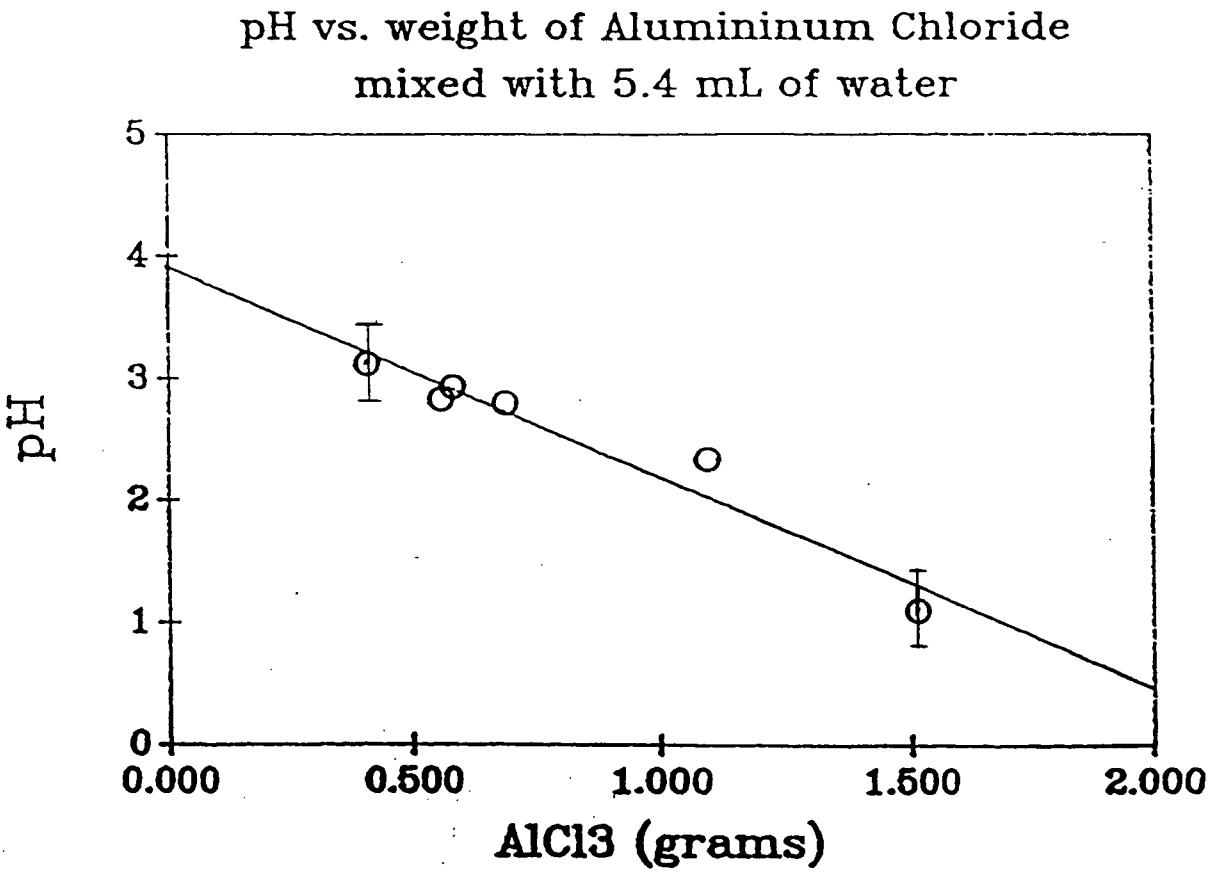


Figure 74) $AlCl_3$ **experiments. pH of final solutions versus initial mass of** $AlCl_3$**.**

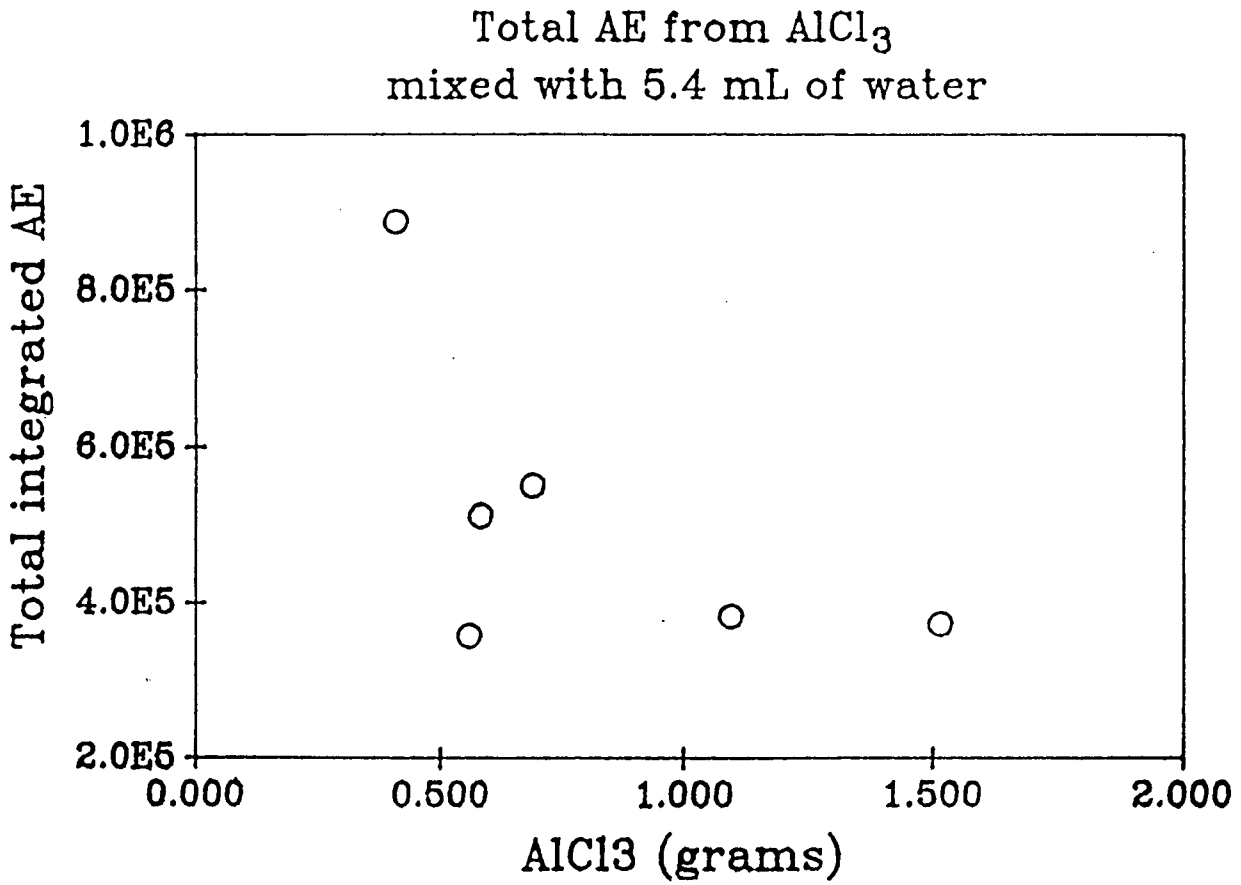


Figure 75) Total acoustic energy recorded versus initial mass of $AlCl_3$.

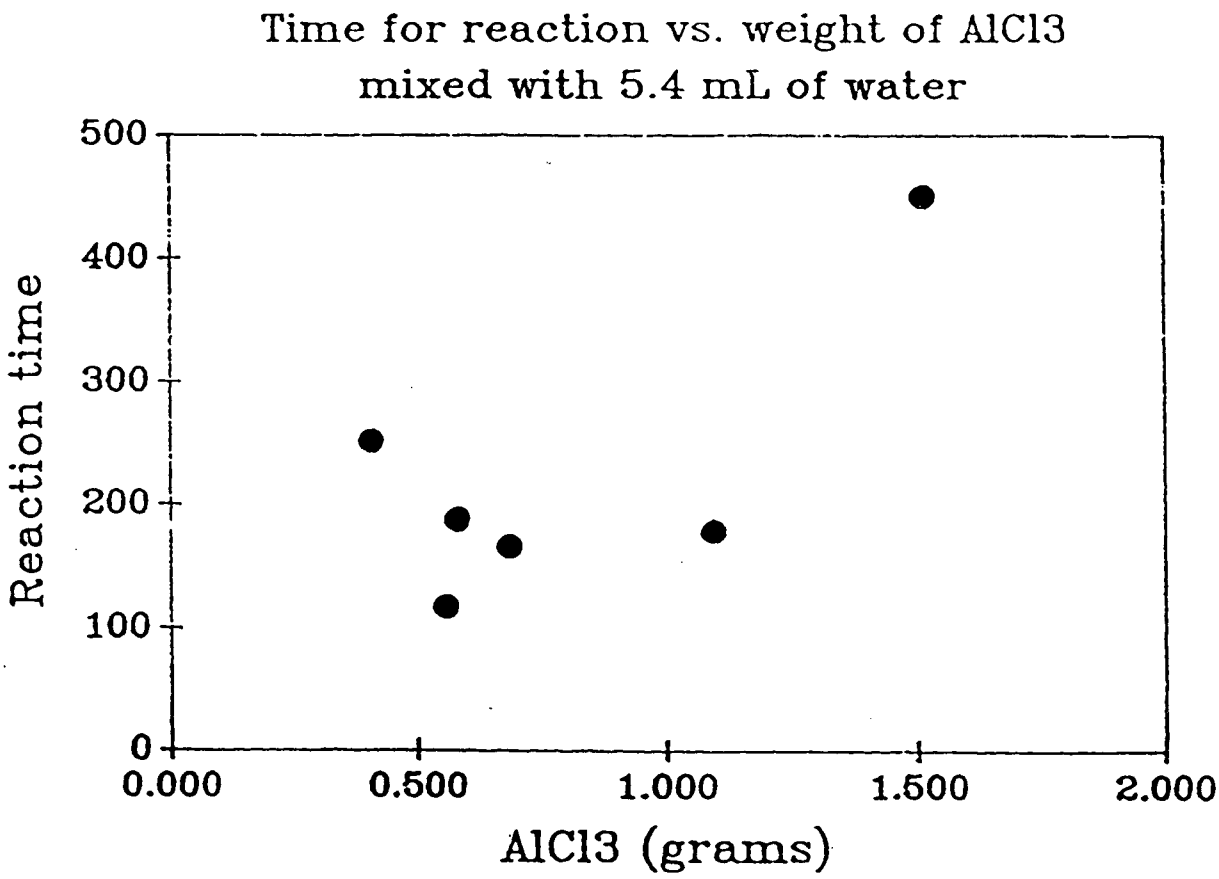


Figure 76) Reaction time (for 90% of total AE emissions) versus mass of AlCl3.

These experiments suggested reasons why the peak output of the amplifier is inadequate for good quantization. This time constant inhibits adequate quantization as, for example, a signal of duration 10 ms may have the same peak level as a signal of duration 100 ms. Obviously, the latter signal will have the greater energy content, but this isn't discernable from the peak output.

By examining the duration of the reaction, a relationship with mass can be surmised. However, since the water was added over a period of time, any dependance of reaction time on mass may be obscured (Figure 76) by the mixing time. Also, the existence of air hydration has been noted from the initial baseline. An added possibility is that the increase in mass may have led to increased heating. A faster reaction would have produced signals more rapidly and therefore, a greater percentage of them would have been "missed" during the "dead time" of the oscilloscope. This would also explain the lack of correlation between total AE and sample mass (Figure 75).

VII.2.5 Liquid Crystals

The sample of α-ω-bis(4-n-decylaniline-benzilidene-4'-oxyhexane) was found to be acoustically active. However, on completion of the experiment, it was found that discernible discoloration of the sample had occurred. It was then not apparent whether the acoustic activity had been due to the phase transformations or decomposition. The highest experimental temperatures, 180-200°C in an enclosed oven, were such that direct observation of the sample temperature wasn't possible. Thus it isn't known exactly the temperature of the sample at the time(s) of acoustic activity.

The lower-temperature phase transitions of the sample of 4-n-pentyloxybenzylidend-4'-n-heptylaniline were such that a temperature probe could be used during the experiment performed by Dr. P. Y. T. Chow. The plot of the RMS

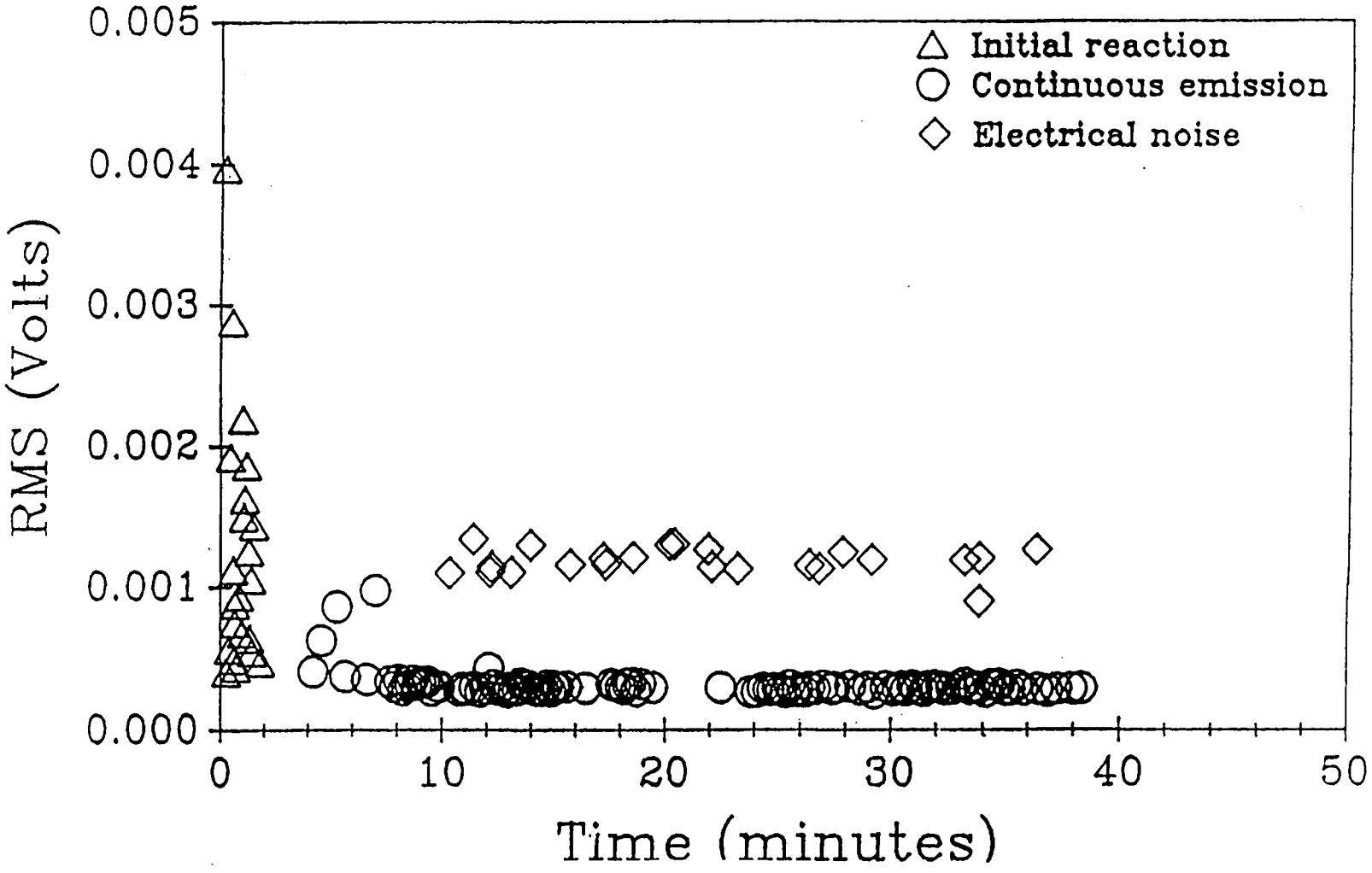


Figure 77) Plot of RMS Voltage versus time for the liquid crystal, 4-n-pentyloxybenzylidene-4'-n-heptylaniline. Signals due to electrical noise are denoted by diamonds and can be seen to be seperate from sample activity.

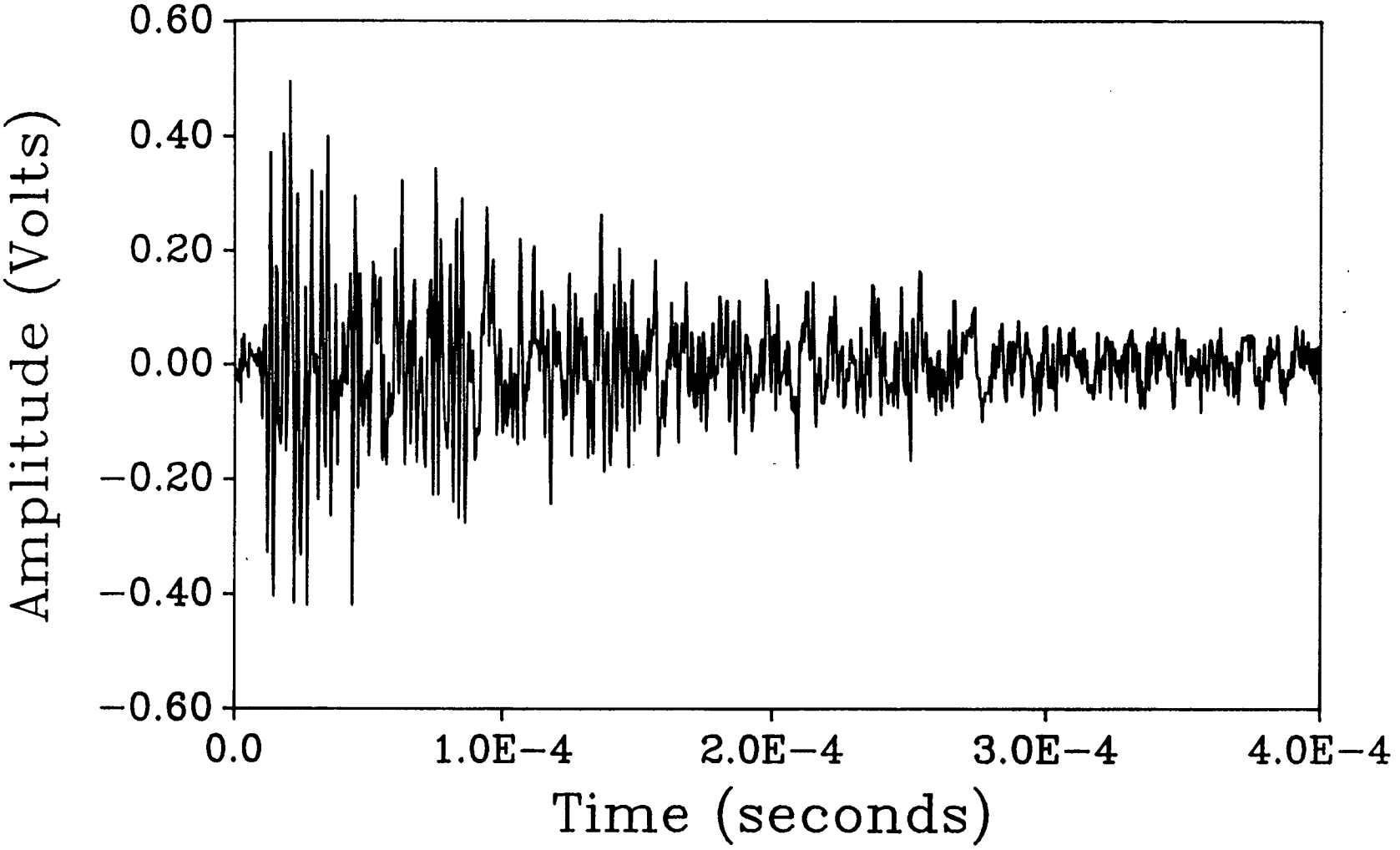


Figure 78) Acoustic signal (short duration burst) during initial acoustic activity of liquid crystal, 4-n-pentyloxybenzylidene-4'-n-heptylaniline.

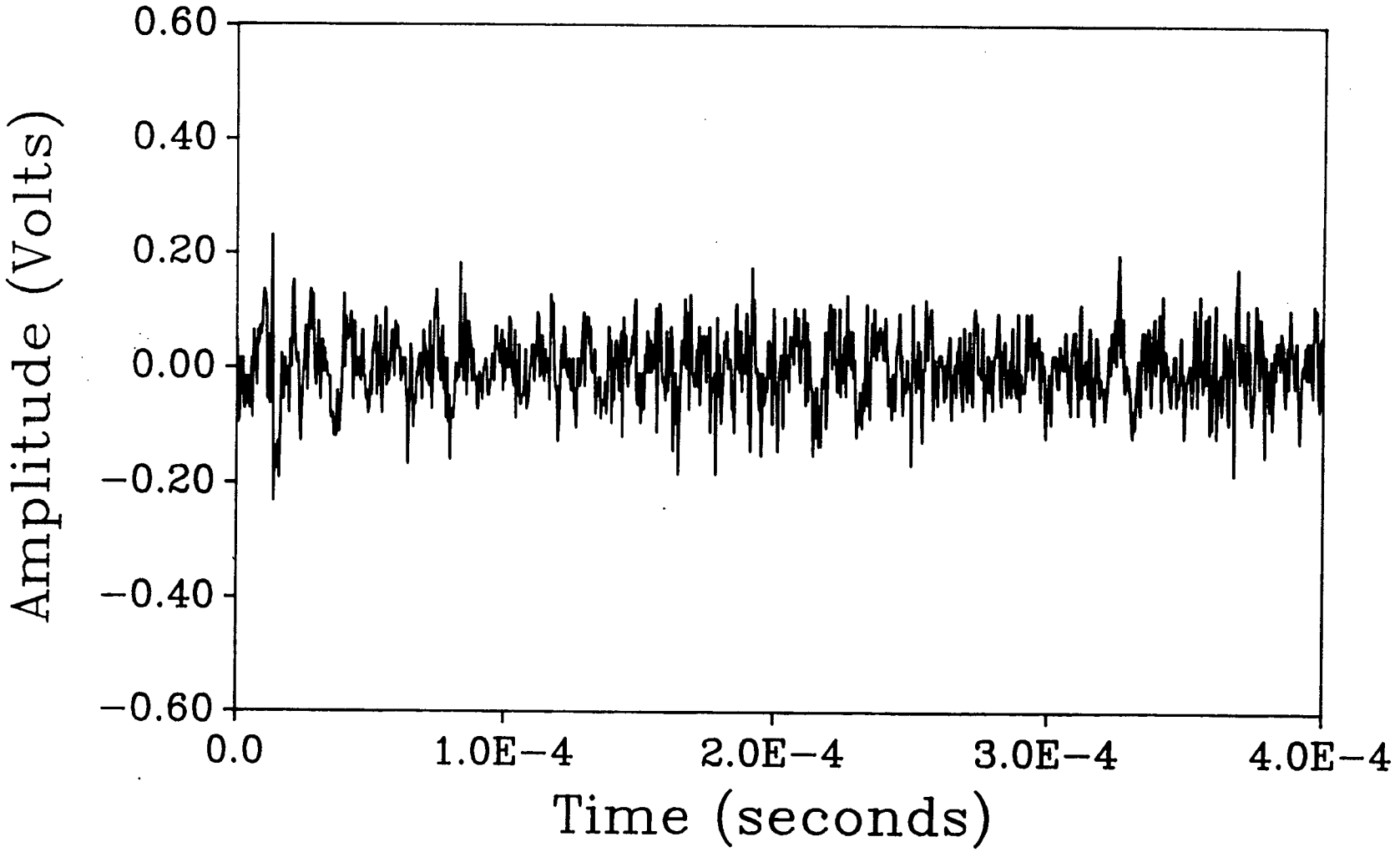


Figure 79) Continuous acoustic activity during "active phase" of 4-n-pentyloxybenzylidene-4'-n-heptylaniline.

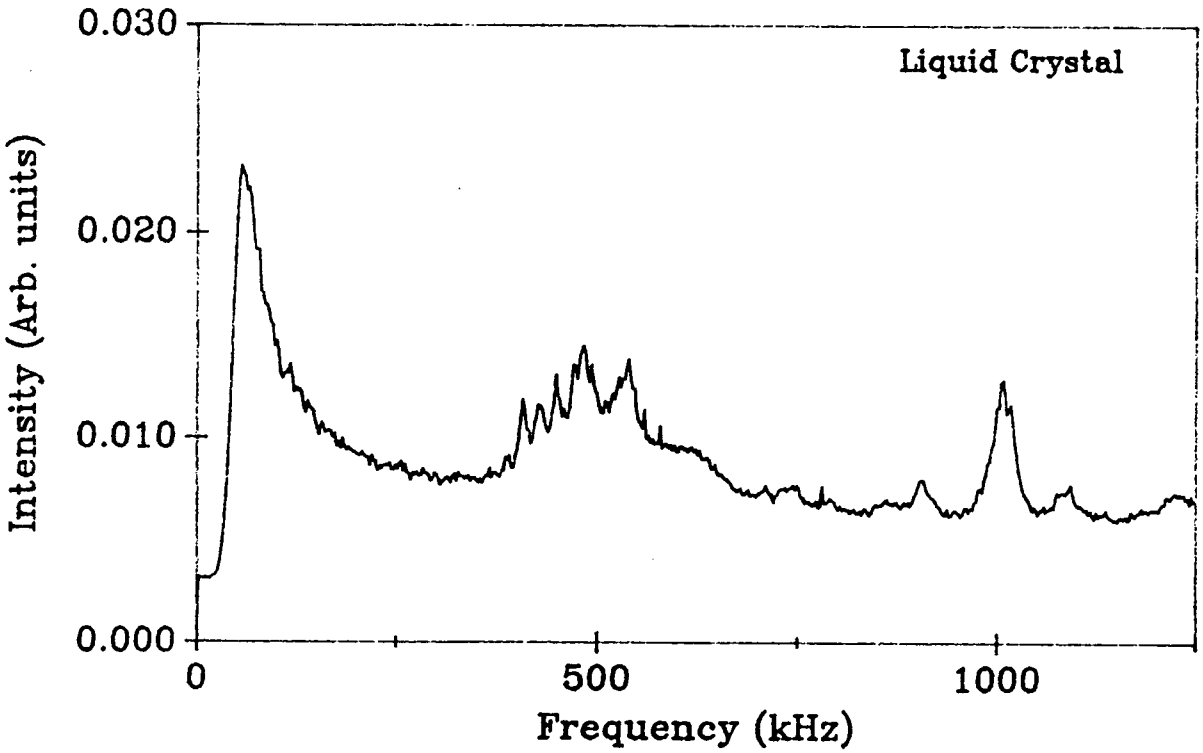


Figure 80) Average power spectrum of cooling liquid crystal 4-n-pentyloxybenzylidene-4'-n-heptylaniline.

of acoustic output versus time is given in Figure 77. There is an initial area of acoustic activity. This initial period is characterized by signals of high intensity compared to the remainder of the experiment. This would suggest that there are two different processes occurring. The activity of the sample after 8 minutes (as the sample was cooling through 67°C) would appear to be due to continuous emission as the signals all have similar RMS values. This is indicative of a sample which is continuously emitting. The initial activity is a series of individual bursts (Figure 78) as compared to the continuous signals captured after 8 minutes Figure 79. Quite noticeable are the signals which have an RMS value at a level above the normal RMS of the continuous emissions (Figure 77). These are due to some electrical noise carried on the line and can be distinguished by the AREA vs RMS plot as seen in Figure 31. Again, the average power spectrum in Figure 80 is different to that of NaOH, and to that of TRIMET.

Further work which may prove useful on this sample of liquid crystal (indeed, on any sample) would be to hold the temperature nearly constant to allow the sample to only cycle through one specific phase transition. This would provide signals specifically from each transformation and may lead to the ability to distinguish amongst them by AE. Improved temperature control is needed to achieve this.

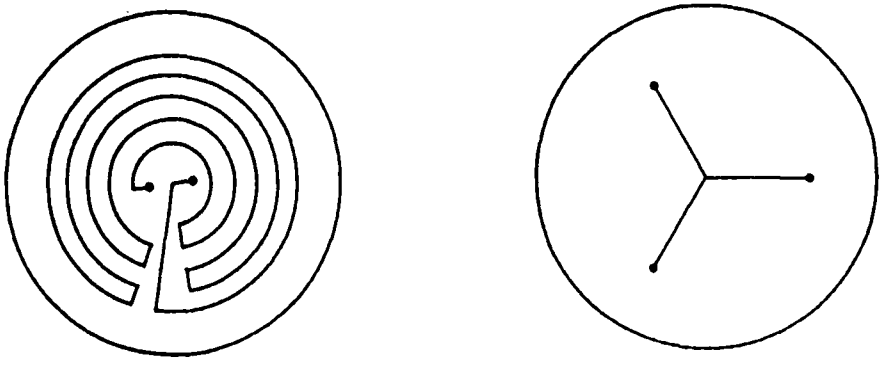
VIII. Further Work

The repeatability of time domain behavior of emission needs to be established. This can be done by ensuring consistent attachment of the transducer to the sample as well as using the same acoustic path (sample to transducer) for all replicate experiments and for those that the data is to be compared between systems.

More accurate quantitation of acoustic energy released by a sample is needed for determinations of total acoustic output. This could be accomplished through the use of a true RMS meter connected to the transducer (in series with the conditioning amplifier). The use of the DC (peak) output of the Bruel and Kjaer amplifier is not ideal for quantitative measurements of the total energy of the acoustic emissions from a sample.

The need for a transducer with linear response has been discussed. This will allow comparison of data taken with different transducers without having to correct for the variation in the frequency response. Also, one would wish to have a wider response range so that higher frequency signal components may be detected, recorded, and analyzed. A means to lower the acoustic detection limit must be pursued - especially when one considers how too high a trigger level can bias results[103].

A video camera may be used to capture the images of the acoustically active processes. In this way, the acoustic signals acquired as data may be related to an observable process that is occurring. It will also be of benefit to hyphenate acoustic emission with techniques such as infrared spectroscopy and nuclear magnetic resonance to correlate acoustic emission with changes in a system which are not directly observable by eye.



1.5 mm (depth and width) groove patterns on surface of two constructed bases. Ends of groove channels are connected to feed tubes through holes in the base by ¼ inch chromatography fittings.

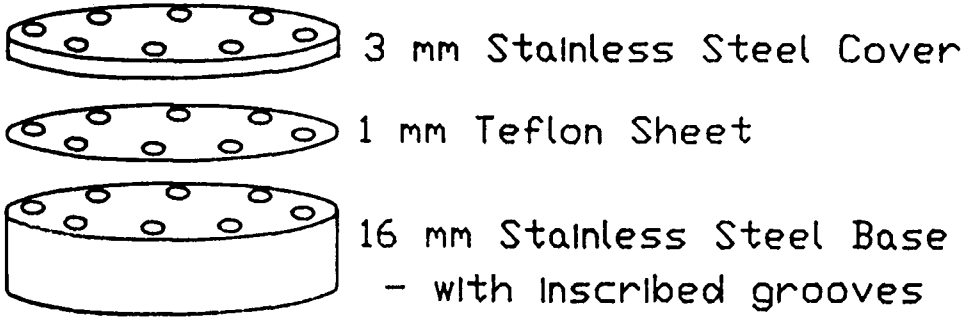


Figure 81) Acoustic flow cell designs. Two stainless steel disks - one with a flow channel inscribed in its surface - are fastened together.

The other principal research interest of Dr. Adrian Wade is the use of flow injection analysis (FIA)[50,104] for the characterization of chemical systems. It is thought that a hyphenated AE-FIA technique might prove useful (as in the $AlCl_3$ work). Towards this end, two flow cells have been designed and constructed in this department from aluminum (Figure 81). Initial experiments with 1.0 M HCl and 3 M Na_2CO_3 were unable to produce detectable acoustic activity. Further flow cells (Figure 82) have since been designed and constructed from glass (also in this department) and should be able to provide better sensitivity.

IX. Conclusions

An apparatus and nescessary data acquisition and analysis sortware has been designed to study chemical acoustic emission. It has been shown that acoustic emission data may readily be recorded from a number of chemical systems. A range of descriptors have been developed and code produced to generate and display these. The use of pattern recognition techniques for the analysis of acoustic emission data appears to be almost mandatory given the large amount and high dimensionality of data produced by even simple AE experiments. Link scaling is proposed and has been shown to be of utility. The use of a relative similarity axis for display of dendrograms is also proposed and appears to be useful in some circumstances[89]

Chemical acoustic emission proves to be a promising means of detecting chemical reaction. Differences in power spectra suggest it has significant analysical potential. Industry has already welcomed the idea of a non-invasive "microphone" on the outside of a reactor vessel. More people should open their ears and put more emphasis on the sounds of chemical reactions as well as the sights.

X. Bibliography

1. "Inaudible Screaming of Thirsty Tomatoes is Chemist's New Tool", (on work by S. Bittman and A. P. Wade), The Globe and Mail, Friday, March 17, 1989, p. A11.

2. Codex Germanicus, ca. 1350 A.D., quoted in J. Read, Prelude to Chemistry, p. 75, G. Bell: London, (1939).

3. F. Förster, E. Scheil, Zeitshrift fur Metalkunde, 24, (1936), 245.

4. J. Kaiser, Arkiv fur das Eisenhuttenwessen, 24, (1953), 43.

5. A. T. Green, C. S. Lockman, R. K. Steele, "Acoustic Verification of Structural Integrity of Polaris Chambers", Modern Plastics, 41, (1963), 137.

6. T. Holroyd, "Acoustic Emission from an Industrial Applications Viewpoint", Journal of Acoustic Emission, 7, (1988), 193.

7. J. A. Simmons, H. N. G. Wadley, "Theory of Acoustic Emission from Phase Transformations", Journal of Research of the National Bureau of Standards, 89, (1984), 55.

8. D. G. Eitzen, H. N. G. Wadley, "Acoustic Emission: Establishing the Fundamentals", Journal of Research of the National Bureau of Standards, 89, (1984), 75.

9. E. F. Carome, P. E. Parks, S.J. Meaz, "Propagation of Acoustic Transients in Water", Journal of the Acoustical Society of America, 36, (1964), 946.

10. C. Allan Boyles, Acoustic Waveguides, J. Wiley and Sons: New York, 1984.

11. I. G. Scott, C. M. Scala, "A Review of Non-destructive Testing of Composite Materials", NDT International, April 1982, 75.

12. D. Mool, R. Stephenson, "Ultrasonic Inspection of a Boron / Epoxy - Aluminum Composite Panel", Material Evaluation, 29, (1971), 159.

13. F. A. Firestone, J. R. Frederick, "Refinements in Supersonic Reflectoscopy. Polarized Sound", Journal of the Acoustical Society of America, 18, (1946), 200.

14. Journal of Acoustic Emission, Acoustic Emission Group, Los Angeles, California, U.S.A., 1982.

15. A. Nozue, T. Kishi, "An Acoustic Emission Study of the Intergranular Cracking of AISI 4340 Steel", Journal of Acoustic Emission, 1, (1982), 1.

16. J. W. McElroy, "Development of Acoustic Emission Testing for the Inspection of Gas Distribution Pipelines", Monitoring Structural Integrity by Acoustic Emission, ASTM STP 571, American society for Testing and Materials, 1975, pp. 59-79.

17. S. V. Hoa, L. Li, "Acoustic Emission During Quasi-Static Loading / Hold / Unloading in Notched Reinforced Fiber Composite Materials", Journal of Acoustic Emission, 7, (1988), 145.

18. G. Sauerbrey, "The Use of Quartz Oscillators for Weighing Thin Layers and for Microweighing", Zeitschrift fur Physik, 155, (1959), 206.

19. J. F. Alder, J. J. McCallum, "Piezoelectric Crystals for Mass and Chemical Measurements", Analyst, 108, (1983), 1169.

20. M. Thompson, G. K. Dhaliwal, C. L. Arthur, G. S. Calabrese, "The Potential of the Bulk Acoustic Wave Device as a Liquid-Phase Immunosensor", IEEE Transactions on Ultrasonics, Ferroelectrics, and Frequency Control, UFFC-34, (1987), 127.

21. F. G. Smith, P. A. Peach, "Apparatus for the Recording of Decrepitation in Minerals", Economics in Geology, 44, (1949), 449.

22. K. Lonvik, "Thermosonimetry", Thermochimica Acta, 110, (1987), 253.

23. G. M. Clark, M. Tonks, M. Tweed, "Thermal Properties of Potassium Dichromate", Journal of Thermal Analysis, 12, (1977), 23.

24. K. Lonvik, "An Experimental Investigation of the Thermal Decomposition of Brucite by Thermosonimetry", Thermochimica Acta, 27, (1978), 27.

25. E. Ranke Madsen, Ph. D. Thesis, University of Copenhagen, 1957.

26. L. M. Belyaev, V. V. Nabatov, Yu. N. Martyshev, "Luminescence Time in the Processes of Tribo- and Crystalloluminescence", Soviet Physics - Crystallography, 7, (1963), 464.

27. J. A. C. van Ooijen, E. van Tooren, J. Reedijk, "Acoustic Emission during the Preparation of Dichloro(pyrazine)zinc(II)", Journal of the American Chemical Society, 100, (1978), 5569.

28. D. Betteridge, M. T. Joslin, and T. Lilley, "Acoustic Emissions from Chemical Reactions", Analytical Chemistry, 53, (1981), 1064.

29. M. T. Joslin, "Analytical Implications of Acoustic Emissions from Chemical Reactions", Ph. D. Thesis, University of Wales, Swansea, Wales, June 1987.

30 R. M. Belchamber, D. Betteridge, P. Y. T. Chow, T. J. Sly, A. P. Wade, "Applications of Computers in Chemometrics and Analytical Chemistry", Analytica Chimica Acta, 150, (1983), 1292-1299.

31. T. Sawada, Y. Gohshi, "Acoustic Emissions Arising from the Gelation of Sodium Carbonate and Calcium Chloride", Analytical Chemistry, 57, (1985), 366.

32. T. Sawada, Y. Gohshi, C. Abe, K. Furaya, "Acoustic Emission from Phase Transition of Some Chemicals", Analytical Chemistry, 57, (1985), 1743.

33. R. M. Belchamber, D. Betteridge, M. P. Collins, T. Lilley, C. Z. Marczewski, A. P. Wade, "Quantitative Study of Acoustic Emission from a Model Chemical Process", Analytical Chemistry, 58, (1986), 1873.

34. O. Lee, Y. Koga, A. P. Wade, "Acoustic Emission Study of the Phase II/III Transformation of Hexachloroethane", Talanta, in press.

35. M. R. Detaevernier, Y. Michote, L. Buydens, M. P. Derde, M. Desmet, L. Kaufman, G. Musch, J. Smeyers-Verbeke, A. Thielemans, L Dryon, D. L. Massart, "Feasibility Study Concerning the Use of Expert Systems for the Development of Procedures in Pharmaceutical Analysis", Journal of Pharmaceutical & Biomedical Analysis, 4, 297, (1986).

36. J. G. Delly, "Sights and Sounds at 250X", The Microscope, 34, (1986), 63.

37. P. D. Wentzell, S. J. Vanslyke, A. P. Wade, "Programming Direct Memory Access Data Acquisition", Trends in Analytical Chemistry, 9, (1990), 3.

38. A. P. Wade, S. J. Vanslyke, P. D. Wentzell, "A Simple Acoustic Emission Monitoring System", Manuscript in preparation.

39. R. M. Belchamber, M. P. Collins, "Mill material physical property determining method - detects noise and converts into electrical signal that is processed by digital bandpass filter and analyzed.", British Petroleum Research Centre, U.S. Patent Application, WPI# 89-0953211/13, (1989).

40. R. M. Belchamber, M. P. Collins, "Machine acousic emission monitoring method - using pattern recognition procedure to develope soft models used as templates to detect fluctuations.", British Petroleum Research Centre, U.S. Patent Application, WPI# 89-152794/21, (1989).

41. P. D. Wentzell, A. P. Wade, "Chemical Acoustic Emission Analysis in the Frequency Domain", Analytical Chemistry, 61, (1989), 2638.

42. D. Betteridge, J. V. Cridland, T. Lilley, N. R. Shoko, M. E. A. Cudby, D. G. M. Wood, Polymer, 23, (1982), 178.

43. D. Betteridge, J. V. Cridland, T. Lilley, N. R. Shoko, M. E. A. Cudby, D. G. M. Wood, Polymer, 23, (1982), 249.

44. D. Betteridge, P. A. Connors, T. Lilley, N. R. Shoko, M. E. A. Cudby, D. G. M. Wood, "Analysis of Acoustic Emissions from Polymers", Polymer, 24, (1983), 1206.

45. T. Lilley, "A Study of Acoustic Emission from Polymers", Ph. D. Thesis, University of Wales, Swansea, Wales, December, 1980.

46. P. Chow, "A Pattern Recognition Study of Acoustic Emissions from Polymers under Stress", M. Sc. thesis, University of Wales, Swansea, Wales, March 1983.

47. S. J. Vanslyke, Chemistry Undergraduate Thesis, University of British Columbia, Canada, April 1989.

48. L. Seungho, M. N. Myers, R. Beckett, J. C. Giddings, "Particle Seperation and Characterization by Sedimentation / Cyclical-Field Field-Flow Fractionation", Analytical Chemistry, 60, (1988), 1129.

49. A. P. Wade, P. T. Palmer, K. J. Hart, C. G. Enke, "Developement of Algorithms for Automated Elucidation of Spectral Feature / Substructure relationships in Tandem Mass Spectroscopy", Analytica Chimica Acta, 215, (1988), 169.

50. P. M. Shiundu, P. D. Wentzell, A. P. Wade, "Spectrophotometric Determination of Palladium with Sulphochlorophendazorhodanine by Flow Injection", Talanta, 37, (1990), 329.

51. D. G. Gelderloos, K. L. Rowlen, J. W. Birks, J. P Avery, C. G. Enke, "Whole Column Detection Chromatography: Computer Simulations", Analytical Chemistry, 58, (1986), 900.

52. R. J. Woodham (U.B.C. Computer Science Department), B.C. Advanced Systems Institute Workshop: "Advanced Systems for Material Sensing and Process Control in the Forest Industry", PAPRICAN Vancouver Laboratories, Vancouver, November 14th, 1988.

53. R. E. Dessy, "Chemists in the Microelectronic Toolbox", 72nd CSC meeting, Victoria, Canada, 1989.

54. C. E. Shannon, "A Mathematical Theory of Communication", Bell Systems Technological Journal, 27, (1948), 379-423 and 623-656.

55. C. C. Sweeley, J. F. Holland, D. S. Towson, B. A. Chamberlin, "Interactive and Multisensory Analysis of Complex Mistures by an Automated Gas Chromatography System", Journal of Chromatography, 399, (1987), 173.

56. R. J. Williams, Biochemical Institute Studies IV : Individual Metabolic Patterns and Human Disease. "An Exploratory Study Utilizing Predominately Paper Chromatographic Methods", University of Texas Publication no. 5109, University of Texas: Austin, TX, (1951), ch. 1.

57. H. Chernoff, "The Use of Faces to Represent Points in k-Dimensional Space Graphically", Journal of the American Statistical Association, 68, (1973), 361.

58. K. Burton, "Cluster Analysis", European Spring School of Chemometrics, Elsevier, Amsterdam, 1988.

59. D. L. Massart, A. Dijkstra, L. Kaufman, Eds., Evaluation and Optimization of Laboratory Methods and Analytical Procedures, Elsevier: Amsterdam, 1978.

60. K. Burton, G. Nickless, "Optimisation via Simplex: Part I. Background, Definitions and a Simple Application", Chemometrics and Intelligent Laboratory Systems, 1, (1987), 135.

61. B. R. Kowalski, "Chemometrics", Analytical Chemistry, 52, (1980), 112R.

62. R. G. Brereton, "Chemometrics in Analytical Chemistry: A Review", Analyst, 112, (1987), 1635.

63. S. D. Brown, T. Q. Barker, R. J. Larivee, S. L. Monfre, H. R. Wilk, "Chemometrics", Analytical Chemistry, 60, (1988), 252R.

64. B. R. Kowalski, Ed., Chemometrics: Theory and Application, ACS Symposium Series 52, American Chemical Society: Washington, D. C., 1977.

65. D. L. Massart, L. Kaufman, The Interpretation of Analytical Chemical Data by the Use of Cluster Analysis, New York: Wiley, 1983.

66. M. A. Sharaf, D. L. Illman, B. R. Kowalski, Chemometrics, New York: Wiley, 1986.

67. S. N. Deming, S. L. Morgan, Experimental Design: A Chemometric Approach, Elsevier: Amsterdam, 1987.

68. R. M. Belchamber, D. Betteridge, Y. T. Chow, T. Lilley, M. E. A. Cudby, D. G. M. Wood, "Evaluation of Pattern Recognition Analysis of Acoustic Emission from Stressed Polymers and Composites", Journal of Acoustic Emission, 4, (1985), 71.

69. R. W. Y. Chan, D. R. Hay, V. Caron, M. Hone, R. D. Sharp, "Classification of Acoustic Emission Signals Generated During Welding", Journal of Acoustic Emission, 4, (1985), 115.

70. A. Maslouhi, C. Roy, "Analysis of AE Signals in Time and Frequency Domains Coupled to Pattern Recognition to Identify Fracture Mechanisms in CFRP", Journal of Acoustic Emission, 8, (1989), S292.

71. R. M. Belchamber, D. Betteridge, M. P. Collins, T. Lilley, C. Z. Marczewski, A. G. Hawkes, "Time Series Analysis of Acoustic Emission Signals from Glass Reinforced Plastics", Acoustic Emission Monitoring and Analysis in Manufacturing - PED-vol. 14, D. A. Dornfeld, Editor, The American Society of Mechanical Engineers: New York, USA, (1984),.

72. M. A. Majeed, C. R. L. Murthy, "An Efficient Unsupervised Pattern Recognition Procedure for Acoustic Emission Signal Analysis", Journal of Acoustic Emission, 8, (1989), S16.

73. R. O. Newman, Personal Communication, Bruel and Kjaer Canada, Richmond, British Columbia, Canada.

74 G. M. Clark, "Instrumentation for Thermosonimetry", Thermochimica Acta, 27, (1978), 19.

75. Bruel and Kjaer amplifier model 2638, technical manual.

76. O. Lee, P. D. Wentzell, D. A. Boyd, A. P. Wade, "Programming Control and Data Acquisition Routines for the IEEE-488 Instrumentation Interface", Trends in Analytical Chemistry, in press.

77. Tektronix 2230 Digital Storage Oscilloscope Operators, Tektronix, Beaverton, Oregon, U.S.A., 1987.

78. Tektronix 2430A Digital Oscilloscope Operators, Tektronix, Beaverton, Oregon, U.S.A., 1988.

79. GPIB-PC User Manual for the IBM Personal Computer and Compatibles, National Instruments Corporation, Austin, Texas, U.S.A., April 1988 Edition.

80. A Complete Guide to TRIMET Brand of Trimethylolethane, Pittman-Moore, Inc: Terra Haute, Illinois, 1988.

81-86. G. Camino, L. Costa, L. Trossarelli, "Study of the Mechanism of Intumescence in Fire Retardant Polymers", Polymer Degradation and Stability
Part I: 6, (1984), 243.
Part II: 7, (1984), 25.
Part III: 7, (1984), 221.
Part IV: 8, (1984), 13.
Part V: 12, (1985), 203.
Part VI: 12, 1985 (213).

87. D. B. Sibbald, P. D. Wentzell, O. Lee, I. H. Brock, K. A. Soulsbury and A. P. Wade, "An Integrated Data Acquisition and Analysis Environment for Chemical Acoustic Emission", manuscript in preparation.

88. P. D. Wentzell, D. B. Sibbald, D. A. Boyd, A. P. Wade, "Chemometric Methods for Acoustic Emission Analysis", 15th FACSS conference, Boston, 1988.

89. D. B. Sibbald, P. D. Wentzell, A. P. Wade, "Display Methods for Dendrograms", Trends in Analytical Chemistry, 8, (1989), 289.

90. D. B. Sibbald, P. D. Wentzell, D. A. Boyd, O. Lee, A. P. Wade, "Is Acoustic Emission Just Going through a Phase?", 15th FACCS conference, Boston, 1988.

91. R. M. Belchamber, D. Betteridge, Y. T. Chow, T. Lilley, M. E. A. Cudby, D. G. M. Wood, "Looking for Patterns in Acoustic Emissions", First International Symposium on Acoustic Emission from Reinforced Composites, The Society of the Plastics Industry, Inc, July 19-21, 1983, 1.

92. A. P. Wade, "Acoustic Emission: Is Industry Listening?", Chemometrics and Intelligent Laboratory Systems, in press.

93. Technical information on broadband sensors types FC500 and FAC500, Acoustic Emission Technology Corporation, Sacramento, California.

94. J. R. Mitchell, "Fundamentals of Acoustic Emission and Applications as an NDT Tool for FRP", 34th Annual Technical Conference, Reinforced Plastics / Composites Institute, The Society of the Plastics Industry, Inc., (1979), Section 3-F, p. 1.

95. T. M. Proctor Jr., "More Recent Improvements on the NBS Conical Transducer", Journal of Acoustic Emission, 5, (1986), 134.

96. Ono, Y. Higo, Progress in Acoustic Emission, 2, (1984), 343.

97. Y. Higo, H. Inaba, "The General Problems of AE Sensors", Journal of Acoustic Emission, 8, (1989), S24

98. H. Hatano, E. Mori, Journal of the Acoustical Society of America, 59, (1976), 344.

99. S. Kallara, P. K. Rajan, J. R. Houghton, "Acoustic Emission Transducer Modelling using System Identification Techniques", Journal of Acoustic Emission, 8, 1989, S28.

100. L. Brekhovskikh, Yu. Lysanov, Fundamentals of Ocean Acoustics, Springer-Verlag: Berlin, 1982, pp. 9-11.

101. T. J. Mason, J. P. Lorimer, "Sonochemistry : Theory, Applications and Uses of Ultrasound in Chemistry", Wiley: New York, 1988.

102 P. D. Wentzell, A. P. Wade, "A Comparison of Pattern Recognition Descriptors", submitted to Journal of Chemometrics, May 1990.

103. A. P. Wade, K. A. Soulsbury, P. Y. T. Chow, I. H. Brock, "Characterization of Chemical Acoustic Emission Near the Conventional Detection Limit", manuscript in preparation for Analytica Chimica Acta.

104. J. Ruzicka, E. H. Hansen, Flow Injection Analysis, 2nd. Ed., Wiley, New York, 1988.

XI.1 Data File Formats

The formats of the data files used have evolved over the course of this work - and indeed are perhaps due for a further change to address the need for greater than 8-bit resolution and longer record lengths. Authors involved in the design of the formats include everybody remotely involved with the acoustic emission experiments and data analysis. The formats have been designed for efficiency and for ease of utility. This work uses the IBM-PC class of computers which run under MicroSoft DOS 3.30 (MicroSoft, Redmond, Washington). Each data file has an eight character name with a specific three character extension corresponding to its type. A file is normally referred to by its type - for example - a .EXT file.

The files used in this work include:

i) .AEA This file is created by the QAQ program and includes the experimental data - signals, times, operating parameters, user comments, etc..

ii) .DS1 This descriptor file contains the descriptors for each of the signals contained in the .AEA file of the same name and is calculated by the AEMUNCH program. (For example, the file DATA.DS1 would be the descriptor file for the experiment file DATA.AEA.) Other versions of the descriptor file are the .DES file, and a .DS2 file.

iii) .DEN This is an ASCII file which contains the linking structure of the dendrogram. It is calculated from the descriptor file by the DENDGRAM program.

iv) .AF2 This ASCII file contains the factor loadings and scores of a principal components analysis of a descriptor file. This file is calculated by the ABSCAT program - which also outputs a results file, .RES, that contains information on the significance of the factors.

XI.1.1 .AEA - Acoustic Emission Experiment Data File

The AEA files are random access files made up of two blocks: a single HEADER block and many SIGNAL blocks. The purpose of the header block is to record general information about the experiment which is common to every signal in the signal block. The header block occupies a minimum of 1040 bytes and is logically organized into 80 byte records, plus a 4 byte record for every extended field present. The signal block contains the digitized signal obtained from the digitization device. This signal is stored as 1024 unsigned integers, thus occupying 1024 bytes. A 16 byte preamble stores important information particular to that signal - id, class, time of acquisition and room for an extra four-byte record. Appended to each signal may be extended records containing experimental data such as temperature, pressure, force, etc..

Field	Field Description	Bytes	Notes
1	Experiment Title	40	
	UNUSED	2	
	Date	10	"MM-DD-YYYY"
	UNUSED	2	
	Sampling Mode	6	"MODE=n" see NOTE (1)
	UNUSED	2	
	Scope ID	18	"T2430Axxxxxxxxxxxx"
2	Comments (1)	80	user comments
3	Comments (2)	80	
4	Transducer Serial #	20	"S/N=xxxxxxxxxxxxxxxx"
	UNUSED	60	
5	Time Per 100 Points	20	"T/D=xxxxxxxxxxxxxxxx"
	Volts per Division	20	"V/D=xxxxxxxxxxxxxxxx"
	Gain	20	"GAIN=xxxxxxxxxxxxxxx"
	Filter Setting	20	"FILTER=xxxxxxxxxxxxx"
6	Trigger Level	20	"TLEV=xxxxxxxxxxxxxxx"
	Pre-Trigger Count	20	"PRET=xxxxxxxxxxxxxxx"
	Sampling Delay Time	20	"DLY=xxxxxxxxxxxxxxxx"
	Extra Field	20	

7 - 12	UNUSED	80	each for total 480 bytes
13	UNUSED	70	
	# of Extended Fields	4	(NEXD$) see NOTE (2)
	spaces	2	
	Extra Field Name	3	
	Extra Field Status byte	1	(EFSB)
14+*	All remaining records are composed of 4 byte "EXTENDED" fields according to VAL(NEXD$). Each is composed of a 3 byte name followed by the status byte for that extended record field (EXSB).		

Signal Blocks

Field	Field Description	Bytes	Field Type / Notes
1	Signal ID	4	STRING
2	Signal Class	4	STRING
3	Time of Signal (sec)	4	SINGLE PRECISION REAL
4	Extra Field Record	4	
5	Digitized Signal	1024	
6+*	Extended Field(s)	4	see NOTE (3)

\+ There may be consecutive multiple occurrences of these fields depending on the number needed in the file.

* These fields may not be present depending on the status of the extra record and the number of extended records.

Note (1) Sampling Modes: 1=Level Triggered; 2=Continuous Trigger; 3=External #1; 4=External #2

Note (2) This string (NEXD$) represents the number of EXTENDED FIELDS. These additional fields are to store additional experimental data particular to an experiment.

Note (3) Beginning of the EXTENDED fields. The number of extended fields is equal to VAL(NEXD$). NEXD$ is found in Record #13 of the Header Block. Each extended field is 4 bytes of the type designated by the status byte in the Header Block.

EFSB: 0 0 0 0 0 0 0 0
0 0 1 0 0 X X Y

Y = 0 if the field is not in use.
= 1 if the field is in use.

XX	field contains data of type	Placement in field	CHR$(EFSB)
00	STRING	"ABCD"	!
01	SINGLE PRECISION REAL	"xxxx"	#
10	INTEGER	"00xx"	%
11	LONG INTEGER	"xxxx"	'

XI.1.2 .DS1 - Descriptor Files

The descriptor files are produced by the AEMUNCH program. They contain the descriptors calculated from each signal in the .AEA file and are stored in random access format in a .DS1 file. The identification, class, time and any extended or axtra records for each signal are stored. There are currently three formats for the descriptor files. The original .DES file, which is a simple ASCII file output by the original descriptor file generating programs, PATCHAR and AECRUNCH, the binary .DS1 file - which contains all the signal information from the .AEA file - and the .DS2 file - an ASCII version of the .DS1 file.

The binary descriptor file (.DS1) consists of 4-byte records and has the following format:

Header Block			
Field	Field Description	Bytes	Field Type / Notes
1	Number of samples	2	NROWS
2	# of variables	2	NCOLS
3	# of extended records	2	NEXD$
4	Status byte for extra record	1	EFSB (0 if not used,1 if used)
5*	Name of extra record field	4	EN$, 3 chars + a space
6+*	Name of extended record	3	EXN$
	UNUSED	4	Blanks for future use
	Extended field status byte	1	EXSB
			This 8 byte record is present once for each extended record (NEXD) "EXN...x"
7+	Descriptor names	8	VARNM$. NCOLS of these for each descriptor

\+ There may be consecutive multiple occurrences of these fields depending on the number needed in the file.

* These fields may not be present depending on the status of the extra record and the number of extended records.

Signal Block - repeated for each signal (NROWS)			
Field	Field Description	Bytes	Field Type / Notes
1	Signal identification	4	ID$ - Character string
2	Signal classification	4	CLSS$ - Character string
3	Time of signal acquisition	4	TIME - Single precision "00xx"
4	Extra record	4	EXTRA$
5+*	Extended record	4	
6+	VAR!	4	NCOLS descriptors single precision real numbers

\+ There may be consecutive multiple occurrences of these fields depending on the number needed in the file.

* These fields may not be present depending on the status of the extra record and the number of extended records.

For the random access of the .DS1 file, the following equations have been calculated. For brevity, the parameters have been shortened for placement in equations.

e = NEXD - number of extended records
y = EFSB
n = NROWS - number of signals
m = NCOLS - number of descriptors

The capitalized parameters refer to the specific element being referred to.

A = signal number (1 to n)
D = descriptor number (1 to m)
T = extended record number (1 to e)

Records are four bytes long, note the difference between records and fields. Some fields take more than one four byte record.

Record	Description
1	Contains the number of signals (n) and number of descriptors per signal (m). Packed as 2-byte integers into the 4-byte records.
2	2-byte integer # of extended records (e), 1-byte (AO) analysis options, Status byte for extra record (x)
3	If (x) < > 32 then a record is present containing three characters for the name of the extra record (EN$) and a zero-byte.

Record	Description
4 & 5	These two records appear once for each extended record (e). Three character bytes, 4 blank bytes and a status byte (EXSB). If (e) = 0 then these records are not present.
2e+y+2 & 2e+y+3	Descriptor name for first descriptor. Maximum of 8 characters packed into two 4-byte records.[y=1 if x < > 32,y=0 otherwise]
2e+y+4 & 2e+y+5	8 character descriptor name for second descriptor.
.	
2m+2e+y+2	Identifier for signal 1 (4 characters).
2m+2e+y+3	Class for signal 1 (4 characters).
2m+2e+y+4	Time for signal 1 (single precision)
2m+2e+y+5	Extra record for signal 1 (4 characters)
2m+2e+y+6	First extended record for signal 1 (Only if (e) < > 0)
.	
2m+3e+y+6	First descriptor value for signal 1 (single precision)
2m+3e+y+7	Second descriptor for signal 1 (single precision)
.	
3m+3e+y+6	Identifier for signal 2 (4 characters)
.	
.	
(1+A)e+y+(A+1)m+4(A-1)+2	Identifier for signal A
(1+A)e+y+(A+1)m+4(A-1)+3	Class for signal A
(1+A)e+y+(A+1)m+4(A-1)+4	Time for signal A
(1+A)e+y+(A+1)m+4(A-1)+5	Extra record for signal A
(1+A)e+y+(A+1)m+4(A-1)+5+T	Extended record T for signal A
(2+A)e+y+(A+1)m+4(A-1)+5+D	Descriptor D for signal A
2e+y+(A+1)m+4*(A-1)+(A-1)n+2	Identifier for signal A

There are three possible ASCII file formats. The .DES and .SCL types are consistent with the original descriptor file format, and have the form,

```
(no. of signals - NROWS),     (no. of descriptors - NCOLS)
(name of descriptor 1)
(name of descriptor 2)
(name of descriptor 3)
.
.
(name of descriptor NVARS)
(signal 1 ID), (signal 1 class), (descriptor 1),    (descriptor 2),    ...
(signal 2 ID), (signal 2 class), (descriptor 1),    (descriptor 2),    ...

(signal NCOLS ID),               (descriptor 1),    (descriptor 2),    ...
```

All of the string variables (descriptor names, ID's, and class) are in double quotes. Note that this format does not facilitate the use of time and extra records. There is also no additional record at the beginning. If this type of file is converted to the .DS1 format, the second record in the .DS1 file is written as all zeros, the time records are written as floating point zeros, and the extra record consists of all spaces.

The other ASCII format is the .DS2 file. This file is essentially the same as the .DES file except that (1) the second record of the .DS1 file appears as two integers on the second line of the .DS2 file, and (2) the time and extra record are inserted after the class for each signal, the latter being written as a string. The .DS2 files are complete and can be converted back to the .DS1 files without loss of information. Note that the ASCII files can be edited and in fact that is one of their intended uses. Great care should be taken in doing this, however, as inconsistencies in file format can cause problems.

XI.1.3 .DEN - Dendrogram Data File

The .DEN file contains the dendrogram structure calculated by DENDGRAM. At the front of the file is the number of signals and the number of descriptors used - along with their names. The file then has three sections. The first contains the linkages needed to construct the dendrogram. A pair of signals is listed along with a dissimilariy (distance). A cluster is referred to by the number of the signal of one of its members. The second section contains the list of signals along with the ID, and CLASS designations. The last section contains information for the user reading the file. This includes contains the name of the original descriptor file, the scaling method used, the method for calculating the dendrogram, and the date and time of calculation.

The .DEN file is in ASCII format and has the following structure.

(Number of signals - NROWS), (Number of descriptors - NCOLS)
(Descriptor name #1)
.
.
(Descriptor name #NCOLS)
(Signal number), (Signal number), (Dissimilarity of linkage)
.
(There will be NROWS-1 linkages)
.
(Signal number), (Signal number), (Dissimilarity of linkage)
(Maximum linkage dissimilarity.@)
(Signal number), (Signal ID), (Signal CLASS)
.
.
.
(Signal number), (Signal ID), (Signal CLASS)
(This is followed by the scaling used, the method of calculation and the name of the original data file.)

@ This is the dissimilarity (distance) between the two most dissimilar signals.

The signals are listed in order that they need to be plotted in. This will not nescessarily (indeed rarely) be the order in which they appear in the .DS1 file. This order is calculated during the DENDGRAM calculation and ensures that the signals are listed in an order for which a dendrogram can be drawn.

As an example, this is the .DEN file used to generate Figure 17.

```
5,2
"x"
"y"
2       1       1.044031
4       3       1.726268
5       4       2.022375
2       4       3.623534
6.382006
2,      "       ",      "   a"
1,      "       ",      "   a"
4,      "       ",      "   a"
3,      "       ",      "   a"
5,      "       ",      "   a"
"No Scaling"
"Single Linkage"
Original Data File =FIG17.DES
Start time      19:18:47
Finished at :  19:18:49
04-12-1990
```

XI1.4 .AF2 - Abstract Factor Analysis Output File

These contain the results on the factor analysis of a descriptor file. There are currently two versions in existence. The original .AFA file and the newer used .AF2 file. They both have the same format and are written as sequential ASCII files. The eigenvalues and eigenvectors appear first. These are followed by the loadings matrix (in the same format as a DES file - ie. ID, CLASS, followed by the NCOLS loadings for that vector). The .AFA file was the first version used for AFA results. When later version of the descriptor file (.DS1) were developed, it was thought that the factor analysis results should also contain the extra information available - namely the time of the signal and any extra or extended records. The .AF2 file then contains TIME, EXTRA record and any extended records so that a .DS1 file created from this (by the ABSCAT program) is complete.

The .AF2 file has the following format for a data set which uses no extended records. If these are present, they are included after the extra record field (ie. before the projected coordinates). The .AFA file is exactly the same except for the second line of the file (NEXD, EFSB), the name of the extra record, and the extra record and the time fields.

NROWS,	NCOLS		
NEXD,	EFSB		
EXN$			
EXNM$	[only if NEXD >0 -NEXD times.]		
(The following information is present for each eigenvector - NCOLS times.)			
EVAL(1)	[first eigenvalue]		
EVEC(1,1),	EVEC(1,2),	....[linear vector containing first e-vector]	
.(The following block contains the projection of the original descriptor file onto .the factor space defined by the eigenvectors.)			
ID,	CLASS,	TIME,	Extra record
C(1),	..,	C(NCOLS)	NCOLS coordinates of signal projected onto factor space

The .AF2 file also contains a trailing line containing the names of all the descriptors used from the original DEScriptor file.

The process of calculating a set of basis vectors leads to a set of eigenvalues. The eigenvalues can be used to determine the significance of the factors. This information is available as a .RES file which is saved with the same name as the .AF2 file. This ASCII file is in column format with the parameter names at the top of the column and each row corresponding to the values for the parameters for each eigenvector. The file can be viewed from DOS by the TYPE command as it is formatted for the 80 column screen display, or it can be viewed from within ABSCAT. (A sample .RES file is included as Table 8).

XI.2 QAQ - Data Acquisition from Digital Storage Oscilloscope

The QAQ program was written in MicroSoft's QuickBASIC 4.0 by D B. Sibbald. Authors contributing to this program include D. A. Boyd (who helped write the original data acquisition program - DATADUMP - in GWBASIC), K. A. Soulsbury and O. Lee (who helped with the addition of the code for the model T2430A), and P. D. Wentzell (who wished for the absence of colour in the screen display). The operation of the program requires a TEKTRONIX Digital Storage Oscilloscope (either model T2230 or T2430A) which is interfaced via an IEEE-488 parallel interface.

Use of the program is straight-forward and is outlined in the following flowchart. The program first looks for a oscilloscope connected as a device on the general purpose interface board (GPIB). Having found one, it the initializes the scope to default settings. The experimental details are asked for from the user. These include experiment title, a brief description of the experiment, and information on how data is to be collected. The scope can be run in triggered mode - where the scope waits for a

signal with amplitude greater than a preset trigger level - or in continuous mode - where the scope is continuously being polled regardless of amplitude. The program is also capable of adding an acquisition delay in between the acquisition of each signal. This is made available for fast emitting systems and for those which emit for a lengthy period of time causing large data storage requirements.

The oscilloscope is then polled for its settings. The user is informed if any of the scope settings are incorrect for the intended mode of operation and is instructed on the correct settings. Once the scope is set up properly, the time-per-division (TDIV) and Volts-per-division (VDIV) settings are read. These parameters will be required to convert the downloaded signals from a string of 8-bit (0-255) integers to a Voltage-time waveform. The user information and the operating parameters are written to an .AEA file. The acquisition of data then follows.

The operation of the scope differs slightly depending on the mode of data acquisition selected. In triggered mode, the scope is armed (via the IEEE-488 interface) and the computer waits for an interrupt from the scope which signifies that a signal has been acquired. The signal is then downloaded from the oscilloscope and is written to the file along with the signal ID, class (same for all signals in an experiment), and the time of acquisition. If an acquisition delay hs been selected, the computer then pauses the specified number of seconds before sending the command to the scope to be armed. This process is repeated until a specified number of signals have been acquired or until the user terminates the experiment manually. The continuous mode of acquisition is similar. The exception being that the scope is not armed but instead runs in untriggered mode. The computer requests that the current contents of the scope's buffer be downloaded as often as possible, or as often as the user specified acquisition delay dictates.

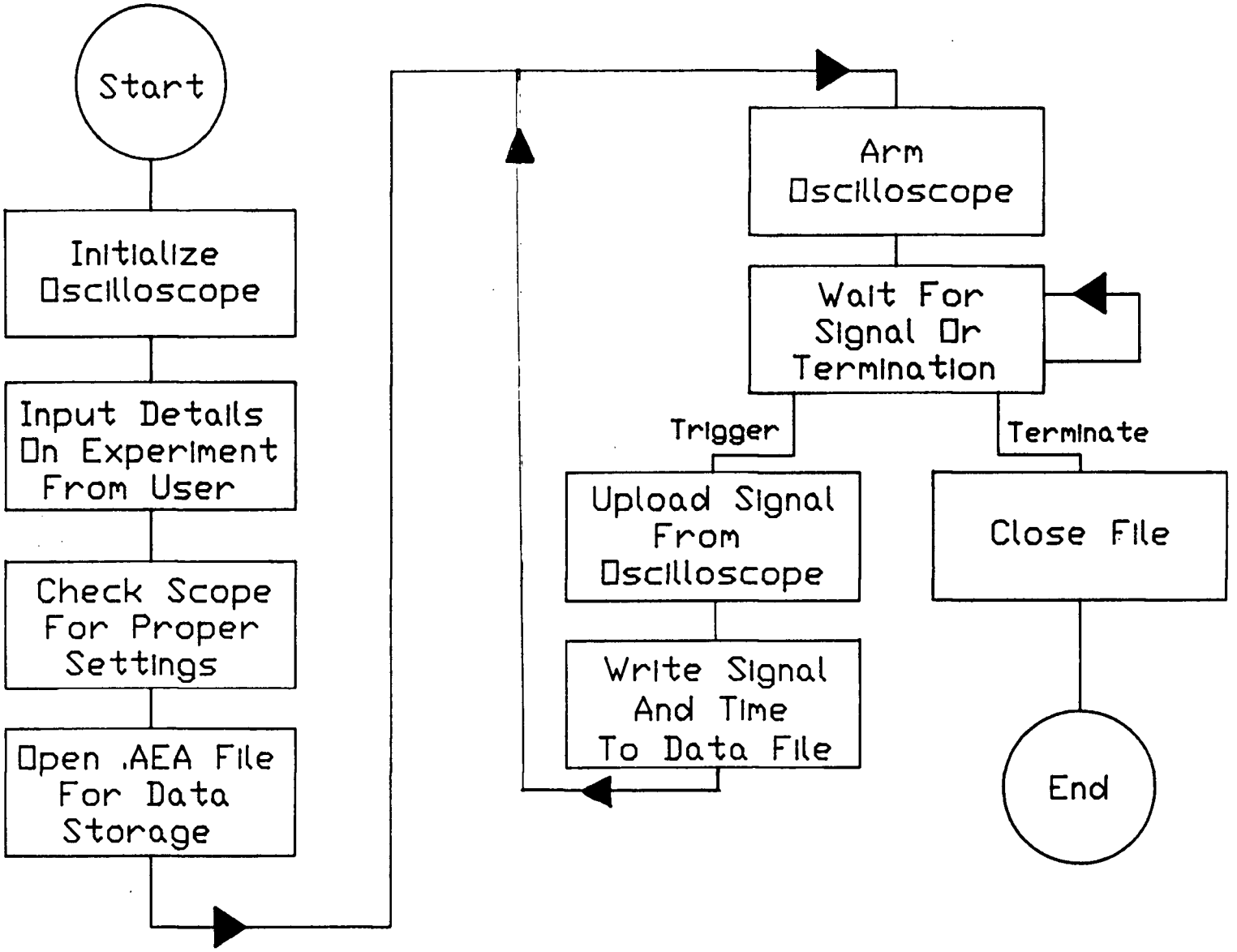


Figure 82) Data acquisition algorithm for QAQ program.

```
'=========================================================================
'     NEWQAQ  - QAQ for MICROSOFT QUICKBasic
'
'      Written by David Sibbald
'      Recoded from the GWBASIC program written by D. Sibbald and Tony
'      Boyd.  September, 1988
'
'      Revision 2.0  - April 1989
'      Written by David Sibbald & Kevin Soulsbury
'      .AEA files produced directly
'      Alterations to headerout to conform to current format
'      Additional sound features
'      E drive now not needed
'
'      Laboratory for Automated Chemical Analysis
'      Department of Chemistry
'      University of British Columbia
'      2036 Main Mall, Vancouver, B.C.
'=========================================================================
'
' NEWQAQ - Data AcQuisition from the TEK 2230 Oscilloscope
'
' Hardware required :
' 1) TEKTRONIX 100MHz Digital Storage Oscilloscope
' 2) GPIB interface.
'
' OPERATION
' =========
' The program first removes the MOUSE drivers from memory. It is assumed
' that the mouse control programs are stored on the C: drive in the
' \MOUSE1 subdirectory. If the location of these drivers is different,
' then the program must be modified. If NO mouse drivers are present,
' then there is no need to worry. The mouse drivers interfere with the
' PEN command and need to be disabled for correct operation of this
' program.
'
' The scope is then polled for it's status. If it is not present, then an
' error is reported. Possible causes are : scope turned off, scope not
```



```
' connected, computer not configured for scope/GPIB drivers
' NOTE : The drivers used for SIGMAPLOT and GPIB are not compatible.
'
' The USER is then bombarded with questions relating to the experiment.
' The amplification settings and filtering options are requested.
' The intended sampling mode is asked for which will affect the required
' scope settings.
'
' The scope is checked for the settings that are relevent to the
' acquisition of signals. The user is informed of any that need
' resetting.
'
' The actual acquisition routines are as follows :
'
' SET scope to assert SRQ when a signal is acquired (for triggered mode)
' The waveform is then downloaded and the SCOPE is armed (for triggered
' mode) If a delay is requested, this is done here first.
'
' Now the signals (maximum 9999) are written directly to the file
' specified as they are processed.
'
' VARIABLES
' =========
' TIMES#() - A linear array containing the TIMER value of each signal
' TEMPS!() - "-------------------------" temperature read from the ADC
' COMMENT$ - An array containing two lines of user comments and a line
'            with the trigger mode.
' WRT$     - T2230 is the identity of the scope for the GPIB interface.
' BD%      - The device number that is assigned to the scope by the GPIB
'            interface (IBFIND 'creates' the actual value)
' MESS$()  - An array for passage to the routine MESSAGE. This array is
'            used to print messages on the screen of the oscilloscope.
' FILT$()  - An array containing all the valid filter settings on the
'            conditioning amplifier.
' FILTER$  - Contains the chosen setting (input by user)
' REP$     - Used to contain the reply from the scope for certain
'            queries.
' EXTITLE$ - The title of the experiment - input by user
```

```
' AMP%      - An integer referring to the amplification setting (user
'             input)
' FROOT$    - The name of the file root. 1 to 5 characters in length. All
'             files are stored with this name. Signals are stored with a
'             three digit number appended to FROOT$ (eg. NAME001.AEB)
' SAMPMODE$- A two digit string referring to the sampling mode.
'             The first letter is a T for time-delay OR a C for
'             continuous.  The second letter is a T for triggered or a C
'             for untriggered.
' DELAY!    - The number of seconds to pause between each acquisition in
'             Time - delayed acquisition mode.
' TIMDIV$   - The seconds per division. The scope sends it back as a
'             string in engineering notation.( eg 50E+3 for 50ms per
'             division)
  VOLDIV$   - The volts per division setting. This is sent in funny form.
' COUNT%    - The current signal number (IE. the number of the NEXT signal
'             to be acquired.
' INITIAL#  - The TIMER reading when the acquisition is started. It is
'             possible that the first signal will be taken at 0.0 seconds.
' DRIVE$    - A two character string representing the drive in use. (D:)
'
' GPIB SUBROUTINES
' ================
' IBFIND    - Opens a device for communication.
' IBWRT     - Writes a string to the device.
' IBRD      - Inputs a string from the device output buffer. The string
'             parameter must be a buffer containing enough space for the
'             device output.
'
' GPIB VARIABLES
' ==============
' IBSTA%    - Returns TRUE (-1) if the IBFIND function fails
'
' DECLARATIONS
' ============
' Common GPIB status variables
       COMMON SHARED ibsta%, IBERR%, ibcnt%
' GPIB Subroutine Declarations
```



```
       DECLARE SUB IBFIND (BDNAME$, bd%)
       DECLARE SUB ibrd (bd%, RD$)
       DECLARE SUB IBRDF (bd%, FLNAME$)
       DECLARE SUB ibwrt (bd%, wrt$)
       DECLARE SUB IBWRTF (bd%, FLNAME$)
' QuickBASIC Subroutine Declarations
       DECLARE SUB CHECKSCOPE (bd%, sampmode$)
       DECLARE SUB GETDTAILS (extitle$, COMMENT$(), AMP%, FILT$(),
               froot$, sampmode$, sennum$, class$)
       DECLARE SUB getdrive (default$, froot$, drive$, dir$)
       DECLARE SUB GETTEMP (TEMP!)
       DECLARE SUB headerout (COMMENT$(), delay, sensornum$)
       DECLARE SUB MESSAGE (bd%, SOURCE$(), NUM%)
       DECLARE SUB SCOPE (bd%, mes$, REP%, REP$)
       DECLARE SUB TTLE ()

       CALL TTLE
       DIM TEMPS!(2000)                    ' stores the temperature
       DIM COMMENT$(1 TO 3)
       CONST wrt$ = "T2230"                ' identity of scope for IBFIND
       DIM MESS$(10)
       DATA 5
       DATA  "15Hello!","13Welcome to QAQ   by  DAVID SIBBALD"
       DATA "12                  and TONY BOYD"
       DATA " 9Please follow the instructions"," 8given by the
              computer."
       READ nummes%
       FOR I% = 1 TO nummes%
         READ MESS$(I%)
       NEXT

       DATA 6
       DATA "Linear","0.1 Hz - 10kHz","50kHz - 2MHz","100kHz - 2MHz"
       DATA "200kHz - 2MHz","400kHz - 2MHz"
       READ numfilts%
       DIM FILT$(0 TO numfilts%)
       FOR I% = 1 TO numfilts%
         READ FILT$(I%)
```

```
NEXT
CALL IBFIND(wrt$, bd%)              ' open scope for communication
REP$ = SPACE$(80)
CALL SCOPE(bd%, "STA?", 0, REP$)        ' check status of scope
IF ibsta% < 0 THEN 4000                 ' ERROR
' Now that we have established communication with the scope, let's
' clear out the MOUSE controllers that interfere with the PEN commands.
SHELL "c:\mouse1\cpanel off  > kill.fil"
SHELL "c:\mouse1\mouse off  > kill.fil"
SHELL "erase kill.fil"
' This will remove the mouse drivers from memory. Please remember to
' place them back or Oliver will be disconcerted.

CALL SCOPE(bd%, "INI", 0, REP$)     ' set scope to initial status
COLOR 14
CALL MESSAGE(bd%, MESS$(), 0)           ' clears scope
CALL MESSAGE(bd%, MESS$(), 5)           ' print MESS$()
CALL GETDTAILS(extitle$, COMMENT$(), AMP%, FILT$(), froot$,
  sampmode$, sennum$, class$)
FILTER$ = FILT$(0)
ERASE FILT$

IF MID$(sampmode$, 2, 1) = "T" THEN
  delay! = VAL(RIGHT$(sampmode$, LEN(sampmode$) - 2))
  sampmode$ = LEFT$(sampmode$, 2)
  delay% = -1
END IF
IF sampmode$ = "CC" AND delay! = 0 THEN
  delay% = -1
  delay! = .5
END IF
CLS
LOCATE 5, 1
PRINT " Please set the following on the scope : -"
LOCATE 7, 1
IF RIGHT$(sampmode$, 1) = "T" THEN
  PRINT "1) Trigger Mode : Single Sweep Mode"
ELSE
```



```
  PRINT "1) Trigger Mode : P.P. AUTO"
END IF
PRINT "2) Vertical Mode : Channel 1 "
PRINT "3) Horizontal Mode : A"
PRINT "4) STORE Mode"
PRINT "5) 1 K Acquisition"
PRINT "6) Please make sure that the SEC/DIV and the VOLTS/DIV are
  in the CAL detents."
PRINT "7) HF REJECT OFF"
INPUT a$

' check the BOZO's new scope settings
CALL CHECKSCOPE(bd%, sampmode$)
CALL MESSAGE(bd%, MESS$(), 0)
' now we have the scope set up for the experiment

LOCATE 2, 15: PRINT "   WELCOME TO 'NEWQAQ' - PLEASE BUCKLE UP!"
LOCATE 3, 15: PRINT "   ****************************************"
COLOR 9
LOCATE 5, 10: PRINT "EXPT. TITLE: "; 'EXTITLE$
LOCATE 7, 10: PRINT "DATE: "; 'DATE$
LOCATE 14, 5: PRINT "Time Scale  = "; 'TIMDIV$
LOCATE 14, 35: PRINT "Volts / DIV   = "; 'VOLDIV$
LOCATE 18, 5: PRINT "Filter      = "; 'FILTER$
LOCATE 18, 35: PRINT "Amplification = "; 'AMP%;" dB"
LOCATE 18, 56: PRINT "dB"
LOCATE 9, 10: PRINT "FILE DESTINATION: "; 'FROOT$;".AEA"
LOCATE 10, 18: PRINT "SIGNAL #:";
LOCATE 20, 35: PRINT "MODE : ";
COLOR 13
LOCATE 12, 24: PRINT "Scope settings :"
LOCATE 13, 24: PRINT "================"
LOCATE 16, 20: PRINT "Conditioning Amplifier :"
LOCATE 17, 20: PRINT "========================"
COLOR 11
LOCATE 5, 23: PRINT extitle$
LOCATE 7, 18: PRINT DATE$
LOCATE 18, 51: PRINT AMP%
```

```
        LOCATE 9, 28: PRINT froot$; ".AEA"
        LOCATE 10, 27: PRINT USING "####"; 1;
        LOCATE 18, 19: PRINT FILTER$
        LOCATE 20, 42
        SELECT CASE sampmode$
        CASE "TT"
          PRINT "Triggered (Delay"; delay!; " sec)"
        CASE "CT"
          PRINT "Untriggered (Delay"; delay!; "sec)"
        CASE "TC"
          PRINT "Triggered (Continuous)"
        CASE "CC"
          PRINT "Constant Sampling"
        END SELECT
        COLOR 14
        LOCATE 21, 14: PRINT "PRESS F10 TO STOP "
        LOCATE 24, 1: COLOR 15
        FOR I% = 0 TO 1000 STEP 100
          SOUND 1000 + I%, (I% + 20) / 100
        NEXT I%
        PRINT" Buckle up, Hold on to your hat ! Press return to start !";
        INPUT ; "", a$
        LOCATE 24, 1
        PRINT SPACE$(75);

' Get Voltage scale and Time scale from scope
        TIMDIV$ = SPACE$(30)
        CALL SCOPE(bd%, "HOR? ASE", -1, TIMDIV$)        ' Get A SEC/DIV
        PS% = INSTR(TIMDIV$, ":")
        LN% = INSTR(TIMDIV$, ";")
        TIMDIV$ = MID$(TIMDIV$, PS% + 1, LN% - PS% - 1)
        VOLDIV$ = SPACE$(30)
        CALL SCOPE(bd%, "CH1? VOL", -1, VOLDIV$)        ' Get VOLTS/DIV
        PS% = INSTR(VOLDIV$, ":")
        LN% = INSTR(VOLDIV$, ";")
        VOLDIV$ = MID$(VOLDIV$, PS% + 1, LN% - PS% - 1)
' It is not nescessary to check for HMAG ON (x10 on SEC/DIV setting on
' the scope because although the display on the scope shows smaller time
```



```
' scale, the 1K signal from the scope is collected at the HOR ASE rate

        count% = 1
        COLOR 11
        LOCATE 14, 20: PRINT TIMDIV$
        LOCATE 14, 51: PRINT VOLDIV$
        count% = 1
        ON KEY(10) GOSUB 2270
        KEY(10) STOP
        signal$ = SPACE$(1037)
        OPEN drive$ + froot$ + ".aea" FOR OUTPUT AS #1
        CALL headerout(COMMENT$(), delay, sennum$)
        SELECT CASE sampmode$
' This is the section that deals with Continuous sampling
' and also time-delay sampling of triggered signals.  A trigger is
' detected by the scope and SRQ is asserted. The computer can't detect an
' SRQ event BUT the SRQ assertion sets the PEN flag. So we need to
' check for PEN events to determine if a trigger has occurred.
        CASE "TC", "TT"
          CALL MESSAGE(bd%, MESS$(), 0)
          COLOR 13
          LOCATE 23, 10: PRINT "CLEARING SCOPE"
          FOR I% = 1 TO 2000: NEXT
          CALL SCOPE(bd%, "MES 0", 0, REP$)
          CALL SCOPE(bd%, "OPC ON", 0, REP$)
          CALL SCOPE(bd%, "EVE?", -1, REP$)    ' Clear first 2 SRQ events
          CALL SCOPE(bd%, "EVE?", -1, REP$)     ' This will be the MES 0
          ON PEN GOSUB 2110
          initial# = TIMER
          count% = 0
          PEN ON
          COLOR 28: LOCATE 23, 10: PRINT "WAITING FOR TRIGGER";
1000        KEY(10) ON
          KEY(10) STOP
          GOTO 1000

' Control comes here when SRQ asserted (PEN)
2110      PEN OFF
```

```
        count% = count% + 1
        COLOR 11
        LOCATE 10, 27: PRINT USING "####"; count%;
        id$ = RIGHT$(STR$(10000 + count%), 4)
        time = TIMER - initial#
        IF time < 0 THEN time = time + 86400
        tim$ = MKS$(time)
        KEY(10) STOP
        COLOR 15: LOCATE 23, 10: PRINT "TRIGGERED....."
        'CALL GETTEMP(TEMP!)
        'TEMPS!(COUNT%) = TEMP!
        CALL ibwrt(bd%, "curv?")
        signal$ = SPACE$(1040)
        CALL ibrd(bd%, signal$)
        signal$ = MID$(signal$, 10, 1024)
        PRINT #1, id$; class$; tim$; "    "; signal$;
        COLOR 28: LOCATE 23, 10: PRINT "WAITING           ";
        LOCATE 23, 18
        KEY(10) ON
        IF delay% THEN       ' pause before ARMing scope
          WHILE TIMER < time + delay!
          WEND
        END IF
        KEY(10) STOP
        CALL SCOPE(bd%, "SGL ARM", 0, REP$)      ' rearms trigger
        CALL SCOPE(bd%, "RQS ON", 0, REP$)       ' set SRQ flag
        PEN ON
        PRINT "FOR TRIGGER"
        IF count% = 9999 GOTO 3000
        RETURN

' This section deals with Continuous sampling of Continuous signals.
        CASE "CC", "CT"
          LOCATE 23, 10
          PRINT "Collecting Data"
          initial# = TIMER
          count% = 1
2000      id$ = RIGHT$(STR$(10000 + count%), 4)
```



```
          CALL ibwrt(bd%, "CURV?")
          signal$ = SPACE$(1040)
          CALL ibrd(bd%, signal$)
          signal$ = MID$(signal$, 10, 1024)
          time = TIMER
          dtime = time - initial#
          IF dtime < 0 THEN
            dtime = dtime + 86400
          END IF
          tim$ = MKS$(dtime)
          PRINT #1, id$; class$; tim$; "    "; signal$;
          'CALL GETTEMP(TEMP!)
          'TEMPS!(COUNT%) = TEMP!

          IF count% = 9999 THEN 3000
          count% = count% + 1
          COLOR 11
          LOCATE 10, 27: PRINT USING "####"; count%;
          KEY(10) ON
          IF delay% THEN
            WHILE TIMER < time + delay!
            WEND
          END IF
          KEY(10) STOP
          GOTO 2000
        END SELECT
        GOTO 3000

2270    KEY(10) OFF
        PEN OFF
        RETURN 3000

3000    PEN OFF
        CLOSE #1
        LOCATE 23, 1
        COLOR 15
        PRINT "Right! That's that then.      "
```

```
' Remember the MOUSE drivers ? Remember the Alamo?
2710      SHELL "c:\mouse1\mouse > kill.fil"
        SHELL "erase kill.fil"
' Doing it this way keeps the message from destroying the screen display.
' You're welcome, Oliver. Stand up when I'm talking to you! (Oh, you are)
        END

4000      CLS
        PRINT "Please check the configuration of computer (run TEK)"
        PRINT "(Also check the connections to the scope.)"
        GOTO 2710

BadDir:
        FOR I% = 1 TO 5
          LOCATE 15 + I%, 1
          PRINT SPACE$(80)
        NEXT I%
        PRINT "No Such Directory - Try again"
        CALL getdrive(default$, froot$, drive$, dir$)
        RESUME NEXT

SUB CHECKSCOPE (bd%, sampmode$)
' This section checks the scope settings to make sure they are such that
' data acquisition is possible.
' Currently checked elements :
'       TRIGGER MODE
'       HORIZONTAL MODE
'       VERTICAL MODE
'       STORE MODE
'       1K ACQUISITION
'       CALIBRATION DETENT OF VERTICAL MODE

100       ERFLAG% = 0
        CALL SCOPE(bd%, "ATR?", -1, REP$)       ' ask for trigger mode
        IF LEFT$(sampmode$, 1) = "T" THEN
          IF MID$(REP$, 15, 6) <> "SGLSWP" THEN
            PRINT "The trigger mode is not set to SINGLE SWEEP."
```



```
            PRINT "The switch is on the right side of the scope below A
              TRIGGER"
            PRINT "Push button labelled SGL SWP in"
            ERFLAG% = 1
          END IF
        ELSE
          IF MID$(REP$, 15, 6) <> "PPAUTO" THEN
            PRINT "The trigger mode is not set to PP AUTO."
            PRINT "The button is on the right side of the scope below A
              TRIGGER"
            PRINT "Push the button labelled P_P AUTO in."
            ERFLAG% = 1
          END IF
        END IF
        CALL SCOPE(bd%, "VMODE?", -1, REP$)    ' vertical mode setting
        IF MID$(REP$, 7, 3) <> "CH1" THEN
          PRINT "The VERTICAL MODE setting is not CH1"
          PRINT "The switch is near the left side of the instrument
            panel"
          PRINT "Move switch fully to the left."
          ERFLAG% = 1
        END IF
        CALL SCOPE(bd%, "STORE?", -1, REP$)    ' ask for store mode status
        IF MID$(REP$, 7, 2) <> "ON" THEN
          PRINT "The scope is not in store mode."
          PRINT "The button is to the left of the knob marked VAR HOLD
            OFF"
          PRINT "Push the button in."
          ERFLAG% = 1
        END IF
        CALL SCOPE(bd%, "HOR? MOD", -1, REP$)
        IF MID$(REP$, 17, 6) <> "ASWEEP" THEN
          PRINT "The horizontal mode is not set to A."
          PRINT "The switch is to the left of the SGL SWP button."
          PRINT "Move the switch full to the left."
          ERFLAG% = 1
        END IF
' There is no way to check for the HF REJECT so we do have to hope that
```

```
' the person has a clue whether he has got it or not!!!

        CALL SCOPE(bd%, "ACQ? POI", -1, REP$)
        IF INSTR(REP$, "1024") = 0 THEN
          PRINT "The Acquisition is not set for 1 K."
          PRINT "Push the button marked 1K under ACQUISTION in."
          ERFLAG% = 1
        END IF

' To check the CALibrate detent on the VOLTS/DIV knob, it is nescessary
' to ask for the V/DIV setting and then see if EVENT 555 is returned. If
' so, then the knob is not in the CALibrate detent.
'- clear SRQ buffer for checking CAL detent on VOLTS/DIV
        DO
          CALL SCOPE(bd%, "EVE?", -1, REP$)
        LOOP UNTIL INSTR(REP$, " 0;")
        CALL SCOPE(bd%, "CH1? VOL", -1, REP$)
        CALL SCOPE(bd%, "EVE?", -1, REP$)
        IF INSTR(REP$, "555") THEN
          PRINT "The Channel 1 VOLTS/DIV knob is not in the CALibrated
          position."
          PRINT "Turn the small knob marked CAL fully clockwise."
          PRINT "Check the CAL knob on the A and B SEC/DIV knob also."
          ERFLAG% = 1
        END IF
' There is no way to check the CAL knob on the SEC/DIV setting. It would
' seem logical to check for an EVENT 555; as in the case for the
' VOLTS/DIV but this capability does not exist in the TEKTRONIX 2230.
        IF ERFLAG% <> 0 THEN
          PRINT
          PRINT "PLEASE make adjustments and press RETURN";
          INPUT a$
          PRINT
          GOTO 100
        END IF
        CLS
' OKAY, now we're ready to get goin'!!!   BaZoom!
END SUB
```



```
SUB getdrive (default$, froot$, drive$, dir$)
        SHELL "dir > dirfil"
        OPEN "i", #1, "dirfil"
        FOR I% = 1 TO 3
          LINE INPUT #1, a$
        NEXT
        CLOSE #1
        SHELL "erase dirfil"
        default$ = MID$(a$, 16)
        IF RIGHT$(default$, 1) <> "\" THEN default$ = default$ + "\"
270      LOCATE 16, 1
        PRINT "Enter filename or path and filename. (file name must not
          exceed 8 characters)"
        LOCATE 17, 1
        PRINT "Default directory:  "; default$
        INPUT froot$
        froot$ = UCASE$(froot$)
        a% = INSTR(froot$, ":")
        IF a% = 2 THEN
          drive$ = LEFT$(froot$, 2)
          froot$ = RIGHT$(froot$, LEN(froot$) - 2)
        ELSEIF a% <> 0 THEN
          PRINT "Do not type any spaces in reply."
          SOUND 800, 1
          GOTO 270
        END IF
        a% = 0
        DO
          n% = a% + 1
          a% = INSTR(n%, froot$, "\")
        LOOP UNTIL a% = 0
        IF n% > 0 THEN
          dir$ = LEFT$(froot$, n% - 1)
          froot$ = RIGHT$(froot$, LEN(froot$) - n% + 1)
        END IF
        IF drive$ = "" AND dir$ = "" THEN default% = 1
        IF LEN(froot$) > 8 THEN
```

```
          SOUND 70, 2
          GOTO 270
        ELSEIF LEN(froot$) < 1 THEN
          PRINT "You must supply a filename."
          SOUND 80, 2
          GOTO 270
        END IF
280       IF drive$ = "" THEN
          SHELL "dir > dirfil"
          OPEN "i", #1, "dirfil"
          LINE INPUT #1, a$
          LINE INPUT #1, a$
          CLOSE #1
          SHELL "erase dirfil"
          drive$ = MID$(a$, 18, 1) + ":"
        END IF
END SUB

SUB GETDTAILS (extitle$, COMMENT$(), AMP%, FLT$(), froot$, SMPMODE$,
        sennum$, class$)
' This is the routine that hassles the user for meaningless details.
' EXTITLE$ - 40 character string containing experiment title
' COMMENT  - 80 character strings containing comments.
' AMP%     - Integer containing amplification of conditioning amplifier
' FLT$()   - Contains the possible filter settings of the C.A.
'            On return, FLT$(0) contains the chosen setting.
' FROOT$   - Contains the filename (max. 5 letters) to be used.
'            On return, the current directory is the intended path.
' SMPMODE$ - A two character string that contains the mode selected.
'            The first character will be a C if continuous sampling is
'            desired as opposed to T for Time delay sampling.  The second
'            letter is a T if triggered signals are to be acquired or a C
'            for constant sampling. If time delay is requested, then the
'            STR$ of the number of seconds is tacked on to the end of
'            SMPMODE$ for return.
' CLASS$   - A four byte character string that contains the default class
'            identification for use when collecting data. Default ="   0"
```



```
SHARED drive$
210     CLS
PRINT "WELCOME TO 'NEWQAQ' - The next generation of DATADUMP"
LOCATE 3, 1
PRINT "Enter experiment title (40 characters) |"
LINE INPUT extitle$
IF LEN(extitle$) > 40 OR LEN(extitle$) < 1 THEN
  SOUND 2000, 1
  GOTO 210
END IF
extitle$ = LEFT$(extitle$ + STRING$(40, " "), 40)
LOCATE 3, 1
PRINT extitle$
PRINT STRING$(80, " ")
LOCATE 5, 1
PRINT "Please enter a description of the experiment."
PRINT "Enter two lines of length no more than eighty characters."
FOR I% = 1 TO 2
220   LOCATE 6 + I%
  LINE INPUT COMMENT$(I%)
  IF LEN(COMMENT$(I%)) > 80 THEN
    SOUND 1200, 1
    GOTO 220
  END IF
  IF COMMENT$(I%) = "" THEN I% = 2
  COMMENT$(I%) = LEFT$(COMMENT$(I%) + STRING$(80, " "), 80)
NEXT
LOCATE 4, 1
FOR I% = 1 TO 2
  PRINT COMMENT$(I%)
NEXT
FOR I% = 1 TO 2
  PRINT STRING$(80, " ")
NEXT
240   LOCATE 8, 1
PRINT"Do you wish Triggered signals or Constant sampling (C/T)?";
a$ = INPUT$(1)
a$ = UCASE$(a$)
```

```
    IF a$ = "C" THEN
      SMPMODE$ = a$
    ELSEIF a$ = "T" THEN
      SMPMODE$ = a$
    ELSE
      SOUND 1000, 1
      GOTO 240
    END IF
230 LOCATE 10, 1
    PRINT "Do you wish to have a time-delay between sampling? (Y/N)";
    a$ = INPUT$(1)
    a$ = UCASE$(a$)
    IF a$ = "N" THEN
      SMPMODE$ = SMPMODE$ + "C"
    ELSEIF a$ = "Y" THEN
      LOCATE 12, 15
      INPUT "Number of seconds to delay "; b$
      IF VAL(b$) = 0 THEN
        SMPMODE$ = SMPMODE$ + "C"
      ELSE
        delay! = VAL(b$)
        SMPMODE$ = SMPMODE$ + "T"
      END IF
    ELSE
      SOUND 1000, 1
      GOTO 230
    END IF
    IF RIGHT$(SMPMODE$,1)="T" THEN SMPMODE$ = SMPMODE$ + STR$(delay!)
    SELECT CASE MID$(SMPMODE$, 2, 1)
    CASE "T"
      COMMENT$(3) = "Time delay (" + RIGHT$(SMPMODE$, LEN(SMPMODE$)
      - 2) + " seconds) : "
    CASE "C"
      COMMENT$(3) = "Continuous sampling : "
    END SELECT
    SELECT CASE LEFT$(SMPMODE$, 1)
    CASE "T"
      COMMENT$(3) = COMMENT$(3) + "Triggered"
```



```
    CASE "C"
      COMMENT$(3) = COMMENT$(3) + "Untriggered"
    END SELECT
    LOCATE 7, 1
    PRINT COMMENT$(3)
    FOR I% = 1 TO 5
      PRINT STRING$(80, " ")
    NEXT
250 LOCATE 9, 1
    INPUT "What is the amplification setting on the conditioning
      amplifier"; a$
    AMP% = VAL(a$)
    IF (AMP% < 1 AND a$ <> "0") OR (AMP% > 60) THEN
      SOUND 900, 1
      GOTO 250
    END IF
    LOCATE 8, 1
    PRINT "Amplification : "; AMP%
    LOCATE 9, 1
    PRINT STRING$(80, " ")
260 LOCATE 10, 1
    PRINT "What is the FILTER setting?"
    FOR I% = 1 TO UBOUND(FLT$)
      PRINT I%; ") "; FLT$(I%)
    NEXT
    PRINT
    INPUT a$
    a% = VAL(a$)
    IF a% < 1 OR a% > UBOUND(FLT$) THEN
      SOUND 800, 1
      GOTO 260
    END IF
    FLT$(0) = FLT$(a%)
    LOCATE 9, 1
    PRINT "FILTER : "; FLT$(0)
    LOCATE 10, 1
    FOR I% = 1 TO UBOUND(FLT$) + 3
      PRINT STRING$(80, " ")
```

```
        NEXT
        LOCATE 11, 1
        INPUT "sensor number : "; sennum$
        LOCATE 13, 1
        INPUT "Enter class prompt: <   0>"; C$
        class$ = LEFT$(C$ + SPACE$(3) + "0", 4)
        LOCATE 14, 1
        PRINT "Class :"; class$
        LOCATE 12, 1
        FOR I% = 1 TO 8
          PRINT STRING$(80, " ")
        NEXT
        LOCATE 12, 1
        PRINT "Class :"; class$
        LOCATE 15, 1
        CALL getdrive(default$, froot$, drive$, dir$)
        IF default% = 1 THEN
          PRINT "File specification = "; default$; froot$; ".AEA"
        ELSE
          PRINT "File specification = "; drive$; dir$; froot$; ".AEA"
        END IF
        LOCATE 20, 1
        INPUT "Is above information correct"; a$
        IF UCASE$(LEFT$(a$, 1)) = "N" THEN 210

' The current directory is changed to that specified
' If the directory does not exist, we will not find a problem until after
' the experiment. To prevent this annoying problem, the directory is
' created. If if already exists an error will occur but the execution
' will continue.
        ON ERROR GOTO BadDir
        IF LEN(dir$) > 1 THEN SHELL "md" + LEFT$(dir$, LEN(dir$) - 1)
        IF LEN(dir$) > 0 THEN SHELL "cd" + LEFT$(dir$, LEN(dir$) - 1)
        ON ERROR GOTO 0
END SUB

SUB GETTEMP (TEMP!)
' This routine one day will poll the RTI-815 ADC and return a value for
```



```
' the temperature. (One day)
END SUB

SUB headerout (COMMENT$(), delay, sensornum$)
' This routine prints out the header file.
' The format of the header file is:' The format of the header file is:
'
'EXPERIMENT TITLE (MAX 40 CHARACTERS)    DATE (8 CHARS)
'TWO comment lines of 80 characters each
'
'This line contains the MODE of sampling.
'T/D=                V/D=                GAIN=                FLT=

        SHARED extitle$, FILTER$, AMP%, TIMDIV$, VOLDIV$, initial#
        PRINT #1, LEFT$(extitle$ + SPACE$(40), 40); SPACE$(2); DATE$;
          SPACE$(10); LEFT$("T2230" + SPACE$(18), 18);
        FOR I% = 1 TO 2
          PRINT #1, LEFT$(COMMENT$(I%) + SPACE$(80), 80);
        NEXT
        PRINT #1, LEFT$("SN=" + sensornum$ + SPACE$(80), 80);
        PRINT #1, LEFT$("T/100pt=" + TIMDIV$ + SPACE$(20), 20);
        PRINT #1, LEFT$("V/D=" + VOLDIV$ + SPACE$(20), 20);
        PRINT #1, LEFT$("GAIN=" + STR$(AMP%) + SPACE$(20), 20);
        PRINT #1, LEFT$("FLT=" + FILTER$ + STRING$(20, " "), 20);
        PRINT #1, SPACE$(40);
        PRINT #1, LEFT$("ACQ DELAY=" + STR$(delay) + SPACE$(40), 40);
        FOR index% = 1 TO 6
            PRINT #1, SPACE$(80);
        NEXT
        EXR$ = "NUL "    ' 3 character name of extra record
        ' status of extra record - see IHB documentation , eh?
        PRINT #1, SPACE$(70); "0    "; SPACE$(2); EXR$;
END SUB

SUB MESSAGE (bd%, SOURCE$(), NUM%)
' Subroutine to print SOURCE$ in format for IBWRT
' The string in source$ must have the first two spaces containing the
' line number of the screen for the message to be written on
```

```
' The numbers are from 16 at the top to 1 at the bottom (upside down)
' If NUM% is zero then "MES 0" is sent which clears the screen and
' returns the screen to display mode. Be aware that the scope screen has
' a limited memory and it is not possible to print on every space on the
' screen at the same time. Yes, spaces at the beginning of strings count
' towards filling the scope's screen buffer so it is advised that all
' messages be left justified.
' SOURCE$() is an array containing the strings to be printed in the
' format discussed above :
'      " 2This message will go on the second line from the bottom"
'      "10This will be printed on the 5th line from the top."
' NUM% : is the number of elements of SOURCE$ that are to be printed.
'        If NUM% = 0 then the screen is cleared and no message is sent.
' BD%  : is the integer expression that refers to the TEK scope for calls
'        to IBWRT.
' GENERIC COMMAND FOR WRITING TO SCREEN :
' MESSAGE$="MES <num>:"+CHR$(34)+"message (40 chars per line)"+chr$(34)
' CALL IBWRT (#,MESSAGE$)
       SOURCE$(0) = "MES 0"
       IF NUM% = 0 THEN
         CALL ibwrt(bd%, SOURCE$(0))
       ELSE
         FOR I% = 1 TO NUM%
           T$ = SOURCE$(I%)
           TLEN = LEN(T$)
           T1$ = "MES " + LEFT$(T$, 2) + ":" + CHR$(34)
           T$ = T1$ + RIGHT$(T$, TLEN - 2) + CHR$(34)
           CALL ibwrt(bd%, T$)
         NEXT I%
       END IF
END SUB

SUB SCOPE (bd%, mes$, REP%, REP$)
' This routine sends MES$ to the device labelled BD%
' If a reply is expected, then REP% should be set TRUE (-1)
' The string is returned in REP$.
' 60 characters is considered enough for most purposes.
       CALL ibwrt(bd%, mes$)
```



```
       IF REP% THEN
         REP$ = SPACE$(60)
         CALL ibrd(bd%, REP$)
       END IF
END SUB

'This routine prints the first title page
SUB TTLE
       COLOR 15, 11
       CLS
       LOCATE 4, 25
       PRINT "NEWQAQ - Data acquisition"
       LOCATE 8, 25
       PRINT "By Dave Sibbald & Kevin Soulsbury"
       LOCATE 14
       PRINT TAB(20); "The Laboratory for Automated Chemical Analysis"
       PRINT TAB(20); "Department of Chemistry"
       PRINT TAB(20); "The University of British Columbia"
       PRINT TAB(20); "Vancouver, B.C., CANADA"
       COLOR , 0
END SUB
```

XI.3 ABSCAT - Abstract Factor Analysis / Scattergram Plotting Utility Program

ABSCAT was written by D. B. Sibbald in QuickBASIC 4.0. The AFA routine was translated from a program written in FORTRAN by P. D. Wentzell. The display of the two-dimensional scattergrams uses an installed enhanced graphics adapter (EGA) card.

The program has many options and any are available from the program in any order, allowing for recall of previous analyses. The procedure for calculation of the AFA analysis is as follows. The user specified descriptor file (with a .DS1, .DES, .SCL) extension is loaded and subjected to the same scaling options as in the DENDGRAM program.

The AFA routine first involves the calculation of the covariance matrix. This is then used to generate each principal component. The principal component is then removed from the covariance matrix and the process continues until a complete set of basis vectors has been calculated. (The complete set of basis vectors will consist of the same number of vectors as there were descriptors in the original data file[F]). The scaled data matrix is then projected onto the new set of basis vectors (factor space). The basis vectors, along with the projected data, are stored in a data file (.AF2). Also, a report file (.RES) is created which lists various parameters which can help determine the number of primary factors present in the data.

The .RES report file can be viewed from within ABSCAT, and help screens are available which explain the significance of parameters such as the root-mean-square (RMS) error, the cumulative percent variance (CPV) and the imbedded error (IE) function. The loadings of the original descriptors on the factors can be displayed

---

F That is, unless some of the descriptors have been removed from analysis by the scaling routine.

graphically. This gives an indication of the relative importance of each descriptor as a descriminating parameter.

ABSCAT also plots two-dimensional scattergrams of data files. These data files can either be descriptor files (with .DS1, .DES, .SCL extensions) or can be the factor analysis data file (.AF2, and .AFA files). Two descriptors / factors can be plotted against each other or they can be plotted against time. Colored plotting is used to allow one to distinguish and assign different classes of signals visually.

On the advice / insistence of the members of this research group who wished to use the program, some elementary DOS functions were incorporated into ABSCAT to allow for people to view the contents of the computer's hard disk (or of floppy disks) to jog a memory which wasn't quite able to keep up with the myriad files created during the data analysis routines.

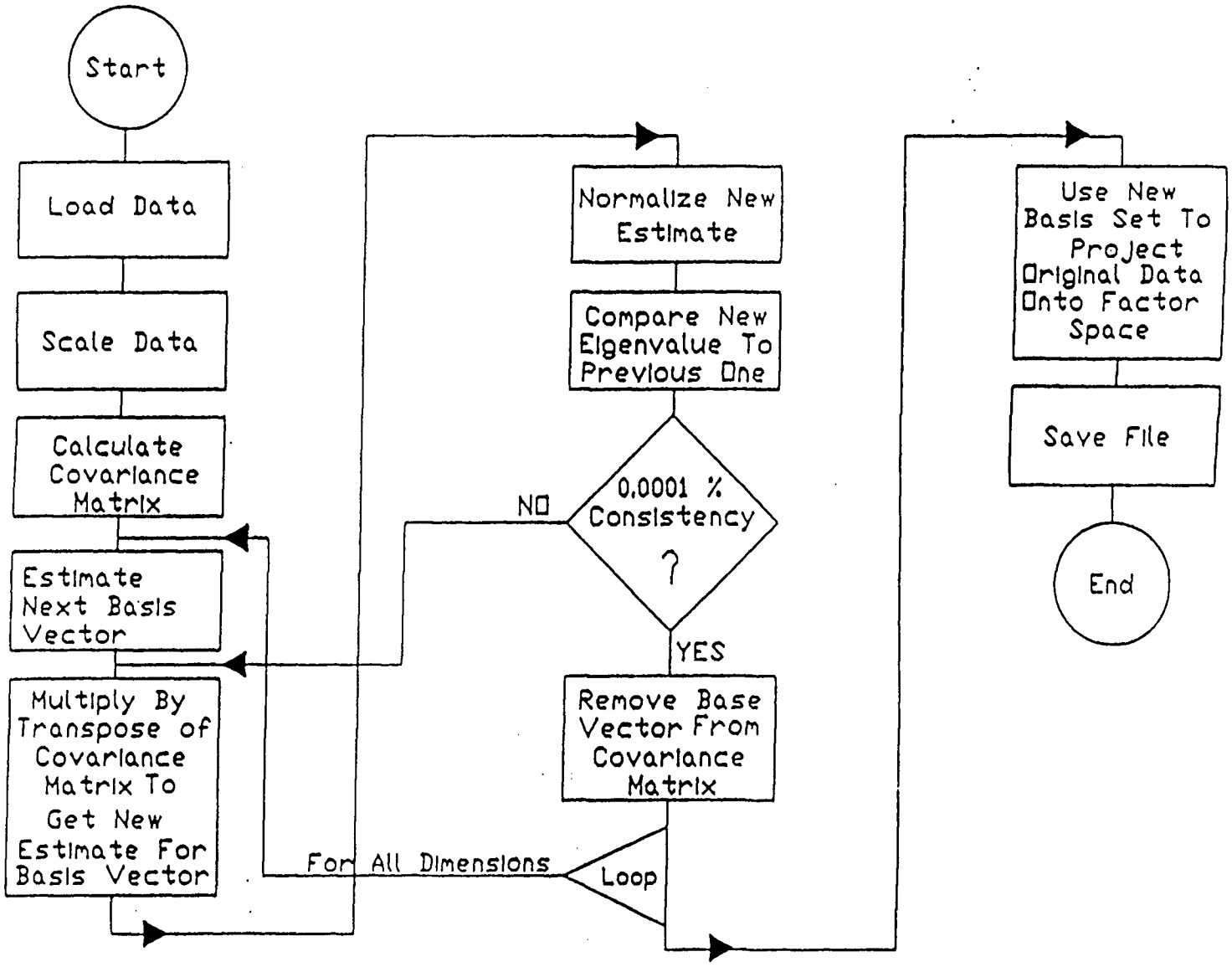


Figure 83) Abstract factor analysis (AFA) algorithm from ABSCAT program.

```
'     ABSCAT  -  Abstract Factor analysis, Scattergram utility. **************
'     Written by David Sibbald
'     (Abstract factor analysis routine by Peter Wentzell from FORTRAN)
'
'     Laboratory for Automated Chemical Analysis
'     Chemistry Department, University of British Columbia.
'     Vancouver , B.C
'
'      A useful chemometric utility program for use with AECRUNCH and
'     DENDGRAM. (See also SIGVIEW)
'
'     The purpose of this program is to :
'       1) Perform abstract factor analysis on a file.
'       2) View the scatterplot of various formats of files.
'       3) Create DEScriptor files for use in DENDGRAM using abstract factors.
'
'
'-------------------------------------------------------------------------------
' Subroutine Declarations
'
DECLARE SUB TITLESCRN ()
DECLARE SUB TITLESCRN2 ()
DECLARE SUB GETKEY (KEY$)
DECLARE FUNCTION FEXIST% (F$, LENGTH&)
DECLARE SUB GETFIL (INFIL$, EXT$, EXIST%)
DECLARE FUNCTION FRTEXT$ (FILE$, ROOT$, EXT$)
DECLARE FUNCTION GETDIR$ ()
DECLARE SUB SLEP ()
'-------------------------------------------------------------------------------
DECLARE SUB READCOL (F$, COL#(), COL%)
DECLARE SUB READVALS (F$, VARNAM$(), MESS#(), ID() AS ANY, CLASS() AS ANY, TIME!
            EXTRA() AS ANY, EXR$, EXNAM$(), EXTS%(), EXTEN$(), SCALED$)
DECLARE SUB READPARS (F$, NUMSAMS%, NUMVARS%, EXT1%, EXT2%)
DECLARE SUB READFACTS (FILE$, NUMFAX%, EVALS!(), EVECS!(), VARNAM$())
'-------------------------------------------------------------------------------
DECLARE SUB NEWDES (FILE$, SUCCESS%)
'-------------------------------------------------------------------------------
DECLARE SUB FACLODISPLAY (FILE$)
```



```
DECLARE SUB FACDISHELP ()
DECLARE SUB verprint (WORD$, X%, Y%)
'-------------------------------------------------------------------------------
' Subroutines used by AFA section
DECLARE SUB AFA (FILE$)
DECLARE SUB ADVSCAL (MESS#(), VARNAM$(), F$, SCALED$, DELVAR%())
DECLARE SUB SCALECOL (MAT#(), COL%, MODE%, VALUE#, DIV#)
DECLARE SUB SCLFORSAV (FILE$, SC$, A%(), B#(), C#())
DECLARE SUB SCLFORLOD (FILE$, SC$, MODE%(), A#(), B#())
DECLARE SUB SCALER (MESS#(), VARNAM$(), FILE$, SCALED$, DELVAR%(), OUTFILE$)
DECLARE FUNCTION COLMAX! (MAT#(), COL%, STARTROW%, ENDROW%, COLMIN!)
DECLARE FUNCTION VARIANCE# (MAT#(), COL%, NUMROWS%, AVE#)
'-------------------------------------------------------------------------------
' Subroutines used by SCATtergram section
DECLARE SUB SCAT (FILE$)
DECLARE FUNCTION INSIDE% (X#, Y#, XD#, YD#, YSCALE#, XSCALE#, RAD%)
DECLARE SUB CROSSHAIRS (X%, Y%, CROSSCOL%)
DECLARE SUB MARK (X%, Y%, MRK%)
DECLARE SUB PMRK (XCORD#, YCORD#)
DECLARE SUB EGGBOX (X%, Y%, RAD%, COLR%, CRCLE%())
DECLARE SUB TIMPRINT (MESS#(), VARNAM$(), F$)
'-------------------------------------------------------------------------------
' Subroutines for explaining the output in the RESults file after performing
' an AFA.
DECLARE SUB RESFILHELP ()
DECLARE SUB CPVHELP ()
DECLARE SUB IEHELP ()
DECLARE SUB INDHELP ()
DECLARE SUB REHELP ()
DECLARE SUB RMSHELP ()
DECLARE SUB EVALHELP ()
DECLARE SUB NACHELP ()
DECLARE SUB EOAHELP ()
DECLARE SUB VARHELP ()
'-------------------------------------------------------------------------------

'     Define some Options , variables, Constants etc.
```

```
1     COLOR FGROUND, bground
      CLS
      LOCATE 2, 20
      PRINT "Abstract Factor Analysis Utility"
      FOR I% = 1 TO numchoices%
      ' change this line to handle more choices than 9
            LOCATE I% * 2 + 7 - numchoices% \ 2, 10
            PRINT I%; ") "; choice$(I%)
      NEXT
10    LOCATE 22, 15
      PRINT "Enter Selection          ";
      slct% = 0
      LOCATE 22, 31
      INPUT ; a$
      a% = VAL(a$)
      IF a% > 0 AND a% <= numchoices% THEN slct% = a%
      IF slct% = 0 THEN 10
      SELECT CASE slct%
      CASE 1 '  Factor Analysis
            CLS
            LOCATE 2, 10
            PRINT "Factor Analysis"
            LOCATE 4, 1
            PRINT "Enter filename (including root and extension) : <";
            PRINT FILE$ + FEXT$; ">"
            COLOR 3
            PRINT "   (Legal extensions = "; Q$; ".DES.DS1.DS2.SCL"; Q$; ")"
            COLOR 11
            INPUT F$
            IF F$ <> "" THEN
                  EXTENS$ = "DS1.DES.DS2.SCL"
                  F$ = UCASE$(F$)
                  CALL GETFIL(F$, EXTENS$, EXIST%)
                  IF F$ = "" THEN
                        PRINT "Illegal file specification"
                        CALL slep
                  ELSEIF NOT EXIST% THEN
```



```
            CLS
            TYPE RECRD
                  R AS STRING * 4
            END TYPE
            CONST blue = 1
            CONST ltblue = 11
            CONST yellow = 14
            CONST red = 4
            CONST purple = 13
            CONST green = 10
            CONST FGROUND = ltblue
            CONST bground = purple
            CONST legalext$ = ".DES.DS1.DS2.SCL.AFA.AF2"
            CONST maxdiml = 23000
            CALL titlescrn
            Q$ = CHR$(34)

      ' number of options at main menu  Max is nine
            DATA 8
            DATA "Abstract Factor Analysis"
            DATA "Print out Results of previous Abstract Factor Analysis"
            DATA "Show AFA factor loadings"
            DATA "Create DEScriptor file from AFA results"
            DATA "View Scattergram"
            DATA "Show directory"
            DATA "Explain file extensions"
            DATA "Exit"
            READ numchoices%
            DIM choice$(1 TO numchoices%)
            FOR I% = 1 TO numchoices%
                  READ choice$(I%)
            NEXT

            CURDIRECTORY$ = GETDIR$
            CALL TITLESCRN2
```

```
                PRINT "Can't find "; F$
                CALL slep
                GOTO 1
        END IF
        FILESPEC$ = F$
    ELSE
        IF F$ = "" AND FILE$ = "" THEN GOTO 1
    END IF
    FILE$ = FRTEXT$(FILESPEC$, FROOT$, FEXT$)
    CALL AFA(FILESPEC$)

CASE 2 ' Copy REP file to printer
    CLS
    LOCATE 2, 10
    PRINT "Print out RES file"
    LOCATE 4, 1
    PRINT "Enter name of .AF2 (AFA) file (including path) :<"; FILE$; ">"
    INPUT F$
    IF F$ <> "" THEN
        EXTENS$ = "RES"
        F$ = UCASE$(F$)
        CALL GETFIL(F$, EXTENS$, EXIST%)
        IF F$ = "" THEN
            PRINT "Illegal file specification"
            CALL slep
        ELSEIF NOT EXIST% THEN
            PRINT "Can't find "; F$
            CALL slep
            GOTO 1
        END IF
        RESFILE$ = F$
    ELSEIF F$ = "" AND FILE$ = "" THEN GOTO 1
    ELSE
        RESFILE$ = FROOT$ + FILE$ + ".RES"
    END IF
    IF NOT FEXIST%(RESFILE$, 1&) THEN
        PRINT "Can't find "; RESFILE$
```



```
        PRINT "(Factor Analysis not performed or File has been moved)"
        CALL slep
        GOTO 1
    END IF
    LOCATE 6, 1
    INPUT "Printer or screen (P/S)", a$
    a$ = UCASE$(LEFT$(a$, 1))
    IF a$ = "P" THEN
        LPRINT "RESULTS FROM ABSTRACT FACTOR ANALYSIS OF FILE :";
        LPRINT FILE$
        SHELL "type " + RESFILE$ + " >prn"
    ELSE
        CLS
        numlines% = 16
        OPEN "I", #1, RESFILE$
        FOR I% = 1 TO 6
            LINE INPUT #1, a$
            PRINT a$
        NEXT
        VIEW PRINT 7 TO 7 + numlines%
        a% = 0
        WHILE NOT EOF(1)
            LINE INPUT #1, a$
            a% = a% + 1
            PRINT a$
            IF a% = numlines% THEN
                a% = 1
                LOCATE 25, 5
                COLOR 15
                PRINT"Press key <?-explain column headings>";
                COLOR 11, 13
                GOSUB 95
                LOCATE 7 + numlines% - 2
            END IF
        WEND
        CLOSE #1
        LOCATE 25, 5
```

```
      COLOR 15
      PRINT "Press a key to continue. <?-explain headings>";
      GOSUB 95
      COLOR 11
      VIEW PRINT
    END IF
    GOTO 1
95  CALL getkey(a$)
    IF a$ = "?" OR a$ = "/" THEN
      CALL resfilhelp
      COLOR 11, 13
      VIEW PRINT 7 TO 7 + numlines%
      GOTO 95
      END IF
    RETURN

    CASE 3
      CLS
      LOCATE 2, 10
      PRINT "Print out Factor Loadings"
      LOCATE 4, 1
      PRINT "Enter name of .AF2 (AFA) file (including path) :<"; FILE$; ">"
      INPUT F$
      IF F$ <> "" THEN
        EXTENS$ = "AF2.AFA"
        F$ = UCASE$(F$)
        CALL GETFIL(F$, EXTENS$, EXIST%)
        IF F$ = "" THEN
          PRINT "Illegal file specification"
          CALL slep
        ELSEIF NOT EXIST% THEN
          PRINT "Can't find "; F$
          CALL slep
          GOTO 1
        END IF
        flsfile$ = F$
      ELSEIF F$ = "" AND FILE$ = "" THEN GOTO 1
```



```
      ELSE
        flsfile$ = FROOT$ + FILE$ + ".AF2"
      END IF
      IF NOT FEXIST%(flsfile$, 1&) THEN
        PRINT "Can't find "; flsfile$
        CALL slep
        GOTO 1
      END IF
      CALL FACLODisplay(flsfile$)
      FILESPEC$ = flsfile$
      FILE$ = FRTEXT(FILESPEC$, FROOT$, FEXT$)

    CASE 4
      CLS
      IF FEXT$ <> ".AFA" THEN FEXT$ = ".AF2"
      LOCATE 2, 10
      PRINT "Create DeScriptor file from abstract factors."
      LOCATE 4, 1
      PRINT "Enter name of .AF2 (AFA) file (including path) :<"; FILE$; ">"
      INPUT F$
      IF F$ <> "" THEN
        EXTENS$ = "AF2.AFA"
        F$ = UCASE$(F$)
        CALL GETFIL(F$, EXTENS$, EXIST%)
        IF F$ = "" THEN
          PRINT "Illegal file specification"
          CALL slep
        ELSEIF NOT EXIST% THEN
          PRINT "Can't find "; F$
          CALL slep
          GOTO 1
        END IF
        DESFILE$ = F$
      ELSEIF F$ = "" AND FILE$ = "" THEN GOTO 1
      ELSE
        DESFILE$ = FROOT$ + FILE$ + FEXT$
      END IF
```

```
   CALL NEWDES(DESFILE$, SUCCESS%)
   IF SUCCESS% THEN FEXT$ = ".DS2"
   FILESPEC$ = FROOT$ + FILE$ + FEXT$

CASE 5
   CLS
   LOCATE 2, 10
   PRINT "View SCATTERGRAM of data file."
   LOCATE 4, 1
   PRINT "Enter name of file (including extension) ";
   IF FILE$ <> "" THEN
      PRINT "<"; FILE$ + FEXT$; ">";
   END IF
   PRINT " :"
   COLOR FGROUND - 8
   PRINT "Legal extensions "; CHR$(238); " "; legalext$
   COLOR FGROUND
   INPUT F$
   F$ = UCASE$(F$)
   IF F$ <> "" THEN
      EXTEN$ = ".DS1.DES.DS2.AF2.AFA.SCL"
      CALL GETFIL(F$, EXTEN$, EXIST%)
      IF F$ = "" THEN
         PRINT "Illegal file specification"
         CALL slep
         GOTO 1
      ELSEIF NOT EXIST% THEN
         PRINT "Can't find "; F$
         CALL slep
         GOTO 1
      END IF
      FILESPEC$ = F$
   ELSEIF F$ = "" AND FILE$ = "" THEN GOTO 1
   ELSE
      FILESPEC$ = FROOT$ + FILE$ + FEXT$
   END IF
   CALL SCAT(FILESPEC$)
```



```
      FILE$ = FRTEXT$(FILESPEC$, FROOT$, FEXT$)

   CASE 6
      CLS
      LOCATE 2, 1
      PRINT "Enter directory to show <"; CURDIRECTORY$; ">"
      PRINT "   (enter "; Q$; ".*"; Q$; " to show ALL files)"
      INPUT d$
      SHELL "DIR " + d$ + " >tempfil.dir"
      CLS
      LOCATE 1, 1
      COLOR yellow
      OPEN "i", #2, "tempfil.dir"
      LINE INPUT #2, a$
      PRINT "  "; a$
      LINE INPUT #2, a$
      PRINT "  "; a$
      LINE INPUT #2, a$
      PRINT "  "; a$
      VIEW PRINT 4 TO 24
      COLOR FGROUND
      WHILE NOT EOF(2)
        linenum% = 1
        DO
          LINE INPUT #2, a$
          IF RIGHT$(a$, 12) <> "TEMPFIL  DIR" THEN
            IF RIGHT$("  " + d$, 2) = ".*" THEN
              PRINT "  "; a$
              linenum% = linenum% + 1
            ELSEIF MID$(a$, 15, 3) = "DIR" THEN
              PRINT "  "; a$
              linenum% = linenum% + 1
            ELSEIF INSTR(legalext$+"REP", MID$(a$,10,3)) <> 0 THEN
              PRINT "  "; a$
              linenum% = linenum% + 1
            END IF
          END IF
```

```
        LOOP UNTIL linenum% = 18 OR EOF(2)
        LOCATE 24, 5
        COLOR yellow + 16, red
        PRINT " press any key to continue... ";
        a$ = INPUT$(1)
        COLOR FGROUND, bground
        CLS
        LOCATE 5, 1
      WEND
      CLOSE #2
      VIEW PRINT
      CLS
      KILL "tempfil.dir"

    CASE 7
      COLOR 15, 0
      CLS
' look for file ABSCAT.TXT - This should contain the latest dirt on the file
' formats.
      OPEN "abscat.txt" FOR RANDOM ACCESS READ AS #1
      l = LOF(1)
      CLOSE #1
      IF l <> 0 THEN
        OPEN "I", #1, "ABSCAT.TXT"
        WHILE NOT (EOF(1))
          LINE INPUT #1, a$
          PRINT a$
        WEND
        CLOSE #1
      ELSE ' it's not there. Print out the standard dirt
PRINT
PRINT "                File Extensions"
PRINT
PRINT ".DES    - Original Descriptor file. Contained descriptors in ASCII
      format."
PRINT
```



```
        PRINT ".DS1    - Current Descriptor file. BINARY file. Contains
      more information."
        PRINT:PRINT ".DS2    - ASCII analog of DS1 file. "
        PRINT:PRINT ".AFA    - Original Abstract Factor Analysis
          file.Contains scores and loadings"
        PRINT:PRINT ".AF2    - Current format of AFA file. Has complete
          info from DS1 file."
        PRINT:PRINT ".RES    - Table of indicator functions for each Factor
          extracted in AF2 file."
        PRINT:PRINT ".SFF    - Scaling format file. Holds USER-defined
          special scaling info."
        PRINT
      END IF
      a$ = INPUT$(1)

    CASE numchoices%
      END     ' exit program - no fanfair - "Dead is dead"... Ian Shildt

    CASE ELSE
    END SELECT
    GOTO 1

SUB advscal (mess#(), varnam$(), F$, scal$, DELVAR%())
'
' This routine is called when the user decides to play God and mess up the
' data with some wierd scaling. If there is some knowledge about the data
' then "user defined" scaling is a good idea but can be disastrous when
' applied indiscriminately. So that's that then.
'
' The columns are shown on the screen and the vital statistics are shown of
' each variable. These are stored in arrays which need not ever be accessed
' again so it is prudent to not declare this routine as STATIC or at least to
' erase the arrays before continuing.
'
' One BUG : Yes I am reporting a bug that I know to be present but could not
' fix. When selecting the mode, pressing the arrow keys 19 times causes a
' STRING FORMULA TOO COMPLEX error on an otherwise perfectly good command. I
```

```
' could discover no obvious or secret reason for this except that the error
' does not occur when the program is operated from executable mode. I feel
' that the QUICKBASIC environment is hostile to people who REALLY can't
' make up their mind.
'
' VARIABLES
' =========
' SCVAL#  - These two variables store values for each column. They are the
' SCDIV#  - parameters passed to SCALCOL. If AUTO scaling or SHIFT/MULTIPLY
'           scaling is used, then SCVAL# is the value to be subtracted from
'           each element (to scale the mean to zero, for example, SCVAL# is
'           equal to the column average.). SCDIV# is the value that each
'           element is divided by. For AUTO scaling, SCDIV# should be the
'           SQuare Root of the variance.
' SCMOD%  - The SCaling MODe for each particular column. The numbers refer to
'           the strings stored in SC().1 - Auto scaling.
'                                     2 - Range scaling.
'                                     3 - User defined Shift/Multiply
'                                     4 - Remove variable from analysis
' VARNZ#  - The variance of each column of the matrix.
' AVE#    - The average of each column.
' COLMINS!- The minimum value of the column.
' COLMAXS!- The maximum value of the column.
' CURROW% - An integer referring to the number of the row on the screen that
'           is currently selected. (NB. each row corresponds to a column of
'           the matrix)
' SCRNROWS- This is a "constant" to give the number of variables that can
'           be displayed on the screen at once. These are displayed starting
'           on line TOP.
' TOP     - This is the first line that the variable information is displayed
'           on. If this is changed, it must be kept in mind that lines 1-3
'           are used for title information and lines 23-25 are used for other
'           info/input.[SCRNROWS + TOP < 23]
' TOPROW% - This is the integer corresponding to the variable # that is to
'           appear on the first line [LOCATE TOP]
' BOTROW% - This is the variable number that appears on the last row on the
'           defined "variable screen". [TOP + SCRNROWS%]
```



```
' CURSTAT%- Refers to the current statistic that is highlighted. This will
'           depend on the SCMOD% for the current column.
' SCAL$   - The string that represents the method of scaling. This gets saved
'           at the end of the SCL file. If SCAL$ = "VIEW MODE" on calling then
'           TIMMODE% is set positive.
' TIMMODE%- Is TRUE (-1) if SCAL$ contains the string "VIEW MODE" on calling.
'           TIMmode is when people like TIM just want to look at the column
'           statistics. Print out is available as well as a future bit to
'           put out a Lotus file. (God knows why)
'           The perfectionist will also notice that MESS# is passed as a blank
'           array with only one dimension.  This may not be acceptable to some
'           COMPUTER GODS but it works to printout TIM'S statistics.

    NUMsams% = UBOUND(mess#, 1)
    NUMVARS% = UBOUND(varnam$)
    CONST SCRNROWS% = 18
    CONST TOP = 4
    CONST yellow = 14
    CONST bgrnd = 13
    CONST hlght = 8
    CONST ntext = 11
    CURROW% = 1
    CURSTAT% = 1
    IF scal$ = "VIEW MODE" THEN TIMmode% = -1
    scal$ = ""
    DIM SC(4) AS STRING * 15
    SC(1) = "Auto scale     "
    SC(2) = "Range scale    "
    SC(3) = "Shift/Multiply "
    SC(4) = "Erase / Remove "
    DIM SCVAL#(1 TO NUMVARS%)
    DIM scdiv#(1 TO NUMVARS%)
    DIM SCMOD%(1 TO NUMVARS%)
    DIM varnz#(1 TO NUMVARS%)
    DIM AVE#(1 TO NUMVARS%)
    DIM colmins!(1 TO NUMVARS%)
    DIM colmaxs!(1 TO NUMVARS%)
```

```
    IF TIMmode% THEN
      DIM col#(NUMVARS%)
    END IF
    pt1$ = " Column ## ,  Mode = "
    pt$= " ## |  ##.###^^^^  |##.####^^^^ | ##.####^^^^  | ##.####^^^^  | "
    DIM pt2$(4)
    pt2$(1) = "  Scale mean to :###.####,Use variable ##  variance  "
    pt2$(2) = "  Range Between :###.####  (min) and   ###.#### (max)"
    pt2$(3) = "  Shift mean to :###.####, Multiply by ###.####      "
    pt2$(4) = " "

' Calculate variable statistics for each column.
    c% = 1
    FOR I% = 1 TO NUMVARS%
      SCVAL#(I%) = 0
      scdiv#(I%) = I%
      SCMOD%(I%) = 1
      IF TIMmode% THEN
        CALL readcol(F$, col#(), I%)
        FOR J% = 1 TO NUMsams%
          mess#(J%, 1) = col#(J%)
        NEXT
      ELSE
        c% = I%
      END IF
      varnz#(I%) = VARIANCE#(mess#(), c%, NUMsams%, AVE#(I%))
      colmaxs!(I%) = colmax!(mess#(), c%, 1, NUMsams%, colmins!(I%))
    NEXT
    COLOR 11, bgrnd
    CLS
    LOCATE 1, 1
    PRINT STRING$(80, "*");
    LOCATE 2, 24
    PRINT "SCALING : "; F$
    LOCATE 3, 1
    PRINT "Variable| Variance |  Average   |   Maximum    |   Minimum    |";
    IF TIMmode% THEN
      PRINT " Name"
    ELSE
      PRINT " Mode "
    END IF
      SOUND 400, 1                                    ' Wake up Bob
      TOPROW% = 1

16  COLOR 11
    LOCATE TOP, 1

' Print out statistics for each column.
    BOTROW% = TOPROW% + SCRNROWS%
    IF BOTROW% > NUMVARS% THEN BOTROW% = NUMVARS%
    FOR I% = TOPROW% TO BOTROW%
      PRINT USING pt$; I%; CSNG(varnz#(I%)); CSNG(AVE#(I%)); colmaxs!(I%);
      PRINT colmins!(I%);
      IF TIMmode% THEN
        PRINT varnam$(I%)
      ELSE
        PRINT SC(SCMOD%(I%))
      END IF
      NEXT I%
      IF NOT TIMmode% THEN
        LOCATE 25, 1
        PRINT "Press L to load in saved Format file. P - Print.";
        PRINT " ESC - Execute"; SPACE$(16);
        LOCATE 23, 1
        COLOR 0, bgrnd
        PRINT " " + varnam$(CURROW%); STRING$(40, " ");
      ELSE
        LOCATE 25, 1
        PRINT "Press ESC to return. P - Print numbers out. F - lotus File";
      END IF
' Highlight current variable (CURROW%)
      COLOR yellow, hlght
      LOCATE CURROW% + TOP - TOPROW%, 1
```

```
        PRINT USING pt$; CURROW%; CSNG(varnz#(CURROW%)); CSNG(AVE#(CURROW%));
            colmaxs!(CURROW%); colmins!(CURROW%);
        IF TIMmode% THEN
          PRINT varnam$(CURROW%);
        ELSE
          PRINT SC(SCMOD%(CURROW%));
        END IF

212     COLOR yellow, bgrnd
        IF NOT TIMmode% THEN
          LOCATE 24, 1
          PRINT USING pt1$; CURROW%;
          IF SCMOD%(CURROW%) = 4 THEN COLOR 14, hlght
          PRINT LEFT$(SC(SCMOD%(CURROW%)), 5);
' Print out the bottom menu line
          IF SCMOD%(CURROW%) <> 4 THEN
            LOCATE 24, 27
            PRINT USING pt2$(SCMOD%(CURROW%)); SCVAL#(CURROW%);
              scdiv#(CURROW%);
          ELSE
            COLOR bgrnd, bgrnd
            PRINT STRING$(54, " ");
          END IF
' Print out the CURSTAT% selection in highlight
          COLOR yellow, hlght
          LOCATE 24, (CURSTAT% - 1) * 22 + 22
          IF SCMOD%(CURROW%) < 4 THEN
            SELECT CASE CURSTAT%
            CASE 1
              PRINT LEFT$(SC(SCMOD%(CURROW%)), 5);
            CASE 2
              PRINT USING "###.####"; SCVAL#(CURROW%);
            CASE 3
              IF SCMOD%(CURROW%) = 1 THEN
                PRINT USING "##"; scdiv#(CURROW%);
              ELSE
                PRINT USING "###.####"; scdiv#(CURROW%);
```



```
              END IF
            END SELECT
          END IF
        END IF

' Get command key and process
1236    CALL getkey(a$)
        IF ASC(a$) = 0 THEN a$ = RIGHT$(a$, 1)
1237    SELECT CASE a$
        CASE CHR$(77)                          ' the RIGHT key was pressed
          CURSTAT% = CURSTAT% + 1
          IF CURSTAT% > 3 THEN CURSTAT% = 1
          GOTO 212
        CASE CHR$(75)                          ' the LEFT key was pressed
          CURSTAT% = CURSTAT% - 1
          IF CURSTAT% = 0 THEN CURSTAT% = 3
          GOTO 212
        CASE CHR$(72)                                ' the UP key was pressed
          CURROW% = CURROW% - 1
          IF CURROW% = 0 THEN CURROW% = NUMVARS%
          IF CURROW% < TOPROW% THEN TOPROW% = CURROW%
          IF CURROW% > BOTROW% THEN TOPROW% = CURROW% - SCRNROWS%
        CASE CHR$(80)                                ' the Down key was pressed
          CURROW% = CURROW% + 1
          IF CURROW% > NUMVARS% THEN CURROW% = 1
          IF CURROW% < TOPROW% THEN TOPROW% = CURROW%
          IF CURROW% > BOTROW% THEN TOPROW% = CURROW% - SCRNROWS%
        CASE "L"
          IF TIMmode% THEN 1236
          LOCATE 25, 1
          INPUT ; "Input name of format file to load : ", a$
          a$ = UCASE$(a$)
          CALL GETFIL(a$, "SFF", EXIST%)
          IF a$ = "" THEN 1236
          IF NOT EXIST% THEN
            LOCATE 25, 1
            PRINT STRING$(80, " ");
```

```
        LOCATE 25, 1
        PRINT "Can't find "; a$;
        CALL slep
        GOTO 1236
      END IF
      CALL SCLFORLOD(a$, scal$, SCMOD%(), SCVAL#(), scdiv#())
    CASE "P", "p"
      GOSUB prinout
    CASE "F", "f"                                                    61
      GOSUB filout
    CASE CHR$(27)                                  ' ESC key pressed
            GOTO 39
    CASE CHR$(13)                                  ' RETURN  CTRL-M
      IF TIMmode% THEN 1236
      COLOR yellow, bgrnd
      LOCATE 25, 1
      PRINT SPACE$(79);
      LOCATE 25, 1
      SELECT CASE CURSTAT%
      CASE 1                                  ' Mode selector
        MD% = SCMOD%(CURROW%)
58      LOCATE 25, 1
        PRINT "Select new mode - ";
        FOR I% = 1 TO 4
          PRINT SC(I%);
        NEXT I%
        LOCATE 25, LEN(SC(1)) * 5 + 5
        LOCATE 25, MD% * 15 + 4
        COLOR yellow, hlght
        PRINT SC(MD%);
        COLOR yellow, bgrnd
        a$ = ""
' Open up menu on bottom line for Mode selection
59      CALL getkey(a$)
        IF a$ = CHR$(13) GOTO 61
        a$ = RIGHT$(a$, 1)
        IF a$ = CHR$(75) THEN 'LEFT
```



```
          MD% = MD% - 1
          IF MD% = 0 THEN MD% = 4
        ELSEIF a$ = CHR$(77) THEN 'RIGHT
          MD% = MD% + 1
          IF MD% = 5 THEN MD% = 1
        END IF
        GOTO 58

      IF MD% <> SCMOD%(CURROW%) THEN            ' change in mode
        SCMOD%(CURROW%) = MD%
        IF MD% = 1 THEN
          SCVAL#(CURROW%) = 0
          scdiv#(CURROW%) = CURROW%
        ELSEIF MD% = 2 THEN
          SCVAL#(CURROW%) = 0
          scdiv#(CURROW%) = 1
        ELSEIF MD% = 3 THEN
          SCVAL#(CURROW%) = 0
          scdiv#(CURROW%) = 1
        ELSE 'md% = 4 == Erase variable
        END IF
      END IF
    CASE 2                                          'value selector
      COLOR yellow, hlght
      PRINT " Change value to :";
      INPUT ; "", a$
      IF SCMOD%(CURROW%) = 2 THEN    ' RANGE scaling : check MIN = MAX
        IF VAL(a$) = scdiv#(CURROW%) THEN
          LOCATE 25, 1
          PRINT "The values for maximum and minimum must be";
          PRINT " different";
          SOUND 37, 3
          GOTO 1236
        END IF
      END IF
      SCVAL#(CURROW%) = VAL(a$)
    CASE 3                                          'div# selector
```

```
    COLOR 14, hlght
    IF SCMOD%(CURROW%) = 1 THEN
      INPUT ; " Scale column to variance of variable #", a$
      dv# = VAL(a$)
      IF dv# > 0 AND dv# < NUMsams% THEN scdiv#(CURROW%) = dv#
    ELSEIF SCMOD%(CURROW%) = 2 THEN
      INPUT ; " Scale maximum to =", a$
      IF VAL(a$) = SCVAL#(CURROW%) THEN
        LOCATE 25, 1
        PRINT SPACE$(79);
        LOCATE 25, 1
        SOUND 90, 1
        PRINT "The values for Maximum and Minimum";
        PRINT " cannot be the same";
        GOTO 1236
      ELSE
        scdiv#(CURROW%) = VAL(a$)
      END IF
    ELSE
      PRINT " Value to multiply by =";
      INPUT ; "", a$
      dv# = VAL(a$)
      IF dv# <> 0 THEN
        scdiv#(CURROW%) = dv#
      ELSE
        LOCATE 25, 1
        COLOR 14, hlght
        PRINT "Use ERASE/REMOVE to eliminate variable";
        PRINT " from analysis";
        SOUND 100, 1
        GOTO 1236
      END IF
    END IF
  CASE ELSE
  END SELECT

CASE ELSE
```



```
    END SELECT
    a$ = ""
    GOTO 16
'GOSUB that prints out statistics to a standard line printer
prinout:
    LOCATE 25, 1
    PRINT STRING$(80, " ");
    LOCATE 25, 1
    PRINT "Set printer to top of form and press any key. ESC aborts.";
    CALL getkey(a$)
    IF a$ = CHR$(27) THEN
      LOCATE 25, 1
      PRINT STRING$(80, " ");
      GOTO 16
    END IF
    pttim$ = " | ##.###^^^^ |##.####^^^^ | ##.####^^^^  | ##.####^^^^  |"
    LPRINT "Variable statistics for : "; F$
    LPRINT "       |Variable     | Variance  | Average   |   Maximum    |
      Minimum    |"
    LPRINT "       |-------------+-----------+-----------+--------------+
      -------------|"
    FOR I% = 1 TO NUMVARS%
      LPRINT STRING$(7, " ") + "|" + LEFT$(varnam$(I%) + SPACE$(12), 12);
      LPRINT USING pttim$; CSNG(varnz#(I%)); CSNG(AVE#(I%)); colmaxs!(I%);
        colmins!(I%)
    NEXT I%
    LPRINT
    LPRINT "         This data packaged on :"; DATE$
    RETURN

' GOSUB which prints statistics to a standard ASCII file for use in LOTUS eg.
filout:
    Q$ = CHR$(34)
    c$ = ","
    QC$ = Q$ + c$ + Q$
    OPEN "o", #1, "Statsfil.vpp"
    PRINT #1, Q$; "Variable statistics for : "; F$; Q$
```

```
      PRINT #1, Q$ + "Variable" + QC$ + "Variance" + QC$;
      PRINT #1, "Average" + QC$ + "Maximum" + QC$ + "Minimum"
      FOR I% = 1 TO NUMVARS%
        PRINT #1, Q$; varnam$(I%); Q$; c$; CSNG(varnz#(I%)); c$;
        PRINT #1, CSNG(AVE#(I%)); c$; colmaxs!(I%); c$; colmins!(I%)
      NEXT I%
      CLOSE #1
      RETURN

'Now we get to do the scaling....
39    IF TIMmode% THEN EXIT SUB     ' bye Tim !
      COLOR 15, 13
      FOR I% = 22 TO 25
        LOCATE I%, 1
        PRINT STRING$(79, " ");
      NEXT I%
      LOCATE 22, 1
      PRINT " Enter User-name of scaling format : <"; scal$; " > ";
      INPUT a$
      IF a$ <> "" THEN scal$ = LEFT$(a$, 20)
      LOCATE 22, 2
      PRINT scal$; SPACE$(60);
      LOCATE 23, 10
      INPUT ; "Save scaling format (Y/N)"; a$
      savscal$ = LEFT$(UCASE$(a$), 1)
      IF savscal$ = "Y" THEN
        LOCATE 23, 2
        INPUT ; "Name of User scaling format file to save :", fi$
        fi$ = UCASE$(fi$)
        SCFILE$ = FRTEXT(fi$, formrt$, formext$)
        IF SCFILE$ = "" THEN savscal$ = "N"
        fi$ = formrt$ + SCFILE$ + formext$
        LOCATE 23, 2
        IF savscal$ = "Y" THEN
          PRINT "Saving Scale format file : "; fi$; SPACE$(25);
          CALL sclforsav(fi$, scal$, SCMOD%(), SCVAL#(), scdiv#())
        END IF
```



```
        FOR I% = 23 TO 25
          LOCATE I%, 1
          PRINT SPACE$(79);
        NEXT
      END IF
      LOCATE 25, 20
      COLOR yellow, bgrnd
      PRINT "Now Scaling the Data";
      FOR I% = 1 TO NUMVARS%
        SELECT CASE SCMOD%(I%)
        CASE 1        ' auto scaling
          vl# = AVE#(I%)
          dv# = SQR(varnz#(scdiv#(I%)))
          MD% = 3
          CALL scalecol(mess#(), I%, MD%, vl#, dv#)
          IF SCVAL#(I%) <> 0 THEN
            vl# = -SCVAL#(I%)
            dv# = 1
            MD% = 3
            CALL scalecol(mess#(), I%, MD%, vl#, dv#)
          END IF

        CASE 2        ' range scaling
          CALL scalecol(mess#(), I%, 2, SCVAL#(I%), scdiv#(I%))
        CASE 3
          MD% = 3
          vl# = SCVAL#(I%)
          dv# = 1 / scdiv#(I%)
          CALL scalecol(mess#(), I%, MD%, vl#, dv#)
        CASE ELSE
          DELVAR%(I%) = -1
        END SELECT
      NEXT I%

END SUB

SUB AFA (FILE$)
```

```
' A Peter Wentzell routine (from MALINOWSKI)
' This subroutine performs the abstract factor analysis on the array D#
' The eigenvectors (column matrix ) are stored in EVEC# with the corresponding
' eigenvalues in EVAL# and the row matrix in D#
'
'       D#()    -  is the array of real data that is read from FILE$
'
'       EVEC#   - is the array that contains the eigenvectors.
'                 (first index indicates vector number)
'       EVAL#   - the eigenvalues
'
' Load in file and check for scaling

      PRINT "Loading file"
      CALL readpars(FILE$, NROWS%, NCOLS%, ext1%, ext2%)
      DIM d#(NROWS%, NCOLS%)
      DIM varnam$(NCOLS%)
      DIM time!(NROWS%)
      DIM cLass(NROWS%) AS RECRD
      DIM id(NROWS%) AS RECRD
      DIM EXTRA(NROWS%) AS RECRD
      DIM EXTEN$(NROWS%, ext2%)
      DIM EXNAM$(ext2%)
      DIM EXTS%(ext2%)
      CALL readvals(FILE$, varnam$(), d#(), id(), cLass(), time!(), EXTRA(),
        EXR$,EXNAM$(), EXTS%(), EXTEN$(), scaled$)

' Perform any desired scaling on the data
      DIM DELVAR%(NCOLS%)
      PRINT STRING$(60, " ");
      OUTFILE$ = FRTEXT$(FILE$, FR$, FX$)
      OUTFILE$ = FR$ + OUTFILE$ + ".AF2"
      CALL scaler(d#(), varnam$(), FILE$, scaled$, DELVAR%(), OUTFILE$)
      IF varnam$(0) = "EXIT" THEN EXIT SUB
      CLS
      LOCATE 3, 5
      LOCATE 3, 15
```



```
      PRINT "Now Performing Abstract Factor Analysis"
      LOCATE 5, 2
' Remove any variables from matrix that are flagged in DELVAR%().
' The columns must be compressed. This will also decrease the number of
' factors removed and speed computation.
      NSHIFT% = 0
      shift% = 0
' Find first variable to be removed and the number to be removed.
      FOR I% = 1 TO NCOLS%
        IF DELVAR%(I%) THEN
          NSHIFT% = NSHIFT% + 1
          IF shift% = 0 THEN shift% = I%
        END IF
      NEXT
      IF NSHIFT% > 0 THEN
        FOR J% = shift% TO NCOLS% - NSHIFT%
          DO
            shift% = shift% + 1
          LOOP UNTIL NOT DELVAR%(shift%)
          FOR K% = 1 TO NROWS%
            d#(K%, J%) = d#(K%, shift%)
          NEXT
        NEXT
      END IF
      NCOLS% = NCOLS% - NSHIFT%

' Calculate covariance (unnormalized) matrix and store in z#

      DIM z#(NCOLS%, NCOLS%)
      LOCATE 20, 5
      PRINT "Calculating covariance matrix :   1 /"; NCOLS%;
      FOR I% = 1 TO NCOLS%
        LOCATE 20, 37
        PRINT USING "###"; I%;
        FOR J% = 1 TO NCOLS%
          sum# = 0
          FOR K% = 1 TO NROWS%
```

```
            sum# = sum# + d#(K%, I%) * d#(K%, J%)
          NEXT
          z#(I%, J%) = sum#
        NEXT
      NEXT

' Covariance matrix calculated.
' Now determine eigenvectors and eigenvalues by iteration

      DIM temp#(NCOLS%)
      DIM evec#(NCOLS%, NCOLS%)
      DIM EVAL#(NCOLS%)
      faprox# = SQR(1# / NCOLS%)
      FOR I% = 1 TO NCOLS%
        FOR J% = 1 TO NCOLS%
          evec#(I%, J%) = faprox#
        NEXT
        EVAL#(I%) = 1
      NEXT

' Now loop to iterate each eigenvector to self-consistency
' Multiply eigenvector approxiamation by covariance matrix and normalise
' to get new eigenvector and eigenvalue. Check eigenvalue for self-consistncy

      LOCATE 22, 1
      PRINT SPACE$(79);
      LOCATE 22, 6
      PRINT "Now calculating vector #     of"; NCOLS%;
      FOR I% = 1 TO NCOLS%
        LOCATE 22, 31
        PRINT USING "###"; I%;
145     sumsq# = 0
        FOR J% = 1 TO NCOLS%
          temp#(J%) = 0
          FOR K% = 1 TO NCOLS%
            temp#(J%) = temp#(J%) + z#(J%, K%) * evec#(I%, K%)
          NEXT
```



```
          sumsq# = sumsq# + temp#(J%) ^ 2
        NEXT
        oldval# = EVAL#(I%)
        EVAL#(I%) = SQR(sumsq#)
        FOR J% = 1 TO NCOLS%
          evec#(I%, J%) = temp#(J%) / EVAL#(I%)
        NEXT
        icnt% = icnt% + 1
        IF icnt% = 1 GOTO 145
        test# = ABS((EVAL#(I%) - oldval#) / EVAL#(I%))
        IF test# > .000001 GOTO 145

' Self consistency attained - now calculate residual matrix to be used in
' determining next eigenvector, unless all have been attained.

        IF I% = NCOLS% GOTO 140
        icnt% = 0
        FOR J% = 1 TO NCOLS%
          FOR K% = 1 TO NCOLS%
            z#(J%, K%) = z#(J%, K%) - EVAL#(I%) * evec#(I%, J%) *
                 evec#(I%, K%)
          NEXT
        NEXT
140   NEXT

' Column matrix calculated, now calculate row matrix by multiplying original
' data matrix by inverse (transpose) of the column matrix. Row matrix is
' stored in original data by utilizing a temporary storage array.
      LOCATE 24, 1
      PRINT SPACE$(79);
      LOCATE 24, 5
      PRINT "Calculating scores matrix. (    /"; NROWS%; ")";
      FOR I% = 1 TO NROWS%
        LOCATE 24, 34
        PRINT USING "###"; I%;
        FOR J% = 1 TO NCOLS%
          temp#(J%) = 0
```

```
        FOR K% = 1 TO NCOLS%
          temp#(J%) = temp#(J%) + d#(I%, K%) * evec#(J%, K%)
        NEXT
      NEXT
      FOR J% = 1 TO NCOLS%
        d#(I%, J%) = temp#(J%)
      NEXT
    NEXT

' Abstract factor analysis complete, save file.
' Output results to file. The file is given the same name(and root)
' but is stored with the extension .AF2
' A file for factor analysis would normally only have the extensions
' .DES; .SCL; .DS1; .DS2

    LOCATE 15, 10
    PRINT "Output file = "; OUTFILE$
    Q$ = CHR$(34)
    OPEN "o", #2, OUTFILE$
    PRINT #2, NROWS%, NCOLS%
    PRINT #2, ext2%, ext1%
    IF ext1% <> 0 THEN PRINT #2, Q$ + EXR$ + Q$
    FOR n% = 1 TO ext2%
      PRINT #2, Q$ + EXNAM$(n%) + CHR$(&H20 OR EXTS%(n%)) + Q$
    NEXT
    FOR I% = 1 TO NCOLS%
      PRINT #2, EVAL#(I%)
      FOR J% = 1 TO NCOLS% - 1
        PRINT #2, evec#(I%, J%),
      NEXT
      PRINT #2, evec#(I%, NCOLS%)
    NEXT
    FOR I% = 1 TO NROWS%
      PRINT #2, Q$ + id(I%).R + Q$; Q$ + cLass(I%).R + Q$;
      PRINT #2, time!(I%), Q$ + EXTRA(I%).R + Q$;
      FOR J% = 1 TO ext2%
        PRINT #2, Q$ + EXTEN$(I%, J%) + Q$;
```



```
      NEXT
      FOR J% = 1 TO NCOLS% - 1
        PRINT #2, d#(I%, J%),
      NEXT
      PRINT #2, d#(I%, NCOLS%)
    NEXT
    PRINT #2, scaled$
    FIRS% = -1
    FOR I% = 1 TO NCOLS% + NSHIFT%
      IF NOT DELVAR%(I%) THEN
        IF NOT FIRS% THEN
          PRINT #2, ",";
        END IF
        PRINT #2, varnam$(I%);
        FIRS% = 0
      END IF
    NEXT
    PRINT #2,
    CLOSE #2

' Print out results file
    REPFILE$ = FRTEXT$(OUTFILE$, a$, B$)
    REPFILE$ = a$ + REPFILE$ + ".RES"
    DIM outfuncs!(5)
    DIM OUTFUNC$(5)
    REpos% = 0:     OUTFUNC$(REpos%) = " RE "
    IEpos% = 0:     OUTFUNC$(IEpos%) = " IE "
    INDpos% = 0:    OUTFUNC$(INDpos%) = " IND"
    CPVpos% = 5:    OUTFUNC$(CPVpos%) = " CPV"
    VARpos% = 4:    OUTFUNC$(VARpos%) = " VAR"
    EoApos% = 2:    OUTFUNC$(EoApos%) = "1/AV"
    rmsPOS% = 3:    OUTFUNC$(rmsPOS%) = " RMS"
    NACpos% = 1:    OUTFUNC$(NACpos%) = "1/1 "
    sum# = 0
    FOR I% = 1 TO NCOLS%
      sum# = sum# + EVAL#(I%)
    NEXT
```

```
    OPEN "O", #5, REPFILE$
    PRINT #5, "Summary of Factor analysis   "; DATE$
    PRINT #5, ""
    PRINT #5, FILE$, "   =>   "; scaled$; "  =>  "; OUTFILE$
    PRINT #5,
    PRINT #5, "  N   E-VALUE(I)    ";
    FOR I% = 1 TO 5
      PRINT #5, OUTFUNC$(I%); "        ";
    NEXT
    PRINT #5,
    PRINT #5, " ==   ==========    ";
    FOR I% = 1 TO 5
      PRINT #5, "====        ";
    NEXT
    PRINT #5,

    FOR I% = 1 TO NCOLS%
      SUM2Y# = 0
      FOR J% = I% + 1 TO NCOLS%
        SUM2Y# = SUM2Y# + EVAL#(J%)
      NEXT
      IF I% < NCOLS% THEN       ' skip over functions that aren't legal

' Real Error
          outfuncs!(REpos%) = SQR(SUM2Y# / (NROWS% * (NCOLS% - I%)))
' imbedded error
          outfuncs!(IEpos%) = SQR(SUM2Y# * I% / (CDBL(NROWS%) * NCOLS% *
               (NCOLS% - I%)))
' indicator function
          outfuncs!(INDpos%) = outfuncs!(REpos%) / (NCOLS% - I%) ^ 2
' change in eigenvalue (N ACceleration)
          outfuncs!(NACpos%) = EVAL#(I%) / EVAL#(I% + 1)
        ELSE 'set them to dummy values - don't worry, we won't print them *8-)
          outfuncs!(REpos%) = 111.111
          outfuncs!(IEpos%) = 111.111
          outfuncs!(INDpos%) = 111.111
          outfuncs!(NACpos%) = 111.111
```



```
        END IF
' Root Mean Square Error
        outfuncs!(rmsPOS%) = SQR(SUM2Y# / (NROWS% * NCOLS%))
' variance
        outfuncs!(VARpos%) = EVAL#(I%) / sum#
' CUMULATIVE PERCENT ERROR
        outfuncs!(CPVpos%) = 100 * (sum# - SUM2Y#) / sum#
' eigenvalue as fraction of average
        outfuncs!(EoApos%) = EVAL#(I%) * NCOLS% / sum#

        prnt$ = " ##  ##.###^^^^  "
        PRINT #5, USING prnt$; I%; EVAL#(I%);
        FOR P% = 1 TO 5
          IF outfuncs!(P%) = 111.111 THEN
            PRINT #5, "   ----     ";
          ELSE
            IF P% = CPVpos% THEN
              PRINT #5, USING " ###.###    "; outfuncs!(P%);
            ELSE
              PRINT #5, USING "##.##^^^^   "; outfuncs!(P%);
            END IF
          END IF
          IF P% = 5 THEN PRINT #5,
        NEXT
      NEXT

      PRINT #5,
      IF NSHIFT% > 0 THEN
        IF NSHIFT% = 1 THEN
          PRINT #5, "Variable removed from analysis : ";
        ELSE
          PRINT #5, "These variables removed from analysis :"
        END IF
        FOR I% = 1 TO NSHIFT%
          DO
            col% = col% + 1
          LOOP WHILE DELVAR%(col%) = 0
```

```
            PRINT #5, varnam$(col%)
          NEXT
        END IF
        CLOSE #5
        ERASE varnam$
        ERASE EVAL#
        ERASE evec#
        ERASE d#
        ERASE temp#
        ERASE DELVAR%
        ERASE z#
        FILE$ = OUTFILE$
END SUB

FUNCTION colmax! (MAT#(), col%, startrow%, ENDROW%, COLMIN!)
' returns the maximum value of the column COL% in matrix MAT# from STARTROW%
' to ENDROW%. COLMIN! is returned as the minimum value of the column

        CONST INIT! = -999
        cmax! = INIT
        COLMIN! = cmax!
        FOR nr% = startrow% TO ENDROW%
          vl# = MAT#(nr%, col%)
          IF vl# < COLMIN! OR COLMIN! = INIT THEN COLMIN! = vl#
          IF vl# > cmax! OR cmax! = INIT THEN cmax! = vl#
        NEXT nr%
        colmax! = cmax!

END FUNCTION

SUB cpvhelp
        COLOR 11, 12
        CLS
        LOCATE 1, 20
        PRINT "Help screen for CPV - Cumulative Percent Variance"
        PRINT
```



```
PRINT "   The cumulative percent variance is a measure of the percent of the
      total"
PRINT " variance in the data which is accounted for by the current set of
      primary"
PRINT " eigenvectors."
PRINT
PRINT "                    n          "
PRINT "              (N *      l )    "
PRINT "                   :1          "
PRINT "  CPV(N) =  ---------------    "
PRINT "                    c          "
PRINT "              (N *      l      "
PRINT "                   :1          "
PRINT "                               "
PRINT
PRINT "    The percent variance criterion accepts the set of largest
      eigenvalues"
PRINT " required to account for the variance within a chosen specification.
      The"
PRINT " problem is that in order to use a 98% (eg.) variance level, one must be
      able"
PRINT " to make an accurate estimate of the true variance in the data."

        CALL slep
        CLS
END SUB

' This routine draws a cross at position x%,y% on the screen. It will also
' erase the cross depending on the setting of CROSSCOL%
' SQUARE%() is an array which a 10x10 square is stored. The next call of this
' routine will replace the original picture (thereby erasing the previous
' crosshairs). OLDGET% is used to determine if there was a previous GET.
' XOLD%, YOLD% contain the coordinates that the array SQUARE refers to.
'
SUB CROSSHAIRS (X%, Y%, CROSSCOL%) STATIC
        DIM square%(45)
        IF oldget% THEN
```

```
        PUT (xold%, yold%), square%, PSET
        oldget% = 0
      END IF
      IF CROSSCOL% >= 0 THEN
                    xold% = X% - 4
        IF xold% < 0 THEN xold% = 0
        yold% = Y% - 5
        IF yold% < YMIN% THEN yold% = YMIN%
        GET (xold%, yold%)-(xold% + 9, yold% + 9), square%
        LINE (X%, Y% + 4)-(X%, yold%), CROSSCOL%
        LINE (xold%, Y%)-(X% + 5, Y%), CROSSCOL%
        oldget% = -1
      END IF
END SUB

' This routine is called by SCAT to draw a circle of color COLR% around
' the position X%, Y%. The radius (in plot units) is RAD%
' The previous image is contained in CRCLE%().
' This routine operates much the same way that CROSSHAIRS operates.
' BOXTHERE% is a boolean value that states whether there is already a box.
SUB eggbox (X%, Y%, RAD%, colr%, crcle%())

      STATIC boxthere%, xold%, yold%
      IF boxthere% THEN
        IF colr% >= 0 THEN
          PUT (xold%, yold%), crcle%, PSET
          boxthere% = 0
        ELSE
          boxthere% = 0
        END IF
      END IF
      IF colr% > 0 THEN
        GET (X% - RAD% - 1, Y% - RAD% - 1)-(X% + RAD% + 1, Y% + RAD% + 1),
      crcle%
        CIRCLE (X%, Y%), RAD%, 15, , , 1
        xold% = X% - RAD% - 1
        yold% = Y% - RAD% - 1
```



```
        boxthere% = -1
      ELSE
        ERASE crcle%
      END IF
END SUB

SUB eoahelp
      COLOR 11, 12
      CLS
      LOCATE 1, 20
      PRINT "Help-screen for 1/AV - ratio to average eigenvalue"
      PRINT

PRINT "   This column contains the eigenvalues divided by the average
      eigenvalue for"
PRINT " the entire factor space."
PRINT
PRINT "   The average eigenvalue proposed by Kaiser is based upon accepting
      all"
PRINT " eigenvalues with values above the average and rejecting those below."

      CALL slep
      CLS
END SUB

SUB evalhelp
      COLOR 11, 12
      CLS
      LOCATE 1, 20
      PRINT "Eigenvalue help-screen"
      PRINT

      PRINT "   The eigenvalues are calculated as the sum of the squares of
        the"
      PRINT " coefficients of the eigenvectors.  Therefore, the larger
        eigenvalues "
```

```
      PRINT " correspond to the eigenvectors (factors) that are more
        important."
      PRINT
      PRINT "   The eigenvectors can be split into two groups : those that
        account for"
      PRINT " real variations in the data (primary eigenvectors) and those
        that are due"
      PRINT " mainly to noise and experimental error (secondary
        eigenvectors)."
      CALL slep
      CLS
END SUB

SUB facdishelp
      COLOR 11
      CLS
      LOCATE 1, 20
      PRINT "Help screen for Factor Loadings Display"
      PRINT

PRINT "   This display shows the loadings of the Abstract Factors on each of
      the"
PRINT " original descriptors. The descriptor names are shown at the bottom of
      the "
PRINT " screen reading vertically. Above each descriptor is plotted the
      absolute "
PRINT " value of the normalized factor loadings for the factor which is listed
PRINT " at the upper right of the screen. The loadings for each factor are
      displayed" .
PRINT " in the same color across all descriptors and in the same color as the"
PRINT " Eigenvalue in the upper right. The factor loadings for a factor are
      always"
PRINT " displayed in the same color regardless of which factor"
PRINT " is the 'current' factor being viewed.  This is convenient when viewing
      a "
PRINT " simultaneous plot of six factors.  This can be done by pressing ";
      CHR$(34);
```



```
PRINT "I"; CHR$(34); "."
PRINT " In this display, the loadings for the six factors are all displayed at
      once."
PRINT " Each factor is displayed in its own color and the loadings are
      consecutive"
PRINT " from left to right for each descriptor. (NB. the loadings for the
      first"
PRINT " factor are always displayed as blue.)"
PRINT "   The vertical scale is normalized (0 to 1) for the current factor."
PRINT " In RELative display (default mode) the factor loadings are scaled
      according"
PRINT " to the ratio of eigenvalues (with the 'current' eigenvalue). "
PRINT " "
PRINT "   Use the left and right arrow keys to change the current factor."
PRINT "   Use the V key to toggle between ABSolute (no scaling) and RELative
      display."
PRINT "   Use the O key to print the factor loadings into a file for printing."

      LOCATE 25, 25
      PRINT "Press Any Key";
END SUB

SUB FACLODisplay (FILE$)

      YMIN% = 30
      YMAX% = 208
      CLS
      lft$ = CHR$(0) + CHR$(75)
      rgt$ = CHR$(0) + CHR$(77)
      c$ = "V-Scale axis for eigenvector D-Digital values for loadings ESC ?"
      com$ = SPACE$(80)
      LSET com$ = c$
      PRINT STRING$(30, "*");
      PRINT "Factor Loadings";
      PRINT STRING$(30, "*");
      PRINT
      PRINT "Reading factors from :"; FILE$
```

```
      CALL readpars(FILE$, NUMsams%, NUMVARS%, ext1%, ext2%)
      DIM EVecs!(NUMVARS%, NUMVARS%)
      DIM EVALS!(NUMVARS%)
      DIM varnam$(NUMVARS%)
      CALL readfacts(FILE$, NUMVARS%, EVALS!(), EVecs!(), varnam$())
      SCREEN 9
      COLOR 14, 0
      LOCATE 1, 1
      PRINT "File : "; FILE$
      LINE (12, YMIN%)-(12, YMAX% + 1), 15
      LINE -(640, YMAX% + 1), 15
      IVANMODE% = -1
' Print out variable names
      colr% = 11
      TB% = 1
      IF NUMVARS% < 39 THEN TB% = 2
      IF NUMVARS% < 25 THEN TB% = 3
      FOR I% = 1 TO NUMVARS%
        IF colr% = 11 THEN
          colr% = 14
        ELSE
          colr% = 11
        END IF
        COLOR colr%
        CALL verprint(varnam$(I%), 2 + TB% * I%, 16)
      NEXT
      FIRSTVEC% = 1
      EVSCALE% = -1
' Plot factor loadings
2
      COLOR 11
      LINE (13, YMIN%)-(649, YMAX%), 0, BF
      IF EVSCALE% THEN
        CALL verprint("ABS LOADING", 1, 4)
      ELSE
        CALL verprint("REL LOADING", 1, 4)
      END IF
```



```
      LOCATE 25, 1
      PRINT com$;
      IF IVANMODE% THEN
        lastvec% = FIRSTVEC%
      ELSE
        lastvec% = FIRSTVEC% + 5
        IF lastvec% > NUMVARS% THEN lastvec% = NUMVARS%
      END IF
      FOR J% = FIRSTVEC% TO lastvec%
        COLOR (J%) MOD 7 + 9
        FACSCALE% = (YMAX% - YMIN%)
        FOR I% = 1 TO NUMVARS%
          Xpos% = (1 + TB% * I%) * 8 + J% - FIRSTVEC%
          Ydis% = ABS(EVecs!(J%, I%)) * FACSCALE%
          IF EVSCALE% THEN
            Ydis% = Ydis% * (EVALS!(J%) / EVALS!(FIRSTVEC%))
          END IF
          IF Ydis% > 0 THEN LINE (Xpos%, YMAX%)-(Xpos%,YMAX% - Ydis% + 1)
          NEXT
        NEXT
        LOCATE 1, 40
        COLOR (FIRSTVEC%) MOD 7 + 9
        PRINT USING "Factor _###    EVAL = ##.###^^^^"; FIRSTVEC%,
            EVALS!(FIRSTVEC%)
        PRINT ev$;
7       CALL getkey(a$)
        SELECT CASE a$
        CASE CHR$(27)
          CLS
          SCREEN 0
          EXIT SUB
        CASE "v", "V"
          EVSCALE% = NOT EVSCALE%
        CASE "I", "i"
          IVANMODE% = NOT IVANMODE%
        CASE "O", "o"
          LOCATE 25, 1
```

```
        PRINT STRING$(80, " ");
        LOCATE 25, 1
        INPUT ; "Enter filename for output :", OF$
        IF OF$ <> "" THEN
          OPEN "o", #1, OF$
          PRINT #1, "Factor loadings for "; FILE$
          PRINT #1,
          START% = 1
1101      STEND% = NUMVARS%
          IF STEND% - START% > 5 THEN STEND% = START% + 4
          PRINT #1, "VARIABLE  ";
          FOR I% = START% TO STEND%
            PRINT #1, USING "FACTOR _###  "; I%;
          NEXT
          PRINT #1,
          PRINT #1, "--------";
          FOR I% = START% TO STEND%
            PRINT #1, "  ----------";
          NEXT
          PRINT #1,
          FOR J% = 1 TO NUMVARS%
            PRINT #1, varnam$(J%);
            FOR I% = START% TO STEND%
              PRINT#1,USING"  ##.###^^^^";CSNG(EVecs!(I%,J%));
            NEXT
            PRINT #1,
          NEXT
          PRINT #1, SPACE$(8);
          FOR I% = START% TO STEND%
            PRINT #1, "  ----------";
          NEXT
          PRINT #1,
          PRINT #1, "E-value:";
          FOR I% = START% TO STEND%
            PRINT #1, USING "  ##.###^^^^"; CSNG(EVALS!(I%));
          NEXT
          PRINT #1,
```



```
            IF STEND% < NUMVARS% THEN
              START% = STEND% + 1
              PRINT #1,
               GOTO 1101
            END IF
            CLOSE #1
          END IF
        CASE "d", "D"
          help% = 0
          SCREEN 9, , 1
          CLS
          PRINT " Variable     Loading on Factor #"; FIRSTVEC%;
          PRINT SPACE$(10); "Eigenvalue ="; EVALS!(FIRSTVEC%)
          PRINT " --------     ----------"
          COLOR 10
          START% = 1
          tabpos% = 1
9001      STEND% = NUMVARS%
          LOCATE 3
          IF STEND% - START% > 18 THEN STEND% = START% + 17
          FOR I% = START% TO STEND%
            P$ = "##) " + varnam$(I%) + "  ##.###^^^^"
            LOCATE , tabpos%
            PRINT USING P$; I%, CSNG(EVecs!(FIRSTVEC%, I%))
          NEXT
          IF STEND% < NUMVARS% THEN
            tabpos% = tabpos% + 20
            START% = STEND% + 1
            GOTO 9001
          END IF
          LOCATE 25, 1
          PRINT "Press any key...";
          SCREEN 9, , , 1
          CALL getkey(a$)
          SCREEN 9, , 0, 0
        CASE lft$
          FIRSTVEC% = FIRSTVEC% - 1
```

```
        IF FIRSTVEC% = 0 THEN FIRSTVEC% = NUMVARS%
      CASE rgt$
        FIRSTVEC% = FIRSTVEC% + 1
        IF FIRSTVEC% > NUMVARS% THEN FIRSTVEC% = 1
      CASE "?"
        IF help% THEN
          SCREEN 9, , 1, 1
        ELSE
          help% = -1
          SCREEN 9, , 1, 1
          CALL facdishelp
        END IF
        CALL slep
        SCREEN 9, , 0, 0
      CASE ELSE
        GOTO 7
      END SELECT
      GOTO 2
END SUB

FUNCTION FEXIST% (F$, LENGTH&)
'
'  This function opens F$ for random access read. LENGTH% is returned as the
' length of the file. If LENGTH& = 0 then F$ is erased and 0 is returned as
' the function value
      IF F$ <> "" THEN
        OPEN F$ FOR RANDOM ACCESS READ WRITE AS #1
        LENGTH& = LOF(1)
        CLOSE #1
        IF LENGTH& > 0 THEN
          FEXIST% = -1
        ELSE
          FEXIST% = 0
          KILL F$
        END IF
      END IF
END FUNCTION
```



```
FUNCTION FRTEXT$ (FILE$, ROOT$, ext$)
' This routine takes FILE$ and breaks it into root,file,extension
' The filename is returned as the function value.
' If no root or extension, then a null string is returned.
' If no filename, the function returns "NONAME$" as the value of FRTEXT$.
' The filename is the first eight charaters after the last '\' character.
' All spaces in FILE$ are removed.

    LENGTH% = LEN(FILE$)
    F$ = FILE$
    FILE$ = ""
    ROOT$ = ""
    FOR a% = 1 TO LENGTH%
      c$ = MID$(F$, a%, 1)
      SELECT CASE c$
      CASE " "
          ' THese are characters that are ignored - removed from FILE$
      CASE "/", ",", ";"
          ' These characters terminate the legal components of FILE$
          ' The string is chopped off here.
        EXIT FOR
      CASE ELSE
          ' The remaining possibilities are legal characters for FILE$
        FILE$ = FILE$ + c$
      END SELECT
    NEXT a%
    ext$ = ""
    a% = 1
    IF LEFT$(FILE$, 2) = ".." THEN a% = 3
    a% = INSTR(a%, FILE$, ".")
    IF a% <> 0 THEN
      ext$ = MID$(FILE$, a%, 4)
      ROOT$ = LEFT$(FILE$, a% - 1)
    ELSE
      ROOT$ = FILE$
    END IF
```

```
' We have the extension taken care of, now kick out the root.

      LENGTH% = LEN(ROOT$)
      a% = 0
      DO
        n% = a% + 1
        a% = INSTR(n%, ROOT$, "\")
      LOOP UNTIL a% = 0
      NAME$ = RIGHT$(ROOT$, LENGTH% - n% + 1)
      ROOT$ = LEFT$(ROOT$, n% - 1)
      a% = INSTR(NAME$, ":")
      IF a% = 2 THEN
        ROOT$ = LEFT$(NAME$, 2)
        NAME$ = RIGHT$(NAME$, LEN(NAME$) - 2)
      END IF
      NAME$ = RTRIM$(LEFT$(NAME$ + "        ", 8))
'       IF LEN(name$) = 0 THEN name$ = ""
' The above line is added so that a DEFAULT filename can be supplied
      FRTEXT$ = NAME$
END FUNCTION

FUNCTION GETDIR$
' This function takes a DIR of the current directory and saves it as
' DIRFIL.TMP. The file is then read into A$. It is assumed that the current
' directory appears on the third line, after a two-space tab. It was seen by
' DOS 3.30 that this is the case. Other versions of DOS may give different
' formats. (Indeed, DOS 4.01 does!)
      SHELL "dir > dirfil.tmp"
      OPEN "i", #1, "dirfil.tmp"
      LINE INPUT #1, a$
      LINE INPUT #1, a$
      LINE INPUT #1, a$
      a% = INSTR(a$, "  ")  ' two-spaces
      GETDIR = LTRIM$(RIGHT$(a$, LEN(a$) - a%))
      CLOSE #1
      KILL "dirfil.tmp"
END FUNCTION
```



```
SUB GETFIL (INFIL$, ext$, EXIST%)
' This routine gets a file name and checks it for proper format.
' FRTEXT$ is called to parse the file name into its parts.
' The extension is checked a/o set according to the value of EXT$ on calling.
'
' EXT$ on calling :
' -----
' - contains a list of legal extensions for FIL$ seperated by periods.
'   eg.  ".DS1.DES.SCL"
' - contains a supplied extension that is the ONLY legal extension.
'   eg.  "RES"
' - contains a supplied default extension AND other acceptable extensions.
'   eg.  "AF2.AFA"
' - is blank - meaning not to check the extension.
'
' NOTE : The difference between formats 1 and 3 is that in format 3, if none
' of the other legal extensions is given (in the example above - .AFA), then
' the extension is taken to be the first three characters preceded by a ".".
' In format 1, if no extension is given, FIL$ is returned as a blank.

      IF ext$ <> "" THEN
        IF LEFT$(ext$, 1) <> "." THEN
          DEFEXTEN$ = "." + LEFT$(ext$, 3)
          DEFEXTEN% = -1
          ext$ = RIGHT$(ext$, LEN(ext$) - 3)
          IF ext$ <> "" THEN
            EXTENSIONS$ = DEFEXTEN$ + ext$
            CHECKEXTEN% = -1
          END IF
        ELSE
          EXTENSIONS$ = ext$
          CHECKEXTEN% = -1
        END IF
      END IF
      F$ = FRTEXT$(INFIL$, FROOT$, FEXT$)
' CHECK FOR BAD EXTENSION
      IF CHECKEXTEN% THEN
```

```
        IF INSTR(EXTENSIONS$, FEXT$) = 0 OR FEXT$ = "" THEN
          IF DEFEXTEN% THEN
            FEXT$ = DEFEXTEN$
          ELSE
            INFIL$ = ""
            EXIST% = 0
            EXIT SUB
          END IF
        END IF
      ELSEIF DEFEXTEN% THEN
        FEXT$ = DEFEXTEN$
      END IF
      INFIL$ = FROOT$ + F$ + FEXT$
      EXIST% = FEXIST%(INFIL$, 1&)
END SUB

SUB getkey (key$)

      a$ = " "
' clear keyboard buffer. This ensures that the program doesn't run away from
      the user.
' It also annoys the pants off of people who are impatient. BWAHAHA
      WHILE a$ <> ""
        a$ = INKEY$
      WEND
      WHILE a$ = ""
        a$ = INKEY$
      WEND
      key$ = a$
END SUB

SUB iehelp
      CLS
      COLOR 11, 12
      LOCATE 1, 20
      PRINT "IE  -  Imbedded Error"
      PRINT
```



```
PRINT "   The imbedded error function can be used to determine the number of
      factors"
PRINT " in a data matrix without relying upon any estimate of the error."
PRINT
PRINT "                      c          1/2"
PRINT "              (N * Σ      l*)   "
PRINT "  IE(N) =          n: N+1       "
PRINT "             ---------------    "
PRINT "              rc * (c - n)      "
PRINT
PRINT " where :       -  N = eigenvector of interest"
PRINT "                  c = number of columns (eigenvectors)"
PRINT "                  r = number of rows (samples)"
PRINT "                  l*= eigenvalue"
PRINT "                   n            n"
PRINT
PRINT " The imbedded error is a function of the secondary eigenvalues."
PRINT " The IE function should decrease as we use more primary eigenvectors.
      However,"
PRINT " once we have exhausted the primary set, the IE should increase.(This
      may be"
PRINT " obscured if non-random errors are present)"

      CALL slep
      CLS
END SUB

SUB indhelp
      COLOR 11, 12
      CLS
      LOCATE 1, 20
      PRINT "Factor Indicator function"
      PRINT
      PRINT " Malinowski discovered an empirical function which appears to be
      more "
      PRINT " sensitive than the IE function."
      PRINT
```

```
        PRINT "                                "
        PRINT "  IND(N) =         RE           "
        PRINT "                 -------------- "
        PRINT "                   (c - N) ^ 2  "
        PRINT
        PRINT " The IND function reaches a minimum when N equals the 'correct'
        number"
        PRINT "of factors."
        PRINT
        PRINT " where :      -  N = eigenvector of interest"
        RINT "                  c = number of columns (eigenvectors)"
        PRINT
        CALL slep
        CLS
END SUB

FUNCTION INSIDE% (X#, Y#, XD#, YD#, YSCALE#, XSCALE#, RAD%)
' This function is used by SCAT to determine if the pixel at X#,Y# is within
        RAD%
' number of pixel (determined by YSCALE# and XSCALE#) of the pixel at XD#,YD#.
        XDST# = ABS(X# - XD#)
        YDST# = ABS(Y# - YD#)
        ydist! = YDST# * YSCALE#
        xdist! = XDST# * XSCALE#
        dst! = SQR(xdist! ^ 2 + ydist! ^ 2)
        IF dst! <= RAD% THEN
          REs% = -1
        ELSE
          REs% = 0
        END IF
        INSIDE% = REs%
END FUNCTION

SUB MARK (X%, Y%, col%) STATIC
'This routine draws a little box at position x,y with color COL%
        PSET (X%, Y%), col%
        LINE (X% - 2, Y% - 2)-(X% + 2, Y% - 2), col%
```



```
        LINE (X% + 2, Y% - 2)-(X% + 2, Y% + 2), col%
        LINE (X% + 2, Y% + 2)-(X% - 2, Y% + 2), col%
        LINE (X% - 2, Y% + 2)-(X% - 2, Y% - 2), col%
END SUB

SUB nachelp
        COLOR 11, 12
        CLS
        LOCATE 1, 20
        PRINT "Help-screen for 1/l' - "
        PRINT
        PRINT "   As factors become less significant, the corresponding
          eigenvalues (l)"
        PRINT " decrease in value. By dividing the eigenvalue (l) by the
          eigenvalue for"
        PRINT " the next factor(l'), one can get an idea for the relative
          significance "
        PRINT "of the individual eigenvectors. "
        PRINT
        PRINT "   The values for 1/l' should change in a pseudo-linear fashion
          while the"
        PRINT " eigenvectors belong to the primary set. The boundary between the
          primary set"
        PRINT " of eigenvectors and the secondary (insignificant) set should be
          noted "
        PRINT "by a large 1/l' value followed by a very low value."
        CALL slep
        CLS
END SUB

' This subroutine takes an AFA file and creates a DEScriptor file of the
' abstract factors. The user is asked for the number of factors to be used.
SUB NEWDES (AFAFILE$, SUCCESS%)

        PRINT
        OPEN "i", #1, AFAFILE$
        ITYP% = INSTR("AFAAF2", RIGHT$(AFAFILE$, 3))
```

```
      INPUT #1, nr%, nc%
      PRINT "The file has"; nc%; " abstract factors."
6100  LOCATE 8, 1
      PRINT "How many do you wish to place in the new DEScriptor file";
      INPUT n%
      IF n% > nc% OR n% < 1 THEN
        PRINT "Legal values ("; 1; " - "; nc%; ")"
        GOTO 6100
      END IF
      PRINT
      afaf$ = FRTEXT$(AFAFILE$, Aroot$, aext$)
43    INPUT " What filename is to be used as the new DES file"; F$
      F$ = UCASE$(F$)
' If the first character is a ~ then the same root is to be used.
      IF LEFT$(F$, 1) = "~" THEN
        F$ = RIGHT$(F$, LEN(F$) - 1)
        OUTFILE$ = FRTEXT$(F$, FROOT$, FEXT$)
        OUTFILE$ = Aroot$ + OUTFILE$ + ".DS2"
      ELSE
        OUTFILE$ = FRTEXT$(F$, FROOT$, FEXT$)
        OUTFILE$ = FROOT$ + OUTFILE$ + ".DS2"
      END IF
      OPEN OUTFILE$ FOR RANDOM ACCESS READ AS #3
      EXIST% = LOF(3) > 0
      CLOSE #3
      IF EXIST% THEN
        PRINT "That file already exists. Overwrite (Y/N) ?"
        a$ = INPUT$(1)
        a$ = UCASE$(a$)
        IF a$ <> "Y" THEN 43
      END IF
      PRINT
      PRINT "New descriptor file = "; OUTFILE$
      PRINT
      Q$ = CHR$(34)
      DIM row#(nc%)
      DIM cLass AS RECRD
```



```
      DIM id AS RECRD
      DIM EXTEN AS STRING * 4
      DIM EXTRA AS STRING * 4
      OPEN "o", #3, OUTFILE$
      PRINT #3, nr%, n%
      INPUT #1, ext2%, ext1%
      PRINT #3, ext2%, ext1%
      IF ITYP% = 4 THEN
        IF ext1% <> 0 THEN
          INPUT #1, EXR$
          PRINT #3, Q$ + EXR$ + Q$
        END IF
        FOR Nn% = 1 TO ext2%
          INPUT #1, EXTEN$
          PRINT #3, Q$ + EXTEN$ + Q$
        NEXT
      END IF
' Skip over eigenvalues and eigenvectors in input file.
      FOR I% = 1 TO nc% * (nc% + 1)
        INPUT #1, row#(1)
      NEXT
' Write out column names to output file
      FOR I% = 1 TO n%
        WRITE #3, "FTR #" + STR$(I%)
      NEXT
' Input entire row, then print the first N% columns.
      FOR I% = 1 TO nr%
        INPUT #1, id.R, cLass.R, time!, EXTRA
        PRINT #3, Q$ + id.R + Q$; Q$ + cLass.R + Q$; time!, Q$ + EXTRA + Q$;
        FOR Nn% = 1 TO ext2%
          INPUT #1, EXTEN$
          PRINT #3, Q$ + EXTEN$ + Q$;
        NEXT
        FOR J% = 1 TO nc%
          INPUT #1, row#(J%)
        NEXT
        FOR J% = 1 TO n% - 1
```

```
            PRINT #3, row#(J%),
          NEXT J%
          PRINT #3, row#(n%)
        NEXT
        PRINT #3, "This file was produced by ABSCAT.BAS"
        PRINT #3, "The complete factor analysis file is : "; AFAFILE$
        PRINT #3, n%; " of "; nc%; " factors."
        PRINT #3, DATE$
        IF NOT EOF(1) THEN
          INPUT #1, scaled$
          PRINT #3, scaled$
        END IF
        CLOSE #3
        CLOSE #1
        SUCCESS% = -1
        AFAFILE$ = OUTFILE$
        LOCATE 24, 1
        PRINT "Press Any Key...";
        CALL slep
END SUB

SUB pmrk (XCORD#, YCORD#)
' This subroutine assumes the plotter is open as file #1
' This routine also assumes that the plotter is set up properly with the P1
' point at the origin of the graph to be drawn (not nescessarily the lower
' left of the paper)
'
' MRK$ is added to allow for user defined characters for labelling the
' points differently. An array will eventually be assigned to correspond
' the seperate points to their plot character.

        MRK$ = "99,4,0,0,8,-4,0,0,-8;"
        PRINT #1, "PA"; XCORD#; ","; YCORD#; ";"
        PRINT #1, "UC" + MRK$
END SUB
```



```
SUB readcol (F$, col#(), col%)
' This routine reads in one column (descriptor of F$)
' It is assumed that the extra records etc. are taken care of by a previous
' call to READPARS with the appropriate ROW#, dimensioning
'
' Parameters.
'============
' F$      - filename including extension
' col#()  - array returned with column values of file
' col%    - column to read in.

      FTYPE% = (INSTR(".DES.DS1.DS2.SCL.AFA.AF2", RIGHT$(F$, 4)) / 4) + 1
      SELECT CASE FTYPE%
      CASE 1, 4    ' DES , SCL
        OPEN "I", #1, F$
        INPUT #1, NUMsams%, NUMVARS%
        FOR n% = 1 TO NUMVARS%
          INPUT #1, varnam$
        NEXT
        FOR NX% = 1 TO NUMsams%
          INPUT #1, id$
          INPUT #1, cLass$
          FOR Ny% = 1 TO col%
            INPUT #1, col#(NX%)
          NEXT
          FOR Ny% = col% + 1 TO NUMVARS%
            INPUT #1, temp#
          NEXT
        NEXT
        CLOSE #1

      CASE 2     ' DS1
        OPEN F$ FOR RANDOM ACCESS READ AS #1 LEN = 4
        FIELD #1, 4 AS REC1$
        GET #1                          ' GET the first record
        tmp$ = LEFT$(REC1$, 2)          ' Parse out the first two bytes
        NUMsams% = CVI(tmp$)            ' Convert to number of signals
```

```
        tmp$ = RIGHT$(REC1$, 2)          ' Parse out the last two bytes
        NUMVARS% = CVI(tmp$)             ' Convert to no. of descriptors
        GET #1                           ' get the extra record
        ext2% = CVI(LEFT$(REC1$, 2))
        ext1% = ASC(RIGHT$(REC1$, 1)) AND &H7
        IF ext1% <> 0 THEN GET #1
        EXR$ = REC1$
        FOR n% = 1 TO ext2%
          GET #1
          GET #1                 ' skip extended record names
        NEXT
' Get the descriptor names
        FOR I% = 1 TO NUMVARS%
          GET #1
          GET #1
        NEXT I%
' Signal / descriptor information
        FOR I% = 1 TO NUMsams%
          GET #1                           ' signal ID
          GET #1                           ' signal class
          GET #1                           ' time of signal
          GET #1                           ' extra record
          FOR n% = 1 TO ext2%
            GET #1                         ' extended records
          NEXT
          FOR J% = 1 TO col%
            GET #1
          NEXT
          col#(I%) = CVS(REC1$)
          FOR J% = col% + 1 TO NUMVARS%
            GET #1
          NEXT
        NEXT
        CLOSE #1
      CASE 3, 3      ' DS2
        OPEN "I", #1, F$
        INPUT #1, NUMsams%, NUMVARS%
```



```
        INPUT #1, ext2%, ext1%
        IF ext1% <> 0 THEN INPUT #1, EXR$
        FOR n% = 1 TO ext2%
          INPUT #1, EXNAM$
        NEXT
        FOR n% = 1 TO NUMVARS%
          INPUT #1, varnam$
        NEXT
        FOR NX% = 1 TO NUMsams%
          INPUT #1, id$
          INPUT #1, clss$
          INPUT #1, time!
          INPUT #1, EXTRA$
          FOR n% = 1 TO ext2%
            INPUT #1, EXTEN$
          NEXT
          FOR Ny% = 1 TO col%
            INPUT #1, col#(NX%)
          NEXT
          FOR Ny% = col% + 1 TO NUMVARS%
            INPUT #1, temp#
          NEXT
        NEXT
        CLOSE #1
      CASE 5     '"AFA"
        OPEN "I", #1, F$
        INPUT #1, NUMsams%, NUMVARS%
        INPUT #1, ext2%, ext1%
'**** READ past eigenvalues and eigenvectors
        FOR I% = 1 TO NUMVARS% + 1
          FOR K% = 1 TO NUMVARS%
            INPUT #1, temp#
          NEXT
        NEXT I%
' ***** READ in all the values here, eh?
        FOR I% = 1 TO NUMsams%
          INPUT #1, id$
```

```
      INPUT #1, cLass$
      INPUT #1, time!
      INPUT #1, EXTRA$
      FOR K% = 1 TO col%
        INPUT #1, col#(I%)
      NEXT
      FOR K% = col% + 1 TO NUMVARS%
        INPUT #1, temp#
      NEXT
    NEXT
    CLOSE #1
CASE 6 '"AF2"
    OPEN "I", #1, F$
    INPUT #1, NUMsams%, NUMVARS%
    INPUT #1, ext2%, ext1%
    IF ext1% <> 0 THEN INPUT #1, EXR$
    FOR n% = 1 TO ext2%
      INPUT #1, EXNAM$
    NEXT
    FOR n% = 1 TO NUMVARS%
      INPUT #1, EIGENVAL!
      FOR M% = 1 TO NUMVARS%
        INPUT #1, EIGENVEC!
      NEXT
    NEXT
    FOR NX% = 1 TO NUMsams%
      INPUT #1, id$
      INPUT #1, cLass$
      INPUT #1, time!
      INPUT #1, EXTRA$
      FOR n% = 1 TO ext2%
        INPUT #1, EXTEN$
      NEXT
      FOR Ny% = 1 TO col%
        INPUT #1, col#(NX%)
      NEXT
      FOR Ny% = col% + 1 TO NUMVARS%
```



```
          INPUT #1, temp#
        NEXT
      NEXT
      CLOSE #1
    END SELECT
END SUB

SUB readfacts (FILE$, NUMFAX%, EVALS!(), EVecs!(), varnam$())
' This subroutine reads in the eigenvalues EVALS!() and eigenvectors EVECS!()
' from FILE$. The variable names VARNAM$() are read in from the file. Older
' files may not have them.
    OPEN "I", #1, FILE$
    INPUT #1, NUMsams%, NUMVARS%
    INPUT #1, ext2%, ext1%
    IF ext1% <> 0 THEN INPUT #1, EXR$
    FOR n% = 1 TO ext2%
      INPUT #1, EXNAM$
    NEXT
    FOR I% = 1 TO NUMFAX%
      INPUT #1, EVALS!(I%)
      FOR J% = 1 TO NUMVARS%
        INPUT #1, EVecs!(I%, J%)
      NEXT
    NEXT
' Skip the factor scores matrix so we can get at the descriptor names.
    FOR I% = 1 TO NUMsams%
      INPUT #1, id$, cLass$, time!, ext$
      FOR J% = 1 TO ext2%
        INPUT #1, extn$
      NEXT
      FOR J% = 1 TO NUMVARS%
        INPUT #1, descr!
      NEXT
    NEXT
    INPUT #1, scaled$
    LINE INPUT #1, DES$
    CLOSE #1
```

```
      a% = 0
      FOR I% = 1 TO NUMVARS%
        B% = a% + 1
        a% = INSTR(a% + 1, DES$, ",")
        IF a% = 0 THEN a% = LEN(DES$) + 1
        varnam$(I%) = MID$(DES$, B%, a% - B%)
      NEXT
END SUB

SUB readpars (FILE$, NUMsams%, NUMVARS%, ext1%, ext2%)
' This routine reads in the header information on the data file.
' NUMSAMS% and NUMVARS% are parameters returned by the routine.
'
' Variables
' =========
' The parameters returned by this subroutine are :
'
' NROWS   - number of samples stored in file
' NCOLS   - number of descriptors for each sample
' EXT1%   - value for the status of the 4-byte extra record. 0 if not used.
'           Otherwise, it conforms to the convention mentioned above.
'           1 = four byte string
'           3 = real number
'           5 = integer of form (00nn)
'           7 = long integer
' NEXT2%  - Number of extended records.
'
' The data files have the following formats
' DES
'=====
' This file is stored as a sequential ASCII file. This file format was later
' upgraded to allow for more info content.
'
'
'   NROWS, NCOLS                    Number of samples, # of variables
'
'   "Name of variable # 1"          NCOLS of these [ 1 to NCOLS ]
```



```
'                                   Name is 8 characters in quotes.
'   "Name of variable #2 "
'
'   ID ,CLSS,var#1,...,var#NCOLS
'                                    ID,CLSS are four-character strings
'                                    that contain the Identity (ie number),
'                                    record (not used). Following these on
'                                    each line are the NCOLS descriptors for
'                                    that sample. There are NROWS such lines.
'
'
' DS1
'=====
' This file is stored as a random access binary file but has more information
' stored. As well as the ID and CLASS of each sample, the time of acquisition,
' and an extra 4-byte record is stored with each signal. These appear on the
' line after the ID and CLASS but before the descriptors. As well, there is a
' status byte for the EXTRA record telling what type of variable it is and
' whether or not it is being used. The 4-bytes are always present.
' The status byte EXSTAT is |00100xxy| where
' y = 0 if record is not used. xx = 00 - four character string.
'     1 if record is used.           01 - real (single precision)
'                                    10 - integer of form (00nn)
'                                    11 - long integer.
' Note: the 1 in the third bit is set to one for compatability with previous
' versions.
' All fields are therefore set up as 4-bytes.
'
'
'   NROWS, NCOLS                     Number of samples, # of variables
'
'   NEXT,EXTSTAT                   ] Number of extended records, status of extra
'                                      record (0 if not used,1 if used)
'   "EXT " [only if EXTSTAT <>0] ] Name of extra record, 3 chars + a space
'
'   "XN1_ _ X" [only if NEXT >0   -NEXT times. Name of extended record.3 chars
'                                      + 4 spaces and status byte X.
```

```
"XN2 _ _X"

"Name of variable # 1"          NCOLS of these [ 1 to NCOLS ]
                                Name is 8 characters. (ie 2-4byte records)
"Name of variable #2 "

ID  CLSS TIME EXTR EXN1 EXN2
                                 NROWS times
var#1,var#2,...,var#NCOLS       ID,CLSS,TIME,EXTR are four-byte records
                                that contain the Identity (ie number),
                                Class,Time of acquisition,extra record.
                                Any extended records are inserted at this
                                point. Following these four (+NEXT2%)
                                fields are the NCOLS variables for that
                                sample. Stored as four-byte reals


DS2
=====
This file is stored as a sequential ASCII file but has the same format as
the DS1 file. The two can be interconverted with no loss of information.


NROWS, NCOLS                    Number of samples, # of variables

NEXT,EXTSTAT                  ] Number of extended records, status of extra
                                  record (0 if not used,1 if used)
"EXT " [only if EXTSTAT <>0] ] Name of extra record, 3 chars + a space

"XN1_ _ X" [only if NEXT >0   -NEXT times. Name of extended record.3 chars
                                + 4 spaces and status byte X. See note in
"XN2 _ _X"                      DS1 file for use of status byte.

"Name of variable # 1"          NCOLS of these [ 1 to NCOLS ]
                                Name is 8 characters in quotes.
"Name of variable #2 "
```



```
" ID " "CLSS" TIME#   "EXTR",
                                        NROWS lines containing
Any extended records go here

var#1,var#2,...,var#NCOLS
                                       ID,CLSS,TIME#,EXTR are four-byte records
                                       that contain the Identity (ie number),
                                       CLASS,time of the signal(real),and the
                                       EXTRa record. Any extended records are
                                       then placed followed by the NCOLS
                                       variables for that sample. There are
                                       NROWS such lines - one for each sample.

AFA,AF2
=======
These files contain the results of the factor analysis of a descriptor file.
They both have the same format and are written as sequential ASCII files.
The eigenvalues and eigenvectors appear first which is followed by the
loadings matrix in the same format as a DES file (ID,CLASS, followed by the
NCOLS loadings for that vector). The AF2 file also contains TIME, EXTRA
record and any extended records so that a DS1 file created from this is
complete.


NROWS, NCOLS                    Number of samples, # of variables

NEXT, EXstat%                   # of extended records, extra record status

NEXT,EXTSTAT                  ] Number of extended records, status of extra
                                  record (0 if not used,1 if used)
"EXT " [only if EXTSTAT <>0] ] Name of extra record, 3 chars + a space

"XN1_ _ X" [only if NEXT >0   -NEXT times. Name of extended record.3 chars
                                + 4 spaces and status byte X. See note in
"XN2 _ _X"                      DS1 file for use of status byte.

EVAL(1)                         - first eigenvalue
```

```
'
'   EVEC(1,1),EVEC(1,2),....        - linear vector containing first e-vector
'                                   - These two lines are repeated for each
'                                     eigenvector. (NFAX times - *)
'
'  " ID " "CLSS" R#1,.., R#NFAX
'                                     NROWS lines containing ID,CLSS and
'                                     NCOLS real values for the loadings.
'                                     AF2 also has extended records and time.
'
'
' NOTE : The AF2 file, as well as containing the extra/ extended records also
' has the added benefit of containing a trailing line containing the names of
' all the descriptors used from the original DEScriptor file.
'-------------------------------------------------------------------------

      SELECT CASE RIGHT$(FILE$, 3)
      CASE "DES"
        OPEN "I", #1, FILE$
        INPUT #1, NUMsams%, NUMVARS%
        CLOSE #1
        ext1% = 0
        ext2% = 0
      CASE "AFA", "AF2"
        OPEN "I", #1, FILE$
        INPUT #1, NUMsams%, NUMVARS%
        INPUT #1, ext2%, ext1%
        CLOSE #1
      CASE "DS1"
        OPEN FILE$ FOR RANDOM ACCESS READ AS #1 LEN = 4
        FIELD #1, 4 AS REC$
        GET #1                          ' First record
        tmp$ = LEFT$(REC$, 2)           ' split into half
        NUMsams% = CVI(tmp$)
        NUMVARS% = CVI(RIGHT$(REC$, 2))
        GET #1
```



```
        ext1% = ASC(RIGHT$(REC$, 1)) AND &H7
        ext2% = CVI(LEFT$(REC$, 2))         ' Convert extra record
        CLOSE #1
      CASE "DS2"
        OPEN "I", #1, FILE$
        INPUT #1, NUMsams%, NUMVARS%
        INPUT #1, ext2%, ext1%
        CLOSE #1
      CASE "SCL"
        OPEN "I", #1, FILE$
        INPUT #1, NUMsams%, NUMVARS%
        CLOSE #1
        ext1% = 0: ext2% = 0
      END SELECT
END SUB

SUB readvals (F$, varnam$(), mess#(), id() AS RECRD, class() AS RECRD, time!(),
      EXTRA() AS RECRD, EXR$, EXNAM$(), EXTS%(), EXTEN$(), scaled$)
' This routine is the second half of the read file sub program. The routine
' is broken in two halves in order for DIMensioning in the main module.
'
'
' Parameters.
'============
' F$      - filename including extension
' VARNAM$ - array returning names of descriptors
' MESS#() - array returning matrix of data
'       - if MESS# is passed as a 1x1 array then the values are not stored
'       - this is useful for reading in extra parameters only
' TIME!() - array returning times of acquisition
' ID()    - array of 4-byte strings of identification
' CLASS() - 4-character string containg class designation
' SCALED$ - string containg method of scaling if extension is ".SCL"
' EXR$    - 3-character string containing name of extra record
' EXTRA$()- array of 4-bytes containg extra records for each signal
' EXN$()  - array containg 7-char names of extended records
' EXNT%() - Array containg 1 byte codes for type of extended records. Same
```

```
'          convention as for EXT1% in READPARS.
' EXTEN$()- 2-D array containing 4-byte extended records for each signal.
'

      DIM EXTRA AS RECRD
      FTYPE% = (INSTR(".DES.DS1.DS2.SCL.AFA.AF2", RIGHT$(F$, 4)) / 4) + 1
      SELECT CASE FTYPE%
      CASE 1, 4    ' DES , SCL
        OPEN "I", #1, F$
        INPUT #1, NUMsams%, NUMVARS%
        FOR n% = 1 TO NUMVARS%
          INPUT #1, varnam$(n%)
        NEXT
        FOR NX% = 1 TO NUMsams%
          INPUT #1, id(NX%).R
          INPUT #1, cLass(NX%).R
          FOR Ny% = 1 TO NUMVARS%
            IF UBOUND(mess#, 2) = 1 THEN
              INPUT #1, temp#
            ELSE
              INPUT #1, mess#(NX%, Ny%)
            END IF
          NEXT
        NEXT
        IF FTYPE% = 4 THEN INPUT #1, scaled$
        CLOSE #1
      CASE 2     ' DS1
        OPEN F$ FOR RANDOM ACCESS READ AS #1 LEN = 4
        FIELD #1, 4 AS REC1$
        GET #1                              ' GET the first record
        tmp$ = LEFT$(REC1$, 2)              ' Parse out the first two bytes
        NUMsams% = CVI(tmp$)                ' Convert to number of signals
        tmp$ = RIGHT$(REC1$, 2)             ' Parse out the last two bytes
        NUMVARS% = CVI(tmp$)                ' Convert to no. of descriptors
        GET #1                              ' get the extra record
        ext2% = CVI(LEFT$(REC1$, 2))
        ext1% = ASC(RIGHT$(REC1$, 1)) AND &H7
```



```
        IF ext1% <> 0 THEN GET #1
        EXR$ = REC1$
        FOR n% = 1 TO ext2%
          GET #1
          EXNAM$(n%) = REC1$
          GET #1
          EXTS%(n%) = ASC(RIGHT$(REC1$, 1)) AND &H7
          EXNAM$(n%) = EXNAM$(n%) + LEFT$(REC1$, 3)
        NEXT
' Get the descriptor names
        FOR I% = 1 TO NUMVARS%
          GET #1
          tmp$ = REC1$
          GET #1
          varnam$(I%) = tmp$ + REC1$
        NEXT I%
' Signal / descriptor information
'
        FOR I% = 1 TO NUMsams%
          GET #1                            ' signal ID
          id(I%).R = REC1$
          GET #1                            ' signal class
          cLass(I%).R = REC1$
          GET #1                            ' time of signal
          time!(I%) = CVS(REC1$)
          GET #1                            ' extra record
          EXTRA(I%).R = REC1$
          FOR n% = 1 TO ext2%
            GET #1
            EXTEN$(I%, n%) = REC1$
          NEXT
          FOR J% = 1 TO NUMVARS%
            GET #1
            IF UBOUND(mess#, 2) > 1 THEN
              mess#(I%, J%) = CVS(REC1$)
            END IF
          NEXT
```

```
    NEXT
    CLOSE #1
  CASE 3       ' DS2
    OPEN "I", #1, F$
    INPUT #1, NUMsams%, NUMVARS%
    INPUT #1, ext2%, ext1%
    IF ext1% <> 0 THEN INPUT #1, EXR$
    FOR n% = 1 TO ext2%
      INPUT #1, EXNAM$(n%)
      EXTS%(n%) = ASC(RIGHT$(EXNAM$(n%), 1)) AND &H7
      EXNAM$(n%) = LEFT$(EXNAM$(n%), LEN(EXNAM$(n%)) - 1)
    NEXT
    FOR n% = 1 TO NUMVARS%
      INPUT #1, varnam$(n%)
    NEXT
    FOR NX% = 1 TO NUMsams%
      INPUT #1, id(NX%).R
      INPUT #1, cLass(NX%).R
      INPUT #1, time!(NX%)
      INPUT #1, EXTRA(NX%).R
      FOR n% = 1 TO ext2%
        INPUT #1, EXTEN$(NX%, I%)
      NEXT
      FOR Ny% = 1 TO NUMVARS%
        IF UBOUND(mess#, 2) > 1 THEN
          INPUT #1, mess#(NX%, Ny%)
        ELSE
          INPUT #1, temp#
        END IF
      NEXT
    NEXT
    CLOSE #1
  CASE 5     '"AFA"
    OPEN "I", #1, F$
    INPUT #1, NUMsams%, NUMVARS%
    INPUT #1, ext2%, ext1%
'**** READ past eigenvalues and eigenvectors
```



```
    FOR I% = 1 TO NUMVARS% + 1
      FOR K% = 1 TO NUMVARS%
        INPUT #1, temp#
      NEXT
    NEXT I%
' ***** READ in all the values here, eh?
    FOR I% = 1 TO NUMsams%
      INPUT #1, id(I%).R
      INPUT #1, cLass(I%).R
      INPUT #1, time!(I%)
      INPUT #1, EXTRA.R
      FOR K% = 1 TO NUMVARS%
        IF UBOUND(mess#, 2) > 2 THEN
          INPUT #1, mess#(I%, K%)
        ELSE
          INPUT #1, temp#
        END IF
      NEXT
    NEXT
    CLOSE #1
    FOR I% = 1 TO NUMVARS%
      varnam$(I%) = "Ftr #" + STR$(I%)
    NEXT
  CASE 6 '"AF2"
    OPEN "I", #1, F$
    INPUT #1, NUMsams%, NUMVARS%
    INPUT #1, ext2%, ext1%
    IF ext1% <> 0 THEN INPUT #1, EXR$
    FOR n% = 1 TO ext2%
      INPUT #1, EXNAM$(n%)
      EXTS%(n%) = ASC(RIGHT$(EXNAM$(n%), 1)) AND &H7
      EXNAM$ = RIGHT$(RIGHT$(EXNAM$(n%), LEN(EXNAM$(n%)) - 1) +
        SPACE$(7), 7)
    NEXT
    FOR n% = 1 TO NUMVARS%
      INPUT #1, EIGENVAL!
      FOR M% = 1 TO NUMVARS%
```

```
            INPUT #1, EIGENVEC!
          NEXT
        NEXT
        FOR NX% = 1 TO NUMsams%
          INPUT #1, id(NX%).R
          INPUT #1, cLass(NX%).R
          INPUT #1, time!(NX%)
          INPUT #1, EXTRA(NX%).R
          FOR n% = 1 TO ext2%
            INPUT #1, EXTEN$(NX%, n%)
          NEXT
          FOR Ny% = 1 TO NUMVARS%
            IF UBOUND(mess#, 2) > 2 THEN
              INPUT #1, mess#(NX%, Ny%)
            ELSE
              INPUT #1, temp#
            END IF
          NEXT
        NEXT
        CLOSE #1
        FOR I% = 1 TO NUMVARS%
          varnam$(I%) = "Ftr #" + STR$(I%)
        NEXT
      END SELECT
END SUB

SUB rehelp
      COLOR 11, 12
      CLS
      LOCATE 1, 20
      PRINT "RE - Real Error (aka Residual Standard Deviation)"
      PRINT
      PRINT "                         c         1/2"
      PRINT "                   (N * Σ     l*)   "
      PRINT "   RE(N) =              n: N+1  n   "
      PRINT "                   ---------------  "
      PRINT "                     r * (c - n)    "
```



```
      PRINT
      PRINT " The Real Error should be less than the estimated error in the
        data (error "
      PRINT " limits) when N = the 'correct' number of factors. RE will then
        decrease with"
      PRINT " N reaching zero only if there is no error in the data."
      CALL slep
      CLS
END SUB

SUB resfilhelp

      SCREEN , , 1, 1
      COLOR 11, 12
      CLS
      LOCATE 1, 20
      PRINT "Help window for RESULTS file"
      PRINT
      hecur% = 1
      LOCATE 22, 15
      PRINT "Use arrows to choose. Press RETURN to select.";
      DIM help$(9)
      help$(1) = " E-VALUE(1)   - Column containing the eigenvalues "
      help$(2) = "    1/1       - Eigenvalue N divided by Eigenvalue N+1"
      help$(3) = "    1/AV      - Eigenvalue N divided by the average
        Eigenvalue"
      help$(4) = "    RMS       - Root Mean Square Error"
      help$(5) = "    VAR       - Variance of data set accounted for by
        primary vectors"
      help$(6) = "    CPV       - Cumulative Percent Variance"
      help$(7) = "    RE        - Real Error"
      help$(8) = "    IE        - Imbedded Error"
      help$(9) = "    IND       - Factor Indicator Function"
93    COLOR 10, 12
      FOR I% = 1 TO 9
        LOCATE 7 + I%, 10
        PRINT help$(I%)
```

```
      NEXT I%
      COLOR 13, hlght
      LOCATE 7 + hecur%, 10
      PRINT LEFT$(help$(hecur%), 12)
      COLOR 10, 12
94    a$ = INKEY$
      IF a$ = "" GOTO 94
      IF a$ = CHR$(13) THEN GOTO 911
      IF a$ = CHR$(27) THEN
        CLS
        SCREEN , , 0, 0
        EXIT SUB
      END IF
      IF ASC(a$) <> 0 GOTO 94
      a$ = RIGHT$(a$, 1)
      SELECT CASE a$
      CASE CHR$(72)
        hecur% = hecur% - 1
        IF hecur% = 0 THEN hecur% = UBOUND(help$)
        GOTO 93
      CASE CHR$(80)
        hecur% = hecur% + 1
        IF hecur% > UBOUND(help$) THEN hecur% = 1
        GOTO 93
        CASE ELSE
        GOTO 94
      END SELECT
911
' call appropriate subroutine
      SELECT CASE hecur%
      CASE 1
        CALL evalhelp
      CASE 2
        CALL nachelp
      CASE 3
        CALL eoahelp
      CASE 4
```



```
        CALL rmshelp
      CASE 5
        CALL varhelp
      CASE 6
        CALL cpvhelp
      CASE 7
        CALL rehelp
      CASE 8
        CALL iehelp
      CASE 9
        CALL indhelp
      CASE ELSE
      END SELECT
      GOTO 93
END SUB

SUB rmshelp
      COLOR 11, 12
      CLS
      LOCATE 1, 20
      PRINT "RMS  -  Root Mean Square Error"
      PRINT
      PRINT "                                    1/2"
      PRINT "                  c                 "
      PRINT "                 Σ (1')             "
      PRINT " RMS(N) =       n:N+1 n            "
      PRINT "                ------------        "
      PRINT "                 r * c              "
      PRINT
      PRINT
      CALL slep
      CLS
END SUB

SUB scalecol (MAT#(), col%, Mode%, VALUE#, DIV#) STATIC
'
' This routine performs the scaling on the COL%'th column of MAT# according to
' the value of MODE% and (if nescessary the) VALUE#.
```

```
' MODE% 1 The column is autoscaled such that the mean of the column is VALUE#
'         and the total variance equals DIV#. For true autoscaling, the mean
'         should be scaled to 0 and the value of DIV# should be 1. To
'         scale a column such that it has the same variance relative to a
'         different column, DIV# should equal the quotient of the variances.
'         DIV# = VARIANCE(of other column) / VARIANCE(of COL%umn)
'       2 The column is range scaled such that the column maximum is VALUE#
'         and the minimum value is DIV#. For range scaling 0 - 1, VALUE#=1 and
'         DIV# = 0. NOTE: it does not matter that VALUE# be larger than DIV#
'         except that the scaled values will all be inverted about the column
'         mean.If VALUE# = DIV# then the values will all be set to VALUE#
'         This is how a column is removed from the calculation of distance.
'       3 Each element of the column has VALUE# subtracted from it and is
'         then divided by DIV#. This MODE% is called recursively from MODE%=1
'         and is included to allow multiplication of a column by a certain
'         value (DIV# = 1/FACTOR) or shifting a column by a VALUE# (NOTE:
'         DIV# should equal 1 to accomplish a shift and VALUE# should equal 0
'         for a multiplication only.

    NSAMS% = UBOUND(MAT#)
    SELECT CASE Mode%
    CASE 1
      varnz# = VARIANCE(MAT#(), col%, NSAMS%, average#)
      vl# = average# - VALUE#
      dv# = SQR(DIV# * varnz#)
      MD% = 3
      CALL scalecol(MAT#(), col%, MD%, vl#, dv#)
    CASE 2
      cmax! = colmax!(MAT#(), col%, 1, NSAMS%, cmin!)
      vl# = cmin!
      dv# = 1
      IF cmax! <> cmin! THEN dv# = (cmax! - cmin!) / (DIV# - VALUE#)
      MD% = 3
      CALL scalecol(MAT#(), col%, MD%, vl#, dv#)
      vl# = VALUE# * SGN(DIV# - VALUE#)
```



```
      dv# = 1
      MD% = 3
      CALL scalecol(MAT#(), col%, MD%, vl#, dv#)
    CASE 3
      FOR NX% = 1 TO NSAMS%
        MAT#(NX%, col%) = (MAT#(NX%, col%) - VALUE#) / DIV#
      NEXT NX%
    CASE ELSE
      STOP
    END SELECT
END SUB

SUB scaler (mess#(), varnam$(), F$, scaled$, DELVAR%(), OUTFILE$)
'
' This program will scale the matrix mess# according to user input. The method
' for scaling the rows is described in the SUBprogram SCaLECOL.

    SCALE% = 1
    scalmax! = 1
    scalmin! = 0
    COLOR 11, 13
    CLS
    SOUND 330, 1
    PRINT STRING$(80, "*")
    LOCATE 3, 25
    PRINT "SCALING FOR FACTOR ANALYSIS"
    LOCATE 5, 20
    PRINT "DATA FILE = "; F$
    LOCATE 7, 20
    PRINT "OUTPUT FILE = "; OUTFILE$
    LOCATE 22, 15
    PRINT "Use arrows to choose. Press RETURN to select.";
    DIM scle$(4)
    scle$(1) = "Auto Scale"
    scle$(2) = "Range Scale"
    scle$(3) = "No Scaling"
    scle$(4) = "Special Scaling Functions"
```

```
3     COLOR 10, 13
      LOCATE 7, 34
      PRINT OUTFILE$ + STRING$(46 - LEN(OUTFILE$), " ");
      FOR I% = 1 TO 4
        LOCATE 7 + I% * 2, 15
        PRINT LEFT$(STR$(I%) + ") " + scle$(I%) + SPACE$(30), 30)
      NEXT I%
      COLOR 13, hlght
      LOCATE 7 + SCALE% * 2, 15
      IF SCALE% > 0 THEN
        PRINT LEFT$(STR$(SCALE%) + ") " + scle$(SCALE%) + SPACE$(30), 30)
      ELSE
        LOCATE , 34
        PRINT OUTFILE$
      END IF
        COLOR 10, 13
        IF SCALE% = 2 THEN
          LOCATE 11, 55
          PRINT "Max = "; LEFT$(STR$(scalmax!) + SPACE$(18), 18)
          LOCATE 12, 55
          PRINT "Min = "; LEFT$(STR$(scalmin!) + SPACE$(18), 18)
        ELSE
          LOCATE 11, 55: PRINT SPACE$(24);
          LOCATE 12, 55: PRINT SPACE$(24);
        END IF
4     CALL getkey(a$)
        IF a$ = CHR$(13) THEN
          IF SCALE% > 0 THEN
            GOTO 11
          ELSE
5           LOCATE 8, 20
            INPUT ; "OUTPUT FILE =", OF$
            FX$ = "AF2"
            CALL GETFIL(OF$, FX$, EXIST%)
            IF OF$ = "" THEN
            GOTO 6
          ELSEIF EXIST% THEN
```



```
            PRINT " already exists. OVERWRITE (Y/N)";
            INPUT B$
            IF UCASE$(LEFT$(B$, 1)) <> "Y" THEN
              GOTO 6
            END IF
          END IF
          OUTFILE$ = OF$
        END IF
6       LOCATE 8, 20
        PRINT STRING$(60, " ");
        GOTO 3
      END IF
      IF a$ = CHR$(27) THEN
        varnam$(0) = "EXIT"
        EXIT SUB
      END IF
      IF ASC(a$) <> 0 GOTO 4
      a$ = RIGHT$(a$, 1)
      SELECT CASE a$
      CASE CHR$(72)
        SCALE% = SCALE% - 1
        IF SCALE% = -1 THEN SCALE% = UBOUND(scle$)
        GOTO 3
      CASE CHR$(80)
        SCALE% = SCALE% + 1
        IF SCALE% > UBOUND(scle$) THEN SCALE% = 1
        GOTO 3
      CASE CHR$(77)      ' the RIGHT key was pressed
        IF SCALE% = 2 THEN
          COLOR 14, hlght
          LOCATE SCALE% * 2 + 7, 61
          PRINT SPC(10);
          LOCATE SCALE% * 2 + 7, 61
          COLOR 14, 13
          INPUT num$
          scalmax! = VAL(num$)
          COLOR 14, hlght
```

```
        LOCATE SCALE% * 2 + 8, 61
        PRINT SPC(10);
        LOCATE , 61
        COLOR 14, 13
        INPUT num$
        scalmin! = VAL(num$)
      END IF
      GOTO 3
    CASE ELSE
      GOTO 4
    END SELECT
11  LOCATE 22, 1
    PRINT SPACE$(79)
    LOCATE 22, 14
    IF FEXIST%(OUTFILE$, 1&) THEN
      PRINT OF$; " already exists, Overwrite (Y/N) ?";
      CALL getkey(a$)
      IF UCASE$(a$) <> "Y" THEN 5
    END IF
    LOCATE 22, 1
    PRINT SPACE$(79)
    LOCATE 22, 30
    IF SCALE% = 4 THEN
      PRINT "Please wait, calculating variables"
    ELSE
      PRINT "Now Scaling ...."
    END IF
    ON SCALE% GOTO 12, 13, 14, 15
' The first choice is selected. This will be the Autoscaling function for all
' columns.
12  vl# = 0
    dv# = 1
    MD% = 1
    FOR nc% = 1 TO NUMVARS%
      CALL scalecol(mess#(), nc%, MD%, vl#, dv#)
    NEXT nc%
    scaled$ = "Auto Scaled"
```



```
    GOTO 40
' The second scaling feature was selected. This will be range scaling.
13  vl# = scalmax#
    dv# = scalmin!
    MD% = 2
    FOR nc% = 1 TO NUMVARS%
      CALL scalecol(mess#(), nc%, MD%, vl#, dv#)
    NEXT nc%
    scaled$ = "Range Scaled:" + STR$(scalmin!) + " to" + STR$(scalmax!)
    GOTO 40
' The third scaling feature is Not to scale. Tricky
14  scaled$ = "No Scaling"
    GOTO 40
' Now it's going to get interesting.
' We have to supply the column statistics
15  CALL advscal(mess#(), varnam$(), F$, scaled$, DELVAR%())
40  LOCATE 21, 30
END SUB

SUB SCAT (FILE$)
' ********************************************
'
'        SCATTER-GRAM By David Sibbald
'
'       University of British Columbia
' Laboratory for Automated Analytical Chemistry
'
'     Created May 30, 1988 (Payday!!)
'
' ********************************************
' QuickBASIC version 4.0
'
' SCATtergram is designed to read in the results
' from ABFACT and display a plot of factor vs.
' factor. Plotting is possible on a HP-ColorPro
' (7440) plotter.
```

```
CONST INIT# = 99999
CONST xmax% = 600
CONST YMIN% = 30
CONST YMAX% = 330
CONST classmax% = 25
COLOR 15, 0
CLS
PRINT "*****************  SCATTERGRAM    by  David Sibbald
  *************************"
PRINT
PRINT "Reading "; FILE$
CALL readpars(FILE$, NROWS%, NCOLS%, ext1%, ext2%)
DIM colnam$(0 TO NCOLS%)
colnam$(0) = "Time     "
DIM Xcol#(1 TO NROWS%)
DIM ycol#(1 TO NROWS%)
DIM cLass(1 TO NROWS%) AS RECRD
DIM id(1 TO NROWS%) AS RECRD
DIM time!(1 TO NROWS%)
DIM EXTEN$(NROWS%, ext2%)
DIM EXTRA(NROWS%) AS RECRD
DIM EXNAM$(ext2%)
DIM EXTS%(ext2%)
DIM row#(NROWS%, 1)' This is a flag to the subroutine not to read in
  ROW#
CALL readvals(FILE$, colnam$(), row#(), id(), cLass(), time!(), EXTRA(),
  EXR$, EXNAM$(), EXTS%(), EXTEN$(), scaled$)
FLNM$ = FRTEXT$(FILE$, FRT$, FEXT$)

' PETE wants to have the different classes plotted as different colors so...
' Search through each CLASS, every time we come on a different one, assign a
' new color. CLASSES() contains all the different classes. COLRS%() contains
' the list of possible colors. COLRMAX% is the number of colors stored in
' COLRS%(). The position of the CLASS in CLASSES() is the key to the color
' contained in COLRS%(). EG: The class found in element 5 of CLASSES() will
' appear as the color number in element 5 in COLRS%().
```



```
DIM CLASSES(0 TO classmax%) AS RECRD
DIM CLPN%(classmax%)
colrs$ = " 7101114131215 9 1 2 3 5 6"
COLRMAX% = LEN(colrs$) / 2 - 1
DIM colrs%(0 TO COLRMAX%)
FOR I% = 0 TO COLRMAX%
  colrs%(I%) = VAL(MID$(colrs$, (I% + 1) * 2 - 1, 2))
NEXT
numclasses% = 0
FOR I% = 1 TO NROWS%
  a% = 0
  WHILE a% <= numclasses% AND CLASSES(a%).R <> cLass(I%).R
    a% = a% + 1
  WEND
  IF a% > numclasses% THEN
    numclasses% = a%
    CLASSES(a%).R = cLass(I%).R
  END IF
  IF numclasses% = classmax% THEN EXIT FOR
NEXT I%

' We now have a linear array of CLASSES. For each point, all we need to do
' is figure out which of CLASSES it belongs to and then assign that color.
' Because there may be more classes than colors, the color assigned is found
' to be the MODulus of the division of the class number divided by COLRMAX%.

1000  SCREEN 0
      COLOR 15, 13
      CLS
      PRINT "FILE = "; FILE$
      PRINT
      PRINT "Select factors to plot -(0 to exit):"
      COLOR 10
      START% = 1
      tabpos% = 5
1001  STEND% = NCOLS%
      LOCATE 5
```

```
IF STEND% - START% > 16 THEN STEND% = START% + 16
FOR I% = START% TO STEND%
  LOCATE , tabpos%
  PRINT I%; ") "; colnam$(I%)
NEXT
IF STEND% < NCOLS% THEN
  tabpos% = tabpos% + 15
  START% = STEND% + 1
  GOTO 1001
END IF
COLOR 14
LOCATE 21, 50
PRINT " To see column statistics,"
LOCATE 22, 50
PRINT "    enter 'S' for Y axis."
COLOR 15
LOCATE 22, 5
PRINT "Enter y-axis (by number)";
INPUT ; " :", a2$
IF UCASE$(LEFT$(a2$, 1)) = "S" THEN
' calculate column stuff here
  CALL TIMPRINT(row#(), colnam$(), FILE$)
  GOTO 1000
END IF
a2% = VAL(a2$)
IF a2% = 0 THEN 4000
IF a2% < 1 OR a2% > NCOLS% GOTO 1000
LOCATE 23, 5
PRINT ; "Enter x-axis ";
IF RIGHT$(FILE$, 3) = "DS1" THEN PRINT "(0 to plot vs. time)";
INPUT ; " :", a1%
IF a1% < 0 OR a1% > NCOLS% GOTO 1000
IF a1% = 0 THEN ' check if TIME info available
  IF RIGHT$(FILE$, 3) <> "DS1" AND RIGHT$(FILE$, 3) <> "DS2" AND
      RIGHT$(FILE$, 3) <> "AF2" THEN GOTO 1000
  FOR I% = 1 TO NROWS%
    Xcol#(I%) = time!(I%)
```



```
    NEXT
  END IF
  xaxnm$ = colnam$(a1%)
  yaxnm$ = colnam$(a2%)

' Now that we know which factors to plot lets try to get them on the screen
' First read them in.
  LOCATE 25, 10
  COLOR 9
  PRINT "Please wait ... reading values from file.";
  COLOR 15
  IF a1% <> 0 THEN CALL readcol(FILE$, Xcol#(), a1%)
  CALL readcol(FILE$, ycol#(), a2%)

' That was easy, now find the max and min values of the factors.
  RXSCALE% = 0
  RYSCALE% = 0
  XSCALE# = 0
  YSCALE# = 0
  MIN1# = INIT#
  min2# = INIT#
  MAX1# = INIT#
  max2# = INIT#
  FOR I% = 1 TO NROWS%
    IF Xcol#(I%) > MAX1# OR MAX1# = INIT# THEN MAX1# = Xcol#(I%)
    IF Xcol#(I%) < MIN1# OR MIN1# = INIT# THEN MIN1# = Xcol#(I%)
    IF a2% > 0 THEN
      IF ycol#(I%) > max2# OR max2# = INIT# THEN max2# = ycol#(I%)
      IF ycol#(I%) < min2# OR min2# = INIT# THEN min2# = ycol#(I%)
    END IF
  NEXT

' Now that's done, find out where on the screen the axes go.
' Then calculate scale factor and offset to plot points on
' screen. X-axis first, hopefully, the y-scale will be the same.

  IF MAX1# * MIN1# >= 0 THEN
```

```
      IF MAX1# - MIN1# = 0 THEN
        XOFFSET% = xmax% \ 2
        XSCALE# = 1
      ELSEIF MAX1# <= 0 THEN
        XOFFSET% = xmax% - 1
        XSCALE# = .9 * (xmax% - 2) / (-MIN1#)
      ELSE
        XOFFSET% = 2
        XSCALE# = .9 * (xmax% - 2) / (MAX1#)
      END IF
    ELSE
      IF (MAX1# - MIN1# = 0) THEN
        XOFFSET% = xmax% / 2
        XSCALE# = .4 * (xmax% - 2) / MAX1#
      ELSE
        XOFFSET% = CDBL(xmax%) * (-MIN1#) / (MAX1# - MIN1#)
        XSCALE# = .9 * (xmax% - 2) / (MAX1# - MIN1#)
      END IF
    END IF
1700  IF a2% = 0 THEN
      YOFFSET% = (YMAX% - YMIN%) / 2
      YSCALE# = 1
    ELSE
      IF max2# * min2# >= 0 THEN
        IF max2# <= 0 THEN
          YOFFSET% = (YMAX% - YMIN% - 1)
          YSCALE# = .9 * (YMAX% - YMIN% - 2) / (-min2#)
        ELSE
          YOFFSET% = 2
          YSCALE# = .9 * (YMAX% - YMIN% - 2) / (max2#)
        END IF
      ELSE
        YOFFSET% = (YMAX% - YMIN%) * (-min2#) / (max2# - min2#)
        YSCALE# = .9 * (YMAX% - YMIN% - 2) / (max2# - min2#)
      END IF
    END IF
```



```
' xoffset% and xscale# are correct, now calculate whether x-scale will do
' for y-scale. ie. check position of min2 and max2 to see if they fall on the
' screen using xscale#

      TYSCALE# = YSCALE#
      TXSCALE# = XSCALE#
      IF a2% = 0 GOTO 2000
      posmax# = max2# * XSCALE# + YOFFSET%
      posmin# = min2# * XSCALE# + YOFFSET%
      IF posmin# > 0 AND posmax# < (YMAX% - YMIN%) THEN
        RYSCALE% = -1
      END IF

' now check if it is possible to scale x-axis with y-scale#. oNLY CHeck if it
' is NOT possible to scale Y-axis.
      IF NOT RYSCALE% THEN
        posmax# = MAX1# * YSCALE# + XOFFSET%
        posmin# = MIN1# * YSCALE# + XOFFSET%
        IF posmin# > 0 AND posmax# < xmax% THEN
          RXSCALE% = -1
        END IF
      END IF
      XCROSS% = XOFFSET%
      YCROSS% = YMAX% - YOFFSET%
2000  CLS
      SCREEN 9, , 0, 0
      keycode% = 0
      COLOR 4
      LINE (0, YMAX% - YOFFSET%)-(xmax%, YMAX% - YOFFSET%)
      LINE (XOFFSET%, YMAX%)-(XOFFSET%, YMIN%)
' draw tick marks on axes
      IF a2% = 0 GOTO 2500
      FOR I# = 0 TO xmax% STEP xmax% / 15
        LINE (XOFFSET% + I#, YMAX% - YOFFSET% - 1)-(XOFFSET% + I#, YMAX% -
            YOFFSET% + 1)
        LINE (XOFFSET% - I#, YMAX% - YOFFSET% - 1)-(XOFFSET% - I#, YMAX% -
            YOFFSET% + 1)
```

```
    NEXT
    stp# = xmax% / 15 * TYSCALE# / TXSCALE#
    IF stp# < 3 THEN stp# = 4
    IF stp# > YMAX% - YMIN% THEN stp# = YMAX%
    FOR I% = 0 TO (YMAX% - YMIN%) STEP stp#
      LINE (XOFFSET%-1,YMAX%-YOFFSET%-I%)-(XOFFSET%+1,YMAX% - YOFFSET% - I%)
      LINE (XOFFSET%-1,YMAX%-YOFFSET%+I%)-(XOFFSET%+1,YMAX% - YOFFSET% + I%)
    NEXT
2500 COLOR 11
    LOCATE 1, 6
    PRINT "Scatter-plot of "; FLNM$;
    COLOR 12
    LOCATE 1, 38
    PRINT " X-Axis = "; xaxnm$; "    ";
    IF a2% <> 0 THEN
      PRINT "Y-Axis = "; yaxnm$
      COLOR 7
      LOCATE 2, 11
      PRINT "Relative scale (Y-axis/X-axis) =";
      SC$ = LTRIM$(STR$(TXSCALE# / TYSCALE#))
      PO% = INSTR(SC$, ".")
      LE% = LEN(SC$)
      IF PO% = 0 THEN
        IF LE% < 6 THEN
          SCF$ = STRING$(LE% + 1, "#")
        ELSE
          SCF$ = "##.##^^^^"
        END IF
      ELSE
        IF PO% < 6 THEN
          SCF$ = STRING$(PO% + 1, "#")
          IF 6-PO% > 0 THEN SCF$ = SCF$ + "." + STRING$(6 - PO%, "#")
        ELSE
          SCF$ = "##.##^^^^"
        END IF
      END IF
      PRINT USING SCF$; CSNG(TXSCALE# / TYSCALE#)
```



```
      END IF
      COLOR 13
      FOR I% = 1 TO NROWS%
        XCORD% = Xcol#(I%) * TXSCALE# + XOFFSET%
        IF a2% <> 0 THEN
          YCORD% = ycol#(I%) * TYSCALE# + YOFFSET%
        ELSE
          YCORD% = YOFFSET%
        END IF
' Find color number for plotting.
        a% = 1
        DO WHILE a% <= numclasses%
          IF class(I%).R <> CLASSES(a%).R THEN
            a% = a% + 1
          ELSE
            EXIT DO
          END IF
        LOOP
        IF a% > numclasses% THEN
          col% = 11
        ELSE
          col% = colrs%(a% MOD COLRMAX%)
        END IF
        CALL MARK(XCORD%, YMAX% - YCORD%, col%)
      NEXT
2650  LOCATE 25, 1
      PRINT STRING$(80, " ");
      COLOR 12
      LOCATE 2, 1
      PRINT USING "##.##^^^^"; CSNG((-YMIN% + YMAX% - YOFFSET%) / TYSCALE#);
      LOCATE 25, 71
      PRINT USING "##.##^^^^"; CSNG((xmax% - XOFFSET%) / TXSCALE#);
      LOCATE 25, 1
      COLOR 15
      PRINT "(N-new axes I-Identify E-exit O-Output K-List Classes";
      IF RYSCALE% OR RXSCALE% THEN PRINT " Y-rescale axes";
      PRINT ")";
```

```
2100  CALL getkey(a$)
      a$ = UCASE$(a$)
      IF a$ = CHR$(27) GOTO 1000
      IF a$ = "E" GOTO 1000
      IF a$ = "I" AND a2% <> 0 GOTO 5000
      IF a$ = "N" GOTO 1000
      IF a$ = "K" GOTO 6000
      IF a$ = "O" GOTO 3000
      IF a$ = "Y" AND (RYSCALE% OR RXSCALE%) THEN
        IF RYSCALE% THEN
          TYSCALE# = YSCALE# + XSCALE# - TYSCALE#
        ELSE
          TXSCALE# = XSCALE# + YSCALE# - TXSCALE#
        END IF
        GOTO 2000
      END IF
      GOTO 2100
' This section deals with identifying individual points on the graph
5000  LOCATE 25, 1
      crsscolor% = 15
      CALL CROSSHAIRS(XCROSS%, YCROSS%, crsscolor%)
5050  LOCATE 25, 1
      PRINT "Use cursor keys to move cross. Press return to place.ESC-exit.?-
        Help.";
5100  LOCATE 2, 56
      XD# = (XCROSS% - XOFFSET%) / TXSCALE#
      YD# = (-YCROSS% - YOFFSET% + YMAX%) / TYSCALE#
      PRINT USING "X =##.##^^^^"; CSNG(XD#);
      PRINT USING " Y =##.##^^^^"; CSNG(YD#);
5101  CALL getkey(a$)
      IF a$ = "?" OR a$ = "/" THEN
        GOSUB 5102
        keycode% = 0
        GOTO 5101
      END IF
      IF LEFT$(a$, 1) = CHR$(0) THEN a$ = RIGHT$(a$, 1)
      SELECT CASE ASC(a$)
```



```
CASE 75    ' left
  IF XCROSS% > 0 THEN XCROSS% = XCROSS% - 1
CASE 77    ' right
  IF XCROSS% < xmax% THEN XCROSS% = XCROSS% + 1
CASE 72    ' up
  IF YCROSS% > YMIN% THEN YCROSS% = YCROSS% - 1
CASE 80    ' down
  IF YCROSS% < YMAX% THEN YCROSS% = YCROSS% + 1
CASE 71    ' home
  IF XCROSS% > XMIN% + 10 THEN
    XCROSS% = XCROSS% - 10
  END IF
CASE 55     ' SHIFT HOME
  IF XCROSS% > XOFFSET% THEN
    XCROSS% = XOFFSET%
  ELSE
    XCROSS% = XMIN%
  END IF
CASE 79   ' end (SHIFT)
  IF XCROSS% < xmax% - 10 THEN
     XCROSS% = XCROSS% + 10
  END IF
CASE 49                          ' SHIFT END
  IF XCROSS% < XOFFSET% THEN
    XCROSS% = XOFFSET%
  ELSE
    XCROSS% = xmax%
  END IF
CASE 73, 56                      ' pg-up, SHIFT UP
  IF YCROSS% > YMIN% + 10 THEN
    YCROSS% = YCROSS% - 10
  ELSE
    YCROSS% = YMIN%
  END IF
CASE 81, 50                      ' pg-dn, SHIFT DOWN
  IF YCROSS% < YMAX% - 10 THEN
    YCROSS% = YCROSS% + 10
```

```
    ELSE
      YCROSS% = YMAX%
    END IF
  CASE 54     ' SHIFT right
    IF XCROSS% < xmax% - 10 THEN
      XCROSS% = XCROSS% + 10
    ELSE
      XCROSS% = xmax%
    END IF
  CASE 52     ' SHIFT left
    IF XCROSS% > 10 THEN
      XCROSS% = XCROSS% - 10
    ELSE
      XCROSS% = 0
    END IF
  CASE 57                                  ' SHIFT PGUP
    IF YCROSS% > YMAX% - YOFFSET% THEN
      YCROSS% = YMAX% - YOFFSET%
    ELSE
      YCROSS% = YMIN% + 1
    END IF
  CASE 51 ' SHIFT PGDN
    IF YCROSS% < YMAX% - YOFFSET% THEN
      YCROSS% = YMAX% - YOFFSET%
    ELSE
      YCROSS% = YMAX% - 1
    END IF
  CASE 13    ' return
    IF XCROSS% > xmax% - 3 OR XCROSS% < 3 THEN
      SOUND 45, 1
      GOTO 5101
    ELSEIF YCROSS% > YMAX% - 3 OR YCROSS% < YMIN% + 3 THEN
      SOUND 45, 1
      GOTO 5101
    END IF
    boxcolor% = 7
    radius% = 2
```



```
5199  LOCATE 25, 1
      PRINT "Use cursor keys to select area.ENTER to select.?-Help.ESC-
        exit.";
      DIM crcle%(maxdim!)
5200  CALL eggbox(XCROSS%, YCROSS%, radius%, boxcolor%, crcle%())
5201  CALL getkey(B$)
      IF B$ = "?" OR B$ = "/" THEN
        keycode% = 0
        GOSUB 5203
        GOTO 5201
      END IF
      IF LEFT$(B$, 1) = CHR$(0) THEN B$ = RIGHT$(B$, 1)
      SELECT CASE ASC(B$)
      CASE 75, 72   ' left,up
        GD% = 0
        IF (XCROSS% - radius% > 1) THEN
          IF (YCROSS% - radius% > YMIN% + 1) THEN
            IF (YCROSS% + radius% < YMAX% - 1) THEN
              radius% = radius% + 1
              GD% = -1
            END IF
          END IF
        END IF
        IF NOT GD% THEN
          SOUND 60, 1
          GOTO 5201
        END IF
      CASE 77, 80   ' right, down
        IF radius% > 2 THEN
          radius% = radius% - 1
        ELSE
          SOUND 60, 1
          GOTO 5201
        END IF
      CASE 71     ' home
        radius% = 6
      CASE 79     ' end
```

```
  CASE 73, 56, 52   ' pg-up,shift up,shift left
    GD% = 0
    IF (XCROSS% - radius% > 10) THEN
      IF (YCROSS% - radius% > YMIN% + 10) THEN
        IF (YCROSS% + radius% < YMAX% - 10) THEN
          radius% = radius% + 10
          GD% = -1
        END IF
      END IF
    END IF
    IF NOT GD% THEN
      SOUND 60, 1
      GOTO 5201
    END IF
  CASE 81, 50, 54   ' pg-dn,shift right,shift down
    IF radius% > 11 THEN
      radius% = radius% - 10
    ELSE
      radius% = 2
    END IF
  CASE 13    ' return
    GOTO lmems
  CASE 27    ' ESC
    CALL eggbox(XCROSS%, YCROSS%, RAD%, 0, crcle%())
    'ERASE crcle%
    GOTO 5050
  CASE ELSE
    GOTO 5201
  END SELECT
  GOTO 5200

CASE 27    ' ESC
  CALL CROSSHAIRS(XCROSS%, YCROSS%, -1)
  GOTO 2650
CASE ELSE
END SELECT
CALL CROSSHAIRS(XCROSS%, YCROSS%, crsscolor%)
```



```
      GOTO 5100

 5102
 ' Print out functions of cursor keys during ID operation
      SCREEN 9, , 1, 1
      COLOR 11, 0
      CLS
      PRINT "          Moving cross"
      PRINT
      PRINT "    KEY            FUNCTION"
      PRINT "    ---            --------"
      PRINT " Up, Down        Moves cursor"
      PRINT " Left, Right       by one pixel"
      PRINT
      PRINT " PgUp, PgDn      Moves cursor up/down/left/right"
      PRINT " Home, End        by ten pixels "
      PRINT " SHIFTed arrows   "
      PRINT
      PRINT " SHIFT Home      Moves cursor to left/right side of screen"
      PRINT " SHIFT End        or to axis"
      PRINT
      PRINT " SHIFT PgUp      Moves cursor to top/bottom of screen"
      PRINT " SHIFT PgDn       or to axis"
      PRINT
      PRINT " RETURN          Opens circle for defining area"
      PRINT
      PRINT " ESCape          Exits to previous menu"
      PRINT
      PRINT " ?               Print this screen"
      CALL slep
      COLOR 15
      CLS
      SCREEN 9, , 0, 0
      RETURN
 5203
 ' Print out functions of cursor keys during ID circle operation
      SCREEN 9, , 1, 1
```

```
        COLOR 11
        CLS
        PRINT "        Defining Area"
        PRINT
        PRINT "    KEY           FUNCTION"
        PRINT "    ---           --------"
        PRINT " Up, Left       Expand circle by one pixel unit"
        PRINT
        PRINT " Right, Down    Contract circle by one pixel unit"
        PRINT
        PRINT " Home           Reset circle to default (6 pixel radius)"
        PRINT
        PRINT " SHIFT Up       Expand circle by ten pixel units"
        PRINT " SHIFT Left    "
        PRINT " PgUp          "
        PRINT
        PRINT " SHIFT Down     Contract circle by ten pixel units"
        PRINT " SHIFT Right"
        PRINT " PgDn"
        PRINT
        PRINT " RETURN         IDentifies points in circle"
        PRINT
        PRINT " ESCape         Exits to previous menu"
        PRINT
        PRINT " ?              Print this screen"
        CALL slep
        COLOR 15
        CLS
        SCREEN 9, , 0, 0
        RETURN

' Display color codes for different classes
6000    IF keycode% THEN
          SCREEN 9, , 1, 1
        ELSE
          SCREEN 9, , 1, 0
          CLS
```



```
          PRINT "CLASS - COLOR #"
          PRINT "=====   ======="
          PRINT
          SCREEN 9, , 1, 1
          START% = 1
          tabpos% = 2
2001      STEND% = numclasses%
          LOCATE 3
          IF STEND% - START% > 18 THEN STEND% = START% + 18
          FOR I% = START% TO STEND%
            LOCATE , tabpos%
            COLOR colrs%(I% MOD COLRMAX%)
            PRINT CLASSES(I%).R; "    "; colrs%(I% MOD COLRMAX%)
          NEXT
          IF STEND% < numclasses% THEN
            tabpos% = tabpos% + 18
            START% = STEND% + 1
            GOTO 2001
          END IF
          IF numclasses% = 25 THEN
            COLOR 11
            LOCATE , tabpos%
            PRINT "OTHER"; "  "; 11
          END IF
          LOCATE 25, 1
          COLOR 13
          PRINT "PRESS ANY KEY TO CONTINUE";
        END IF
6001    a$ = INPUT$(1)
        keycode% = -1
        SCREEN 9, , 0, 0
        GOTO 2100

lmems:
' This routine prints out the members that are contained in the circle
' defined by X%, Y%, and RAD%. This involves a tricky function that should
' only be attempted by a professional. DO NOT try this at home. (Yes, dear.)
```

```
' SCREEN 9 page two is used for display to allow concurrent viewing of the
' elements and their ID's
'
      keycode% = 0
      LOCATE 25, 1
      PRINT "press any key... "; SPACE$(54);
      SCREEN 9, , 1, 0
      CLS
      PRINT "Identification of points centered about ";
      COLOR 12
      PRINT RIGHT$(STRING$(8, " ") + colnam$(a1%), 8);
      COLOR 15
      PRINT USING " = ##.####^^^^ "; CSNG(XD#);
      PLUSMINUS$ = CHR$(241) + "##.###^^^^ "
      PRINT USING PLUSMINUS$; CSNG(radius% / TXSCALE#);
      LOCATE 2, 41
      COLOR 12
      PRINT RIGHT$(STRING$(8, " ") + colnam$(a2%), 8);
      COLOR 15
      PRINT USING " = ##.####^^^^ "; CSNG(YD#);
      PRINT USING PLUSMINUS$; CSNG(radius% / TYSCALE#);
      LOCATE 3, 1
      PRINT " ID   CLASS    X-value      Y-value          Distance from
        crosshairs"
      LINE (0, 41)-(640, 41)
      LOCATE 24, 1
      PRINT "Space bar to see next page, B for previous page, P to see plot,
      LOCATE 25, 1
      PRINT "F to create list file, C to change classification, ESC to exit.
' Find all members that belong, print them out in batches of 18, allow for
' scanning up and down through them PgUP, PgDn and flipping back and forth
' between the two screens.
      NMEMS% = 0
      I% = 1
      backend% = 1
5250  LOCATE 4, 1
      backstart% = backend%
```



```
      backend% = I%
      NUMMEMS% = 0
      WHILE NUMMEMS% < 19 AND I% <= NROWS%
        X# = Xcol#(I%)
        Y# = ycol#(I%)
        IF INSIDE%(X#, Y#, XD#, YD#, TYSCALE#, TXSCALE#, radius%) THEN
          NUMMEMS% = NUMMEMS% + 1
          PRINT id(I%).R; " "; cLass(I%).R;
          PRINT USING "    ##.####^^^^ "; CSNG(X#);
          PRINT USING " ##.####^^^^     "; CSNG(Y#);
          delta$ = CHR$(127)
          PRINT USING delta$ + "X=##.###^^^^     "; CSNG(ABS(XD# - X#));
          PRINT USING delta$ + "Y=##.###^^^^ "; CSNG(ABS(YD# - Y#))
        END IF
        I% = I% + 1
      WEND
      NMEMS% = NMEMS% + NUMMEMS%
      IF NUMMEMS% < 1 THEN
        SCREEN 9, , 0, 0
        SOUND 45, 2
        LOCATE 25, 1
        PRINT "No points selected...    Press any key...";
        CALL getkey(a$)
        GOTO 5199
      END IF
      SCREEN 9, , 1, 1
' clear screen below last printed line (but above command lines)
      LINE (0, (3 + NUMMEMS%) * 14)-(639, 320), 0, BF
' get some input here about whether to go to 5250 and what to change i% to.
' We need to keep track of the first member of each set of 18 so that we can
' start at the right place quickly, ie without scanning through the damn thing
' each time. Going forward is no problem. Perhaps it is best to allow going
' forward only. It is no problem to allow scanning backwards each time by one
' page only.
5260  LOCATE 25, 67
      COLOR 13
      PRINT CHR$(2);
```

```
      COLOR 15
5261  CALL getkey(a$)
      LOCATE 25, 67
      PRINT " ";
      SELECT CASE UCASE$(a$)
      CASE " "
        IF I% >= NROWS% THEN I% = 1
        GOTO 5250
      CASE "B"
        I% = backstart%
        NMEMS% = NMEMS% - NUMMEMS%
        GOTO 5250
      CASE "C"
        LOCATE 24, 1
        PRINT SPACE$(79);
        LOCATE 25, 1
        PRINT SPACE$(79);
        LOCATE 24, 1
        PRINT "Enter new classification for ALL points identified.";
        LOCATE 25, 1
        INPUT ; ":", B$
        IF B$ <> "" THEN
          classchange% = -1
          B$ = LEFT$(B$ + "    ", 4)
          EXIST% = 0
          a% = 1
          WHILE NOT EXIST% AND a% <= numclasses%
            EXIST% = (B$ = CLASSES(a%).R)
            a% = a% + 1
          WEND
          IF NOT EXIST% THEN
            numclasses% = numclasses% + 1
            CLASSES(numclasses%).R = B$
          END IF
          a% = 1
          WHILE B$ <> CLASSES(a%).R AND a% <= numclasses%
            a% = a% + 1
```



```
          WEND
          IF a% > numclasses% THEN
            col% = 11
          ELSE
            col% = colrs%(a% MOD COLRMAX%)
          END IF
          SCREEN 9, , 0, 1
          FOR ik% = 1 TO NROWS%
            X# = Xcol#(ik%)
            Y# = ycol#(ik%)
            IF INSIDE%(X#, Y#, XD#, YD#, TYSCALE#,TXSCALE#,radius%) THEN
              cLass(ik%).R = B$
              X% = X# * TXSCALE# + XOFFSET%
              Y% = -Y# * TYSCALE# - YOFFSET% + YMAX%
              CALL MARK(X%, Y%, col%)
            END IF
          NEXT ik%
          SCREEN 9, , 1, 1
        END IF
        GOTO lmems
      CASE "F"
        LOCATE 25, 1
        PRINT "Enter filename to create (ENTER to abort)";
        INPUT ; "", LIST$
        IF LIST$ = "" THEN 5262
        LIST$ = UCASE$(LIST$)
        OPEN "o", #5, LIST$
        PRINT #5, "Elements of : "; FILE$
        PRINT #5, "X axis = "; colnam$(a1%), "Y axis = "; colnam$(a2%)
        PRINT #5, USING "X radius = ##.###^^^^ "; XD#;
        PRINT #5, USING PLUSMINUS$; radius% / TXSCALE#
        PRINT #5, USING "y radius = ##.###^^^^ "; YD#;
        PRINT #5, USING PLUSMINUS$; radius% / TYSCALE#
        PRINT #5, "  ID    CLASS   X-value       Y-value   Distance from
          crosshairs"
        PRINT #5, "  __    _____  _________    _________ _____________
          _________"
```

```
        J% = 1
        backend% = 1
        WHILE J% <= NSAMS%
          X# = Xcol#(J%)
          Y# = ycol#(J%)
          IF INSIDE%(X#, Y#, XD#, YD#, TYSCALE#, TXSCALE#, radius%) THEN
            PRINT id(J%).R; " "; cLass(J%).R;
            PRINT #5, USING " ##.###^^^^"; X#;
            PRINT #5, USING " ##.###^^^^  "; Y#;
            PRINT #5, USING CHR$(127) + "X=##.###^^^^ "; XD# - X#;
            PRINT #5, USING CHR$(127) + "Y=##.###^^^^ "; YD# - Y#
          END IF
          J% = J% + 1
        WEND
        CLOSE #5
5262    LOCATE 25, 1
        PRINT"F to create list file, C to change class,ESC to exit.
      CASE CHR$(27)
        SCREEN 9, , 1, 0
        CLS
        SCREEN 9, , 0, 0
        GOTO 5299
      CASE "P"
        SCREEN 9, , 1, 0
        a$ = INPUT$(1)
        SCREEN 9, , 1, 1
      CASE ELSE
      END SELECT
      GOTO 5260
5299  CALL CROSSHAIRS(XCROSS%, YCROSS%, -1)
      CALL eggbox(XCROSS%, YCROSS%, RAD%, -1, crcle%())
      'ERASE crcle%
      GOTO 2000

' This section deals with output
3000  LOCATE 25, 1
      PRINT STRING$(80, " ");
```



```
      LOCATE 25, 1
      COLOR 11
      PRINT "(P- HP ColorPro plotter  S- SIGMAPLOT file  H- printer)";
      a$ = INKEY$
      a$ = ""
      WHILE a$ = ""
        a$ = INKEY$
      WEND
      IF UCASE$(a$) = "P" THEN 3004
      IF UCASE$(a$) = "S" THEN 3100
      IF UCASE$(a$) = "H" THEN 2650     ' NOT YET
      GOTO 2650

3004  CLS
      PRINT "Please make sure the plotter is turned on and paper is loaded."
      INPUT "Press RETURN (Q - to abort)", R$
      IF R$ = "q" OR R$ = "Q" THEN 3999
      PRINT
3005  OPEN "COM1:9600,S,7,1,RS,CS65535,DS,CD" FOR RANDOM AS #1
      PRINT #1, "IN;"
3006  PRINT #1, "OS;"

' pause while waiting for plotter to respond to OS command
      PRINT "Talking to plotter.";
      FOR I! = 1 TO 5000
      NEXT
      LOCATE 4, 1
      PRINT

      IF EOF(1) THEN
        LOCATE 4, 1
        PRINT "The plotter is not loaded with paper."
        PRINT "Please do so now and press return."
        PRINT "(Enter 'Q' to abort plot.)";
        INPUT R$
        IF UCASE$(LEFT$(R$, 1)) = "Q" GOTO 3999
```

```
' Control is now sent to 3005. Originally, the COM1 port was not closed and
' the control was passed back to 3006. However, at this point if the turkey
' sat and repeatedly pressed return without loading paper (3 or 4 times), then
' the plotter buffer would fill and the computer would hang - waiting for
' space in the plotter buffer. The pause is added to account for the delay
' time between OS command and the plotter response.
        CLOSE #1
        GOTO 3005
      END IF
      LINE INPUT #1, a$
      PRINT #1, CHR$(27) + ".P3:"
      CLS
      PRINT
      PRINT "Axes will be labelled. Filename and date will also appear on the
      plot."
      PRINT
3040  PRINT "Enter TITLE (maximum 42 characters)      |"
      LOCATE , 43
      PRINT "|";
      LOCATE , 1
      INPUT "", title$
      IF LEN(title$) > 42 GOTO 3040
' center title in field of 42 spaces
      cnt% = 42 - LEN(title$)
      cnt% = cnt% \ 2
      title$ = SPACE$(cnt%) + title$
3010  DIM pn%(3)
      LOCATE 12, 1
      INPUT "Enter pen number for plot title (0-8): ", pn%(1)
      IF pn%(1) < 1 OR pn%(1) > 8 GOTO 3010
      INPUT "         Pen number for date         : ", pn%(3)
3020  LOCATE 14, 1
      INPUT "         Pen number for axes         : ", pn%(2)
      IF pn%(2) < 1 OR pn%(2) > 8 GOTO 3020
3030  LOCATE 16, 1

' enter input here for labels on points
```



```
      FOR I% = 1 TO numclasses%
3031    PRINT "     Pen for signals of class  <"; CLASSES(I%).R; ">: ";
        INPUT "", CLPN%(I%)
        IF CLPN%(I%) < 1 OR CLPN%(I%) > 8 GOTO 3031
      NEXT
      INPUT "Pen number for signals with no class : ", B$
      CLPN%(0) = CLPN%(1)
      IF B$ <> "" THEN CLPN%(0) = VAL(B$)
' set dimensions on plotter. Please see HP7440A program manual for a list of
' what is being done here. If you want to change anything, you'll need it!!!
      PRINT #1, "pa0,6900;"
      PRINT #1, "DR;"
      PRINT #1, "SI.4,.5;"
      PRINT #1, "SP"; pn%(1)
      PRINT #1, "LB" + title$ + CHR$(3) + ";"
      PRINT #1, "PA0,7290;"
      PRINT #1, "SI.2,.2;"
      PRINT #1, "PA9000,7290;"
      IF LEN(FILE$ + FEXT$) > 11 THEN
        PRINT #1, "CP"; 11 - LEN(FILE$ + FEXT$); ",0;"
      END IF
      PRINT #1, "LB" + FILE$ + FEXT$ + CHR$(3) + ";"
      PRINT #1, "SP"; pn%(3); ";"
      PRINT #1, "LB" + DATE$ + CHR$(3) + ";"
      PRINT #1, "PA300,6675;"
      PRINT #1, "SI.3,.3;"
      PRINT #1, "LBPlot of " + CHR$(3)
      PRINT #1, "CP"; LEN(RTRIM$(colnam$(a2%))); ",0;"
      PRINT #1, "LB vs. " + CHR$(3)
      PRINT #1, "SP"; pn%(2)
      PRINT #1, "CP "; -LEN(RTRIM$(colnam$(a2%))) - 5; ",0;"
      PRINT #1, "LB" + RTRIM$(colnam$(a2%)) + CHR$(3) + ";"
      PRINT #1, "CP 5,0;"
      PRINT #1, "LB" + RTRIM$(colnam$(a1%)) + CHR$(3) + ";"
' Now that all the title and crap is printed let's draw some axes.

      PRINT #1, "LT;"
```

```
      PRINT #1, "PA0,"; YOFFSET% * 50 / 3 + 700; ";"
      PRINT #1, "PD;PR10000,0;"
      PRINT #1, "PU;"
      PRINT #1, "PA"; XOFFSET% * 50 / 3; ",700;"
      PRINT #1, "PD;PR0,5000;"
      PRINT #1, "PU;"

' First, how about remapping the paper to correspond to screen units.
      PRINT #1, "IW 0,700,10000,5700;"
      PRINT #1, "IP";XOFFSET%*50/3;",";YOFFSET% * 50/3+700; ",10000,5700;";
      YPLOT = CSNG((YMAX% - YOFFSET%) / TYSCALE#)
      IF YPLOT < 1 THEN YPLOT = 1
      PRINT #1, "SC0,"; CSNG((xmax% - XOFFSET%) / TXSCALE#); ",0,"; YPLOT; ";"

' The plotter will now directly map the coordinates of xcol# and ycol#
      PRINT #1, "SI .1,.1;"
      FOR c% = 1 TO numclasses%
        PRINT #1, "SP"; CLPN%(c%); ";"
        FOR s% = 1 TO NROWS%
          IF cLass(s%).R = CLASSES(c%).R THEN
            CALL pmrk(Xcol#(s%), ycol#(s%))
          END IF
        NEXT
      NEXT
' now plot all points that have no class in plot color (1)
      PRINT #1, "SP"; STR$(CLPN%(0)); ";"
      FOR I% = 1 TO NROWS%
        FOR c% = 1 TO numclasses%
          IF cLass(I%).R = CLASSES(c%).R THEN
            GOTO 901
          END IF
        NEXT
        CALL pmrk(Xcol#(I%), ycol#(I%))
901   NEXT
      PRINT #1, "SP;"
      PRINT #1, "IN;"
      PRINT #1, "PA0,7479;"
```



```
      CLOSE #1
3999  GOTO 2000

3100  LOCATE 25, 1
      PRINT STRING$(80, " ");
      LOCATE 25, 1
      INPUT ; "Enter filename for Sigmaplot output :"; spf$
      'put error checking here
      ' extension = ".ASP"
      LOCATE 25, 1
      PRINT STRING$(80, " ");
      LOCATE 25, 1
      PRINT ">TYPE "; spf$; " for SIGMAPLOT loading instructions.";
      Q$ = CHR$(34)
      OPEN "O", #1, spf$
      PRINT #1, Q$+"Output file from ABSCAT.  Source file  :"+FILE$+ext$ + Q$
      PRINT #1, Q$ + "x-axis = "; colnam$(a1%);" ; y-axis = ";colnam$(a2%); Q$
      PRINT #1, Q$+"Read into SIGMAPLOT with ";1+numclasses%;"columns and ";
      PRINT #1, "use ',' "; " as delimiter" + Q$
      PRINT #1, Q$+"Skip 5 fields. Column 1 = x-axis. Other columns are y-";
      PRINT #1, "axis" + Q$
      PRINT #1, Q$ + "x-axis" + Q$;
      FOR I% = 1 TO numclasses%
        PRINT #1, "," + Q$ + "CL=" + CLASSES(I%).R + Q$;
      NEXT
      PRINT #1,
      FOR I% = 1 TO NROWS%
        PRINT #1, CSNG(Xcol#(I%)); ",";
        FOR a% = 1 TO numclasses% - 1
          IF cLass(I%).R = CLASSES(a%).R THEN
            EXIT FOR
          ELSE
            PRINT #1, Q$ + " - " + Q$; ",";
          END IF
        NEXT
        PRINT #1, CSNG(ycol#(I%));
        FOR B% = a% + 1 TO numclasses%
```

```
            PRINT #1, ","; Q$ + " - " + Q$;
          NEXT
          PRINT #1,
        NEXT
        CLOSE #1
        COLOR 13
        PRINT "...Press any key...";
        CALL slep
        GOTO 2650

' If Mr. User has changed some of the classes of the signals, then it seems
' only fair to ask him/her/other if they should be written to the file. And
' HEY! the word "other" is sexist!!!!  Should be spelled "otheir" for
' neutrality. Right persiblings?

4000    IF classchange% AND RIGHT$(FILE$, 3) = "DS1" THEN
          LOCATE 25, 1
          PRINT "Do you wish the new classes to be written to the file";
          INPUT ; ""; a$
          IF UCASE$(LEFT$(a$, 1)) = "Y" THEN
            OPEN FILE$ FOR RANDOM ACCESS WRITE AS #1 LEN = 4
            FIELD #1, 4 AS REC$
            FOR I% = 1 TO NROWS%
              LSET REC$ = cLass(I%).R
              spot& = (I% + 1) * NCOLS% + 4 * (I% - 1) + 4
              PUT #1, spot&
            NEXT
            CLOSE #1
          END IF
        END IF
        SCREEN 0
END SUB

SUB SCLFORLOD (FILE$, SC$, Mode%(), a#(), B#())
' This subprogram is designed to read in a file saved by SCLFORSAV and parse
' it into the ADVSCAL subprogram so that the lazy Joe's don't have to type so
' much. Also it is nice to be able to ensure that scaling is consistent
```



```
' between data sets. If the number of descriptors is different between the
' current data set and the saved format file, then (uh oh) the program reads
' in only enough to use (ie. the extra formatting is ignored). If there is not
' enough in the file the variables left over are left as the default

        OPEN "i", #3, FILE$
        INPUT #3, SC$
        FOR I% = 1 TO UBOUND(Mode%)
          INPUT #3, Mode%(I%), a#(I%), B#(I%)
        NEXT
        CLOSE #3
END SUB

SUB sclforsav (FILE$, SC$, a%(), B#(), c#())
' See the notes on SCLFORLOD. And see if they aren't straight forward. The file
' has a simple ASCII format.
        OPEN "O", #2, FILE$
        WRITE #2, SC$
        FOR I% = 1 TO UBOUND(a%)
          WRITE #2, a%(I%), B#(I%), c#(I%)
        NEXT I%
        PRINT #2, "This file created on " + DATE$ + " at " + TIME$
        CLOSE #2
END SUB

SUB slep
' My version of QB 4.5's SLEEP command. Had to write this when 4.5 kept
' crashing.
        a$ = INKEY$: a$ = "": WHILE a$ = "": a$ = INKEY$: WEND
END SUB

SUB TIMPRINT (mess#(), varnam$(), F$)
' This routine was inspired by Timothy Crowthers' attempts at getting out
' the column statistics. So....     we need to call ADVSCAL and have the
' format changed slightly. Allow print screen and lotus file output if time
' permits
'
```

```
' The trick is, how do we call advscal without having the thing actually
' scale the data? Well, how about if we pass scal$ as "VIEW MODE". Now, if
' the ADVSCAL routine checks for "VIEW MODE" and behaves appropriately...

      scal$ = "VIEW MODE"
      DIM DELVAR%(1)      ' dummy array for parameter consistency
      CALL advscal(mess#(), varnam$(), F$, scal$, DELVAR%())
END SUB

SUB titlescrn
      COLOR 15, 11
      CLS
      LOCATE 3, 15
      PRINT "Astract Factor Analysis and Scattergram Utility."
      LOCATE 5, 35
      PRINT "ABSCAT"
      LOCATE 10, 18
      PRINT "Laboratory for Automated Chemical Analysis"
      LOCATE 12, 28
      PRINT "Department of Chemistry"
      LOCATE 12, 24
      PRINT "University of British Columbia"
      LOCATE 13, 25
      PRINT "Vancouver, British Columbia"
      LOCATE 23, 27
      COLOR 11
      PRINT "Press any key to continue."
END SUB

' This is a small attempt at a sense of humour. Approximately 6% of the time
' the front screen will piss somebody off for three seconds. Hope it isn't
' Adrian. (Note, this routine is obviously not mandatory - but I can think of
' better ones!)
SUB TITLESCRN2
      a$ = INPUT$(1)
      RANDOMIZE TIMER
      a% = RND(1) * 100 + 1
```



```
      IF a% < 6 THEN
        SOUND 45, 32
        LOCATE 24, 30
        COLOR 28
        PRINT "EXCEPT THAT ONE !!!";
        a# = TIMER
        WHILE TIMER < a# + 3
        WEND
      END IF
END SUB

SUB varhelp
      COLOR 11, 12
      CLS
      LOCATE 1, 20
      PRINT "Help-screen for VAR - Variance of eigenvector"
      PRINT
      PRINT "   In practise, eigenvectors having large variances are
        considered primary"
      PRINT " eigenvectors, whereas eigenvectors having small variances are
        considered "
      PRINT " to be secondary eigenvectors. "
      PRINT "                                "
      PRINT "                    1           "
      PRINT "               ---------------- "
      PRINT "   VAR(N) =        c            "
      PRINT "                  Σ    1       "
      PRINT "                 n:1   n       "
      PRINT "                                "
      PRINT
      PRINT " where :      -  N = eigenvector of interest"
      PRINT "                 c = number of columns (eigenvectors)"
      PRINT "                 1 = eigenvalue"
      PRINT "                  n           n"
      PRINT
      CALL slep
      CLS
```

```
END SUB

FUNCTION VARIANCE# (MAT#(), col%, numrows%, AVE#)
' returns the variance of the COL%'th column of the MAT#rix from the first to
' the NUMROWS%'th row.
      sum# = 0
      FOR NX% = 1 TO numrows%
        sum# = sum# + MAT#(NX%, col%)
      NEXT NX%
      AVE# = sum# / numrows%
      vr# = 0
      FOR NX% = 1 TO numrows%
        vr# = vr# + (MAT#(NX%, col%) - AVE#) ^ 2
      NEXT NX%
      IF vr# THEN
        VARIANCE# = vr# / (numrows% - 1)
      ELSE
        VARIANCE# = 0
      END IF
END FUNCTION

SUB verprint (WORD$, X%, Y%)
' This routine prints out the string WORD$ vertically with the first letter
' at X%,Y%. If the word is to be printed upwards, give X% as negative.
      LN% = LEN(WORD$)
      IF X% < 1 THEN
        X% = -X%
        en% = Y%
        st% = Y% + LN% - 1
        dir% = -1
      ELSE
        st% = Y%
        en% = Y% + LN% - 1
        dir% = 1
      END IF
      ps% = 0
      FOR I% = st% TO en% STEP dir%
        ps% = ps% + 1
        LOCATE I%, X%
        PRINT MID$(WORD$, ps%, 1);
      NEXT
END SUB
```


XI.4 Hierarchical Clustering / Dendrogram Analysis Program

DENDGRAM was written in QuickBASIC 4.0 entirely by D. B. Sibbald. The knowledge for the dendrogram algorithms is attributed to the course presented by Dr. Ken Burton at the 2nd Spring School of Chemometrics in April, 1988 at Bristol, U.K.. The attendance of the course was precipitated by the coincidence of the dates of the course with that of the birthday of a special friend living in Norfolk.

The program has been divided into two halves because of size. The first half calculates the dendrogram from a data file (.DS1) created by P. D. Wentzell's AEMUNCH program. The second half (DENDPLOT) is dedicated to displaying the dendrogram on the screen and plotting it on a Hewlett Packard plotter (HP-Color Pro Plotter HP7440A, Hewlett Packard, San Diego, CA).

The operation of the program is as follows. The user specifies a data file with the extension of .DS1 - extensions of .DES, .SCL, are also permitted. These correspond to files created from earlier versions of AEMUNCH (most notably AECRUNCH, and a version written by D. A. Boyd in QuickBASIC 3.0 - PATCHAR) and to a prescaled data file. The data file is then loaded in and scaling options are presented. These include auto scaling, and range scaling, as well as allowing the user to define a specific scaling algorithm. It is through the use of the user-defined scaling menu that LINK scaling is achieved.

After scaling the data file, the user is first asked if a scaled data file should be saved for future use. Then, the calculations follow. The distance matrix is calculated. All signals are then clustered together successively until all data points (signals) have been joined into one cluster. This involves selecting the closest pair of objects (data

points and/or clusters) and fusing them into one object. The distance matrix is then recalculated and the process continues.

The resulting dendrogram is stored in a file (with the extension of .DEN) as a list of object pairs and the distances (or dissimilarities) at which they were joined. This file can then be used by DENDPLOT to generate a graphical display (using an installed enhanced graphics adapter (EGA) card) or to produce a hardcopy.

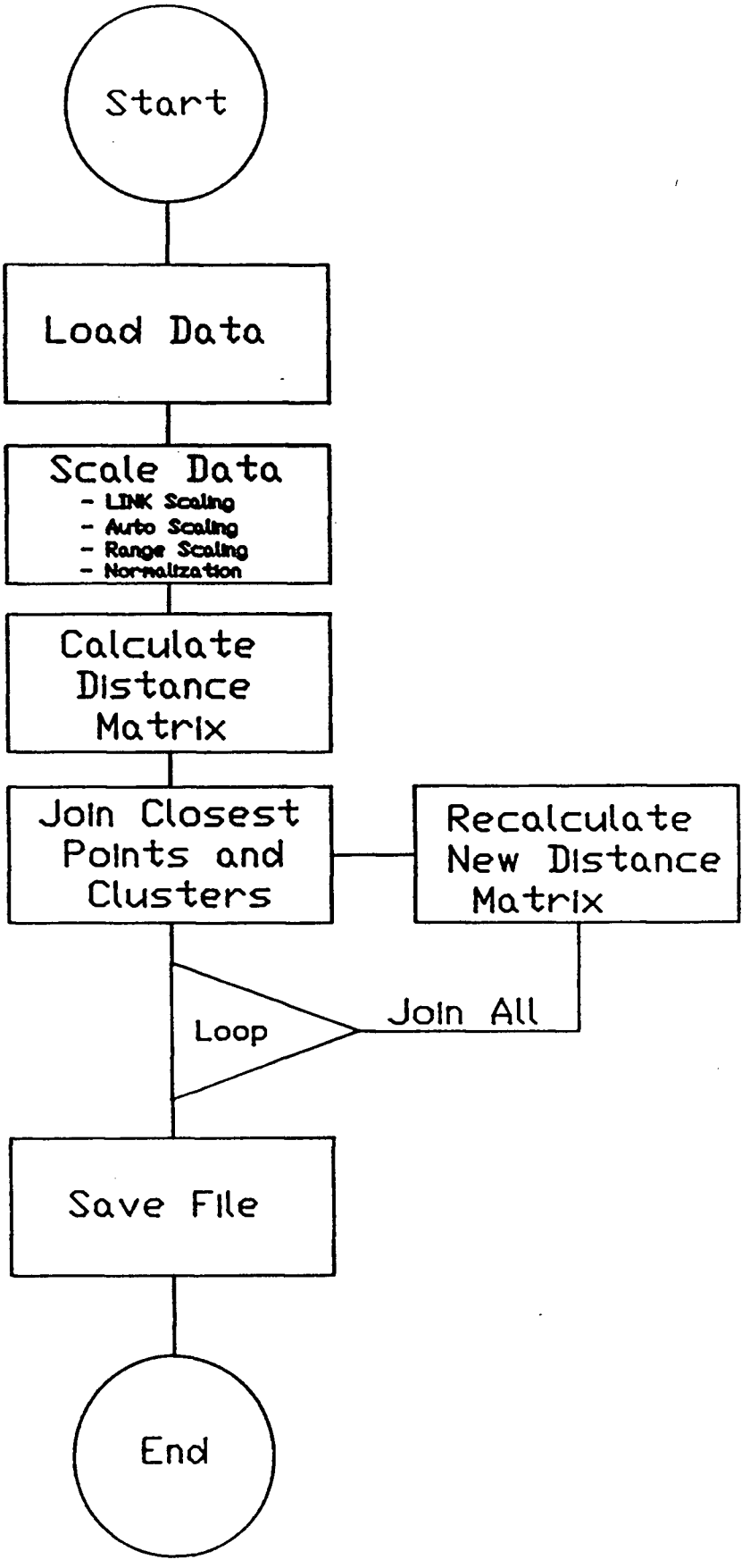


Figure 84) Dendrogram algorithm from DENDGRAM program.

```
'************* DENDGRAM    by David Sibbald **********************
'*                                                                *
'* Written January 14, 1988 at University of British Columbia     *
'
' Update #7 March 1988          1 -
'                               2 -
'                               3 -
'                               4 -
'                               5 -
'                               6 - implimentation of new file formats
'                                 - use of single precision
'                                 - use of bubble sort during the
'                                   calculation (removed in v. 8)
'                               7 - addition of plotting algorithm for
'                                   HP Color Pro 7440
'                                 - Use of qb version 4.0
'                                 - MINDPI in algorithm (see below)
' Update #8 June 28, 1988       8 - calculation of distance matrix to
'                                   speed execution
'                                 - addition of classifier to file.
' Update #9 July 4th, 1988      9 - addition of subroutines FIXCSD and
'                                   FIXCCD - use of matrices for
'                                   distances between clusters and
'                                   samples.
'                                 - Menu for choices of SEVEN
'                                   different methods of calculating
'                                   Dendrogram.
'                                 - Menu for selection of scaling
'                                   functions. - Ability to scale
'                                   columns individually.
'                                 - Ability to scale columns to meet
'                                   the variance of other columns.
'                                   Shifting and Multiplication.
'                                 - Ability for BATch file control.
' Update #9.1  January 5, 1989   - Use of new(er) file formats for
'                                   DES file. DS1 and DS2 files now
'                                   exist. SC2 file format parallel
```



```
'                                   with DS2.
'                                 - Future plans to allow for use of
'                                   disk as RAM to increase memory
'                                   capacity of prog.
'
'================================================================================
' This program is designed to load in a data matrix
' [ samples , variables ] from a .DES file
' The format of the DES file is as follows:
'
'            numsams% - the number of samples
'            NUMVARS - the number of variables per sample
'            VARNAM$(NUMVARS) - a linear array of strings
'                               containing the descriptors of
'                               the variables
'            these are followed by numsams% rows of :
'                 "id..","clss",var!(1),var!(2),.....,var!(numvars%)
'
' With version 9.1, it is possible to use the newer descriptor file
' formats.
'
' DS1 File format.
' ================
' The DS1 file is a binary random access file. It consists of 4-byte
' records and has the following format:
'
' RECORD        DESCRIPTION
' ======        ===========
'   1           Contains the number of signals (n) and number of
'               descriptors per signal (m). Packed as 2-byte integers
'               into the 4-bytes.
'   2           Blank record for future modification. Currently zero.
' 3 & 4         Descriptor name for first descriptor. 8 characters
'               packed into two 4-byte records.
' 5 & 6         Second descriptor.
'    .
'    .
```

```
'   .
' 2m + 3        Identifier for signal 1 (4 characters)
' 2m + 4        Class for signal 1 (4 characters)
' 2m + 5        Time for signal 1 (single precision)
' 2m + 6        Extra record for signal 1 (4 characters)
' 2m + 7        First of m descriptors for signal 1 (single precision)
'   .
'   .
'   .
' 3m + 6        m'th descriptor for signal 1 (single precision)
' 3m + 7        Identifier for signal 2
' 3m + 8        Class for signal 2
'   .
'   .
' (y+1)m+4y+2+x    Descriptor x for signal y (single precision)
' (y+1)m+4(y-1)+3   Identifier for signal y
' (y+1)m+4(y-1)+4   Class for signal y
' (y+1)m+4(y-1)+5   Time for signal y
' (y+1)m+4(y-1)+6   Extra record for signal y
'
' The alternate ASCII descriptor file has the extention DS2 and has the
' same format as the DES file except that the second record of the DS1
' file appears as two integers on the second line and that the time and
' extra record are inserted after the class for each signal. The file
' is therefore the ASCII counterpart of the DS1 file. The DS2 files are
' thus complete and a transformation between DS1 and DS2 files are
' therefore complete with no loss of information.
'
' SCALING
'========
' Scaling is then performed on the data matrix
' - Auto-scaling :  mean-centering each variable vector to
'                   have unit variance.
' - Range scaling : scaling each variable vector to have values
'                   from 0 to +1
' Other options such as weighting are now available with the special
' functions option in the scale menu. The variable rows can be shifted,
```



```
' multiplied, ranged between any two values (including an inversion
' about  the median or about the minimum value.
' The transformed matrix is stored as a file in the same format as the
' original data file with a one line string at the end giving
' some information as to the method of scaling. A DS1 file having been
' scaled is saved as a SC2 file. (NB. A DES file will result in an SCL
' file being saved and a DS1 file will result in a SC2 file.)
'=================================================================
' BATCH FILE control:
' ===================
' For BATCH file control, the command line must be set up in the
' following format (so there!):
'>DENDGRAM /filename /S=scopt /M=method [/Minkowski value] [/O=outfile]
' where
'      1) filename is the file (including root and extension) for
'         input. Valid extensions (SCL,DES,DS1,DS2)
'      2) scopt is the scaling function desired. Valid parameters :
'         /S=AUTO      - Auto scaling
'         /S=RANGE     - Range scaling (0 to 1)
'         /S=NO        - No scaling
'         /S=YES       - The file is taken to be already scaled. No
'                        further scaling is done and the SCLE$ at the
'                        end of the .SCL file is read.
'         /S=sclfile   - User defined scaling. SCLFILE must be a
'                        previously saved scaling format file.
'         if no /S= option is given, then no scaling is performed.
'      3) method is the method for calculating the DENDROGRAM. Valid
'         options:
'         /M=x         - x is an integer between 1 and 7
'                        1 - Single Linkage
'                        2 - Complete Linkage
'                        3 - Weighted Average Linkage
'                        4 - Average Linkage (unweighted)
'                        5 - Centroid
'                        6 - Median Centroid
'                        7 - Ward's Method
'         if no /M= option is given then Single Linkage is used.
```

```
'       4) The Minkowski factor can be supplied . Default = 2.
'       5) The output root can be supplied if it is required to be
'          different.
'
' Sample command line for a complete linkage dendrogram from TEST.DES
' in the active directory, to be scaled according to LINK.SFF in a
' lower directory, using a Minkowski factor of 1 (Manhattan city block
' distances) and saved in the root directory.
' DENDGRAM /TEST /S=SCALE\LINK.SFF /M=2 /1 /O=\TEST
'=======================================================================
DECLARE SUB ADVSCAL (MESS!(), ID() AS ANY, CLASS() AS ANY, F$, SCLE$)
DECLARE SUB BATCHGUYS (batchCALL%, BATOPT$())
DECLARE SUB CALCDIST (MESS!(), NUMSAMS%, NUMVARS%, SSDIST!(), MAXDST!)
DECLARE FUNCTION clustmem% (CLUSTNUM%, MEM%)
DECLARE FUNCTION colmax! (MAT!(), COL%, startrow%, endrow%, COLMIN!)
DECLARE SUB DOPT (FILE$, method%, method$(), Minkowski!, OUTFILE$)
DECLARE SUB fixcsd (CSD!(), PRTR%, PTR2%, OMEM%, NMEM%, method%)
DECLARE SUB fixccd (CCD!(), PTR1%, PTR2%, OMEM%, NMEM%, method%)
DECLARE SUB FRTEXT (FILE$, FROOT$, FEXT$)
DECLARE FUNCTION MATCH (X%, Y%)
DECLARE SUB READPARS (NUMSAMS%, NUMVARS%, EXT1%, EXT2%, F$)
DECLARE SUB READVALS (VARNAM$(), MESS!(), ID() AS ANY, CLASS() AS ANY,
        SCALED$, F$)
DECLARE SUB scalecol (MAT!(), COL%, mode%, value!, div!)
DECLARE SUB scaler (MESS!(),ID() AS ANY,CLASS() AS ANY, SCLE$(), FILE$)
DECLARE SUB sclforlod (FILE$, scal$, mode%(), a!(), B!())
DECLARE SUB sclforsav (FILE$, scal$, a%(), B!(), C!())
DECLARE SUB sclsave (MAT!(),ID() AS ANY,CLASS() AS ANY,sc$(),spec%, F$)
DECLARE FUNCTION VARIANCE! (MAT!(), COL%, numrows%, AVE!)

' *** FNTXT% returns the print line on screen corresponding
' ***        to graphics line n
        DEF fntxt% (N)
           IF N < 8 THEN fntxt% = 1 ELSE fntxt% = (N + 9) * 25 / 350
        END DEF

        CONST init = 3E+38
```



```
        CONST bgrnd = 11
        CONST hlght = 4
        CONST YELLOW = 14
        CONST dblue = 9
        CONST ntext = 15
        CONST legext$ = ".DES.DS1.DS2.SCL.SC2"
        TYPE RECRD
           R AS STRING * 4
        END TYPE

1       CLEAR
        RESTORE
        DATA 7,"Single Linkage","Complete Linkage"
        DATA "Average Linkage (Weighted)","Average Linkage
          (Unweighted)"
        DATA "Centroid","Weighted Centroid (Median)","Ward's Method"
        READ nummethods%
        DIM SHARED method$(1 TO nummethods%)
        FOR I% = 1 TO nummethods%
           READ method$(I%)
        NEXT I%
        REM $DYNAMIC
        DIM BATOPT$(5)
' Check to see if this is done by remote-control. (Don't let my Dad
' have  the remote). If the COMMAND line contains parameters, the
' BATCHYES%  will be TRUE and the program can run automatically.
        CALL BATCHGUYS(batchyes%, BATOPT$())
5       COLOR ntext, bgrnd
        CLS
        IF batchyes% THEN
           F$ = BATOPT$(1)
           FILE$ = BATOPT$(1)
           IF BATOPT$(2) = "YES" THEN md% = 2
           GOSUB 100
           IF BATOPT$(2) = "YES" THEN 600
           GOTO 500
        END IF
```

```
      PRINT STRING$(22,"*");" DEND-GRAM by David Sibbald ";
        STRING$(30, "*")
      LOCATE 5, 19
      PRINT "1) Calculate dendrogram"
      LOCATE 6, 19
      PRINT "2) Show directory"
      LOCATE 7, 19
      PRINT "3) Draw dendrogram from DEN file"
      LOCATE 8, 19
      PRINT "4) Jump out of a moving automobile"
8     LOCATE 11, 16
      PRINT "Select number";
      a$ = INPUT$(1)
      IF a$ < "1" OR a$ > "4" GOTO 8
      md% = VAL(a$)
      ON md% GOTO 10, 20, 30, byby

10    LOCATE 13, 9
      PRINT "Enter data file to calculate dendrogram (include
        extension)..."
      LOCATE , 10
      PRINT "[ Legal extensions "; CHR$(238); " {";
      Q$ = CHR$(34)
      PRINT Q$; MID$(legext$, 2, 3); Q$;
      FOR I% = 2 TO LEN(legext$) / 4
         PRINT ","; Q$; MID$(legext$, (I% - 1) * 4 + 2, 3); Q$;
      NEXT
      PRINT "} ]"
      LOCATE 16, 20
      INPUT "File =", FILE$
      IF FILE$ = "" THEN 5
      FILE$ = UCASE$(FILE$)
      CALL FRTEXT(FILE$, FROOT$, FEXT$)
      IF (INSTR(".DES.SCL.DS1.DS2.SC2",FEXT$) = 0) OR FEXT$ = "" THEN
         SOUND 300, 2
         GOTO 10
      END IF
```



```
      F$ = FROOT$ + FILE$ + FEXT$
      GOSUB 100
      GOTO 500
20    LOCATE , 10
      PRINT "Directory to show [RETURN for current directory]"
      LOCATE , 10
      INPUT ; a$
      shell$ = "DIR " + a$ + " > TEMPFIL.DIR"
      SHELL shell$
      OPEN "I", #1, "TEMPFIL.DIR"
      CLS
      l% = 0
      WHILE NOT EOF(1)
         LINE INPUT #1, a$
         LOCATE , 2
         PRINT a$
         l% = l% + 1
         IF l% = 18 THEN
            LOCATE 24, 15
            COLOR 14
            PRINT "Press any key";
            COLOR ntext
            a$ = INPUT$(1)
            l% = 0
            CLS
            LOCATE 5, 2
         END IF
      WEND
      LOCATE 24, 15
      COLOR 14
      PRINT "Press any key";
      COLOR ntext
      a$ = INPUT$(1)
      CLOSE #1
      SHELL "ERASE TEMPFIL.DIR"
      GOTO 5
30    LOCATE 13, 8
```

```
        PRINT " Enter DEN file to draw dendrogram (no extension)"
        LOCATE 15, 20
        INPUT "File =", FILE$
        FILE$ = UCASE$(FILE$)
        CALL FRTEXT(FILE$, FROOT$, FEXT$)
        FEXT$ = ".DEN"
        F$ = FROOT$ + FILE$ + FEXT$
        GOSUB 200
        GOTO 899

' ******   Reads data from file$ into mess!
' Open file and get the vital information for dimensioning variables.
100     CALL READPARS(NUMSAMS%, NUMVARS%, EXT1%, EXT2%, F$)

' Dimension variables that need to be read from file
        DIM VARNAM$(1 TO NUMVARS%)
        DIM MESS!(1 TO NUMSAMS%, 1 TO NUMVARS%)
        DIM CLASS(1 TO NUMSAMS%)  AS RECRD
        DIM ID(1 TO NUMSAMS%) AS RECRD

' Read in data from file. No need to pass NUMSAMS, NUMVARS because they
' are read by the subroutine.
        CALL READVALS(VARNAM$(), MESS!(), ID(), CLASS(), SCALED$, F$)

' Dimension variables needed for calculation of dendrogram
        DIM DEND%(1 TO NUMSAMS% - 1, 1 TO 2)
        DIM DENDD!(1 TO NUMSAMS% - 1)
        DIM quashvar%(1 TO NUMVARS%)
        RETURN

' ****** Reads dendrogram into dend%
200     OPEN "i", #2, F$
        INPUT #2, NUMSAMS%, NUMVARS%
        DIM VARNAM$(1 TO NUMVARS%)
        DIM MESS!(1 TO NUMSAMS%, 1 TO NUMVARS%)
        DIM CLASS(1 TO NUMSAMS%) AS RECRD
        DIM ID(1 TO NUMSAMS%) AS RECRD
```



```
        FOR N% = 1 TO NUMVARS%
           INPUT #2, VARNAM$(N%)
        NEXT

        DIM PLOT%(1 TO NUMSAMS%) ' Contains numbers of samples in order
          of t.t.b
        DIM DEND%(1 TO NUMSAMS% - 1, 1 TO 2)
        DIM DENDD!(1 TO NUMSAMS% - 1)
        FOR np% = 1 TO NUMSAMS% - 1
           FOR N% = 1 TO 2
              INPUT #2, DEND%(np%, N%)
           NEXT
           INPUT #2, DENDD!(np%)
        NEXT
        INPUT #2, MAXDST!
        FOR np% = 1 TO NUMSAMS%
           INPUT #2, PLOT%(np%)
           INPUT #2, ID(PLOT%(np%)).R
           INPUT #2, CLASS(PLOT%(np%)).R
        NEXT
        INPUT #2, SCALED$
        INPUT #2, method$(1)
        method% = 1
        CLOSE #2
        RETURN
'****** Scales data according to choice
500     CLS
        DATA 4
        DATA "Auto-scaling "
        DATA "Range scale (0 - +1)"
        DATA "Perform no scaling ","Special functions."
        READ numofscales%
        DIM SCLE$(numofscales%)
        FOR I% = 1 TO numofscales%
           READ SCLE$(I%)
        NEXT
        INFILE$ = F$
```

```
        OUTFILE$ = FROOT$ + FILE$ + ".DEN"
        F$ = FROOT$ + FILE$ + ".SCL"
        CALL scaler(MESS!(), ID(), CLASS(), SCLE$(), F$)
        IF NUMSAMS% < 2 THEN GOTO 1
        SCALED$ = SCLE$(1)
        ERASE SCLE$

' **********     Make Dendrogram        by David Sibbald **********
' This program is designed to calculate the joints and correlations
' between the stupid points in the scaled matrix mess!
' The dendrogram is stored in a file with the extension .DEN
'
' The algorithm works as follows:
' 0) calculate distance matrix (DIST!(,)). This is added in version 8.
'       It was an element of the earlier versions but removed to save
'       computer memory for the larger NUMSAM sizes. Now that version
'       4.0 can handle matrices of larger than 64 K it seems wise to
'        include DIST! again. The function FNDSTSS! has been changed.
'
' 1) find shortest joint - the clusters are compared to each other
'                        - the clusters are compared to loose samples
'                        - the loose samples are compared to each other
'
' (This order is arbitrary, the shortest fusion is made regardless)
'
' 2) set dend% to appropriate variables based on type of joint in 1) -
'                    - merge two clusters
'                    - add sample into cluster
'                    - join two samples into new cluster
'
' 3) Calculate the new matrices for the distances between samples and
'       clusters CSDIST!() and between clusters CCDIST!().
'
' 4) repeat 1) to 3) until the number of joints (nnaybs) is equal to
'       the number of samples (numsams%) minus one (eg. 4 samples == 3
'       joints)
'
```



```
' 5) For output to the screen or printer, the order (toptobottom)
'       of the samples must be known. This is stored in PLOT%.
'       The order will always end up in the array SAMPPTR() starting
'       with element CLSN%(1). See the discussion in the CLUSTMEM%
'       function for the details of this MAGICAL side effect.
'
' 6) Output : The data file (.DEN) is stored as
'          dend%(1,1) ,          dend%(1,2) ,          DENDD!(1)
'          ...
'          DEND%(NUMSAMS%-1,1), DEND%(NUMSAMS%-1,2), DENDD!(NUMSAMS%-1)
'          PLOT%(1),         ID$*4(1),         CLASS$*4(1)
'          ...
'          PLOT%(NUMSAMS%), ID$*4(NUMSAMS%) ,CLASS$*4(NUMSAMS%)
'
'    For output to screen, each sample is assigned a point in (x,y)
'       space. These points are joined by a ]-type shape following the
'       order of dend%.  As a point is joined it is assigned the new
'       position at the center of the cross-line.


600     IF NOT batchyes% THEN
           LOCATE 25, 1
           PRINT "Press return to continue.....";
           SLEEP: a$ = INKEY$
           method% = 1
           Mink! = 2
           CALL DOPT(INFILE$, method%, method$(), Mink!, OUTFILE$)
        ELSE
           method% = VAL(BATOPT$(3))
           Mink! = VAL(BATOPT$(5))
        END IF
        CLS
        PRINT "DENDGRAM for ..."; INFILE$;
        OFILE$ = OUTFILE$
        CALL FRTEXT(OFILE$, OUTROOT$, OUTEXT$)
        IF OFILE$ <> FILE$ THEN
           PRINT " ==> "; OUTROOT$ + OFILE$ + OUTEXT$
```

```
    END IF
    LOCATE 6, 15: PRINT SCALED$
    LOCATE 10, 15: PRINT method$(method%)
    LOCATE 23, 1
    PRINT "Calculating distance matrix";
    time1$ = TIME$
    DIM SSDIST!(1 TO NUMSAMS%, 1 TO NUMSAMS%)
    CALL CALCDIST(MESS!(), NUMSAMS%, NUMVARS%, SSDIST!(), MAXDST!)
    IF method% > 4 THEN
       DIM clstvar!(1 TO NUMSAMS% \ 2, 1 TO NUMVARS%)
    ELSE
       ERASE MESS!
    END IF

' CSDIST!(S,C) contains the distance between sample S and cluster C.
' The matrix is completely calculated (except where the number of
' clusters is less than NUMSAMS% /2). In addition, the first row
' (CSDIST!(0,X)) contains the number of members of cluster X. The first
' column (CSDIST!(S,0)) contains a BOOLEAN value for [sample S joined
' into any cluster]<-1 = TRUE,0 = FALSE>.
    DIM csdist!(0 TO NUMSAMS%, 0 TO NUMSAMS% \ 2)

' CCDIST!(X,Y) contains the distance matrix between clusters X and Y.
' The matrix is only half calculated (because of symmetry) so for the
' correct results X must be less than Y.
    DIM ccdist!(1 TO NUMSAMS% \ 2, 1 TO NUMSAMS% \ 2)

' See the comments on the FUNCTION CLUSTMEM% for the description of the
' use of the arrays CLSN% and SAMPPTR%.
    DIM clsn%(1 TO NUMSAMS% / 2)
    DIM sampptr%(0 TO NUMSAMS%)
    LOCATE 23, 1
    PRINT "Calculating Dendrogram      "
    nnaybs% = 1
    nclsts% = 0
    WHILE nnaybs% < NUMSAMS%    ' *** Main loop
       LOCATE 24, 1
```



```
       t$ = " ###/### "
       PRINT USING t$; nnaybs%; NUMSAMS% - 1;
       mind! = MAXDST! + 1

' ***  find shortest distance between clusters
       FOR nC% = 2 TO nclsts%
          IF csdist!(0, nC%) = 0 GOTO 610
          FOR nY% = 1 TO nC% - 1
           IF csdist!(0, nY%) <> 0 THEN
              dst! = ccdist!(nY%, nC%)
              IF dst! < mind! THEN
                mind! = dst!
                nmin1% = nY%
                nmin2% = nC%
                newjoint% = 1
              END IF
           END IF
         NEXT
610    NEXT

' *** find shortest distance between cluster and point

       FOR nC% = 1 TO nclsts%
          IF csdist!(0, nC%) = 0 GOTO 620
          FOR NS% = 1 TO NUMSAMS%
           IF csdist!(NS%, 0) = 0 THEN
              dst! = csdist!(NS%, nC%)
              IF dst! < mind! THEN
                newjoint% = 2
                nmin1% = nC%
                nmin2% = NS%
                mind! = dst!
              END IF
           END IF
         NEXT
620    NEXT
```

```
' *** find shortest distance between two samples

          FOR NS% = 2 TO NUMSAMS%
             IF csdist!(NS%, 0) = 0 THEN
              FOR nY% = 1 TO NS% - 1
                 IF csdist!(nY%, 0) = 0 THEN
                   dst! = SSDIST!(NS%, nY%)
                   IF dst! < mind! THEN
                     mind! = dst!
                     nmin1% = NS%
                     nmin2% = nY%
                     newjoint% = 3
                   END IF
                 END IF
              NEXT
            END IF
          NEXT

' *** Set up new values in dend%

660       ON newjoint% GOTO 700, 710, 720

' *** Two clusters to be joined
' NMIN1% = cluster NUMBER
' NMIN2% = cluster NUMBER          NMIN2% < NMIN1%
' We need to join the members of cluster NMIN2% onto the back of
' cluster NMIN2.  This is done by setting SAMPPTR() of the last member
' of cluster NMIN1 to be equal to the first member of cluster NMIN2 (or
' CLSN%(NMIN2%)).
700       DEND%(nnaybs%, 1) = clustmem%(nmin1%, 1)
          DEND%(nnaybs%, 2) = clustmem%(nmin2%, 1)
          DENDD!(nnaybs%) = mind!
          OMEM% = csdist!(0, nmin1%)
          NMEM% = csdist!(0, nmin2%)
          lastsam% = clustmem%(nmin1%, OMEM%)
          sampptr%(lastsam%) = clsn%(nmin2%)
```



```
' Set appropriate values in the "cluster" vector of CSDIST!()
          csdist!(0, nmin2%) = 0
          csdist!(0, nmin1%) = NMEM% + OMEM%
          GOTO 750

710     ' *** Add sample to cluster
' NMIN1% = cluster number
' NMIN2% = sample number
          DEND%(nnaybs%, 1) = nmin2%
          DEND%(nnaybs%, 2) = clustmem%(nmin1%, 1)
          DENDD!(nnaybs%) = mind!
          OMEM% = csdist!(0, nmin1%)
          NMEM% = 1
          lastmem% = clustmem%(nmin1%, OMEM%)
          sampptr%(lastmem%) = nmin2%

' set sample NMIN2% to "used" in the sample vector of CSDIST
          csdist!(nmin2%, 0) = -1
' add one member to cluster NMIN1%
          csdist!(0, nmin1%) = OMEM% + 1
' set NMIN2% to negative for passing into FIX's
          nmin2% = -nmin2%
          GOTO 750

720 ' *** Two points to make new cluster
          DEND%(nnaybs%, 1) = nmin1%
          DEND%(nnaybs%, 2) = nmin2%
          DENDD!(nnaybs%) = mind!
          NMEM% = 1
          OMEM% = 1
          nclsts% = nclsts% + 1
          clsn%(nclsts%) = nmin1%
          sampptr%(nmin1%) = nmin2%

' set two samples to "used" and set the number of members to TWO in new
' cluster.
          csdist!(nmin1%, 0) = -1
```

```
            csdist!(nmin2%, 0) = -1
            csdist!(0, nclsts%) = 2
            nmin2% = 0
            nmin1% = nclsts%

750         CALL fixcsd(csdist!(), nmin1%, nmin2%, OMEM%, NMEM%,
method%)
            CALL fixccd(ccdist!(), nmin1%, nmin2%, OMEM%, NMEM%,
method%)
            nnaybs% = nnaybs% + 1
         WEND

' Now that dend% is calculated, figure order for PLOT%ing
' This is simple because the order of the samples in CLSN%(1) IS the
' order!!  This little trick was figured out by David Sibbald during a
' particularly bad bought of mental instability. Serendipitously, an
' algorithm based on some of the delirium tremens resulted that made
' life much easier.
         ERASE csdist!, ccdist!, SSDIST!, clstvar!, quashvar%
         DIM PLOT%(1 TO NUMSAMS%) ' Contains #'s of samples in order of
'                                    t.t.b
         PLOT%(1) = clsn%(1)
         FOR I% = 2 TO NUMSAMS%
            PLOT%(I%) = sampptr%(PLOT%(I% - 1))
         NEXT
         CLS
         PRINT "Dendrogram calculated."
         ERASE clsn%, sampptr%
         IF batchyes% THEN
            F$ = BATOPT$(4)
         ELSE
            F$ = OUTFILE$
         END IF
         SOUND 2000, 5
         OPEN "o", #4, F$
         WRITE #4, NUMSAMS%, NUMVARS%
         FOR nv% = 1 TO NUMVARS%
```



```
            WRITE #4, VARNAM$(nv%)
         NEXT
         FOR nd% = 1 TO NUMSAMS% - 1
            PRINT #4, DEND%(nd%, 1), DEND%(nd%, 2), DENDD!(nd%)
         NEXT
         IF method% = 7 THEN MAXDST! = DENDD!(NUMSAMS% - 1)
         PRINT #4, MAXDST!
         FOR np% = 1 TO NUMSAMS%
            WRITE #4, PLOT%(np%), ID(PLOT%(np%)).R, CLASS(PLOT%(np%)).R
         NEXT
' Printout stuff such as scaling method, calculation method, Minkowski
'  factor, date, etc..
         WRITE #4, SCALED$
         PRINT #4, CHR$(34) + method$(method%) + CHR$(34),
         IF Mink! <> 2 THEN
            PRINT #4, CHR$(34)+"Minkowski factor ="+STR$(Mink!)+CHR$(34)
         ELSE
            PRINT #4,
         END IF
         PRINT #4, "Original Data File =" + FROOT$ + FILE$ + FEXT$
         PRINT #4, "Start time", time1$
         PRINT #4, "Finished at :", TIME$
         PRINT #4, DATE$
         CLOSE #4
         PRINT " Dendrogram saved as "; F$
         LOCATE 23, 15
         PRINT "Press any key to view graph... [ESC to exit]"
         IF batchyes% THEN GOTO byby
         a$ = INPUT$(1)
         IF a$ = CHR$(27) THEN GOTO byby

' Now plot dendrogram
899      CLS
         SHELL "dendplot " + F$
' The following line is an alternative to the above line.  The file
' DENDPLOT.BAS needs to be in the current directory with QuickBASIC in
' the path.
```

```
        'SHELL "QB /RUN DENDPLOT "+F$
byby:   SCREEN 0
END

REM $STATIC
SUB ADVSCAL (MESS!(), ID() AS RECRD, CLASS() AS RECRD, F$, scal$)
' This routine is called when the user decides to play God and mess up
' the data with some wEird scaling. If there is some knowledge about
' the data then "user defined" scaling is a good idea but can be
' disastrous when applied indiscriminately. So that's that then.
' (Eh, Richard?)
'
' The columns are shown on the screen and the vital statistics are
' shown of each variable. These are stored in arrays which need not
' ever be accessed again so it is prudent to not declare this routine
' as STATIC or at least to erase the arrays before continuing.
'
' One BUG : Yes I am reporting a bug that I know to be present but
' could not fix. When selecting the mode, pressing the arrow keys 19
' times causes a STRING FORMULA TOO COMPLEX error on an otherwise
' perfectly good command. I could discover no obvious or secret reason
' for this except that the error does not occur when the program is
' operated from executable mode. I feel that the QUICKBASIC environment
' is hostile to people who REALLY can't make up their mind - like Sara.

        SHARED NUMVARS%, NUMSAMS%, VARNAM$(), quashvar%()
        SHARED batchyes%, BATOPT$()
        SCRNROWS% = 16
        TOP = 5
        TOPROW% = 1
        COLOR ntext, bgrnd
        CURROW% = 1
        curstat% = 1
        scal$ = ""
        DIM scval!(1 TO NUMVARS%)
        DIM scdiv!(1 TO NUMVARS%)
        DIM scmod%(1 TO NUMVARS%)
```



```
        DIM varnz!(1 TO NUMVARS%)
        DIM AVE!(1 TO NUMVARS%)
        IF batchyes% GOTO skip
        DIM colmins!(1 TO NUMVARS%)
        DIM colmaxs!(1 TO NUMVARS%)
        DIM sc(4) AS STRING * 15
        sc(1) = "Auto scale     "
        sc(2) = "Range scale    "
        sc(3) = "Shift/Multiply "
        sc(4) = "Erase / Remove "
skip:   FOR I% = 1 TO NUMVARS%
          varnz!(I%) = VARIANCE!(MESS!(), I%, NUMSAMS%, AVE!(I%))
          IF NOT batchyes% THEN
            scval!(I%) = 0
            scdiv!(I%) = I%
            scmod%(I%) = 1
            colmaxs!(I%) =colmax!(MESS!(),I%,1,NUMSAMS%,colmins!(I%))
          END IF
        NEXT
        IF batchyes% THEN
          CALL sclforlod(BATOPT$(2),scal$,scmod%(),scval!(), scdiv!())
          GOTO doit
        END IF
        pt1$ = " Column ## ,  Mode = "
        DIM pt2$(4)
        pt2$(1)="  Scale mean to :###.####,Use variable ##  variance  "
        pt2$(2)="  Range Between :###.####  (min) and  ###.#### (max)"
        pt2$(3)="  Shift mean to :###.####, Multiply by ###.####      "
        pt2$(4) = " "
        CLS
        LOCATE 1, 1
        PRINT STRING$(80, "*");
        LOCATE 2, 24
        PRINT "SCALING : "; F$
        LOCATE 4, 1
        pt$=" ## |  ##.###^^^^  |##.####^^^^ | ##.####^^^^  |
          ##.####^^^^  | "
```

```
      PRINT "Variable| Variance |  Average   |   Maximum    |
        Minumum    |    Mode"
      SOUND 400, 1
16    COLOR 1, bgrnd
      LOCATE TOP, 1
      BOTROW% = TOPROW% + SCRNROWS%
      IF BOTROW% > NUMVARS% THEN BOTROW% = NUMVARS%
      FOR I% = TOPROW% TO BOTROW%
        PRINT USING pt$; I%; varnz!(I%); AVE!(I%); colmaxs!(I%);
          colmins!(I%);
        PRINT sc(scmod%(I%))
      NEXT I%
      LOCATE 25, 1
      PRINT "Press L to load in saved Format file. ESC to execute.";
        SPACE$(26);
      LOCATE 23, 1
      COLOR ntext, bgrnd
      PRINT " " + VARNAM$(CURROW%); STRING$(40, " ");
      COLOR YELLOW, hlght
      LOCATE CURROW% + TOP - TOPROW%, 1
      PRINT USING pt$; CURROW%; varnz!(CURROW%); AVE!(CURROW%);
        colmaxs!(CURROW%); colmins!(CURROW%);
      PRINT sc(scmod%(CURROW%));
212   COLOR ntext, bgrnd
      LOCATE 24, 1
      PRINT USING pt1$; CURROW%;
      IF scmod%(CURROW%) = 4 THEN COLOR YELLOW, hlght
      PRINT LEFT$(sc(scmod%(CURROW%)), 5);
      IF scmod%(CURROW%) <> 4 THEN
         LOCATE 24, 27
         PRINT USING pt2$(scmod%(CURROW%)); scval!(CURROW%);
        scdiv!(CURROW%);
      ELSE
         COLOR ntext, bgrnd
         PRINT STRING$(54, " ");
      END IF
      COLOR YELLOW, hlght
```



```
      LOCATE 24, (curstat% - 1) * 22 + 22
      IF scmod%(CURROW%) < 4 THEN
         SELECT CASE curstat%
         CASE 1
            PRINT LEFT$(sc(scmod%(CURROW%))              PRINT LEFT$(sc(scmod%(CURROW%)),
         CASE 2
            PRINT USING "###.####"; scval!(CURROW%);
         CASE 3
            IF scmod%(CURROW%) = 1 THEN
              PRINT USING "##"; scdiv!(CURROW%);
            ELSE
              PRINT USING "###.####"; scdiv!(CURROW%);
            END IF
         END SELECT
      END IF
1236  a$ = INKEY$
      IF a$ = "" GOTO 1236
      IF ASC(a$) = 0 THEN a$ = RIGHT$(a$, 1)
1237  SELECT CASE a$
      CASE CHR$(77)      ' the RIGHT key was pressed
         curstat% = curstat% + 1
         IF curstat% > 3 THEN curstat% = 1
         GOTO 212
      CASE CHR$(75)      ' the LEFT key was pressed
         curstat% = curstat% - 1
         IF curstat% = 0 THEN curstat% = 3
         GOTO 212
      CASE CHR$(72)      ' the UP key was pressed
         CURROW% = CURROW% - 1
         IF CURROW% = 0 THEN CURROW% = NUMVARS%
         IF CURROW% < TOPROW% THEN TOPROW% = CURROW%
         IF TOPROW%<CURROW%-SCRNROWS% THEN TOPROW%=CURROW%-SCRNROWS%
      CASE CHR$(80)      ' the Down key was pressed
         CURROW% = CURROW% + 1
         IF CURROW% > NUMVARS% THEN CURROW% = 1
         IF CURROW% > BOTROW% THEN TOPROW% = CURROW% - SCRNROWS%
         IF TOPROW% > CURROW% THEN TOPROW% = CURROW%
```

```
CASE "L"
   LOCATE 25, 1
   INPUT ; "Input name of format file to load : ", a$
   a$ = UCASE$(a$)
   CALL FRTEXT(a$, roota$, exta$)
   IF a$ = "" THEN 1236
   a$ = roota$ + a$ + exta$
   CALL sclforlod(a$, scal$, scmod%(), scval!(), scdiv!())
CASE CHR$(27)
   GOTO 39
CASE CHR$(13)    ' RETURN
   COLOR ntext, bgrnd
   LOCATE 25, 1
   PRINT SPACE$(79);
   LOCATE 25, 1
   SELECT CASE curstat%
   CASE 1         ' Mode selector
      md% = scmod%(CURROW%)
58    LOCATE 25, 1
      PRINT "Select new mode - ";
      FOR I% = 1 TO 4
       PRINT sc(I%);
      NEXT I%
      LOCATE 25, LEN(sc(1)) * 5 + 5
      LOCATE 25, md% * 15 + 4
      COLOR YELLOW, hlght
      PRINT sc(md%);
      COLOR ntext, bgrnd
      a$ = ""
59    a$ = INKEY$
      IF a$ = "" GOTO 59
      IF a$ = CHR$(13) GOTO 61
      a$ = RIGHT$(a$, 1)
      IF a$ = CHR$(75) THEN 'LEFT
       md% = md% - 1
       IF md% = 0 THEN md% = 4
      ELSEIF a$ = CHR$(77) THEN 'RIGHT
```



```
       md% = md% + 1
          IF md% = 5 THEN md% = 1
      END IF
      GOTO 58
61   IF md% <> scmod%(CURROW%) THEN '  change in mode
       IF scmod%(CURROW%) = 0 THEN quashvar%(CURROW%) = 0
       scmod%(CURROW%) = md%
       IF md% = 1 THEN
         scval!(CURROW%) = 0
         scdiv!(CURROW%) = CURROW%
       ELSEIF md% = 2 THEN
         scval!(CURROW%) = 0
         scdiv!(CURROW%) = 1
       ELSEIF md% = 3 THEN
         scval!(CURROW%) = 0
         scdiv!(CURROW%) = 1
       ELSE 'md% = 4 == Erase variable
      END IF
   END IF
   CASE 2          'value selector
      COLOR YELLOW, hlght
      PRINT " Change value to :";
      INPUT ; "", a$
      IF scmod%(CURROW%)=2 THEN 'RANGE scaling : check MIN=MAX
       IF VAL(a$) = scdiv!(CURROW%) THEN
         LOCATE 25, 1
         PRINT"The values for maximum and minimum must be
           different";
         SOUND 37, 3
         GOTO 1236
       END IF
      END IF
      scval!(CURROW%) = VAL(a$)
   CASE 3          'div! selector
      COLOR YELLOW, hlght
      IF scmod%(CURROW%) = 1 THEN
       INPUT ; " Scale column to variance of variable #", a$
```

```
      dv! = VAL(a$)
      IF dv!>0 AND dv! <= NUMVARS% THEN scdiv!(CURROW%) = dv!
     ELSEIF scmod%(CURROW%) = 2 THEN
      INPUT ; " Scale maximum to =", a$
      IF VAL(a$) = scval!(CURROW%) THEN
        LOCATE 25, 1
        PRINT SPACE$(79);
        LOCATE 25, 1
        SOUND           90, 1
        PRINT "The values for Maximum and Minumum cannot be
          the same";
        GOTO 1236
      ELSE
        scdiv!(CURROW%) = VAL(a$)
      END IF
     ELSE
      PRINT " Value to multiply by =";
      INPUT ; "", a$
      dv! = VAL(a$)
      IF dv! <> 0 THEN
        scdiv!(CURROW%) = dv!
      ELSE
        LOCATE 25, 1
        COLOR YELLOW, hlght
        PRINT "Use ERASE/REMOVE to eliminate variable from
          analysis";
        SOUND 100, 1
        GOTO 1236
      END IF
     END IF
   CASE ELSE
   END SELECT

CASE ELSE
END SELECT
a$ = ""
GOTO 16
```



```
'Now we get to do the scaling....
39      COLOR ntext, bgrnd
        FOR I% = 22 TO 25
          LOCATE I%, 1
          PRINT STRING$(79, " ");
        NEXT I%
        LOCATE 22, 1
        PRINT " Enter User-name of scaling format : <"; scal$; " > ";
        INPUT a$
        IF a$ <> "" THEN scal$ = LEFT$(a$, 20)
        LOCATE 22, 2
        PRINT scal$; SPACE$(60);
        LOCATE 23, 10
        INPUT ; "Save scaling format (Y/N)"; a$
        savscal$ = LEFT$(UCASE$(a$), 1)
        IF savscal$ = "Y" THEN
          LOCATE 23, 2
          INPUT ; "Name of User scaling format file to save :", fi$
          fi$ = UCASE$(fi$)
          CALL FRTEXT(fi$, formrt$, formext$)
          IF fi$ = "" THEN savscal$ = "N"
          fi$ = formrt$ + fi$ + formext$
          LOCATE 23, 2
          PRINT "Saving Scale format file : "; fi$; SPACE$(25);
          CALL sclforsav(fi$, scal$, scmod%(), scval!(), scdiv!())
          FOR I% = 23 TO 25
            LOCATE I%, 1
            PRINT SPACE$(79);
          NEXT
        END IF
        LOCATE 25, 20
        COLOR blue, hlght
        PRINT "Now Scaling the Data";
doit:   DIM scprnt$(NUMVARS%)
        FOR I% = 1 TO NUMVARS%
          SELECT CASE scmod%(I%)
```

```
      CASE 1        ' auto scaling
         vl! = AVE!(I%)
         dv! = varnz!(scdiv!(I%)) / varnz!(I%)
         md% = 1
         CALL scalecol(MESS!(), I%, md%, vl!, dv!)
         scprnt$(I%) = " Auto Scaled to mean =" + STR$(scval!(I%))
         IF scdiv!(I%) <> I% THEN
          scprnt$(I%) = scprnt$(I%) + ", and for variance of
            variable " + STR$(scdiv!(I%))
         END IF
      CASE 2        ' range scaling
         CALL scalecol(MESS!(), I%, 2, scval!(I%), scdiv!(I%))
         scprnt$(I%) = " Range scaled : Min = " + STR$(scval!(I%))
          + ", Max = "
         scprnt$(I%) = scprnt$(I%) + STR$(scdiv!(I%))
      CASE 3
         md% = 3
         vl! = scval!(I%)
         dv! = 1 / scdiv!(I%)
         CALL scalecol(MESS!(), I%, md%, vl!, dv!)
         scprnt$(I%) = " Shifted by " + STR$(vl!) + " and
          multiplied by "
         scprnt$(I%) = scprnt$(I%) + STR$(1 / dv!)
      CASE ELSE
         quashvar%(CURROW%) = 1
         scprnt$(I%) = " Removed from analysis "
      END SELECT
   NEXT I%
   scprnt$(0) = scal$
   IF NOT batchyes% THEN
      CALL sclsave(MESS!(), ID(), CLASS(), scprnt$(), 1, F$)
   END IF
END SUB

SUB BATCHGUYS (batchCALL%, BATOPT$())
' This dumb stupid routine explodes the command line into parameters.
' If there are the proper number of nescessary parameters then the
```



```
' BATCHCALL% flag is set.
' BATOPT$ contains the following :
' BATOPT$(1) = the filename of the DES file. Including root + extension
'       (2) = the scaling option . AUTO, RANGE, NO, YES or the name of
'             the scaling format file. AUTO is the default.
'       (3) = the method of DENDROGRAM-ATION. An integer between 1 and
'             7. 1 is the default.
'       (4) = output filename for DEN file. The default is the same
'             name as the input filename but with the proper
'             extension.
'       (5) = The Minkowski factor. The default is two.
   BATOPT$(1) = ""
   BATOPT$(2) = "AUTO"
   BATOPT$(3) = "1"
   BATOPT$(4) = ""
   BATOPT$(5) = "2"
   C1$ = UCASE$(COMMAND$)
   lc% = LEN(C1$)
   N% = 1
   batchCALL% = INSTR(C1$, "/") > 0
' Go through command$ looking for '/'
   DO
      a% = INSTR(N%, C1$, "/")
      IF a% = 0 THEN GOTO THERE
' Check for type of parameter. The first 2 characters after the / is
' the key.  If there is no equal sign, then a letter indicates that the
'  parameter is the input DES file. A number indicates the desired
'  Minkowski factor. If there is an equal sign, then the letter
'  preceeding it tells of the key.

      C$ = MID$(C1$, a% + 2, 1)
      IF C$ = "=" THEN
         C$ = MID$(C1$, a% + 1, 1)
         SELECT CASE C$
         CASE "S"           ' Scaling option
          s$ = ""
          C$ = ""
```

```
            N% = a% + 3
            DO
              s$ = s$ + C$
              C$ = MID$(C1$, N%, 1)
              N% = N% + 1
            LOOP UNTIL C$ = " " OR C$ = "/" OR N% > lc% + 1

' Now check for AUTO, RANGE, NO, YES
            IF INSTR("AUTO.RANGE.NO.YES", s$) THEN
              BATOPT$(2) = s$
            ELSE
' the string is a filename. Check for its existence.

              s$ = UCASE$(s$)
              CALL FRTEXT(s$, sroot$, sext$)
              F$ = sroot$ + s$ + sext$
              OPEN "R", #1, F$
              le& = LOF(1)
              CLOSE #1
              IF le& = 0 THEN
                BATOPT$(2) = "AUTO"
                KILL F$
              ELSE
                BATOPT$(2) = F$
              END IF
            END IF
          CASE "M"
            C$ = MID$(C1$, a% + 3, 1)
            N% = a% + 3
            meth% = VAL(C$)
            IF meth% >= 1 AND meth% <= 7 THEN BATOPT$(3) = C$
          CASE "O"
            s$ = ""
            C$ = ""
            N% = a% + 3
            DO
              s$ = s$ + C$
```



```
              C$ = MID$(C1$, N%, 1)
              N% = N% + 1
            LOOP UNTIL C$ = " " OR C$ = "/" OR N% > lc% + 1

' the string is a filename.
              s$ = UCASE$(s$)
              CALL FRTEXT(s$, sroot$, sext$)
              F$ = sroot$ + s$ + ".DEN"
              BATOPT$(4) = F$
            CASE ELSE
            END SELECT
          ELSE
' There is no equal sign so the thing is either a filename
' or it is the Minkowski factor.
            N% = a% + 1
            s$ = ""
            C$ = ""
            DO
              s$ = s$ + C$
              C$ = MID$(C1$, N%, 1)
              N% = N% + 1
            LOOP UNTIL (C$ = " " OR C$ = "/" OR lc% + 1 < N%)
            vals = VAL(s$)
            IF vals = 0 THEN
' The string is a filename.
' We need to check for the extension .DES or .SCL. Also if file exists.
              s$ = UCASE$(s$)
              CALL FRTEXT(s$, FROOT$, FEXT$)
              IF NOT INSTR(".DES.SCL.DS1.DS2",FEXT$) THEN
                FEXT$=".DES"
              END IF
              F$ = FROOT$ + s$ + FEXT$
              OPEN "R", #1, F$
              le& = LOF(1)
              CLOSE #1
              IF le& = 0 THEN PRINT "We have a FUCK UP"
              BATOPT$(1) = F$
```

```
            ELSE
              BATOPT$(5) = s$
            END IF
          END IF
        LOOP UNTIL N% > lc%
THERE:
        IF BATOPT$(4) = "" THEN
' There was no input for the output file. Therefore we will give it the
' same as the input file but with the extension DEN
          F$ = BATOPT$(1)
          CALL FRTEXT(F$, FROOT$, FEXT$)
          F$ = FROOT$ + F$ + ".DEN"
          BATOPT$(4) = F$
        END IF
END SUB

SUB CALCDIST (mss!(), NUMSAMS%, NUMVARS%, dst!(), maximum!)
' This routine calculates the distance matrix
' The simple euclidian distance is calculated, unless WARD'S method is
' used
        SHARED Mink!, method%, quashvar%()
        IF method% <> 7 THEN
          FOR X% = 2 TO NUMSAMS%
            FOR Y% = 1 TO X% - 1
              dt! = 0
              FOR numy% = 1 TO NUMVARS%
                IF NOT quashvar%(numy%) THEN
                  dt!=dt!+ABS((mss!(X%,numy%)-mss!(Y%,numy%)))^Mink!
                END IF
              NEXT
              dt! = dt! ^ (1 / Mink!)
              dst!(X%, Y%) = dt!
              dst!(Y%, X%) = dt!
              IF dst!(X%, Y%) > maximum! THEN maximum! = dst!(X%, Y%)
            NEXT Y%
          NEXT X%
        ELSE
```



```
          DIM tempvar!(1 TO NUMVARS%)
          FOR X% = 2 TO NUMSAMS%
            FOR Y% = 1 TO X% - 1
              ess! = 0
              FOR I% = 1 TO NUMVARS%
                tempvar!(I%) = (mss!(X%, I%) + mss!(Y%, I%)) / 2
                ess! = ess! + (mss!(X%, I%) - tempvar!(I%)) ^ 2
                ess! = ess! + (mss!(Y%, I%) - tempvar!(I%)) ^ 2
              NEXT I%
              dst!(X%, Y%) = ess!
              dst!(Y%, X%) = ess!
            NEXT Y%
          NEXT X%
          maximum! = init
        END IF
END SUB

FUNCTION clustmem% (CLUSTNUM%, MEM%)
' This function returns the MEM%'th member of cluster CLUSTNUM%.
' The cluster members are not stored specifically in any array but are
' coded into SAMPPTR%. CLSN%(X) points to the first member of cluster
' X. The value stored in SAMPPTR%(CLSN%(X)) points to the second
' member. The value stored in SAMPPTR%(SAMPPTR%(CLSN%(X))) points to
' the third member, etc.. The SAMPPTR%() array is initialized to zero
' so any non-existant member is returned as ZERO. For this reason, the
' array SAMPPTR%() is dimensioned with a ZERO'th element even though it
' is not used specifically.
        SHARED csdist!(), clsn%(), sampptr%()
        IF MEM% > csdist!(0, CLUSTNUM%) THEN
          clstmm% = 0
        ELSE
          clstmm% = clsn%(CLUSTNUM%)
          FOR I% = 2 TO MEM%
            clstmm% = sampptr%(clstmm%)
          NEXT I%
          clustmem% = clstmm%
        END IF
```

```
END FUNCTION

FUNCTION colmax! (MAT!(), COL%, startrow%, endrow%, COLMIN!)
' returns the maximum value of the column COL% in matrix MAT! from
' startrow% to ENDROW%. COLMIN! is returned as the minimum value of the
' column.

        cmax! = init
        COLMIN! = cmax!
        FOR nr% = startrow% TO endrow%
           vl! = MAT!(nr%, COL%)
           IF vl! < COLMIN! OR COLMIN! = init THEN COLMIN! = vl!
           IF vl! > cmax! OR cmax! = init THEN cmax! = vl!
        NEXT nr%
        colmax! = cmax!
END FUNCTION

SUB DOPT (INFILE$, method%, method$(), Minkowski!, OUTFILE$)
' This routine selects the method for DENDGRAM. The Minkowski factor
' can also be changed here. The output file can also be modified here.
        methmax% = UBOUND(method$)
        COLOR YELLOW, 3
pnt:    CLS
        PRINT STRING$(80, "*")
        COLOR 10
        LOCATE 18, 5
        PRINT "Use arrow keys to choose, C to change Minkowski factor"
        LOCATE 19, 5
        PRINT "? for a brief summary of the different techniques, ESC
          to select."
prnt:   COLOR ntext, bgrnd
        LOCATE 3, 20
        PRINT "DENDROGRAM for :"; INFILE$
        LOCATE 4, 20
        PRINT "Output file    : "; OUTFILE$
        FOR I% = 1 TO methmax%
           LOCATE I% + 6, 20
```



```
           PRINT LEFT$(STR$(I%) + ": " + method$(I%) + SPACE$(60), 60)
        NEXT
        LOCATE 15, 30
        IF method% <> 7 THEN
           PRINT "Minkowski Factor :"; Minkowski!; SPACE$(10);
        ELSE
           PRINT SPC(69);
        END IF
        COLOR ntext, hlght
        IF method% THEN
           LOCATE method% + 6, 20
          OUT$=LEFT$(STR$(METHOD%) + ": " +METHOD$(METHOD%)+SPACE$(40)
           PRINT LEFT$(OUT$(40)
        ELSE
           LOCATE 4, 37
           PRINT OUTFILE$;
        END IF
        COLOR 9, bgrnd
        LOCATE 21, 10
        PRINT SPACE$(65)
        LOCATE 21, 10
1234    a$ = INKEY$
        IF a$ = "" GOTO 1234
        IF ASC(a$) = 0 THEN a$ = RIGHT$(a$, 1)
1235    SELECT CASE a$
        CASE CHR$(77)      ' the RIGHT key was pressed
        CASE CHR$(75)      ' the LEFT key was pressed
        CASE CHR$(13)      ' the RETURN key
           IF method% THEN 1234
           LOCATE 5, 10
           COLOR 0
           INPUT "Name of DEN file to save :", a$
           a$ = UCASE$(a$)
           CALL FRTEXT(a$, roota$, exta$)
           IF a$ <> "" THEN OUTFILE$ = roota$ + a$ + ".DEN"
           COLOR ntext%
           LOCATE 5, 10
```

```
   PRINT SPACE$(69);
   LOCATE 4, 37
   PRINT SPACE$(40);
CASE CHR$(72)      ' the UP key was pressed
   method% = method% - 1
   IF method% = -1 THEN method% = methmax%
CASE CHR$(80)      ' the Down key was pressed
   method% = method% + 1
   IF method% > methmax% THEN method% = 1
CASE "1" TO RIGHT$(STR$(methmax%), 1)
   method% = VAL(a$)
CASE CHR$(27)
   IF method% GOTO bye
CASE "?"
   COLOR ntext, 3
   CLS
   PRINT "The explanation of the various methods can be found
     in the article by Ken"
   PRINT "Burton in the Chemometrics notes from Bristol (ref
     M.SC. Thesis by"
   PRINT " David Sibbald, UBC Chemistry Dept. 1990).  Briefly,
     the difference in"
   PRINT " the methods is in the way in which the distance
     between clusters is calculated."
   LOCATE 5, 19
   PRINT" - Distance taken as shortest distance between any
     member of"
   PRINT "    the cluster."
   LOCATE 7, 21
   PRINT " - Distance taken as LONGEST distance between any
     members"
   PRINT "    of cluster."
   LOCATE 9, 21
   PRINT " - When a join is made, the distance to the new
     cluster is"
   PRINT "    taken to be the weighted average of the distanc
     to the original components."
```



```
LOCATE 11, 31
PRINT " - Same as 3 above but the distance to a new"
PRINT "   cluster is taken as half the sum of the distances
  to the previous components."
LOCATE 13, 13
PRINT " - The actual "; CHR$(34); "location"; CHR$(34); " of
  the cluster is found"
PRINT "    in the variable-space. Distances are calculated
  to this point."
LOCATE 15, 35
PRINT " - The centroid is calculated to be at the"
PRINT "    half way point between the two previous
  components."
LOCATE 17, 18
PRINT " - The distances are calculated as the resulting loss
  of info"
PRINT "    if an arbitrary join is made. The Error of Sum of
  Squares (ESS) is the sum"
PRINT "    of the sqares of the residuals of each variable
  in the possible centroid."
LOCATE 20, 17
PRINT " - The distance is calculated as the";
PRINT SPC(5); "root of the sum "
PRINT "    of the";
PRINT SPC(5);
PRINT "power of the differences in each variable."
PRINT "    For example, the Euclidean distance is given
  when";
PRINT SPC(3); "equals 2.0."
PRINT "    The Manhattan City Block distance is given when";
PRINT SPC(3); "equals 1.0.";
COLOR 12, 7
LOCATE 5, 1: PRINT "1 - Single Linkage";
LOCATE 7, 1: PRINT "2 - Complete Linkage";
LOCATE 9, 1: PRINT "3 - Weighted Average";
LOCATE 11, 1: PRINT "4 - Unweighted Average Linkage";
LOCATE 13, 1: PRINT "5 - Centroid";
```

```
        LOCATE 15, 1: PRINT "6 - Growers Method (Mean Centroid)";
        LOCATE 17, 1: PRINT "7 - Ward's Method"
        LOCATE 20, 1: PRINT "Minkowski factor";
        COLOR YELLOW, 3
        LOCATE 20, 53: PRINT " rth"
        LOCATE 21, 11: PRINT " rth"
        LOCATE 22, 54: PRINT " r"
        LOCATE 23, 52: PRINT " r";
        LOCATE 25, 1
        INPUT ; "continue...", a$
        CLS
        GOTO pnt
     CASE "c", "C"
        IF method% < methmax% THEN
           LOCATE 23, 10
           INPUT "Enter Minkowski factor "; a$
           v1! = VAL(a$)
           IF v1! = 0 THEN
            Minkowski! = 2
           ELSE
            Minkowski! = v1!
           END IF
           LOCATE 23, 10
           PRINT SPACE$(65)
        END IF
     CASE ELSE
     END SELECT
     GOTO prnt:
bye: CLS
END SUB

SUB fixccd (CCD!(), PTR1%, PTR2%, OMEM%, NMEM%, method%)
' This subroutine has a function.  Okay, here it is...
' The cluster-cluster distance matrix needs to be fixed now that a new
' fusion has been made.  This guy does it. The parameters pass are
' similar to those in the other routines (FIXCSD).
' CCD! -The cluster-cluster distance matrix
```



```
' PTR1% The cluster being added to
' PTR2% The cluster being absorbed
' OMEM% The number of members in the original cluster
' NMEM% The number of members of the newly formed cluster. The number
'        added is obviously NMEM%-OMEM%

     SHARED csdist!(), MESS!(), nclsts%, clsn%(), NUMVARS%, NUMSAMS%
     SHARED Mink!, clstvar!(), sampptr%(), quashvar%()
     SELECT CASE method%
     CASE 1, 2
        FOR cl% = 1 TO nclsts%
           IF csdist!(0, cl%) <> 0 AND cl% <> PTR1% THEN
            cl1% = cl%
            cl2% = PTR1%
            IF cl1% > cl2% THEN SWAP cl1%, cl2%
            IF PTR2% > 0 THEN
              IF PTR2% < cl% THEN
                dst2! = CCD!(PTR2%, cl%)
              ELSE
                dst2! = CCD!(cl%, PTR2%)
              END IF
              dst1! = CCD!(cl1%, cl2%)
            ELSEIF PTR2% = 0 THEN
              dst1! = csdist!(clustmem%(PTR1%, 1), cl%)
              dst2! = csdist!(clustmem%(PTR1%, 2), cl%)
            ELSE
              dst2! = csdist!(-PTR2%, cl%)
              dst1! = CCD!(cl1%, cl2%)
            END IF
            IF (method% = 1) XOR (dst1! < dst2!) THEN
              CCD!(cl1%, cl2%) = dst2!
            ELSE
              CCD!(cl1%, cl2%) = dst1!
            END IF
          END IF
        NEXT cl%
```

```
CASE 3, 4
   nfac! = .5: ofac! = .5
   IF method% = 3 THEN
      nfac! = NMEM% / (NMEM% + OMEM%)
      ofac! = OMEM% / (NMEM% + OMEM%)
   END IF
   FOR cl% = 1 TO nclsts%
      IF csdist!(0, cl%) <> 0 AND cl% <> PTR1% THEN
       cl1% = cl%
       cl2% = PTR1%
       IF cl1% > cl2% THEN SWAP cl1%, cl2%
       IF PTR2% > 0 THEN
         dst1! = CCD!(cl1%, cl2%)
         IF cl% < PTR2% THEN
           dst2! = CCD!(cl%, PTR2%)
         ELSE
           dst2! = CCD!(PTR2%, cl%)
         END IF
       ELSEIF PTR2% = 0 THEN
         dst1! = csdist!(clustmem%(PTR1%, 1), cl%)
         dst2! = csdist!(clustmem%(PTR1%, 2), cl%)
       ELSE
         dst1! = CCD!(cl1%, cl2%)
         dst2! = csdist!(-PTR2%, cl%)
       END IF
       CCD!(cl1%, cl2%) = dst1! * ofac! + dst2! * nfac!
     END IF
   NEXT cl%

CASE 5, 6
   FOR cl% = 1 TO nclsts%
      IF cl% <> PTR1% AND csdist!(0, cl%) <> 0 THEN
       dst! = 0
       FOR I% = 1 TO NUMVARS%
         IF NOT quashvar%(I%) THEN
           dst! = dst! + ABS((clstvar!(cl%,I%) -
             clstvar!(PTR1%,I%)))^Mink!
```



```
         END IF
       NEXT I%
       dst! = (dst!) ^ (1 / Mink!)
       IF cl% < PTR1% THEN
         CCD!(cl%, PTR1%) = dst!
       ELSE
         CCD!(PTR1%, cl%) = dst!
       END IF
     END IF
   NEXT cl%

CASE 7
   DIM tempvar!(1 TO NUMVARS%)
   FOR cl% = 1 TO nclsts%
      IF csdist!(0, cl%) <> 0 AND cl% <> PTR1% THEN
       ess! = 0
       cfc! = csdist!(0,cl%)/(csdist!(0,cl%)+csdist!(0,PTR1%))
       pfc!=csdist!(0,PTR1%)/(csdist!(0,cl%)+csdist!(0,PTR1%))
       FOR I% = 1 TO NUMVARS%
         IF NOT quashvar%(I%) THEN
           tempvar!(I%) = clstvar!(cl%, I%) * cfc!
           tempvar!(I%) =clstvar!(PTR1%,I%)*pfc!+tempvar!(I%)
         END IF
       NEXT
       FOR MEM% = 1 TO csdist!(0, cl%)
         curmem% = clustmem%(cl%, MEM%)
         FOR I% = 1 TO NUMVARS%
           IF NOT quashvar%(I%) THEN
             ess!=ess!+(MESS!(curmem%,I%) - tempvar!(I%)) ^ 2
           END IF
         NEXT
       NEXT MEM%
       FOR MEM% = 1 TO csdist!(0, PTR1%)
         curmem% = clustmem%(PTR1%, MEM%)
         FOR I% = 1 TO NUMVARS%
           IF NOT quashvar%(I%) THEN
             ess! = ess! + (MESS!(curmem%,I%)-tempvar!(I%))^2
```

```
            END IF
          NEXT I%
        NEXT MEM%
        IF cl% < PTR1% THEN
          CCD!(cl%, PTR1%) = ess!
        ELSE
          CCD!(PTR1%, cl%) = ess!
        END IF
      END IF
    NEXT cl%
    ERASE tempvar!
  END SELECT
END SUB

SUB fixcsd (CSD!(), PTR1%, PTR2%, OMEM%, NMEM%, method%)
' Similar to FIXCCD, but fixes the cluster-sample matrix. Only the
' elements actually affected need be changed (according to METHOD%)
  SHARED MESS!(), SSDIST!(), nclsts%, clsn%(), NUMVARS%, NUMSAMS%
  SHARED clstvar!(), Mink!, sampptr%(), quashvar%()
  nfac! = .5: ofac! = .5
  IF method% = 3 OR method% = 5 THEN
    nfac! = NMEM% / (NMEM% + OMEM%)
    ofac! = OMEM% / (NMEM% + OMEM%)
  END IF
  IF method% > 4 AND method% < 8 THEN
    FOR I% = 1 TO NUMVARS%
      IF NOT quashvar%(I%) THEN
        IF PTR2% = 0 THEN ' join two points to make new cluster
          var1! = MESS!(clustmem%(PTR1%, 1), I%)
          var2! = MESS!(clustmem%(PTR1%, 2), I%)
        ELSEIF PTR2% < 0 THEN
          var2! = MESS!(-PTR2%, I%)
          var1! = clstvar!(PTR1%, I%)
        ELSE
          var1! = clstvar!(PTR1%, I%)
          var2! = clstvar!(PTR2%, I%)
        END IF
```



```
        var1! = var1! * ofac!
        var2! = var2! * nfac!
        clstvar!(PTR1%, I%) = var1! + var2!
      END IF
    NEXT
  END IF
' clstvar! is now correct for the new join.
' Calculate distances between all single points and the new cluster.
  SELECT CASE method%
  CASE 1, 2
    FOR SAMP% = 1 TO NUMSAMS%
      IF CSD!(SAMP%, 0) = 0 THEN
        IF PTR2% > 0 THEN
          dst1! = CSD!(SAMP%, PTR1%)
          dst2! = CSD!(SAMP%, PTR2%)
        ELSEIF PTR2% = 0 THEN
          dst1! = SSDIST!(SAMP%, clustmem%(PTR1%, 1))
          dst2! = SSDIST!(SAMP%, clustmem%(PTR1%, 2))
        ELSE
          dst1! = CSD!(SAMP%, PTR1%)
          dst2! = SSDIST!(SAMP%, -PTR2%)
        END IF
        IF (method% = 1) XOR (dst1! < dst2!) THEN
          CSD!(SAMP%, PTR1%) = dst2!
        ELSE
          CSD!(SAMP%, PTR1%) = dst1!
        END IF
      END IF
    NEXT SAMP%

  CASE 3, 4
    FOR SAMP% = 1 TO NUMSAMS%
      IF CSD!(SAMP%, 0) = 0 THEN
        IF PTR2% > 0 THEN
          dst1! = CSD!(SAMP%, PTR1%)
          dst2! = CSD!(SAMP%, PTR2%)
        ELSEIF PTR2% = 0 THEN
```

```
            dst1! = SSDIST!(SAMP%, clustmem%(PTR1%, 1))
            dst2! = SSDIST!(SAMP%, clustmem%(PTR1%, 2))
          ELSE
            dst1! = CSD!(SAMP%, PTR1%)
            IF SAMP% < -PTR2% THEN
              dst2! = SSDIST!(SAMP%, -PTR2%)
            ELSE
              dst2! = SSDIST!(-PTR2%, SAMP%)
            END IF
          END IF
          CSD!(SAMP%, PTR1%) = dst1! * ofac! + dst2! * nfac!
        END IF
      NEXT SAMP%

CASE 5, 6
      FOR SAMP% = 1 TO NUMSAMS%
        IF CSD!(SAMP%, 0) = 0 THEN
          dst! = 0
          FOR I% = 1 TO NUMVARS%
            IF NOT quashvar%(I%) THEN
              dst!=dst!+ABS((MESS!(SAMP%,I%)-
                clstvar!(PTR1%,I%)))^Mink!
            END IF
          NEXT I%
          CSD!(SAMP%, PTR1%) = dst! ^ (1 / Mink!)
        END IF
      NEXT SAMP%

CASE 7
      DIM tempvar!(1 TO NUMVARS%)
      FOR SAMP% = 1 TO NUMSAMS%
        IF CSD!(SAMP%, 0) = 0 THEN
          ess! = 0
          FOR I% = 1 TO NUMVARS%
            IF NOT quashvar%(I%) THEN
              tempvar!(I%) = clstvar!(PTR1%, I%) * ofac!
              tempvar!(I%) = tempvar!(I%)+MESS!(SAMP%,I%)*nfac!
```



```
              ess! = ess! + (MESS!(SAMP%, I%) - tempvar!(I%))^2
            END IF
          NEXT I%
          FOR MEM% = 1 TO CSD!(0, PTR1%)
            FOR I% = 1 TO NUMVARS%
              IF NOT quashvar%(I%) THEN
                curmem% = clustmem%(PTR1%, MEM%)
                ess! = ess! + (MESS!(curmem%,I%)-tempvar!(I%))^2
              END IF
            NEXT I%
          NEXT MEM%
          CSD!(SAMP%, PTR1%) = ess!
        END IF
      NEXT SAMP%
      ERASE tempvar!
    CASE ELSE
      ' error
    END SELECT
END SUB

SUB FRTEXT (FILE$, FROOT$, FEXT$) STATIC
' It is this routines function to break a file name into root,file, and
' extension. file$ is the input string.
' FILE$ contains filename on return (maximum 8 characters)
' FROOT$ contains path information
' FEXT$ contains extension
'
' Loop to remove spaces from file name
    LENGTH% = LEN(FILE$)
    F$ = FILE$
    FILE$ = ""
    FOR I% = 1 TO LENGTH%
      IF MID$(F$, I%, 1) <> " " THEN FILE$ = FILE$ + MID$(F$,I%,1)
    NEXT I%
    FROOT$ = ""
    FEXT$ = ""
    a% = INSTR(FILE$, ".")
```

```
        IF a% = 0 GOTO nroot ' no "." means no extension
        IF a% = LENGTH% THEN
          FEXT$ = ""
          FILE$ = LEFT$(FILE$, LENGTH% - 1)
          LENGTH% = LEN(FILE$)
          GOTO nroot
        END IF
        FEXT$ = MID$(FILE$, a%, 4)
        FILE$ = LEFT$(FILE$, a% - 1)
' extension taken care of, now peel off root

nroot:  LENGTH% = LEN(FILE$)
        a% = INSTR(FILE$, "\")
        IF a% = LENGTH% THEN
          FROOT$ = FROOT$ + FILE$
          FILE$ = ""
          GOTO fin
        END IF
        IF a% = 0 GOTO fin
        FROOT$ = FROOT$ + LEFT$(FILE$, a%)
        FILE$ = RIGHT$(FILE$, LENGTH% - a%)
        GOTO nroot
fin:    END SUB

FUNCTION MATCH (X%, Y%)
' MATCH tells us if sample x% is one of the samples in node Y%
        SHARED DEND%()
        IF Y% > 0 THEN
          MATCH = (X% = DEND%(Y%, 1) OR X% = DEND%(Y%, 2))
        END IF
END FUNCTION

SUB READPARS (NUMSAMS%, NUMVARS%, EXT1%, EXT2%, F$)
' This routine is the first half of the file reading section. The
' second half is called READVALS. This section reads in the two
' parameters at the head of the file F$ (F$ is opened according to
' FTYPE%). NUMSAMS% is the number of signals recorded in the file and
```



```
' NUMVARS is the number of descriptors per signal. EXT1 and EXT2 are
' not currently used but are included for future use.

        FTYPE% = (INSTR("DESDS1DS2SCLAFAAF2", RIGHT$(F$, 3)) - 1) / 3 +
1
        SELECT CASE FTYPE%
        CASE 1, 4, 5
          OPEN "i", #1, F$
          INPUT #1, NUMSAMS%
          INPUT #1, NUMVARS%
          EXT1% = 0
          EXT2% = 0
          CLOSE #1
        CASE 2
          OPEN F$ FOR RANDOM ACCESS READ AS #1 LEN = 4
          FIELD #1, 4 AS REC1$
          GET #1                          ' GET the first record
          TMP$ = LEFT$(REC1$, 2)          ' Parse out the 1ST 2 bytes
          NUMSAMS% = CVI(TMP$)            ' Convert to # of signals
          TMP$ = RIGHT$(REC1$, 2)         ' Parse out last 2 bytes
          NUMVARS% = CVI(TMP$)            ' Convert # of descriptors
          GET #1                          ' get the extra record
          TMP$ = REC1$                    ' convert in future *******
          EXT1% = &H7 AND ASC(RIGHT$(TMP$, 1))
          EXT2% = CVI(LEFT$(TMP$, 2))
          CLOSE #1
        CASE 3, 6
          OPEN "i", #1, F$
          INPUT #1, NUMSAMS%
          INPUT #1, NUMVARS%
          INPUT #1, EXT2%, EXT1%
          CLOSE #1
        END SELECT
END SUB

SUB READVALS (VARNAM$(), MESS!(), ID() AS RECRD, CLASS() AS RECRD,
SCALED$, F$)
```

```
' This routine is the second half of the read file sub program. The
' routine is broken in two halves in order for DIMensioning in the main
' module.

      DIM EXTRA AS RECRD
      FTYPE% = (INSTR("DESDS1DS2SCLAFAAF2",RIGHT$(F$,3)) - 1) / 3 + 1
      SELECT CASE FTYPE%
      CASE 5
         STOP ' NOT PRESENT IN THIS VERSION
         ' TALK TO DAVE - OR USE ABSCAT TO CREATE DS2 FILE
      CASE 1, 4
         OPEN "I", #1, F$
         INPUT #1, NUMSAMS%, NUMVARS%
         FOR N% = 1 TO NUMVARS%
            INPUT #1, VARNAM$(N%)
         NEXT
         FOR nx% = 1 TO NUMSAMS%
            INPUT #1, ID(nx%).R
            INPUT #1, CLASS(nx%).R
            FOR nY% = 1 TO NUMVARS%
             INPUT #1, MESS!(nx%, nY%)
            NEXT
         NEXT
         IF FTYPE% = 4 THEN INPUT #1, SCALED$
         CLOSE #1
      CASE 2
         OPEN F$ FOR RANDOM ACCESS READ AS #1 LEN = 4
         FIELD #1, 4 AS REC1$
         GET #1                          ' GET the first record
         TMP$ = LEFT$(REC1$, 2)          ' Parse out the 1ST 2 bytes
         NUMSAMS% = CVI(TMP$)            ' Convert to # of signals
         TMP$ = RIGHT$(REC1$, 2)         ' Parse out last two bytes
         NUMVARS% = CVI(TMP$)            ' Convert # of descriptors
         GET #1                          ' get the extra record
         TMP$ = REC1$                    ' convert in future ******
         EXT1% = &H7 AND ASC(RIGHT$(TMP$, 1))
         EXT2% = CVI(LEFT$(TMP$, 2))
```



```
         IF EXT1% > 0 THEN GET #1        ' skip extra record name
         FOR N% = 1 TO EXT2%
            GET #1                       ' skip extended records
            GET #1
         NEXT
' Get the descriptor names
         FOR I% = 1 TO NUMVARS%
            GET #1
            TMP$ = REC1$
            GET #1
            VARNAM$(I%) = TMP$ + REC1$
         NEXT I%
'
' Signal / descriptor information
         FOR I% = 1 TO NUMSAMS%
            GET #1                       ' signal ID
            ID(I%).R = REC1$
            GET #1                       ' signal class
            CLASS(I%).R = REC1$
            GET #1                       ' time of signal
            GET #1                       ' extra record
            FOR J% = 1 TO EXT2%
             GET #1                      ' skip extended records
            NEXT
            FOR J% = 1 TO NUMVARS%
             GET #1
             MESS!(I%, J%) = CVS(REC1$)
            NEXT
         NEXT
         CLOSE #1

      CASE 3
         OPEN "I", #1, F$
         INPUT #1, NUMSAMS%, NUMVARS%
         INPUT #1, EXT2%, EXT1%
         IF EXT1% <> 0 THEN INPUT #1, EXT$   ' read extra record name
         FOR N% = 1 TO EXT2%
```

```
      INPUT #1, EXNM$              ' read extended record names
   NEXT
   FOR N% = 1 TO NUMVARS%
      INPUT #1, VARNAM$(N%)
   NEXT
   FOR nx% = 1 TO NUMSAMS%
      INPUT #1, ID(nx%).R
      INPUT #1, CLASS(nx%).R
      INPUT #1, TIME!
      INPUT #1, EXTRA.R
      FOR N% = 1 TO EXT2%
       INPUT #1, EXTEN$
      NEXT
      FOR nY% = 1 TO NUMVARS%
       INPUT #1, MESS!(nx%, nY%)
      NEXT
   NEXT
   CLOSE #1
CASE 6
   OPEN "I", #1, F$
   INPUT #1, NUMSAMS%, NUMVARS%
   INPUT #1, EXT2%, EXTI%
   IF EXT1% <> 0 THEN INPUT #1, EXR$
   FOR N% = 1 TO EXT2%
      INPUT #1, EXTNM$
   NEXT
   FOR I% = 1 TO NUMSAMS%
      INPUT #1, ID(I%).R
      INPUT #1, CLASS(I%).R
      INPUT #1, TIME!
      INPUT #1, EXTRA.R
      FOR N% = 1 TO EXT2%
       INPUT #1, EXTEN$
      NEXT
      FOR J% = 1 TO NUMVARS%
       INPUT #1, MESS!(I%, J%)
      NEXT
```



```
      NEXT
   CASE ELSE
      'BLOWUP
   END SELECT
END SUB

SUB scalecol (MAT!(), COL%, mode%, value!, div!) STATIC
' This routine performs the scaling on the COL%'th column of MAT!
' according to the value of MODE% and (if nescessary the) VALUE!.
'
' MODE% 1 The column is autoscaled such that the mean of the column is
'         VALUE! and the total variance equals DIV!. For true
'         autoscaling, the mean should be scaled to ZERO and the value
'         of DIV! should be ONE. To scale a column such that it has the
'         same variance relative to a different column, DIV! should
'         equal the quotient of the variances.
'         DIV! = VARIANCE(of other column) / VARIANCE(of COL%umn)
'       2 The column is range scaled such that the column maximum is
'         VALUE! and the minumum value is DIV!. For range scaling
'         0 - 1, VALUE!=1 and DIV! = 0. NOTE: it does not matter that
'         VALUE! be larger than DIV! except that the scaled values will
'         all be inverted about the column mean.If VALUE! = DIV! then
'         the values will all be set to VALUE!
'       3 Each element of the column has VALUE! subtracted from it and
'         is then divided by DIV!. This MODE% is called recursively
'         from MODE%=1 and is included to allow multiplication of a
'         column by a certain value (DIV! = 1/FACTOR) or shifting a
'         column by a VALUE! (NOTE: DIV! should equal 1 to accomplish a
'         shift and VALUE! should equal 0 for a multiplication only.
'
   SHARED NUMSAMS%
   SELECT CASE mode%
   CASE 1
      varnz! = VARIANCE(MAT!(), COL%, NUMSAMS%, average!)
      vl! = average! - value!
      dv! = SQR(div! * varnz!)
      md% = 3
```

```
        CALL scalecol(MAT!(), COL%, md%, vl!, dv!)
      CASE 2
        cmax! = colmax!(MAT!(), COL%, 1, NUMSAMS%, cmin!)
        vl! = cmin!
        dv! = 1
        IF cmax!<>cmin! THEN dv! = (cmax! - cmin!) / (div! - value!)
        md% = 3
        CALL scalecol(MAT!(), COL%, md%, vl!, dv!)
        vl! = value! * SGN(div! - value!)
        dv! = 1
        md% = 3
        CALL scalecol(MAT!(), COL%, md%, vl!, dv!)
      CASE 3
        FOR nx% = 1 TO NUMSAMS%
          MAT!(nx%, COL%) = (MAT!(nx%, COL%) - value!) / div!
        NEXT nx%
      CASE ELSE
        STOP
      END SELECT
END SUB

SUB scaler (MESS!(), ID() AS RECRD, CLASS() AS RECRD, SCLE$(), F$)
' This program will scale the matrix mess! according to user input. The
' method for scaling the rows is described in the SUBprogram SCaLECOL.
      SHARED NUMVARS%, NUMSAMS%
      SHARED batchyes%, BATOPT$()
      IF batchyes% THEN
        IF BATOPT$(2) = "AUTO" GOTO 12
        IF BATOPT$(2) = "RANGE" GOTO 13
        IF BATOPT$(2) = "NO" GOTO 14
        GOTO 15
      END IF
      scale% = 1
      scalmax! = 1
      scalmin! = 0
      COLOR 1, bgrnd
      CLS
```



```
      SOUND 330, 1
      PRINT"************************************************************
        ********************"
      LOCATE 3, 30
      PRINT "SCALING"
      LOCATE 22, 15
      PRINT "Use arrows to choose. Press RETURN to select.";

3     COLOR 10, bgrnd
      FOR I% = 1 TO UBOUND(SCLE$)
        LOCATE 7 + I% * 2, 15
        PRINT LEFT$(STR$(I%) + ") " + SCLE$(I%) + SPACE$(30), 30)
      NEXT I%
      COLOR ntext, hlght
      LOCATE 7 + scale% * 2, 15
      PRINT LEFT$(STR$(scale%) + ") " + SCLE$(scale%) +SPACE$(30),30)
      COLOR 10, bgrnd
      IF scale% = 2 THEN
        LOCATE 11, 55
        PRINT "Max = "; LEFT$(STR$(scalmax!) + SPACE$(18), 18)
        LOCATE 12, 55
        PRINT "Min = "; LEFT$(STR$(scalmin!) + SPACE$(18), 18)
      ELSE
        LOCATE 11, 55: PRINT SPACE$(24);
        LOCATE 12, 55: PRINT SPACE$(24);
      END IF
4     a$ = INKEY$
      IF a$ = "" GOTO 4
      IF a$ = CHR$(13) GOTO 11
      IF a$ = CHR$(27) THEN
        NUMSAMS% = -1
        EXIT SUB  ' ESC was pressed
      END IF
      IF ASC(a$) <> 0 GOTO 4
      a$ = RIGHT$(a$, 1)
      SELECT CASE a$
      CASE CHR$(72)
```

```
      scale% = scale% - 1
      IF scale% = 0 THEN scale% = UBOUND(SCLE$)
      GOTO 3
   CASE CHR$(80)
      scale% = scale% + 1
      IF scale% > UBOUND(SCLE$) THEN scale% = 1
      GOTO 3
   CASE CHR$(77)      ' the RIGHT key was pressed
      IF scale% = 2 THEN
         COLOR YELLOW, hlght
         LOCATE scale% * 2 + 7, 61
         PRINT SPC(10);
         LOCATE scale% * 2 + 7, 61
         COLOR YELLOW, bgrnd
         INPUT num$
         scalmax! = VAL(num$)
         COLOR YELLOW, hlght
         LOCATE scale% * 2 + 8, 61
         PRINT SPC(10);
         LOCATE , 61
         COLOR YELLOW, bgrnd
         INPUT num$
         scalmin! = VAL(num$)
      END IF
      GOTO 3
   CASE ELSE
      GOTO 4
   END SELECT
11 LOCATE 22, 1
   PRINT SPACE$(79)
   LOCATE 22, 30
   IF scale% = 4 THEN
      PRINT "Please wait, calculating variables"
   ELSE
      PRINT "Now Scaling ...."
   END IF
   ON scale% GOTO 12, 13, 14, 15
```



```
' The first choice is selected. This will be the Autoscaling function
' for all columns.
12    vl! = 0
      dv! = 1
      md% = 1
      FOR nC% = 1 TO NUMVARS%
        CALL scalecol(MESS!(), nC%, md%, vl!, dv!)
      NEXT nC%
      SCLE$(1) = "Auto Scaled"
      GOTO 40

' The second scaling feature was selected. This will be range scaling.
13    vl! = scalmax!
      dv! = scalmin!
      md% = 2
      FOR nC% = 1 TO NUMVARS%
        CALL scalecol(MESS!(), nC%, md%, vl!, dv!)
      NEXT nC%
      SCLE$(1) = "Range Scaled:" + STR$(scalmin!)+" to "
        +STR$(scalmax!)
      GOTO 40

' The third scaling feature is Not to scale. Tricky
14    SCLE$(1) = "No Scaling"
      GOTO 40

15 ' Now it's going to get interesting. We have to supply the column
' statistics
      CALL ADVSCAL(MESS!(), ID(), CLASS(), F$, SCLE$(1))
40    LOCATE 21, 30
END SUB

SUB sclforlod (FILE$, sc$, mode%(), a!(), B!())
' This subprogram is designed to read in a file saved by SCLFORSAV and
' parse it into the ADVSCAL subprogram so that the lazy Joe's don't
' have to type so much. Also it is nice to be able to ensure that
```

```
' scaling is consistent between data sets. If the number of descriptors
' is different between the current data set and the saved format file,
' then (uh oh) the program reads in only enough to use (ie. the extra
' formatting is ignored). If there is not enough in the file the
' variables left over are left as the default.

            OPEN "i", #3, FILE$
            INPUT #3, sc$
            FOR I% = 1 TO UBOUND(mode%)
              INPUT #3, mode%(I%), a!(I%), B!(I%)
            NEXT
            CLOSE #3
END SUB

SUB sclforsav (FILE$, sc$, a%(), B!(), C!())
' This is the sister routine to SCLFORLOD.
        OPEN "O", #2, FILE$
        WRITE #2, sc$
        FOR I% = 1 TO UBOUND(a%)
          WRITE #2, a%(I%), B!(I%), C!(I%)
        NEXT I%
        PRINT #2, "This file created on " + DATE$ + " at " + TIME$
        CLOSE #2
END SUB

SUB sclsave (MAT!(), ID() AS RECRD, CLASS() AS RECRD, sc$(), spec%, F$)
' This routine saves the matrix as a scaled file in the same format as
' the original data file. (Note: a scaled DS1 file is saved as a SC2
' file)
        SHARED NUMSAMS%, NUMVARS%
        SHARED VARNAM$()

' ***** Output matrix to file$
        COLOR ntext, hlght
        LOCATE 25, 1
        PRINT SPACE$(79);
        LOCATE 25, 10
```



```
INPUT ; "Save scaled data file"; a$
IF UCASE$(LEFT$(a$, 1)) = "Y" THEN
  EXT$ = RIGHT$(F$, 3)
  F$ = LEFT$(F$, LEN(F$) - 4)
  LOCATE 25, 10
  PRINT "File Name = <"; F$; ">";
  INPUT ; a$
  a$ = UCASE$(a$)
  CALL FRTEXT(a$, roota$, exta$)
  IF a$ = "" THEN
    a$ = F$
  ELSE
    a$ = roota$ + a$
  END IF
  IF EXT$ = "DS1" OR EXT$ = "DS2" THEN
    exta$ = ".SC2"
  ELSE
    exta$ = ".SCL"
  END IF
  a$ = a$ + exta$
  OPEN "o", #3, a$
  WRITE #3, NUMSAMS%
  WRITE #3, NUMVARS%
  FOR nv% = 1 TO NUMVARS%
    WRITE #3, VARNAM$(nv%)
  NEXT
  Q$ = CHR$(34)
  FOR nx% = 1 TO NUMSAMS%
    PRINT #3, Q$ + ID(nx%).R + Q$, Q$ + CLASS(nx%).R + Q$,
    FOR nY% = 1 TO NUMVARS% - 1
      PRINT #3, MAT!(nx%, nY%),
    NEXT
    PRINT #3, MAT!(nx%, NUMVARS%)
  NEXT
  WRITE #3, sc$(0)
  IF spec% THEN
    lim% = UBOUND(sc$)
```

```
            FOR I% = 1 TO lim%
              WRITE #3, "Variable #" + STR$(I%) + sc$(I%)
            NEXT I%
          END IF
          CLOSE #3
' This is a pause put in for no obviously apparent reason. Perhaps some
' people  were distressed at the speed of scaling and saving of small
' data sets and this is put in to ease these people's anxieties. (Kim
' knows who these people are.)
          FOR I% = 1 TO 20000
          NEXT I%
        END IF
END SUB

FUNCTION VARIANCE! (MAT!(), COL%, numrows%, AVE!)
' returns the variance of the COL%'th column of the MAT!rix from the
' first to the NUMROWS%'th row.
        sum! = 0
        FOR nx% = 1 TO numrows%
          sum! = sum! + MAT!(nx%, COL%)
        NEXT nx%
        AVE! = sum! / numrows%
        vr! = 0
        FOR nx% = 1 TO numrows%
          vr! = vr! + (MAT!(nx%, COL%) - AVE!) ^ 2
        NEXT nx%
        IF numrows% > 1 THEN
          VARIANCE! = vr! / (numrows% - 1)
        ELSE
          VARIANCE! = 0
        END IF
END FUNCTION

' We're done! Time to go out on the boat and skip Kathleen's cookies!
```

```
'************************************************************************
' DENDPLOT - By David Sibbald
'
' The Laboratory for Automated Chemical Analysis
' Department of Chemistry
' University of British Columbia
'
' November 1989
'************************************************************************
'
' This is part two of the DENDOGRAM program package.  This program loads
' in the dendrogram data from a .DEN file created by the DENDOGRAM
' program and displays it on the screen.  EGA with 256K of memory is
' required.
'************************************************************************

DECLARE FUNCTION DFCALC! (POSIT!, MAX!, FORM$)
DECLARE SUB GETFILE (FILE$)
DECLARE SUB PMOVE (px%(), PY%(), nx%, nY%, oy1%, oy2%)
DECLARE SUB LST (CURSOR%, MARK%(), id() AS ANY, CLASS() AS ANY,
        PLOT%(), C%())
DECLARE SUB MRK (X%, Y%, Y1%, YL%, C%)
DECLARE SUB CMRK (X%, Y%, C%)
DECLARE SUB plotter (DEND%(), DENDD!(), PLOT%(), id() AS ANY, CLASS()
        AS ANY, MAXDST!)
DECLARE SUB DLINE (X1%, Y1%, X2%, Y2%, YMAX%, TWOPAGES%, PL7440%)
DECLARE SUB FRTEXT (FILE$, FROOT$, FEXT$)
DECLARE FUNCTION WADEDST! (NODE%)

' *** FNTXT% returns the print line on screen corresponding
' ***        to graphics line n
        DEF fntxt% (N)
           IF N < 8 THEN fntxt% = 1 ELSE fntxt% = (N + 9) * 25 / 350
        END DEF
        TYPE RECRD
           R AS STRING * 4
        END TYPE
```



```
        CONST bgrnd = 11
        CONST hlght = 4
        CONST YELLOW = 14
        CONST dblue = 9
        CONST ntext = 15

        REM $DYNAMIC
        DATA LINEAR,,LOGARITHMIC,QUADRATIC
        DMETHS% = 4
        DIM DSMETH$(DMETHS%)
        FOR I% = 1 TO DMETHS%
           READ DSMETH$(I%)
        NEXT

' Check COMMAND$ to see if this program has been called from DENDOGRAM
' with a  file name. If so, we can read it straight in, otherwise, get
' one.
        FILE$ = COMMAND$
        IF FILE$ = "" THEN
           CALL GETFILE(FILE$)
           IF FILE$ = "" THEN GOTO BYBY
        END IF
        IF INSTR(FILE$, ".") = 0 THEN FILE$ = FILE$ + ".DEN"
' Okay, now we have a file name. "We're not sure where the file name
' came from, all we're sure is: we have one now" - apologies to Rod,
' the fascist geographer.
' So, read it in
        OPEN "I", #1, FILE$
        INPUT #1, NUMSAMS%, NUMVARS%
        DIM VARNAM$(NUMVARS%)
        FOR I% = 1 TO NUMVARS%
           INPUT #1, VARNAM$(I%)
        NEXT
' Dimension some variables to hold the stuff
        DIM DENDD!(NUMSAMS% - 1) 'The array of the dissimilarities
        DIM DEND%(NUMSAMS% - 1, 2)       'The list of pairs of fusions
```

```
        DIM PLOT%(NUMSAMS%)       'The order in which the samples lie
                                  'on the vertical axis.
        DIM id(NUMSAMS%) AS RECRD 'The ID's of the samples
        DIM CLASS(NUMSAMS%) AS RECRD  'The CLASS designations
        FOR I% = 1 TO NUMSAMS% - 1
           INPUT #1, DEND%(I%, 1)
           INPUT #1, DEND%(I%, 2)
           INPUT #1, DENDD!(I%)
        NEXT
        INPUT #1, MAXDST!
        FOR I% = 1 TO NUMSAMS%
           INPUT #1, PLOT%(I%)
           INPUT #1, id(PLOT%(I%)).R
           INPUT #1, CLASS(PLOT%(I%)).R
        NEXT
        INPUT #1, scaled$
        INPUT #1, method$
        CLOSE #1
' It should be said that at this point, the ID of sample number one -
' ID(1) is not the same as the ID of the first sample. (Huh?).  ID(1)
' contains the ID of the  sample (signal) which is displayed at the top
' (or left) of the dendrogram - ie. it will be one of the two samples
' involved in the very first fusion made when constructing the
' dendrogram.  So be careful when converting between this program and
' the other data analysis program. This is confusing, but more useful
' than a screen door on a battleship.

' Now plot dendrogram
899     CLS
        DIM SHARED SCRN%
        DIM SHARED VEEW%
        SCRN% = 9
        CALL plotter(DEND%(), DENDD!(), PLOT%(), id(), CLASS(),
        MAXDST!)
BYBY:   SCREEN 0
        CLS
END
```



```
REM $STATIC
SUB CMRK (X%, Y%, C%)
' Subroutine to make a cursor at position x,y on the screen with color
        C%
        ch$ = " H3BD6E3L5"        ' bit mapping for a cross
        mk$ = "C" + STR$(C%) + "BM" + STR$(X% - 1) + "," + STR$(Y%) +
        ch$
        DRAW mk$
END SUB

FUNCTION DFCALC! (POSIT!, MAX!, FORM$)
' This function calculates a position for the  plotting of a dendral
' link occuring at POSIT! dissimilarity value. MAX! should be the
' maximum value (ie - the value of DENDD!(NUMSAMS%-1) OR MAXDST!) from
' main module). FORM$ is a number of the plot method desired. The
' VAL(FORM$) is taken. This is an integer refering to the format of
' display. If an extra parameter is required to be passed (such as the
' R! value for quadratic display) then this is stored in FORM$ after a
' decimal point.
'
' The function value is returned as a single precision value between 0
' and 1.  This value corresponds to the plot position relative to MAX!.
' Eg. a position calculated to be shown at 1/4 the displacement from
' the bottom the the MAX! should have would return .25 as the function
' value. So the point of all this is that the returned value must still
' be multiplied by the plot window size (XMAX%-XMIN%) and added to the
' offset (XMIN%)
' CURRENTLY:
' 1  :   Linear  (normal)
' 2  :   Linear  (relative axis - affectionately called the Wade-
        Wentzell )
' 3  :   Logarithmic
' 4  :   Quadratic of the form x^Rth power

        SELECT CASE INT(VAL(FORM$))
        CASE 1
```

```
         D! = POSIT! / MAX!
      CASE 2
         D! = POSIT! / MAX!
' The actual calculation for the Wade- Wentzell method is
' done in the WADEDST! routine called by the PLOTTER subroutine.
      CASE 3
         D! = LOG(POSIT! + 1) / LOG(MAX! + 1)
         IF D! < 0 THEN D! = 0
      CASE 4
         R! = VAL(RIGHT$(FORM$, LEN(FORM$) - INSTR(FORM$, ".")))
         D! = (POSIT! ^ R!) / (MAX! ^ R!)
      CASE ELSE
         STOP
      END SELECT
      IF D! < 0 THEN D! = 0
      IF D! > 1 THEN D! = 1
      DFCALC! = D!
END FUNCTION

SUB DLINE (X1%, Y1%, X2%, Y2%, YMAX%, TWOPAGES%, PL7440%)
' *** DLINE draws a line between two points
'     If the y-coordinate is off scale then page two is used if
'     TWOPAGES% = - 1

      SHARED SCRN%, VEEW%
      IF PL7440% THEN
         PRINT #1, "PA"; Y1%; ","; X1%;'";PD;PA"; Y2%; ","; X2%;
      ";PU;"
         GOTO 3000
      END IF
      IF NOT TWOPAGES% THEN
         SCREEN SCRN%, , 0, VEEW%
         LINE (X1%, Y1%)-(X2%, Y2%)
      ELSE
         IF Y1% > YMAX% THEN
            IF Y2% > YMAX% THEN
              SCREEN SCRN%, , 1, VEEW%
```



```
              LINE (X1%, Y1% - 2 * YMAX% + 350)-(X2%,Y2%-2*YMAX%+350)
            ELSE
              slope! = (X1% - X2%) / (Y1% - Y2%)
              xmid% = slope! * (YMAX% - Y2%) + X2%
              SCREEN SCRN%, , 0, VEEW%
              LINE (X2%, Y2%)-(xmid%, YMAX% + 10)
              SCREEN SCRN%, , 1, VEEW%
              LINE (xmid%, 350 - YMAX% - 10)-(X1%, Y1% -2*YMAX%+350)
            END IF
         ELSE ' y1% <= ymax%
            IF Y2% <= YMAX% THEN
              SCREEN SCRN%, , 0, VEEW%
              LINE (X1%, Y1%)-(X2%, Y2%)
            ELSE
              slope! = (X1% - X2%) / (Y1% - Y2%)
              xmid% = slope! * (YMAX% - Y2%) + X2%
              SCREEN SCRN%, , 0, VEEW%
              LINE (X1%, Y1%)-(xmid%, YMAX% + 10)
              SCREEN SCRN%, , 1, VEEW%
              LINE (xmid%, 350 - YMAX% - 10)-(X2%, Y2% - 2*YMAX%+350)
            END IF
         END IF
      END IF
3000
END SUB

SUB FRTEXT (FILE$, FROOT$, FEXT$) STATIC
' It is this routines function to break a file name into root,file, and
' extension. file$ is the input string.
' FILE$ contains filename on return (maximum 8 characters)
' FROOT$ contains path information
' FEXT$ contains extension

' Loop to remove spaces from file name
      LENGTH% = LEN(FILE$)
      F$ = FILE$
      FILE$ = ""
```

```
        FOR I% = 1 TO LENGTH%
           IF MID$(F$, I%, 1) <> " " THEN FILE$ = FILE$ + MID$(F$,I%,1)
        NEXT I%
        FROOT$ = ""
        FEXT$ = ""
        A% = INSTR(FILE$, ".")
        IF A% = 0 GOTO nroot ' no "." means no extension
        IF A% = LENGTH% THEN
           FEXT$ = ""
           FILE$ = LEFT$(FILE$, LENGTH% - 1)
           LENGTH% = LEN(FILE$)
           GOTO nroot
        END IF
        FEXT$ = MID$(FILE$, A%, 4)
        FILE$ = LEFT$(FILE$, A% - 1)

' extension taken care of, now peel off root
nroot:  LENGTH% = LEN(FILE$)
        A% = INSTR(FILE$, "\")
        IF A% = LENGTH% THEN
           FROOT$ = FROOT$ + FILE$
           FILE$ = ""
           GOTO fin
        END IF
        IF A% = 0 GOTO fin
        FROOT$ = FROOT$ + LEFT$(FILE$, A%)
        FILE$ = RIGHT$(FILE$, LENGTH% - A%)
        GOTO nroot
fin:    END SUB

SUB GETFILE (FILE$)
' This routine is called if DENDPLOT is called without any file being
' passed to it.

        CLS
        LOCATE 3, 20
        PRINT "DENDPLOT - Dendrogram plotting utility"
```



```
        LOCATE 10, 15
        INPUT "Enter filename (no extension)"; FILE$
        FILE$ = UCASE$(FILE$)
        IF FILE$ = "" THEN GOTO by
' Put error checking in here to check for a legal file name.  In case
' the user is not paying attention while watching Don Cherry's Coach's
' Corner during Hockey Night in Canada. Eh, Duane?
        A% = INSTR(FILE$, ".")
        IF A% > 0 THEN FILE$ = LEFT$(FILE$, A% - 1)
        FILE$ = FILE$ + ".DEN"
        OPEN FILE$ FOR RANDOM ACCESS READ AS #1
        L& = LOF(1)
        CLOSE #1
        IF L& = 0 THEN
           KILL FILE$
           PRINT FILE$; " not found."
           STOP
        END IF
by:
END SUB

SUB LST (CURSOR%,MARK%(),id() AS RECRD,CLASS() AS RECRD,PLOT%(),
        colr%())
' This routine prints out the dendrogram samples in order. The sample
' at the CURSOR% position is kept in the middle of the display (unless
' the cursor is at the top or bottom).

        NUMSAMS% = UBOUND(MARK%)
        SCREEN 9, , 0, 0
        LOCATE 25, 1
        PRINT STRING$(80, " ");
        LOCATE , 1
        COLOR 14
        PRINT "Press any key...";
        SCREEN 9, , 1, 0
        CLS
```

```
' Determine what range to print out
        TOPSAM% = 1
        IF NUMSAMS% > 23 THEN
           IF CURSOR% > 12 THEN
              TOPSAM% = CURSOR% - 11
           END IF
        END IF
        BOTSAM% = TOPSAM% + 22
        IF BOTSAM% > NUMSAMS% THEN BOTSAM% = NUMSAMS%

' Print out ID and CLASS's
        SCREEN 9, , 1, 1
        LOCATE 1, 1
        COLOR 14
        PRINT " ID  CLASS"
        FOR I% = TOPSAM% TO BOTSAM% - 1
           CURSCOLR% = colr%(-MARK%(I%))
           IF I% = CURSOR% THEN CURSCOLR% = colr%(-MARK%(I%) + 2)
           IF CURSCOLR% = 0 THEN CURSCOLR% = 15
           COLOR CURSCOLR%
           PRINT id(PLOT%(I%)).R; " "; CLASS(PLOT%(I%)).R
        NEXT
        PRINT id(PLOT%(BOTSAM%)).R; " "; CLASS(PLOT%(BOTSAM%)).R;
        LOCATE 25, 1
        COLOR 14
        PRINT "(D - SHOW DENDROGRAM, ESC - EXIT)";
        SCRN% = 1
530     A$ = INKEY$
        A$ = ""
        WHILE A$ = ""
           A$ = INKEY$
        WEND
        A$ = UCASE$(A$)
        IF A$ = CHR$(27) THEN
           SCREEN 9, , 1, 1
           CLS
           SCREEN 9, , 0, 0
```



```
           EXIT SUB
        END IF
        SCRN% = -SCRN% + 1
        SCREEN 9, , SCRN%, SCRN%
        GOTO 530
END SUB

SUB MRK (X%, Y%, Y1%, YL%, C%) STATIC
' This routine draws a box around the cursor area for the current page.
' The box has the dimensions 7 by YL% with the upper right corner being
' positioned at X%,Y1%. The subroutine CMRK is called to draw the
' cursor at position X%,Y% with color C%.

        DIM AREA%(1000)
        DIM colr%(3)
        colr%(0) = 0            ' Background
        colr%(1) = 12           ' Marked
        colr%(2) = 11           ' Cursor
        colr%(3) = 10           ' Marked with cursor

        IF PREV% THEN
           PUT (xold% - 7, yold%)-(xold%, yold%+ YL%), AREA%
           PREV% = 0
        END IF
' Store area into AREA% for redrawing later
        GET (X% - 7, Y1%)-(X%, Y1% + YL%), AREA%
        CALL CMRK(X%, Y%, C%)
        IF C% = colr%(1) THEN C% = colr%(2)
        IF C% > 8 THEN C% = C% - 8
        LINE (X% - 1, Y1%)-(X% - 7, Y1% + YL%), C%, B
        PREV% = -1
        xold% = X%
        yold% = Y1%
END SUB

SUB plotter (DEND%(), DENDD!(), PLOT%(),id() AS RECRD,CLASS() AS RECRD,
        MAXDST!)
```

```
'******   Print-Dendrogram      by   David  Sibbald
'  This is the program which converts the dendrogram file
'  created by DENDO  to a visual display

      SHARED SCRN%
      SHARED FILE$, method$, scaled$, DSMETH$()
      XMAX% = 640 ' *****    These are the values for
      YMAX% = 320 ' *****    the window on the
      XMIN% = 50  ' *****    graphics screen
      YMIN% = 20
      NUMSAMS% = UBOUND(id)
      DIM MARK%(1 TO NUMSAMS%)
      DIM POSY%(1 TO NUMSAMS%)  '** y-coordinate on screen of samples
      DIM POSX%(1 TO NUMSAMS%)  '** x
      DIM colr%(3)
      colr%(0) = 0              ' Background
      colr%(1) = 12             ' Marked
      colr%(2) = 11             ' Cursor
      colr%(3) = 10             ' Marked with cursor
      DFORM$ = "1"
      curexp! = 2
      COLOR , colr%(0)
      distmax! = MAXDST!
      IF NUMSAMS% < 100 THEN WADEMAX! = WADEDST!(NUMSAMS% - 1)
      by1$ = "(Q-quit T-two page display P-plot I-Identify WLN^-
        change display format)          "
      by2$ = "(O-other page S-single page P-plot Q-quit WLN^-change
        display format)"
901   IF PL7440% THEN  ' *** set values for plotter output
         YMIN% = 75
         YMAX% = 975
         DVERT! = (YMAX% - YMIN%) / (NUMSAMS%)
         XMIN% = 50
         XMAX% = 600
         DOPRINT% = 0
         GOTO 920
      END IF
```



```
      by$ = by1$
' Figure out where the plot is allowed to go, and what type of display.
      IF TWOPAGES% <> 0 THEN
         YMAX% = 314
         IF NUMSAMS% <= (YMAX% - 14) / 14 THEN
            YMIN% = 28 * ((YMAX% - 28) / 28 - NUMSAMS% / 2) + 28
            tbtop% = fntxt%(YMIN%)
            XMIN% = 70
            DVERT! = 28
            DOPRINT% = -1
         ELSEIF NUMSAMS% < (YMAX% - 28) / 7 THEN
            YMIN% = 14 * ((YMAX% - 14) / 14 - NUMSAMS% / 2) + 21
            XMIN% = 70
            DVERT! = 14
            DOPRINT% = -1
         ELSE
            YMIN% = 20
            DVERT! = 2 * (YMAX% - YMIN%) / (NUMSAMS%)
            XMIN% = 10
            DOPRINT% = 0
         END IF
      ELSE '***  one page
         IF NUMSAMS% > 23 THEN
            YMIN% = 20
            YMAX% = 325
            DVERT! = (YMAX% - YMIN%) / (NUMSAMS% - 1)
            XMIN% = 10
            DOPRINT% = 0
         ELSEIF NUMSAMS% > 11 THEN
            DVERT! = 14
            YMIN% = 7 * (25 - NUMSAMS%) + 5
            XMIN% = 70
            DOPRINT% = -1
         ELSE
            DVERT! = 28
            YMIN% = 3.5 * (25 - NUMSAMS%) - 5
            XMIN% = 70
```

```
            DOPRINT% = -1
          END IF
        END IF

' ****  Graphics to draw dendrogram on screen
920
        SCREEN SCRN%, , 1, VEEW%
        CLS
        SCREEN SCRN%, , 0, VEEW%
        CLS
        PAGLEN% = DVERT! * 23
' Set the Ycords of the samples based on plot order
        FOR N% = 1 TO NUMSAMS%
           POSY%(PLOT%(N%)) = YMIN% + DVERT! * (N% - 1)
        NEXT
        IF PL7440% THEN
           GOSUB 2000
           IF PL7440% = 0 THEN
              A$ = "S"
              VEEW% = 0
              GOTO 1021
           END IF
        END IF
' Set xcords of all samples to the left axis
        FOR N% = 1 TO NUMSAMS%
           POSX%(PLOT%(N%)) = XMIN%
        NEXT
        COLOR 15
        IF DOPRINT% THEN
           FOR N% = 1 TO NUMSAMS%          ' Print out samplenames
              IF POSY%(PLOT%(N%)) > YMAX% THEN
               VTAB% = fntxt%(POSY%(PLOT%(N%)) + 350 - 2 * YMAX%)
               SCREEN SCRN%, , 1, VEEW%
               colr% = colr%(-MARK%(N%))
               CALL CMRK(XMIN%, POSY%(PLOT%(N%)) + 350-2*YMAX%,colr%)
               LOCATE VTAB%, 1
              ELSE
```



```
               SCREEN SCRN%, , 0, VEEW%
               VTAB% = fntxt%(POSY%(PLOT%(N%)))
               colr% = colr%(-MARK%(N%))
               CALL CMRK(XMIN%, POSY%(PLOT%(N%)), colr%)
               LOCATE VTAB%, 1
             END IF
              out1$ = SPACE$(4)
              RSET out1$ = id(PLOT%(N%)).R
              out2$ = SPACE$(4)
              RSET out2$ = RTRIM$(CLASS(PLOT%(N%)).R)
              out$ = out1$ + out2$
              PRINT out$;
           NEXT
        END IF
1000
        DSCALE! = ABS(XMIN% - XMAX%) * .98
        FOR p% = 1 TO NUMSAMS% - 1
           pos1% = DEND%(p%, 1)
           pos2% = DEND%(p%, 2)
           OLDy1% = POSY%(pos1%)
           OLDy2% = POSY%(pos2%)
           NEWY% = (OLDy1% + OLDy2%) / 2
           IF WWMETH% THEN
              NEWX%=DSCALE!*DFCALC!(WADEDST!(p%),MAXDST!,DFORM$)+XMIN%
           ELSE
              NEWX%=DSCALE! * DFCALC!(DENDD!(p%), MAXDST!,DFORM$)+XMIN%
           END IF
           CALL DLINE(POSX%(pos1%), OLDy1%, NEWX%, OLDy1%, YMAX%,
             TWOPAGES%, PL7440%)
           CALL DLINE(POSX%(pos2%), OLDy2%, NEWX%, OLDy2%, YMAX%,
             TWOPAGES%, PL7440%)
           CALL DLINE(NEWX%, OLDy1%, NEWX%, OLDy2%, YMAX%, TWOPAGES%,
                PL7440%)
           CALL PMOVE(POSX%(), POSY%(), NEWX%, NEWY%, OLDy1%, OLDy2%)
        NEXT
        CALL DLINE(NEWX%, NEWY%, XMAX%, NEWY%, YMAX%, TWOPAGES%,
        PL7440%)
```

```
        SCREEN 9, , -TWOPAGES%, VEEW%
        LOCATE 1, 1
        PRINT FILE$
        LOCATE 1, 6 + LEN(FILE$)
        PRINT "Display method = ";
        PRINT DSMETH$(INT(VAL(DFORM$)));
' Patch in here if you wish the number on the end of DSFORM$
' printed
        SELECT CASE INT(VAL(DFORM$))
        CASE 4
           PRINT SPC(3);
           PRINT "POWER ="; RIGHT$(DFORM$, LEN(DFORM$) - INSTR(DFORM$,
        "."));
           PRINT " Use + and - keys.";
        CASE ELSE
        END SELECT
        IF WWMETH% THEN
           LOCATE 1, 65
           PRINT "Relative axis"
        END IF

' Draw similarity axis and print out command lines
           COLOR 11
           IF NOT TWOPAGES% THEN
              LOCATE 25, 1
              PRINT by1$;
              COLOR 13
              LINE (XMIN%, YMIN% - 5)-(XMAX%, YMIN% - 5)
           ELSE
              SCREEN SCRN%, , 0, VEEW%
              LOCATE 1, 1
              PRINT by2$;
              COLOR 13
              LINE (0, YMAX% + 11)-(XMAX%, YMAX% + 11)
              SCREEN SCRN%, , 1, VEEW%
              LOCATE 25, 1
              COLOR 15
```



```
              PRINT by2$;
              COLOR 13
              LINE (0, 350 - YMAX% - 11)-(XMAX%, 350 - YMAX% - 11)
              SCREEN SCRN%, , 0, VEEW%
           END IF
' Draw hashmarks on similarity axis
        IF NOT PL7440% THEN
           IF TWOPAGES% THEN
              SCREEN SCRN%, , 1, VEEW%
              COLOR 12
              Y% = 350 - YMAX% - 12
              GOSUB scale
              SCREEN SCRN%, , 0, VEEW%
              COLOR 12
              Y% = YMAX% + 9
              GOSUB scale
           ELSE
              SCREEN SCRN%, , 0, VEEW%
              COLOR 12
              Y% = YMIN% - 8
              GOSUB scale
           END IF
        ELSE
           Y% = YMIN% - 10
           GOSUB scale
           PRINT #1, "PA0,0;"
           PRINT #1, "SP;"
           CLOSE #1
           TWOPAGES% = 0
           VEEW% = 0
           PL7440% = 0
           GOTO 901
        END IF
        GOTO 1020

' This GOSUB prints out the hashmarks on the axes. Specifically, it
' draws a line 5 pixels long (Y% to Y%+4) at the position calculated
```

```
' using the DFCALC function
scale:
        FOR I! = 0 TO MAXDST! STEP MAXDST! / 10
           D! = DFCALC!(I!, MAXDST!, DFORM$) * DSCALE! + XMIN%
           LINE (D!, Y%)-(D!, Y% + 4)
        NEXT
        RETURN

1020    INIT! = TIMER
        A$ = INKEY$
        A$ = ""
' Pause and wait for key. If the Bozo doesn't play with the display, it
' goes away.  The time constant should be changed for longer delays, or
' removed for people with three cars who are never on time (but no
' mentioning Darcy's name.)
        WHILE A$ = "" AND TIMER < INIT! + 10000
           A$ = INKEY$
        WEND
        IF A$ = "" THEN A$ = "Q"
        A$ = UCASE$(A$)
1021    SELECT CASE A$
        CASE "Q"
           SCREEN SCRN%, , 0, 0
           CLS
           SCREEN SCRN%, , 0, 1
           CLS
           GOTO 1030
        CASE "T"
           IF TWOPAGES% = 0 THEN
              TWOPAGES% = -1
              GOTO 901
           END IF
        CASE "W"
           IF NUMSAMS% < 100 THEN
              IF WWMETH% THEN
                MAXDST! = distmax!
                WWMETH% = 0
```



```
              ELSE
                MAXDST! = WADEMAX!
                WWMETH% = 1
              END IF
           END IF
           GOTO 920
        CASE "N"
           IF DFORM$ = "1" THEN
              SOUND 50, 1
              GOTO 1020
           END IF
           DFORM$ = "1"
           GOTO 920
        CASE "L"
           IF DFORM$ = "3" THEN
              SOUND 50, 1
              GOTO 1020
           END IF
           DFORM$ = "3"
           GOTO 920
        CASE "^"
           IF LEFT$(DFORM$, 1) = "4" THEN
              SOUND 51, 1
              GOTO 1020
           END IF
           DFORM$ = "4." + STR$(curexp!)
           GOTO 920
        CASE "+", "="
           IF INT(VAL(LEFT$(DFORM$, 1))) = 4 THEN
              curexp! = curexp! + .1
              IF curexp! > 2 THEN curexp! = 2
              DFORM$ = "4." + STR$(curexp!)
              GOTO 920
           ELSE
              SOUND 70, 1
              GOTO 1020
           END IF
```

```
      CASE "-", "_"
         IF INT(VAL(LEFT$(DFORM$, 1))) = 4 THEN
            curexp! = curexp! - .1
            IF curexp! = 0 THEN curexp! = .1
            DFORM$ = "4." + STR$(curexp!)
            GOTO 920
         ELSE
            SOUND 70, 1
            GOTO 1020
         END IF
      CASE "X"    ' change power
' This could be implemented inputing a power from the keyboard
' but I was busy.
      CASE "S"
         IF TWOPAGES% THEN
            TWOPAGES% = 0
            VEEW% = 0
            GOTO 901
         END IF
      CASE "P"
         CLS
         IF TWOPAGES% THEN
            VEEW% = (VEEW% + 1) MOD 2
            SCREEN SCRN%, , VEEW%, VEEW%
            CLS
         END IF
         PL7440% = -1
         TWOPAGES% = -1
         GOTO 901
      CASE "O", " "
         IF TWOPAGES% THEN
            VEEW% = (VEEW% + 1) MOD 2
            SCREEN SCRN%, , VEEW%, VEEW%
         END IF
      CASE "I"
         IF NOT TWOPAGES% THEN
            CURSOR% = 1
```



```
            GOSUB IDCL
            LOCATE 25, 1
            PRINT by$;
         ELSE
            LOCATE 25, 1
            COLOR 14
            PRINT STRING$(80, " ");
            LOCATE 25, 1
            PRINT "You must be in single page mode to identify
points.";
            A$ = INKEY$
            A$ = ""
            INIT! = TIMER
            WHILE A$ = "" AND INIT! + 10 > TIMER
              A$ = INKEY$
            WEND
            GOTO 920
         END IF
      CASE ELSE
      'burp
      END SELECT
      GOTO 1020

' This section deals with identifying and changing the class of signals
' For a one page display, we will use page two to print out the
' dendrogram ID's in PLOT order. We will allow the selection of
' specific elements on the dendrogram itself and highlight these on the
' print out of ID's.  The ID page will contain the ID and CLASS of each
' sample.

IDCL:
      CURSOR% = 1
IDCL2:
      LOCATE 25, 1
      PRINT SPACE$(79);
      LOCATE 1, 65
      PRINT STRING$(15, " ");
```

```
        LOCATE 25, 1
        PRINT "Use cursor keys to move, M-to mark sample, L-to list
              samples, N-next marked.";
        IF FEXT$ = ".DS1" THEN PRINT ", C to change classifier";
        '******
        IF NUMSAMS% > 23 THEN
           PGLNTH% = DVERT! * 23 + 2
        ELSE
           PGLNTH% = NUMSAMS% * 14 + 2
        END IF
        ext% = 0
1110    WHILE ext% = 0
           LOCATE 1, 70
           CURSCOLR% = colr%(-MARK%(CURSOR%) + 2)
           COLOR CURSCOLR%
           PRINT id(PLOT%(CURSOR%)).R; " "; CLASS(PLOT%(CURSOR%)).R;
           TB% = (CURSOR% - 1) * DVERT! + YMIN%
           TOP% = 1
           IF NUMSAMS% > 23 AND NUMSAMS% - CURSOR% < 12 THEN
              TOP% = (NUMSAMS% - 23 - .5) * DVERT! + YMIN%
           ELSE
              IF CURSOR% > 11 AND NUMSAMS% > 23 THEN
               TOP% = (CURSOR% - 11 - .5) * DVERT! + YMIN%
              ELSE
               TOP% = (-.5 * DVERT!) + YMIN%
              END IF
           END IF
           IF TOP% < 1 THEN TOP% = 1
           CALL MRK(XMIN%, TB%, TOP%, PGLNTH%, CURSCOLR%)
           A$ = INKEY$
           A$ = ""
           INIT! = TIMER
           WHILE A$ = "" AND TIMER < INIT! + 1000
              A$ = INKEY$
           WEND
           IF A$ = "" THEN A$ = CHR$(27)
           IF ASC(A$) = 0 THEN A$ = RIGHT$(A$, 1)
```



```
           SELECT CASE UCASE$(A$)
           CASE "M"                    ' the M key - Duh, thanks Kelly.
              MARK%(CURSOR%) = NOT MARK%(CURSOR%)
           CASE "N"                              ' the N key
              I% = CURSOR% + 1
              M% = 0
              DO WHILE NOT M%
               M% = MARK%(I%)
               I% = I% + 1
               IF I% > NUMSAMS% AND NOT M% THEN
                 M% = -1
                 I% = 0
               END IF
              LOOP
              IF I% = 0 THEN
               SOUND 105, 2
              ELSE
               colr% = colr%(-MARK%(CURSOR%))
               CURSOR% = I% - 1
               CALL MRK(XMIN%, TB%, TOP%, PGLNTH%, colr%)
              END IF
           CASE "L"                              ' the L key
              CALL LST(CURSOR%, MARK%(), id(), CLASS(),PLOT%(),colr%())
              GOTO IDCL2
           CASE CHR$(80)                    ' the DOWN key was pressed
              colr% = colr%(-MARK%(CURSOR%))
              IF CURSOR% < NUMSAMS% THEN CURSOR% = CURSOR% + 1
              CALL CMRK(XMIN%, TB%, colr%)
           CASE CHR$(72)                         ' the UP key
              colr% = colr%(-MARK%(CURSOR%))
              IF CURSOR% > 1 THEN CURSOR% = CURSOR% - 1
              CALL MRK(XMIN%, TB%, TOP%, PGLNTH%, colr%)
           CASE CHR$(56),CHR$(73),CHR$(57) ' the shift UP, PgUp,SH PgUp
              colr% = colr%(-MARK%(CURSOR%))
              IF CURSOR% > 11 THEN
               CURSOR% = CURSOR% - 10
              ELSE
```

```
            CURSOR% = 1
          END IF
          CALL MRK(XMIN%, TB%, TOP%, PGLNTH%, colr%)
       CASE CHR$(27)                          ' the ESCape key
          ext% = -1
       CASE CHR$(50), CHR$(81), CHR$(51) ' shft DOWN ,PgDn,shftPgDn
          colr% = colr%(-MARK%(CURSOR%))
          IF CURSOR% < NUMSAMS% - 11 THEN
            CURSOR% = CURSOR% + 10
          ELSE
            CURSOR% = NUMSAMS%
          END IF
          CALL MRK(XMIN%, TB%, TOP%, PGLNTH%, colr%)
       CASE CHR$(71), CHR$(55)                ' HOME, Shift HOME
          colr% = colr%(-MARK%(CURSOR%))
          CALL MRK(XMIN%, TB%, TOP%, PGLNTH%, colr%)
          CURSOR% = 1
       CASE CHR$(79), CHR$(49)                ' END, Shift END
          colr% = colr%(-MARK%(CURSOR%))
          CALL MRK(XMIN%, TB%, TOP%, PGLNTH%, colr%)
          CURSOR% = NUMSAMS%
       CASE ELSE
       END SELECT
    WEND
    LOCATE 1, 70
    PRINT SPACE$(9)
    CALL MRK(XMIN%, TB%, TOP%, PGLNTH%, colr%(-MARK%(CURSOR%)))
    COLOR 15
    RETURN

' *********  Opens the port for the plotter
' Selects pen 3 for labelling the dendrogram
' Selects pen 2 for labelling each sample
' Selects pen 1 before returning control back to PRIN-DEND
2000
    CLS
    PRINT "Make sure that HP ColorPro is connected"
```



```
PRINT "Load paper"
PRINT "Pen 1 - black (0.3mm)  Pen 2 - blue (0.3mm)   Pen 3 -
       purple"
PRINT "Press Return";
INPUT "", A$
A$ = LEFT$(A$, 1)
IF A$ = "q" OR A$ = "Q" THEN
   PL7440% = 0
   RETURN
END IF
OPEN "COM1:9600,S,7,1,RS,CS65535,DS,CD" FOR RANDOM AS #1
'OPEN "O", #1, "HPPLOT.DPL"
PRINT #1, "SP3;"
PRINT #1, "SC0,1000,0,700;"
PRINT #1, "PA 10,670;DR1,0;SI.4,.4;"
CLS
PRINT " Enter  Title  -  maximum 40 characters|"
LINE INPUT TI$
TI$ = LEFT$(TI$, 40)
CLS
PRINT "Now plotting ... "; TI$
PRINT #1, "LB" + TI$ + CHR$(3)
PRINT #1, "PA0,695;SI.2,.2;"
PRINT #1, "SP2;"
PRINT #1, "LB" + DATE$ + CHR$(3)
PRINT #1, "DR0,1;"
PRINT #1, "SI.17,.17;"
PRINT #1, "PA1000,70;"
PRINT #1, "LB" + FILE$ + CHR$(3)
PRINT #1, "PA1000,200;"
PRINT #1, "LB" + scaled$ + CHR$(3)
PRINT #1, "PA1000,350;"
PRINT #1, "LB" + method$ + CHR$(3)
IF WWMETH% THEN
   PRINT #1, "PA1000,550;"
   PRINT #1, "LBww" + CHR$(3)
END IF
```

```
size! = 7 / NUMSAMS%
IF size! > .3 THEN size! = .3
PRINT #1, "DR 1,0;"
IF size! < .045 THEN size! = .045
PRINT #1, "SI "; size!; ",.2;"
CLASLEN% = 0
FOR N% = 1 TO NUMSAMS%
   L% = LEN(RTRIM$(LTRIM$(CLASS(N%).R)))
   IF L% > CLASLEN% THEN CLASLEN% = L%
NEXT
FOR N% = 1 TO NUMSAMS%
   N1$ = SPACE$(4)
   RSET N1$ = (RTRIM$(LTRIM$(id(PLOT%(N%)).R)))
   N2$ = RTRIM$(LTRIM$(CLASS(PLOT%(N%)).R))
   LETPOS% = POSY%(PLOT%(N%)) + size! * 35
   PRINT #1, "PA "; LETPOS%; ",0;"
   IF NUMSAMS% < 30 THEN
      BS% = LEN(N2$)
      IF BS% > 0 THEN PRINT #1, "CP" + STR$(-BS%) + ",0;"
      PRINT #1, "LB" + N2$ + CHR$(3) + ";"
   LETPOS% = POSY%(PLOT%(N%)) + size! * 5
      PRINT #1, "PA "; LETPOS%; ",0;"
      PRINT #1, "CP 0,1;"
      N1$ = RTRIM$(LTRIM$(N1$))
      BS% = LEN(N1$)
      IF BS% > 0 THEN PRINT #1, "CP" + STR$(-BS% / 2) + ",0;"
      PRINT #1, "LB"; N1$ + CHR$(3) + "; "; ""
      N$ = "12" ' 2 characters
   ELSE
      N2$ = LEFT$("     " + N2$, CLASLEN%)
      N$ = N2$ + N1$
      FOR ln% = 4 + CLASLEN% TO 1 STEP -1
       PRINT #1, "CP-1,1;"
       PRINT #1, "LB" + MID$(N$, ln%, 1) + CHR$(3) + ";"
      NEXT
   END IF
NEXT
```



```
      XMIN% = 19 * LEN(N$)
      PRINT #1, "SP1;"
      RETURN
1030
END SUB

SUB PMOVE (px%(), PY%(), nx%, nY%, oy1%, oy2%)
' PMOVE looks through PY%() [posy%() in main module] for any% "members"
' that have y-coordinates OY1% or OY2% [oldy1,2%]. Any% matches have
' their POSitions moved to NX%,NY% [newx,newy]
' |ns% = numsams%|

      NS% = UBOUND(PY%)
      FOR I% = 1 TO NS%
         IF PY%(I%) = oy1% OR PY%(I%) = oy2% THEN
            PY%(I%) = nY%
            px%(I%) = nx%
         END IF
      NEXT
END SUB

FUNCTION WADEDST! (NODE%)
' This routine recursively WADEs through the dendrogram tree structure
' to find the maximum height needed to display the dendrogram stored in
' dend% as in the newly proposed format.
' The routine takes the current distance stored in dendd!(node%) and
' adds it to the maximum of the values returned from the recursive
' calls of WADEDST(leftnode%) and WADEDST(rightnode%).

      SHARED DEND%(), DENDD!()
      MAX! = DENDD!(NODE%)
      rnode% = 0
      rmax! = 0
      lnode% = 0
      lmax! = 0
      cnode% = NODE%
      DO
```

```
        cnode% = cnode% - 1
      LOOP UNTIL cnode% = 0 OR MATCH(DEND%(NODE%, 1), cnode%)
      IF cnode% THEN
        lnode% = cnode%
        lmax! = WADEDST!(lnode%)
      END IF
      cnode% = NODE%
      DO
        cnode% = cnode% - 1
      LOOP UNTIL cnode% = 0 OR MATCH(DEND%(NODE%, 2), cnode%)
      IF cnode% THEN
        rnode% = cnode%
        rmax! = WADEDST!(rnode%)
      END IF
      IF rmax! > lmax! THEN
        MAX! = MAX! + rmax!
      ELSE
        MAX! = MAX! + lmax!
      END IF
      WADEDST! = MAX!
END FUNCTION

SUB WRITECLASS (clss AS RECRD, NUMSAMS%, NUMVARS%, SAMP%, F$, op%)
' This subroutine writes the class of sample # SAMP% to the DS1 file
' F$.  (See opening credits for format of DS1 file)
' The formula for finding the record number of the CLASS record of
' sample SAMP% in a dataset containing NUMVARS% descriptors is:
'
' REC! = (SAMP%+1)*NUMVARS%+4(SAMP%-1)+4
'
' CLSS      = 4 byte string to be written
' NUMSAMS%  = The maximum number of signals in the file.
' NUMVARS%  = The number of descriptors per signal.
' SAMP%     = the sample to be written
' F$        = the name of the file (.DS1 extention)
' OP%       = BOOLEAN variable. True if file F$ is already open.
```



```
      REC1$ = SPACE$(4)
      IF NUMSAMS% < SAMP% THEN GOTO nope
      rec! = (SAMP% + 1) * NUMVARS% + 4 * (SAMP% - 1) + 4
      IF NOT op% THEN
        OPEN F$ FOR RANDOM ACCESS WRITE AS #1 LEN = 4
        FIELD #1, 4 AS REC1$
      END IF
      LSET REC1$ = clss.R
      PUT #1, rec!
      IF NOT op% THEN CLOSE #1
nope:
END SUB
```