DETECTION OF RAMAN SPECTRAL PEAKS BY THE USE OF MATCHED FILTERING AND SEQUENTIAL (WALD) TESTING

by

DAMON M. MICK

B.S., Kansas State University, 1984

A THESIS

submitted in partial fulfillment of the

requirements for the degree

MASTER OF SCIENCE

Department of Electrical and Computer Engineering

KANSAS STATE UNIVERSITY
Manhattan, Kansas

1987

Approved by:

Stephen A. Dyer

Major Professor

CONTENTS

LIST OF FIGURES

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENTS

I would like to thank Dr. Stephen Dyer for his help throughout this project, and my parents, Ardwin and Barbara Mick, for their support during my college years.

I. INTRODUCTION

1.1 General Remarks

The purpose of the research presented here was to develop an automatic detection scheme for signals resembling those encountered in Raman spectroscopy. The detection of Raman spectral peaks presents a special problem because of the low signal-to-noise ratios (SNRs) that are often dealt with and because the shapes of the peaks, which are Lorentzian or Gaussian in form, make them difficult to detect.

Various methods have been used for peak enhancement. Matched filtering, ensemble averaging, and boxcar averaging are different methods for improving signal-to-noise ratios of signals. A matched filter is a filter whose impulse response is a time-reversed version of the profile of the expected signal. A matched filter can be shown to maximize the signal-to-noise ratio when the noise is white. However, the effectiveness of the filter depends on how closely the signal matches the filter response. Ensemble averaging consists of making a number of runs and averaging the resulting sequences. When N runs are used the SNR is improved by a factor of $\sqrt{N}$. The problem with ensemble averaging is that a longer time is needed to obtain results. In boxcar averaging the data sequence is divided into groups of n adjacent points and the points in each group are averaged The SNR ratio is improved by a factor of $\sqrt{n}$. However, an improvement in the SNR requires decreased resolution.

The adaptive peak detector [1] uses an adaptive filter to detect changes in the statistics of the input which would indicate the presence of a signal. It has been demonstrated to be capable of detecting peaks at SNRs of from 1.0 down to 0.3; the main problem is that there are several parameters that must be adjusted.

1.2 Summary of Principal Results

The general strategy was to first enhance the peaks by use of a filter matched to a typical pulse shape, and to then follow this with an automatic detector. The detector was developed by an application of statistical decision theory. The decision criteria chosen for study were the Neyman-Pearson test and the Wald (or sequential) test, mainly because they do not require knowledge of the a priori probabilities of the signal. The Wald test, which has the feature of using a variable number of samples to make decisions, was chosen for implementation.

An addition to the scheme that was found to be necessary was to fit and remove a trend from the output of the matched filter, so that the threshold comparison procedure in the detector could work properly in identifying individual peaks that are closely grouped.

The detection scheme has some limitations in that an estimate of the noise variance must be available, and the performance of the filter depends on how well the peaks match its shape. Some parameters involved in the sequential detector, particularly the theoretical probabilities of false detection and missed detection, have an effect on the detector output. However, it may be possible for the detector to work without need for adjustment of these parameters on the part of the observer.

1.3 Structure of the Thesis

The theoretical background for the detector is given in Section II. This begins with a general discussion of hypothesis testing, followed by a detailed presentation of the Neyman-Pearson and Wald (sequential) tests, including the development of these tests for the case of an unknown signal in Gaussian noise, the situation of interest here.

Section III presents some details of how the detection procedure and the testing of it are performed in software. Included are a discussion of the generation of the data sequences, the matched filter, and the detector.

Section IV discusses first how tests on several data sequences were used to determine how the performance of the detector depends on the selection of the detector parameters. Then a set of tests is presented and some conclusions are drawn about the detector's performance in terms of its ability to detect pulses at low SNRs and its ability to avoid false detections.

II. MATHEMATICAL BACKGROUND

2.1 Statistical Decision Theory

Statistical decision theory, or hypothesis testing, was first developed by Neyman and Pearson and later expanded by Wald and others [2]. The purpose of hypothesis testing is to test two or more alternative hypotheses and to determine which is the best choice, according to some criterion of optimality. Some of the more common examples of decision criteria include the Bayes, the Ideal Observer, and the Neyman-Pearson criteria. What follows is a discussion of decision theory in terms of its application to the problem of signal detection.

In a detection problem the first hypothesis (denoted H_0) is that noise alone is present, and the second hypothesis (H_1) is that signal plus noise is present. The situation discussed here is actually an example of composite hypothesis testing [3], where the second hypothesis is composite; i.e., one or more parameters of the signal (such as the amplitude) may have a range of possible values rather than a single value. There may be more than two hypotheses if a decision is to be made from among various classes of signals; a binary detection problem is one in which there are two hypotheses.

The set of signals that are possible is called the signal space (Fig. 2.1), while the observations that are made are considered to be elements of the observation space. An observation will consist of one or more samples of the received signal, and when there is more than one sample the observation may be represented as a vector $z = [z_1 \ z_2 \ \ldots \ z_n]^T$ of the individual samples. For a binary test the observation space is divided into two decision regions Z_0 and Z_1, corresponding to the two hypotheses. Some kind of decision criterion must be used to decide which hypothesis should be chosen for any given value of z, that is, to decide the boundary between the decision regions.

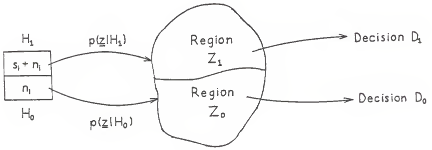

Figure 2.1. Signal space and observation space.

In some cases application of one of these criteria leads to a likelihood ratio test. The likelihood ratio $\Lambda(z)$ is a ratio of conditional probabilities, i.e.

$$\Lambda(z) = p(z|H_1)/p(z|H_0) \tag{2-1}$$

A likelihood ratio test consists of comparing this ratio to some threshold value so as to decide from among the hypotheses. A larger liklihood ratio means that H_1 is more likely to be true.

In a binary detection problem, there are two possible kinds of errors: deciding 'signal present' when there is no signal (Type I error) and deciding 'signal absent' when a signal is present (Type II error).

Some tests require knowledge of the a priori signal statistics, and may also require that costs be assigned to each type of error and to each type of correct decision (signal present or signal absent). For example, the Bayes criterion requires an assignment of all costs, and the Minimum-Probability-of-Error criterion is a special case of the Bayes criterion in which the cost of either type of error is 1 and the cost of a correct decision is 0 [4]. Two tests which are useful when this information is not available are the Neyman-Pearson test and the Wald test.

2.2 Neyman-Pearson Test

The Neyman-Pearson test does not require knowledge of the a priori probabilities because it is only concerned with the conditional probabilities of error $p(D_1|H_0)$ and $p(D_0|H_1)$, and not the total probabilities. The objective of the test is to keep both the probability P_f of false detection and the probability P_m of a miss as low as possible. Since decreasing one of these probabilities is likely to mean increasing the other, the strategy that is used is to select a value for P_f and to then minimize P_m. This is the same as maximizing the probability P_d of detection where $P_d = 1 - P_m$. The miss probability is minimized subject to the constraint $P_f = a'$, where a' is the chosen value for P_f, using the calculus of extrema [5]. From this is developed a likelihood ratio test where the decision strategy is:

$$p(z|H_1)/p(z|H_0) < \lambda \quad \text{decide } H_0 \tag{2-2a}$$

$$p(z|H_1)/p(z|H_0) > \lambda \quad \text{decide } H_1 \tag{2-2b}$$

where λ is the threshold of the test.

The threshold is determined by the choice of a' and is calculated from

$$a' = \int_{Z_1} p(z|H_0)dz \tag{2-3}$$

The noise process in Raman spectroscopy can be considered to be Gaussian with a mean of zero. If a known variance σ^2 is assumed, then for a single sample z the conditional probability distribution functions for the two hypotheses will be:

$$p(z|H_0) = (1/\sqrt{2\pi}\sigma)\exp\{-z^2/2\sigma^2\} \tag{2-4a}$$

$$p(z|H_1) = (1/\sqrt{2\pi}\sigma)\exp\{-(z-m)^2/2\sigma^2\} \tag{2-4b}$$

where m is the signal amplitude.

The likelihood ratio then becomes:

$$\Lambda(z) = \exp\{(-(z-m)^2 + z^2)/2\sigma^2\}$$

$$= \exp\{(2mz-m^2)/2\sigma^2\} \tag{2-5}$$

It is common to work with the natural logarithm of the likelihood ratio, $L(z)$, so that the test becomes:

$$L(z) = (2mz-m^2)/2\sigma^2 < \ln(\lambda) \qquad \text{decide } H_0 \qquad (2\text{-}6a)$$

$$L(z) \geq \ln(\lambda) \qquad \text{decide } H_1 \qquad (2\text{-}6b)$$

or

$$z < [2\sigma^2 \ln(\lambda) + m^2]/2m = z_t \qquad \text{decide } H_0 \qquad (2\text{-}7a)$$

$$z \geq z_t \qquad \text{decide } H_1 \qquad (2\text{-}7b)$$

The threshold z_t is determined from the probability of false detection as shown in Eqn. (2-3):

$$a' = \int_{z_t}^{\infty} (1/\sqrt{2\pi}\sigma) \exp\{-z^2/2\sigma^2\}\, dz = \operatorname{erfc}(z_t/\sigma) \qquad (2\text{-}8)$$

Thus, for a chosen value of a' the threshold can be calculated from the inverse erfc function. Note that since $z_t = \sigma \cdot \operatorname{erfc}^{-1}(a')$, the noise variance must be known.

2.3 Neyman-Pearson Test With Multiple-Sample Observations

Usually, a decision is based on a set of samples rather than a single sample. In this case we work with the joint conditional pdf's, e.g. $p(z_1, z_2, \ldots, z_k | H_0)$ for k samples. The set of samples can be treated as a vector $z = [z_1\ z_2 \ldots z_k]^T$. The observation space is then a k-dimensional vector space which is divided into decision regions Z_0 and Z_1 by a decision surface.

Again, for the case of an unknown signal in Gaussian noise, the joint conditional probabilities would be

$$p(z|H_0) = 1/(2\pi\sigma^2)^{k/2} \exp\{-z^T z/2\sigma^2\} \qquad (2\text{-}9a)$$

$$p(z|H_1) = 1/(2\pi\sigma^2)^{k/2} \exp\{-(z-m)^T(z-m)/2\sigma^2\} \qquad (2\text{-}9b)$$

where $m = [m_1\ m_2\ \dots\ m_k]^T$, and the likelihood ratio is

$$\Lambda(z) = \exp\{(m^T z - .5m^T m)/\sigma^2\} \qquad (2\text{-}10)$$

The decision rule is then

$$m^T z \geq \sigma^2 \ln(\lambda) + 0.5m^T m \qquad \text{decide } H_1 \qquad (2\text{-}11a)$$

$$m^T z \leq \sigma^2 \ln(\lambda) + 0.5m^T m \qquad \text{decide } H_0 \qquad (2\text{-}11b)$$

It can be seen that some knowledge of m, i.e. of the shape of the signal, may be required, or at least some estimate must be made. This will also be seen when the Wald test is developed later.

If a constant but unknown signal amplitude is assumed, then a test is developed comparable to that of Eqns. (2-7) and (2-8), involving the sum of the sample values. This result will be referred to in Section III.

It may be difficult to decide what sample size will be best when the signals are not precisely known. In such cases it might be more useful to employ a modified form of the Neyman-Pearson test known as sequential testing, in which the sample size varies according to the number of samples required to reach a decision.

2.4 Sequential Detection (Wald's Test)

In the sequential detection procedure the number of samples per decision is not fixed, but varies according to number needed to reach a decision with the desired probabilities of Type I and Type II errors. The test involves comparing the joint likelihood ratio of all the samples taken with two thresholds. If the likelihood ratio is above the higher threshold, a signal is declared present; if it is below the lower threshold, it is decided that no signal is present; and if the likelihood ratio is between the two, no decision is made and another sample is taken.

If the samples are statistically independent, then the joint conditional density function $p(z_j|H_k)$ can be calculated as $\prod_{i=1}^{j} p(z_i|H_k)$, and therefore $\Lambda(z_j) = \prod_{i=1}^{j} \Lambda(z_i)$.

The desired probabilities for both Type I and Type II error must be specified. If we assign $P_f = a'$ and $P_m = b'$ then the thresholds for the likelihood-ratio test that were derived by Wald [6] are

$$n_1 \leq (1-b)'/a' \tag{2-12a}$$

and

$$n_0 \geq b'/(1-a') \tag{2-12b}$$

where n_1 is the upper threshold and n_0 is the lower threshold.

The inequality relationships arise from the requirements that $\Lambda(z_j) \geq n_1$ and $\Lambda(z_j) \leq n_0$ for acceptance or dismissal, respectively, of a signal [7]. If it can be assumed that the log-likelihood ratio $L(z_j)$ has small increments $L(z_j)$, then the thresholds can be defined by the equalities

$$\ln(n_1) \cong \ln[(1-b')/a'] \tag{2-13a}$$

and

$$\ln(n_0) \cong \ln[b'/(1-a')] \tag{2-13b}$$

The average number of samples required to reach a decision under each hypothesis has been calculated [8], and it has been shown that Wald's test will on the average require fewer samples to detect a signal than a fixed-sample test with the same probabilities of error.

The form of the Wald test for the case of an unknown signal in Gaussian noise will now be developed. The noise pdf is again described by a mean of zero and a standard deviation of σ. The density functions for a single sample z_i are as in Eqns. (2-4), with a mean of m_i, and the joint conditional density functions for the jth sample are

$$p(z_j|H_0) = [1/(2\pi\sigma^2)^{j/2}]\exp\{-(1/2\sigma^2 \sum_{i=1}^{j}(z_i)^2)\} \tag{2-14a}$$

and

$$p(z_j|H_1) = [1/(2\pi\sigma^2)^{j/2}]\exp\{-(1/2\sigma^2) \sum_{i=1}^{j}(z_i-m_i)^2)\} \tag{2-14b}$$

and the joint likelihood ratio is

$$\begin{aligned}\Lambda(z_j) &= \exp\{-(1/2\sigma^2) \sum_{i=1}^{j}[(z_i-m_i)^2 - (z_i)^2]\} \\ &= \exp\{-(1/2\sigma^2) \sum_{i=1}^{j}[2m_i z_i - (m_i)^2]\}\end{aligned} \tag{2-15}$$

the likelihood ratio test could then be expressed as

$$\sum_{i=1}^{j}m_i z_i \le \sigma^2\ln(n_0) + 0.5 \sum_{i=1}^{j}(m_i)^2 \qquad \text{decide } H_0 \tag{2-16a}$$

$$\sum_{i=1}^{j}m_i z_i \ge \sigma^2\ln(n_1) + 0.5 \sum_{i=1}^{j}(m_i)^2 \qquad \text{decide } H_1 \tag{2-16b}$$

If it is assumed that all the m_i are identical, i.e., the signal amplitude is constant, then this can be simplified to

$$\sum_{i=1}^{j} z_i \leq (\sigma^2/m)\ln(n_0) + 0.5jm \qquad \text{decide } H_0 \tag{2-17a}$$

$$\sum_{i=1}^{j} z_i \geq (\sigma^2/m)\ln(n_1) + 0.5jm \qquad \text{decide } H_1 \tag{2-17b}$$

with the sum of the samples being compared to the thresholds on the right-hand sides of Eqns. (2-17).

III. SIMULATION OF THE DETECTION PROCEDURE

A program was written to simulate the matched filter and the detector and to generate signals for testing. The complete program listings are in the Appendix.

Signal Generation. Two forms for the signal pulses that are of interest in Raman spectroscopy are the Lorentzian and the Gaussian forms, since the Lorentzian shape is typical of the spectra of solids, and the Gaussian shape, for liquids. The Gaussian form is

$$s(\nu) = A \exp\{-(\nu-\nu_0)^2/4\ln(2)\alpha^2\} \qquad (3\text{-}1)$$

where ν is the wavenumber, ν_0 is the wavenumber of the center of the peak, and α is the half-width at half-height (hwhh) of the pulse. The Lorentzian form is:

$$s(\nu) = A/[1+(\nu-\nu_0)^2/\alpha^2] \qquad (3\text{-}2)$$

For the Gaussian shape the relationship of the hwhh to standard deviation σ is $\sigma = \alpha/\sqrt{2\ln 2}$.

The program allows the generation of two types of signal sequences. In one type, the heights, widths, and spacings of the pulses are all randomly chosen, being uniformly distributed between selected maximum and minimum values. For position and hwhh these are set within the program. The maximum amplitude is specified by the user, and the minimum amplitude is one-half of the maximum. The limits now used are 50 to 150 wavenumbers (or cm^{-1}) for the spacing between the centers of pulses, and 5 to 20 wavenumbers for the hwhh. Examples can be seen in Figs. 4.9 and 4.17. In the second type of sequence, the amplitude and spacing are constant, although the width varies, advancing in even steps from 5 to 20 wavenumbers for the set of 10 pulses (Fig. 4.1).

After both the signal and noise have been generated, the signal-to-noise ratio (SNR) for each pulse is calculated. The definition of SNR used here is

$$SNR = \sum_{i=\nu_0-2\alpha}^{\nu_0+2\alpha} s(\nu_i)^2 / \sum_{i=\nu_0-2\alpha}^{\nu_0+2\alpha} n(\nu_i)^2. \tag{4-3}$$

The estimate for the variance of the noise is the mean-square noise level, calculated as

$$\hat{\sigma}^2 = (1/N) \sum_{i=1}^{N} n(\nu_i)^2 \tag{4-4}$$

<u>Matched Filtering</u>. Before the statistical decision procedure is carried out, the input signal is passed through a filter that is matched to a Lorentzian pulse of typical hwhh.

The discrete-time impulse response of the filter is 128 points (wavenumbers) in length with the pulse peak at 63, and it has a maximum amplitude of 1.0 and a hwhh of 10.0.

The assumption is made that the noise component of the signal at the filter output has a Gaussian density function. This assumption should hold if the filter is linear. It has been verified that the noise at the filter output is roughly Gaussian by computing histograms of the output of the filter resulting from an input of noise alone.

The process of filtering involves performing a discrete convolution of the filter response and the signal. The overlap-add method of fast convolution is used [9].

High-Pass Filtering (Trend Removal). A problem that appeared in early tests was that if pulses are so close that in the matched filter output the signal level does not decrease far enough, the detector may not resolve them as separate pulses. One idea for overcoming this problem is to filter the output so as to remove very low-frequency components, since large pulses that are close together create a general trend which rises during the occurrence of such pulses.

This was done by having the program write the output sequence from the matched filter to a file, and then using RALPH, a general-purpose signal-processing program [10], to remove a polynomial trend. A fifth-degree polynomial was used. The effect can be seen by comparing Figs. 4.3 and 4.4 or Figs. 4.11 and 4.12. The result was that the large peaks were lowered enough so that they could be separated more easily. One problem, as will be seen in the results, is that smaller pulses are sometimes lowered so much that they are more difficult to detect.

Sequential Detector. The algorithm for carrying the sequential detection procedure is shown in Fig. 3.1.

The program originally allowed the sampling to continue

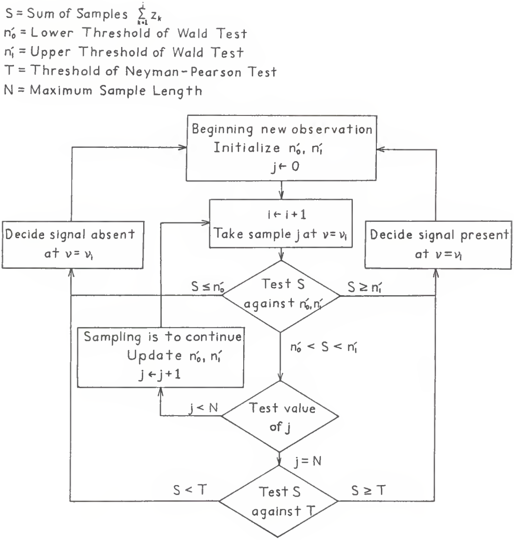

Figure 3.1. Sequential detection algorithm.

indefinitely until a decision was reached. However, this might cause a problem in that the thresholds are increased for each additional sample, and a peak that appears after many samples have been taken may be harder to detect. A good suggestion for a maximum sample length might be the half-width at half-height or the width at half-height of a typical pulse, and a maximum length of 15 samples is now used. Once this maximum length is reached, a decision would be made on the basis of the samples already taken.

This decision is made using a multiple-sample Neyman-Pearson test, with a probability of false detection of $a' = 10^{-4}$. When a constant but unknown signal amplitude is assumed, the test shown in Eqns. (2-11) can be developed so that the sum of k samples is compared to the threshold $T_1 = k\ \mathrm{erfc}^{-1}(a')\sigma$[11]. For $k = 15$ and $a' = 10^{-4}$ this becomes $T_1 = 3.8\sqrt{15}\,\sigma$.

IV. EXPERIMENTAL RESULTS

4.1 Effect of Detector Parameters

The sequential detection procedure presented above involves a requirement that some parameters be selected. In setting the threshold, the theoretical probabilities of error must be decided upon, as must an estimate for the signal amplitude, as shown by Eqns. (2-12) and (2-17). The rate of sampling must also be chosen. It is desired that the detector should give consistent results on pulses that vary in amplitude or width when a single set of parameters is chosen. Some discussion of these parameters follows.

Signal Level. The sequential test, as described in Section II, is based on the assumption of a known signal, and an assumed signal level is used in setting the threshold as shown in Eqns. (2-17). Any signal which exceeds this level will be detected sooner and a signal which stays below this level will not be detected as soon and will have a lower probability of being detected. In the tests it appeared that a higher assumed signal level actually seemed to help in detection of weak signals, and also made false detections more likely in a noise-only sequence. This occurs because, as can be seen from Eqns. (2-17), a higher value of m lowers the upper threshold initially, although it is then increased more quickly as more samples are taken. A preferred value has not been decided upon, but a signal level equal to the rms noise level or one-half this value was used for most tests.

<u>Sampling Rate</u>. The sampling interval used in the tests that are presented was the shortest possible, being the sample interval of the discrete signal representation, or one sample per cm^{-1}. Slower sampling greatly reduces the ability to detect low peaks, so the faster sampling rate was maintained.

<u>Theoretical Error Probabilities</u>. The intended probabilities of Type I and Type II errors, denoted a' and b' respectively, are used in setting the thresholds as shown in Eqns. (2-12) and (2-17). These equations show that the effect of a smaller a' (probability of false detection) is to raise the upper threshold, without significantly affecting the lower threshold, thus making decisions in favor of H_1 more difficult. The main effect of a smaller b' (probability of false dismissal) is to decrease the lower threshold, making dismissals of H_1 more difficult. In the actual results the effect of decreasing a' seemed to be to make false detection less likely and also to make detection of signals more difficult. The effect of changing b' was not as clear.

4.2 Results of Tests

Complete sets of plots for three different signal sequences, showing signal alone, signal with noise, filter output, and detector output, are presented in Figs. 4.1 through through 4.22. The numbers printed along the upper edges of the plots are the SNRs for the respective peaks. For each sequence several detector outputs are shown to demonstrate the effect of changing some of the detector parameters. In all of the sequences the pulses are of the Lorentzian shape. The sequences are as follows:

(1) A set of uniform pulses with an amplitude of 0.7.

(2) A set of randomly generated pulses with amplitudes between 1.5 and 3.0. The seed for random number generation was 11.

(3) A set of randomly generated pulses with amplitudes between 1.0 and 2.0. The seed was 11, following the generation of Sequence (2).

The amplitudes referred to here are set with respect to the rms noise level.

Sequence (1). This sequence was used to test the detector performance at low SNRs. The peaks vary in SNR from 0.13 to 0.25. What appears to be the first peak in the filter output (at about 80 or 90 cm^{-1}) is actually a noise pulse, as can be seen from the fact that a similar pulse appears at the same position in the filter outputs for the other two sequences. The actual signal peak follows it and is too low to be detected.

Here peaks as low as 0.2 can be detected. The second and sixth peaks, with SNRs of 0.13 and 0.18 respectively, cannot be detected with values of a' anywhere from 10^{-5} (Figs. 4.5 and 4.6) to 10^{-2}, but they can be detected when a' = 10^{-1} (Figs. 4.7 and 4.8).

Changing b' from 10^{-3} to 10^{-1} (Figs. 4.5 and 4.6) caused the secondary peak on the seventh pulse to be detected separately from the main peak, instead of being missed. Otherwise the choice of b' did not appear to have a significant effect.

When the assumed amplitude m was changed to 0.5 instead of 1.0, with a' at 10^{-1} (Fig. 4.8) the sixth peak seemed to be detected easier, but there was not much effect on the other peaks.

Sequence (2). Using a larger value of a' generally helps the detector performance, but in some earlier tests it appeared that this also made it more likely that some larger pulses would be merged together in the detector output. To test for this problem in the final version of the detector, a sequence with amplitudes of between 1.5 and 3.0 and SNRs of 0.6 to 2.7 was tested.

Changing a' from 10^{-5} to 10^{-1} did have the effect of causing less separation between some of the output pulses, but no problems resulted (Figs. 4.13 and 4.15). At a' = 10^{-5}, a choice of b' = 10^{-3} caused the noise peak preceding the first signal peak to be detected separately (Fig. 4.13), while with values of both b' = 10^{-5} (Fig. 4.14) and b' = 10^{-1} (not shown), it was merged with the first peak; so again the effect of b' was hard to determine. Changing m to 0.5 had little effect except that the first peak was again merged with the noise peak (Fig. 4.16).

The third peak, with a SNR of 0.91, was not detected in any of the tests, since it was lowered too much by the trend removal.

Sequence (3). This sequence was used to test the detector performance on a varied sequence such as Sequence (2) but at lower SNRs. Here the SNRs range from 0.37 to 1.50. The plots shown are for a' = 10^{-1} and b' = 10^{-3} at both m = 1.0 and m = 0.5 (Figs. 4.20 and 4.21), as well as one for a' = 10^{-5} and m = 1.0 (Fig. 4.22). All of the pulses were detected when using both values of a', although value of 10^{-5} results in weaker detection. There was also a false detection at the beginning due to the effect of the trend removal. However, the noise peak which was detected as a signal in the other two sequences was not detected here.

Noise-Only Sequence. It was seen in tests on a noise-only sequence that false detections were likely to result from the noise peaks that appear when the noise is filtered (Figs. 4.23 to 4.25). These false detections appear even at very small values of a', such as 10^{-10}. Since the value of a' does not seem to have much effect on these false detections, it may be best to keep a' around 10^{-5} or higher to help in the detection of low-level signals. In the other tests, when signals were present, these noise peaks usually did not have a significant effect. These results suggest that there may be a greater problem with false detections when peaks are farther apart.

TABLE 4.1

Data for Sequence (1)

Pulse Number	Center Position	Amplitude	Half-Width at Half-Height	SNR
1	51	0.7	5.0	0.146
2	151	0.7	6.7	0.127
3	251	0.7	8.3	0.240
4	351	0.7	10.0	0.236
5	451	0.7	11.7	0.188
6	551	0.7	13.3	0.180
7	651	0.7	15.0	0.246
8	751	0.7	16.7	0.167
9	851	0.7	18.3	0.182
10	951	0.7	20.0	0.233

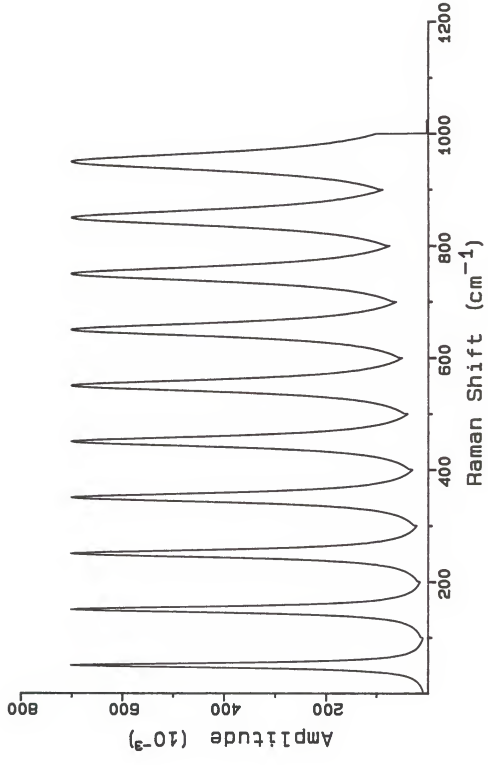

Figure 4.1. Sequence (1).

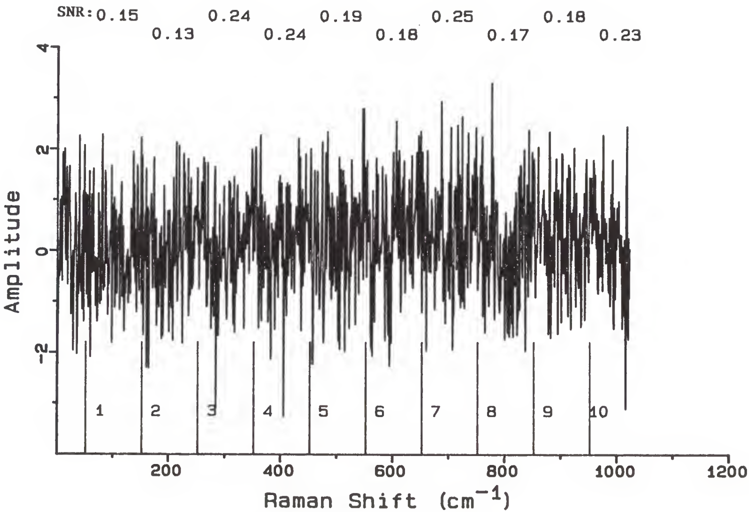

Figure 4.2. Sequence (1) with noise added.

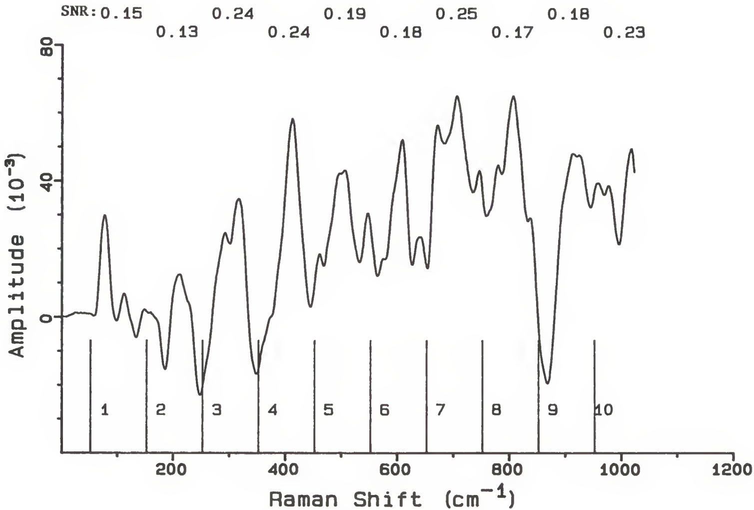

Figure 4.3. Matched filter output.

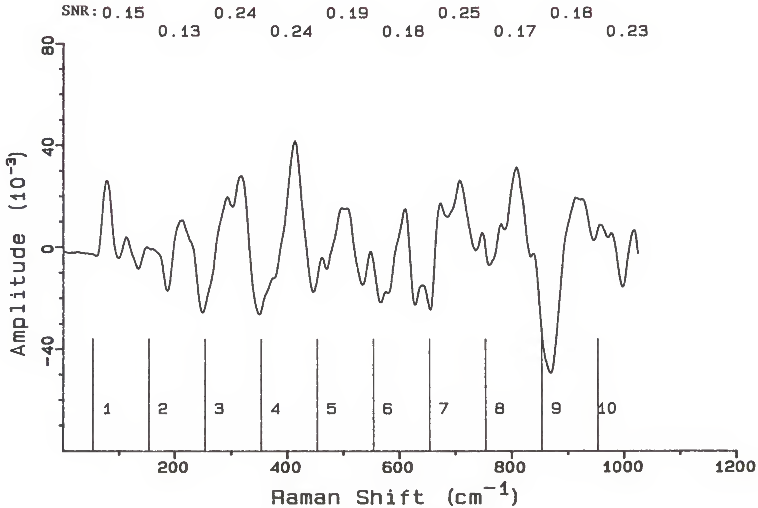

Figure 4.4. Matched filter output, with trend removed.

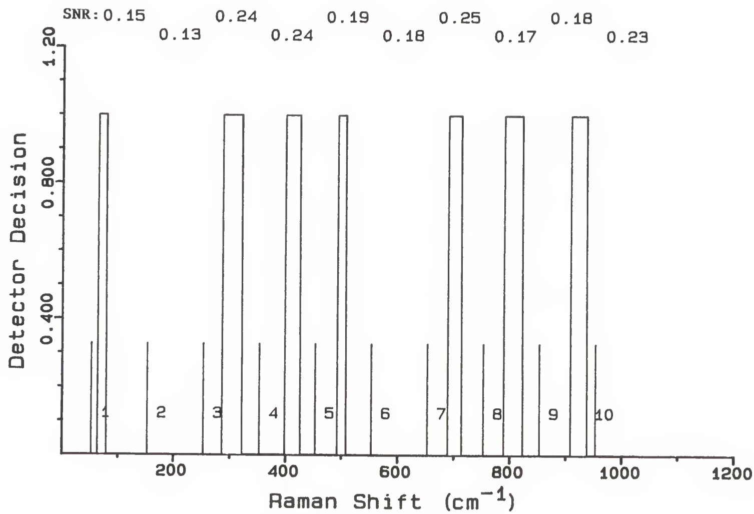

Figure 4.5. Detector output ($a' = 10^{-5}$, $b' = 10^{-3}$, $m = 1.0$).

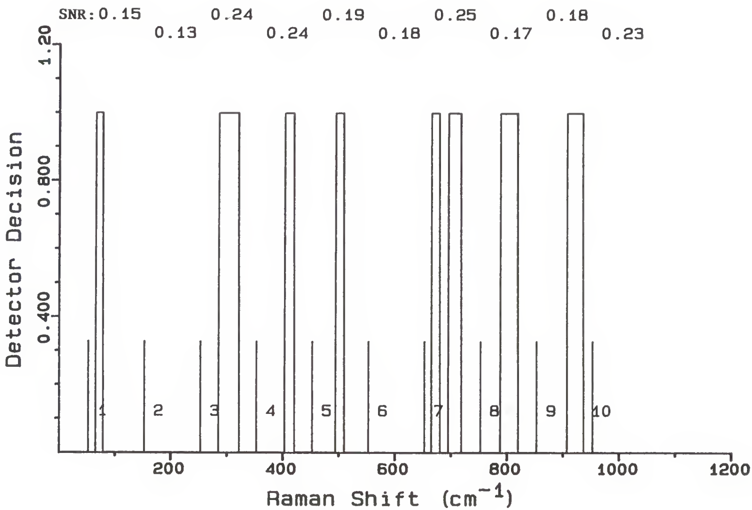

Figure 4.6. Detector output (a' = 10^{-5}, b' = 10^{-1}, m = 1.0).

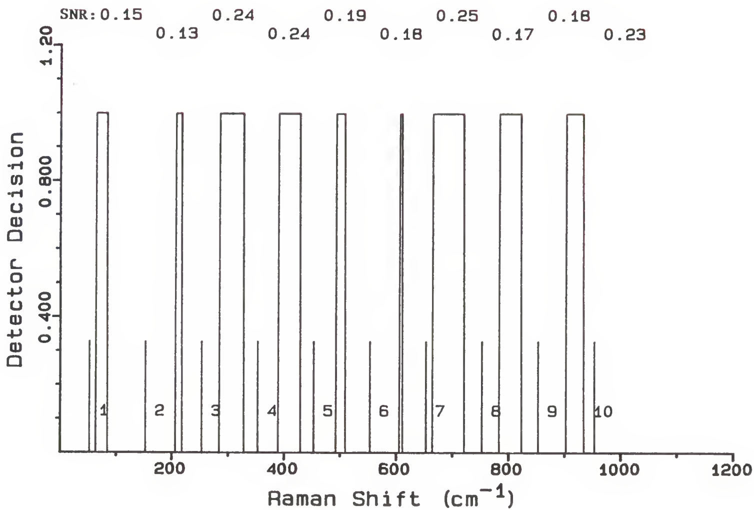

Figure 4.7. Detector output (a' = 10^{-1}, b' = 10^{-3}, m = 1.0).

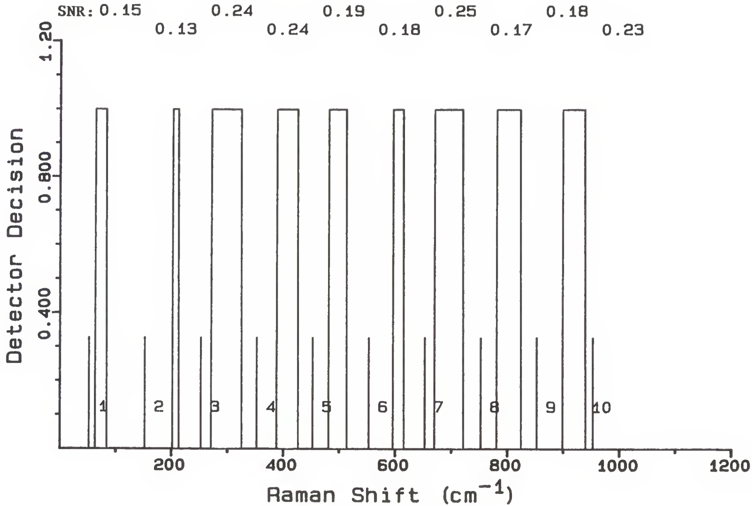

Figure 4.8. Detector output ($a' = 10^{-1}$, $b' = 10^{-3}$, $m = 0.5$).

TABLE 4.2

Data for Sequence (2)

Pulse Number	Center Position	Amplitude	Half-Width at Half-Height	SNR
1	50	1.45	5.0	0.555
2	101	1.63	5.6	1.461
3	191	1.45	9.0	0.905
4	303	1.94	14.7	1.725
5	377	1.57	10.6	1.267
6	442	2.15	7.2	1.680
7	551	2.55	7.7	1.921
8	695	1.49	16.1	0.862
9	800	2.77	19.6	2.542
10	925	2.53	8.9	2.605
11	995	2.58	11.5	2.652

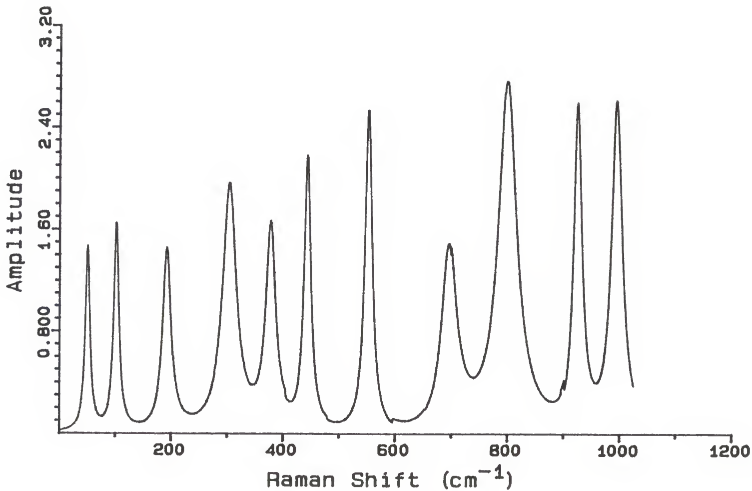

Figure 4.9. Sequence (2).

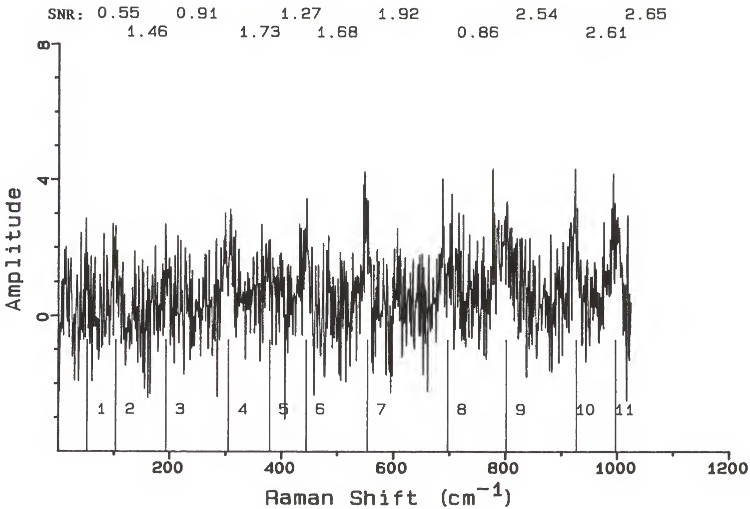

Figure 4.10. Sequence (2) with noise added.

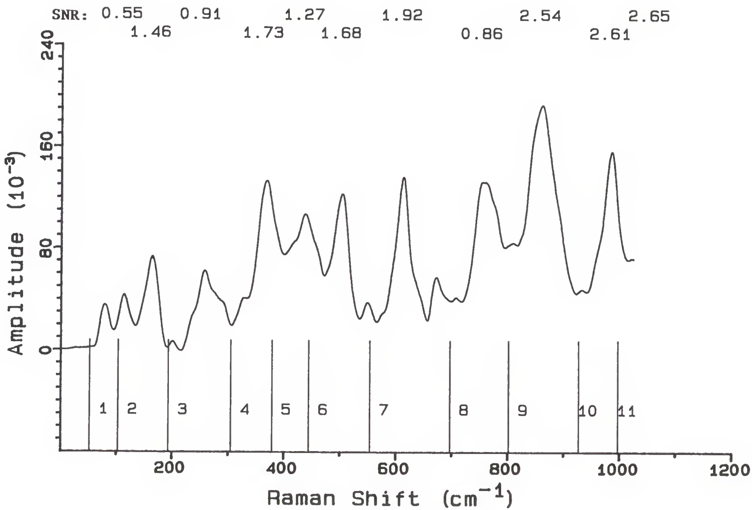

Figure 4.11. Matched filter output.

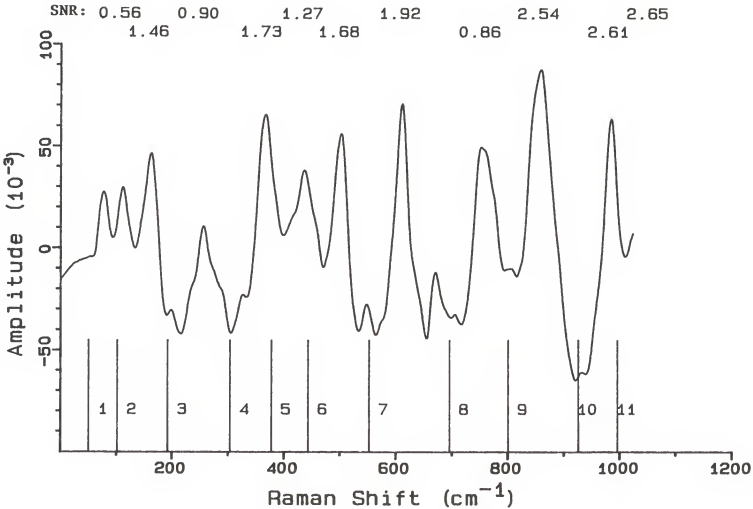

Figure 4.12. Matched filter output, with trend removed.

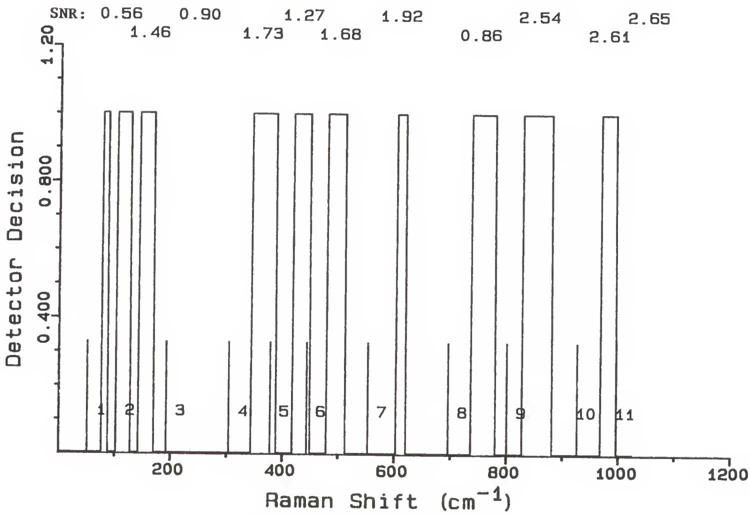

Figure 4.13. Detector output ($a' = 10^{-5}$, $b' = 10^{-3}$, $m = 1.0$).

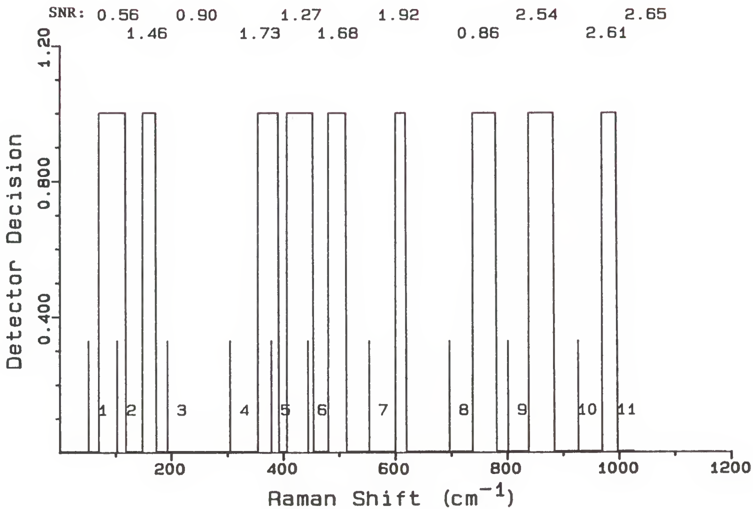

Figure 4.14. Detector output (a' = 10^{-5}, b' = 10^{-5}, m = 1.0).

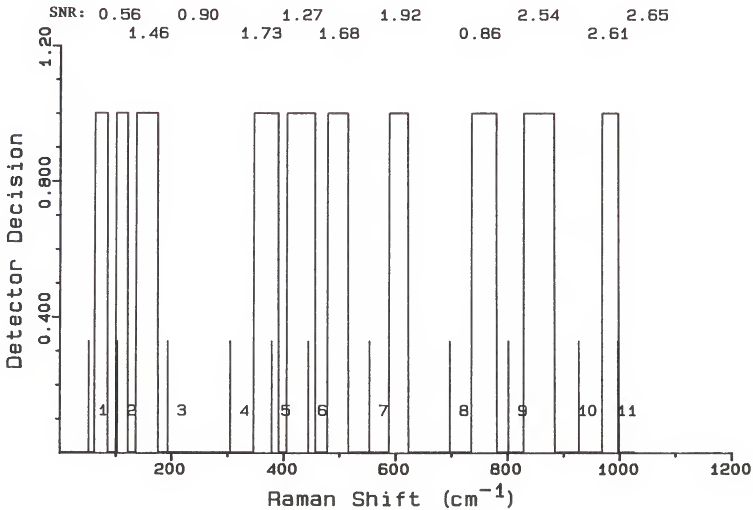

Figure 4.15. Detector output (a' = 10^{-1}, b' = 10^{-3}, m = 1.0).

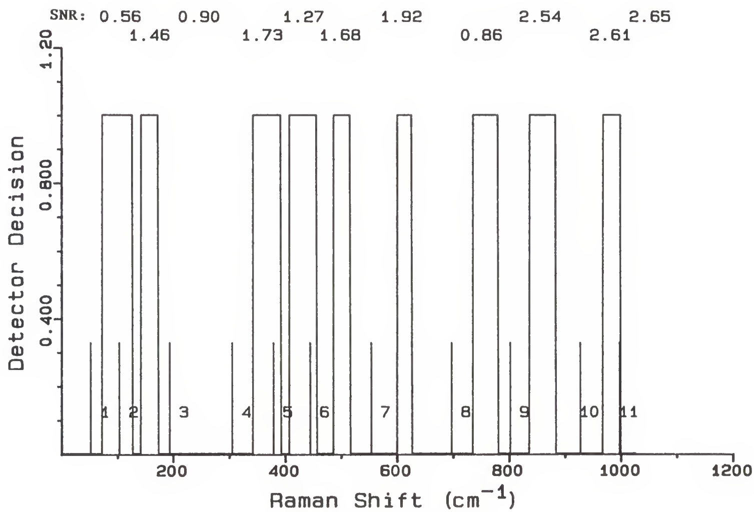

Figure 4.16. Detector output (a' = 10^{-1}, b' = 10^{-3}, m = 0.5).

TABLE 4.3

Data for Sequence (3)

Pulse Number	Center Position	Amplitude	Half-Width at Half-Height	SNR
1	90	1.61	10.6	1.180
2	206	1.81	19.7	1.503
3	300	1.56	16.5	1.189
4	433	1.72	10.5	1.769
5	524	1.52	10.8	0.857
6	673	1.45	15.8	0.822
7	775	1.01	14.5	0.405
8	883	0.98	5.5	0.336
9	1015	1.22	16.8	0.599

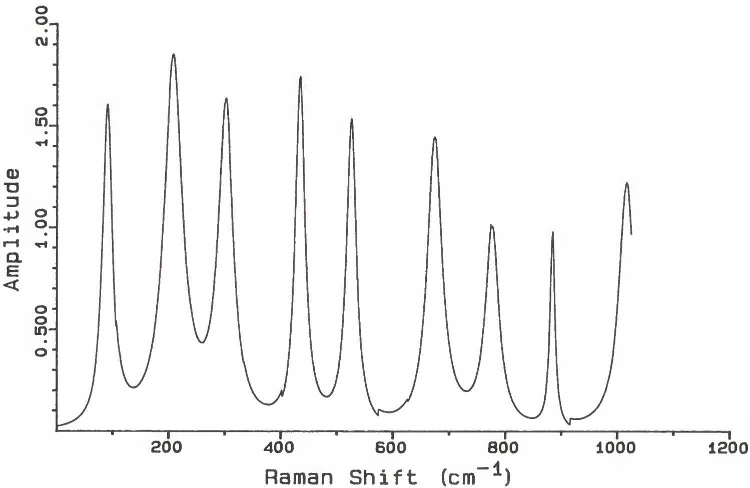

Figure 4.17. Sequence (3).

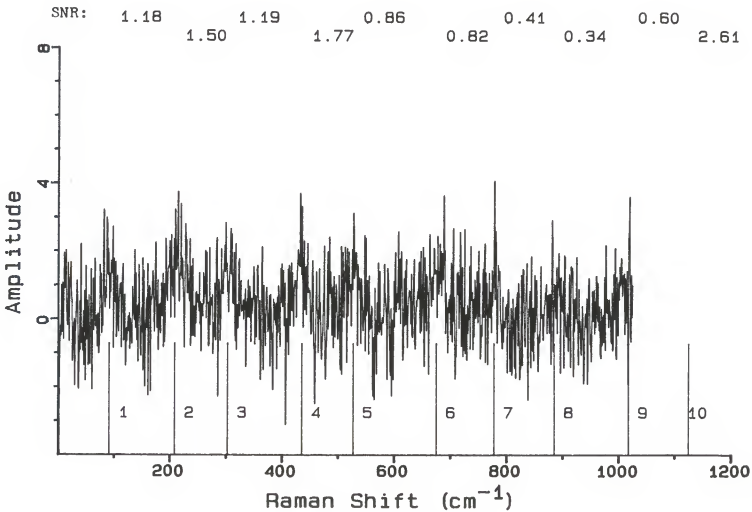

Figure 4.18. Sequence (3) with noise added.

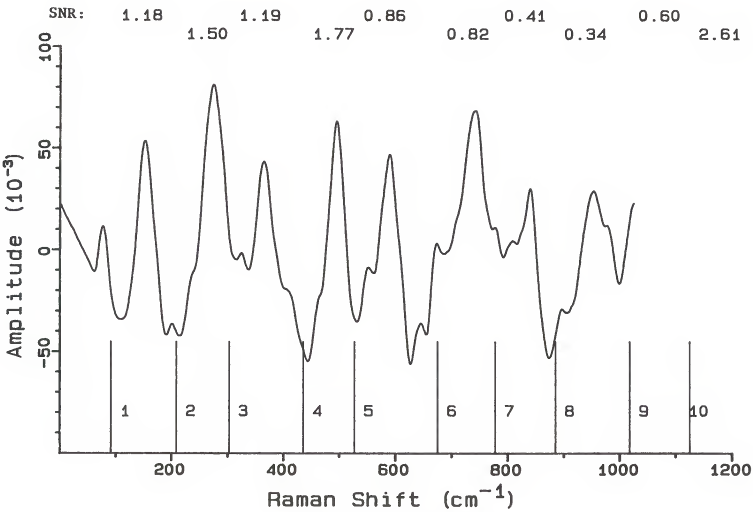

Figure 4.19. Matched filter output, with trend removed.

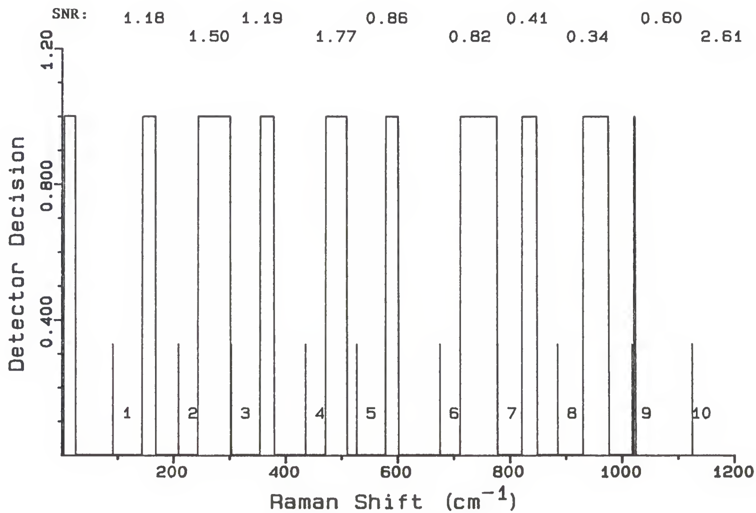

Figure 4.20. Detector output (a' = 10^{-1}, b' = 10^{-3}, m = 1.0).

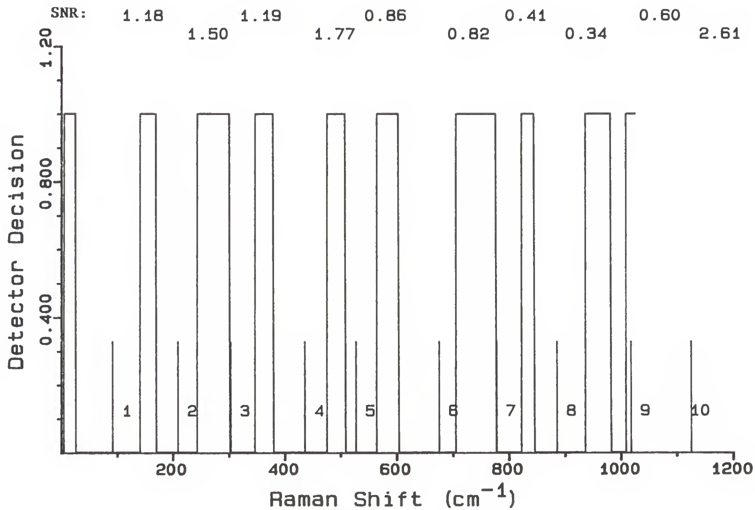

Figure 4.21. Detector output (a' = 10^{-1}, b' = 10^{-3}, m = 0.5).

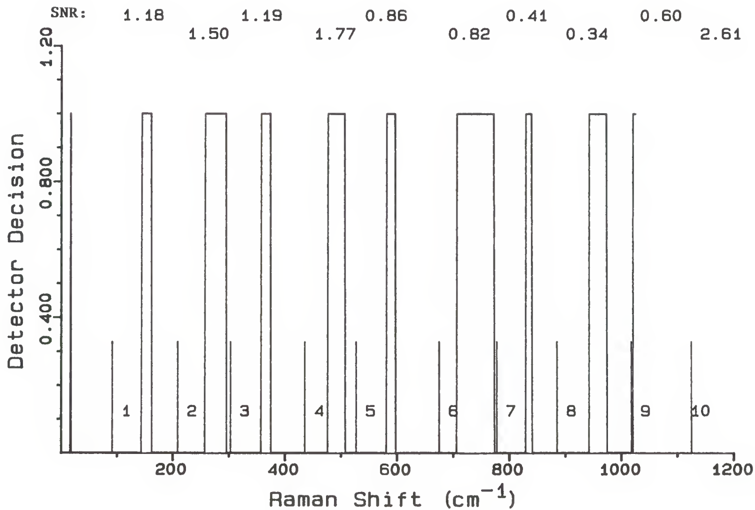

Figure 4.22. Detector output (a' = 10^{-5}, b' = 10^{-3}, m = 1.0).

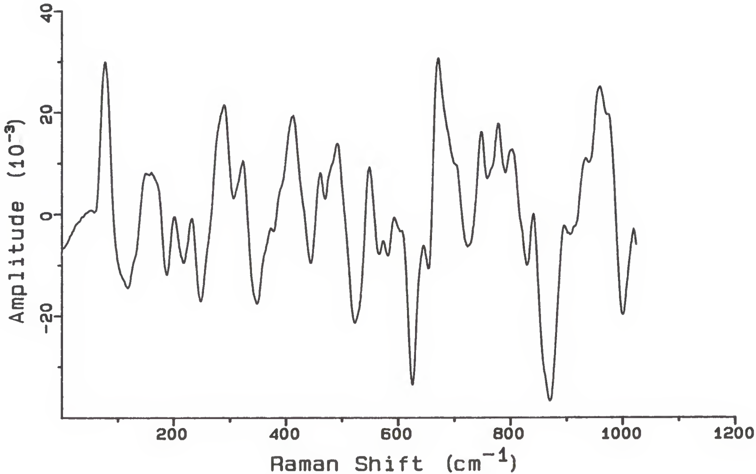

Figure 4.23. Filter output from noise-only sequence.

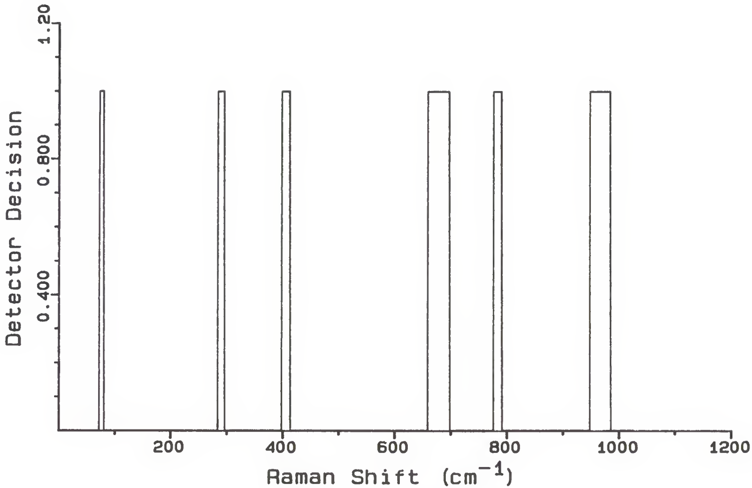

Figure 4.24. Detector output (a' = 10^{-5}, b' = 10^{-3}, m = 1.0).

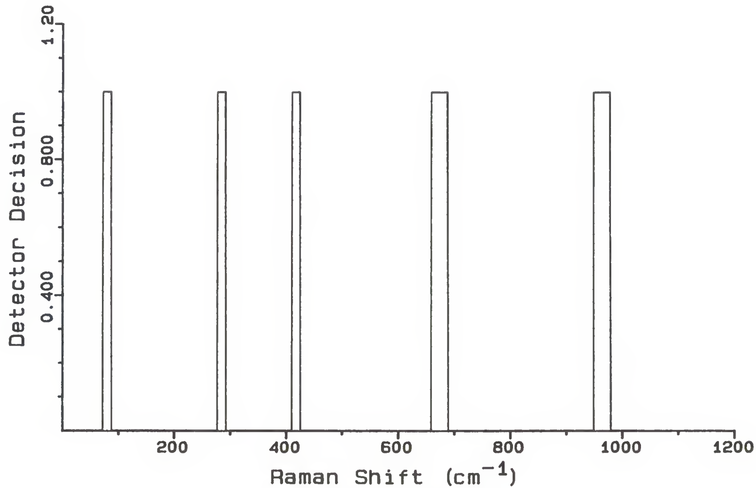

Figure 4.25. Detector output (a' = 10^{-10}, b' = 10^{-3}, m = 1.0).

V. CONCLUSION

5.1 Summary of Results

A study has been made of a system for the detection of signals typical of those encountered in Raman spectrocopy. The system consists of a matched filter followed by a detector based on Wald testing. The detector has been demonstrated to be capable of detecting peaks with SNRs as low as 0.25 to 0.5. However, the reliability of the detector appears to be dependent on the particular form of the sequence, which is to say that when there are many peaks close together having varying amplitudes, some may be lost due to the trend removal action, even some of those with SNRs as high as 1.0 or 2.0.

It was seen from the first sequence shown that peaks as low as 0.13 can be detected by allowing a large value for the false-detection probability, such as 10^{-1}. However, selecting large values of a' will probably increase the likelihood of false detections. Generally it appeared that larger values of a' increase both the probability of detection and the probability of false detections, as would be expected, while b' and the assumed signal level m did not have as significant an effect, although a lower value of m did appear to help detection in some cases.

In general, this system may be a useful addition to the methods of peak enhancement that have already been employed in Raman spectroscopy, in that it provides for automatic detection of spectral peaks and gives fairly reliable results at signal-to-noise ratios as low as 1.0 or less.

5.2 Recommendations for Further Work

Some further testing of this system would be needed to fully understand its capabilities. Tests might be made on sequences more closely resembling actual spectra. Also, the effect on the performance when the noise variance is greater than the estimate should be tested.

The performance of a detector based on a multiple-sample Neyman-Pearson test might be tested for comparison. Such a test would be essentially the same as the test used in the Wald detector to make decisions once the maximum sample size was reached.

It might be useful to study the class of detectors known as nonparametric or distribution-free detectors, which have been applied to problems in radar and communications. The characteristic of such detectors is that their performance is not highly dependent on the distribution functions of the noise or signal-plus-noise. Such a detector would be useful when the noise or signal statistics are not well known. However, if these statistics can be estimated with some accuracy, then it would be advantageous to use a detector, such as that presented here, which makes use of this knowledge. Reference [3] is one source for information on nonparametric detection.

REFERENCES

1. S.A. Dyer, D.S. Hardin, and E.B. Bradley, "An Adaptive Detection System for Raman Spectral Peaks at Low Signal-to-Noise Ratios", Journal of Raman Spectroscopy, Vol. 15, No. 6, 400-405 (1984).

2. J.V. DiFranco and W.L. Rubin, Radar Detection, Prentice-Hall, Englewood Cliffs, New Jersey, 1968. p.253.

3. J.D. Gibson and J.L. Melsa, Introduction to Nonparametric Detection with Applications, Academic Press, New York, 1975, p. 10.

4. DiFranco and Rubin, p.261.

5. M.D. Srinath and P.K. Rajasekaran, An Introduction to Statistical Signal Processing with Applications, John Wiley and Sons, New York, 1979, p. 70.

6. ibid, pp. 90-91.

7. ibid, pp. 90-91.

8. ibid, pp. 91-93.

9. N. Ahmed and T. Natarajan, Discrete-Time Signals and Systems, Reston Publishing Co., Reston, Virginia, 1983, pp. 214-216.

10. Ralph Users Guide, Department of Electrical and Computer Engineering, Manhattan, Kansas, 1986.

11. Gibson and Melsa, pp. 39-41.

BIBLIOGRAPHY

1. B. Dimitriadis and B. Kazakos, "Nonparametric Sequential Test for Data with Markov Dependence", IEEE Transactions on Aerospace and Electronics Systems, Vol. AES-19, No. 3, 338-347 (1983).

2. J.C. Hancock and P.A. Wintz, Signal Detection Theory, McGraw-Hill, New York, 1966.

3. D.P. Meyer and H.A. Mayer, Radar Target Detection: Handbook of Theory and Practice, Academic Press, New York, 1973.

4. H.B. Raemer, Statistical Communication Theory and Applications, Prentice-Hall, Englewood Cliffs, New Jersey, 1969.

5. S. Tantaratana, "Sequential Probability Ratio Test and Sequential Linear Detector Compared for Autoregressive Noise", IEEE Transactions on Information Theory, Vol. IT-31, No. 5, 693-697 (1985).

6. I. Vrana, "On a Direct Method of Analysis and Synthesis of the SPRT", IEEE Transactions on Information Theory, Vol. IT-28, No. 6, 905-911 (1982).

7. A. Wald, Sequential Analysis, John Wiley and Sons, New York, 1947.

```
C*************************************************************************
C
C       PROGRAM DETECTOR_TEST
C
C       VAX-11 FORTRAN SOURCE FILENAME:          DETECTOR_TEST.FOR
C
C       DEPARTMENT OF EECE                       KANSAS STATE UNIVERSITY
C
C       AUTHOR:  Damon Mick
C
C       DATE CREATED:  July 1987 (Final version)
C
C*************************************************************************
C
C       PURPOSE:  Main routine to generate signal data, carry out filtering
C       and the detection procedure, write data to and read data from disk,
C       and plot data sequences.
C
C       ROUTINE(S) CALLED OR ACCESSED BY THIS ROUTINE:
C
C       DETECT
C       FILTER
C       PLOT
C       READ_DATA
C       RMS_CALC
C       SIGNAL_GEN
C       WRITE_DATA
C
C*************************************************************************

      PROGRAM DETECTOR_TEST

      IMPLICIT NONE

      REAL      SIGNAL(1024), NOISE(1024), SIGNOI(1024), OUTPUT(1024),
     -          Y_DATA(1024), MEAN(16), AMPL(16), HWHH(16),
     -          SNR(16), RMS, RMS_2, AMAX, ALPHA, BETA, LEVEL

      INTEGER   DET(1024), OPER, SEQ_TYPE, SHAPE, SEED, ARRAY_NUM,
     -          I, T, DEVICE_NUM

      CHARACTER*1 A
      CHARACTER*20 FILE_NAME
C
C--   Main menu.
C
      DO WHILE (OPER.NE.6)
         TYPE *, 'ENTER NUMBER:'
         TYPE *, ' (1) Generate a signal, add noise, and filter'
         TYPE *, ' (2) Read a filtered signal sequence from a file'
         TYPE *, ' (3) Set threshold and perform detection'
         TYPE *, ' (4) Plot data'
         TYPE *, ' (5) Write a data sequence to a file'
         TYPE *, ' (6) Exit the program'
         READ *, OPER

         IF (OPER.EQ.1) THEN
C
C--         Read noise sequence and obtain measurement of rms noise level.
C
```

```
          TYPE*, 'Reading noise file.'
          CALL READ_DATA(NOISE)
          CALL RMS_CALC(NOISE, RMS)
C
C--       Get information from the user for the generation of pulses.
C
          TYPE*, 'Enter shape of pulses: (1) Gaussian  (2) Lorentzian'
          READ*, SHAPE
          TYPE*, 'Enter: (1) Random pulses  (2) Uniform pulses'
          READ*, SEQ_TYPE
          TYPE*, 'Enter maximum amplitude for pulses:'
          READ*, AMAX
          IF (SEQ_TYPE.EQ.1) THEN
             TYPE*, 'Enter seed for signal generation:'
             READ*, SEED
          END IF

          CALL SIGNAL_GEN(SIGNAL, NOISE, SHAPE, SEQ_TYPE, AMAX,
     -       RMS, MEAN, AMPL, HWHH, SNR, SEED)

          DO  I=1, 1024
             SIGNOI(I) = SIGNAL(I) + NOISE(I)
          END DO
C
C--       Obtain noise variance at filter output for use in detection routine
C
          CALL FILTER (NOISE, OUTPUT)
          CALL RMS_CALC(OUTPUT, RMS_2)
C
C--       Obtain output of matched filter.
C
          CALL FILTER(SIGNOI, OUTPUT)
C
C--       Write pulse data to a file.
C
          OPEN (UNIT=1, STATUS='NEW', FILE='PULSES.DAT')
          DO I = 1, 15
             IF (MEAN(I).NE.0.) THEN
                WRITE(1, 1) I, MEAN(I), AMPL(I), HWHH(I), SNR(I)
    1           FORMAT (I4.2, 4F10.3)
             END IF
          END DO
          WRITE (1, 1) 16, 0.0, 0.0, 0.0, 0.0

       END IF

       IF (OPER.EQ.2) THEN
C
C--        Noise seqence is needed to calculate estimate of noise
C--        standard deviation.
C
          TYPE*, 'Require noise sequence and signal type used in ',
     -       'original signal.'
          CALL READ_DATA(NOISE)
          CALL FILTER(NOISE, OUTPUT)
          CALL RMS_CALC(OUTPUT, RMS_2)
C
C--       Obtain file of pulse data so that SNRs can be plotted.
C
          TYPE*, 'Enter filename for pulse data.'
```

```
            READ(5, 3) FILE_NAME
    3       FORMAT (A20)

            OPEN(UNIT=1, STATUS='OLD', FILE=FILE_NAME)
            I = 0
            DO WHILE (I.NE.16)
               READ(1,1) I, MEAN(I), AMPL(I), HWHH(I), SNR(I)
            END DO
C
C--         Read file of filtered signal.
C
            CALL READ_DATA(OUTPUT)
         END IF

         IF (OPER.EQ.3) THEN
            TYPE*, 'Enter desired error probabilities (Type I, Type II)'
            READ*, ALPHA, BETA
            TYPE*, 'Enter assumed signal level:'
            READ*, LEVEL
            TYPE*, 'Enter sampling interval:'
            READ*, T
            CALL DETECT(OUTPUT, DET, RMS_2, ALPHA, BETA, LEVEL, T)
         END IF

         IF ((OPER.EQ.4).OR.(OPER.EQ.5)) THEN
   15       TYPE*, 'Select data array:'
            TYPE*, ' (1) Signal without noise'
            TYPE*, ' (2) Signal with noise'
            TYPE*, ' (3) Filter output'
            TYPE*, ' (4) Detector output'
            READ*, ARRAY_NUM
         END IF

         IF (OPER.EQ.4) THEN

            IF (ARRAY_NUM.EQ.1) THEN
               DO I = 1, 1024
                  Y_DATA(I)=SIGNAL(I)
               END DO

            ELSE IF (ARRAY_NUM.EQ.2) THEN
               DO I = 1, 1024
                  Y_DATA(I)=SIGNOI(I)
               END DO

            ELSE IF (ARRAY_NUM.EQ.3) THEN
               DO I = 1, 1024
                  Y_DATA(I)=OUTPUT(I)
               END DO

            ELSE IF (ARRAY_NUM.EQ.4) THEN
               DO I = 1, 1024
                  Y_DATA(I)=DET(I)
               END DO

            END IF

            TYPE*, 'Enter device number: (1) 7475  (2) 4014'
            READ*, DEVICE_NUM
            CALL PLOT(Y_DATA, DEVICE_NUM, ARRAY_NUM, MEAN, SNR)
```

```
      TYPE *, 'Do you want to make another plot?'
      READ(5, 2) A
2     FORMAT(A1)
      IF (A.EQ.'Y') GO TO 15
   END IF

   IF (OPER.EQ.5) THEN
       IF (ARRAY_NUM.EQ.1) THEN
          CALL WRITE_DATA(SIGNAL)
       ELSE IF (ARRAY_NUM.EQ.2) THEN
          CALL WRITE_DATA(SIGNOI)
       ELSE IF (ARRAY_NUM.EQ.3) THEN
          CALL WRITE_DATA(OUTPUT)
       ELSE IF (ARRAY_NUM.EQ.4) THEN
          CALL WRITE_DATA(DET)
       END IF
   END IF

END DO

STOP
END
```

```
C*****************************************************************************
C
C       SUBROUTINE READ_DATA
C
C       VAX-11 FORTRAN SOURCE FILENAME:           RW_DATA.FOR
C
C       DEPARTMENT OF EECE                        KANSAS STATE UNIVERSITY
C
C       AUTHOR:  Damon Mick
C
C       DATE CREATED:  July 1987 (Final version)
C
C*****************************************************************************
C
C       CALLING SEQUENCE:  CALL READ_DATA (SEQUENCE)
C
C       PURPOSE:  This routine reads a data sequence from a disk file.
C
C       ROUTINE(S) CALLED OR ACCESSED BY THIS ROUTINE:
C         DISKIN
C
C       ARGUMENTS REQUIRED:
C
C         SEQUENCE         output     real
C
C*****************************************************************************

      SUBROUTINE READ_DATA(SEQUENCE)

      REAL SEQUENCE(1024)

      CALL DISKIN('REAL', SEQUENCE, 1024, LENACT, 'PROMPT', 'PROMPT',
     &  IERR, 0)

      RETURN
      END
```

```
C*****************************************************************************
C
C       SUBROUTINE WRITE_DATA
C
C       VAX-11 FORTRAN SOURCE FILENAME:          RW_DATA.FOR
C
C       DEPARTMENT OF EECE                       KANSAS STATE UNIVERSITY
C
C       AUTHOR:  Damon Mick
C
C       DATE CREATED:  July 1987 (Final version)
C
C*****************************************************************************
C
C       CALLING SEQUENCE:  CALL WRITE_DATA (SEQUENCE)
C
C       PURPOSE:  This routine writes a data sequence to a disk file.
C
C       ROUTINE(S) CALLED OR ACCESSED BY THIS ROUTINE:
C         DISKOUT
C
C       ARGUMENTS REQUIRED:
C
C         SEQUENCE          input      real
C                           Array from which data is to be written.
C*****************************************************************************

      SUBROUTINE WRITE_DATA(SEQUENCE)

      REAL SEQUENCE(1024)

      CALL DISKOU('REAL', SEQUENCE, 1024, 'PROMPT', 'PROMPT',
     &  IERR, 0)

      RETURN
      END
```

```
C*******************************************************************************
C
C       SUBROUTINE SIGNAL_GEN
C
C       VAX-11 FORTRAN SOURCE FILENAME:          SIGNAL_GEN.FOR
C
C       DEPARTMENT OF EECE                       KANSAS STATE UNIVERSITY
C
C       AUTHOR:  Damon Mick
C
C       DATE CREATED:  July 1987 (Final version)
C
C*******************************************************************************
C
C       CALLING SEQUENCE:  CALL SIGNAL_GEN (SIG, NOI, SHAPE, SEQ_TYPE, AMAX,
C       RMS, MEAN, AMPL, HWHH, SNR, IX)
C
C       PURPOSE:  This routine generates series of pulses having either
C       Lorentzian or Gaussian shapes.
C
C       ROUTINE(S) CALLED OR ACCESSED BY THIS ROUTINE:
C         RANDU
C
C       ARGUMENTS REQUIRED:
C
C         SIG                output    real
C                            The array containing the signal sequences, without
C                            noise added.
C
C         NOI                input     real
C                            The array containing the noise sequence.
C
C         SHAPE              input     integer
C                            Indicates the shape of the pulses to be generated.
C                              1:  Gaussian pulses.
C                              2:  Lorentzian pulses.
C
C         SEQ_TYPE           input     integer
C                            Indicates the type of series to be generated.
C                              1:  Randomly generated pulses.
C                              2:  Pulses uniform in height and spacing.
C
C         AMAX               input     real
C                            The maximum amplitude for the pulses.
C
C         RMS                input     real
C                            The rms noise level.
C
C         MEAN               output    real
C                            The array of the center positions of all the pulses.
C
C         AMPL               output    real
C                            The array of the amplitudes of all the pulses.
C
C         HWHH               output    real
C                            The array of the hwhh's of all the pulses.
C
C         SNR                output    real
C                            The array of the SNRs of all the pulses.
C
```

```
C         IX               input     integer
C                          The seed for random number generation.
C
C*************************************************************************
      SUBROUTINE SIGNAL_GEN(SIG, NOI, SHAPE, SEQ_TYPE, AMAX, RMS,
     -  MEAN, AMPL, HWHH, SNR, IX)

      IMPLICIT NONE

      REAL     SIG(1024), NOI(1024), SGPEAK(15), MEAN(*),
     -         AMPL(*), HWHH(*), SNR(*), AMAX, YFL, RMS,
     -         SSUM, NSUM, SDEV

      INTEGER  SEQ_TYPE, SHAPE, IX,
     -         IY, PNUM, I, J, MIN_POINT, MAX_POINT
      PRINT*, IX
C
C--   Remove any previously generated signal from the array
C--   SIG and any pulse data from the pulse data arrays.
C
      DO I = 1, 1024
         SIG(I) = 0.0
      END DO

      PNUM = 0
      DO  I = 1, 15
         SGPEAK(I)=0.
         MEAN(I)=0.
         HWHH(I)=0.
         AMPL(I)=0.
      END DO

      IF (SEQ_TYPE.EQ.1) THEN
C
C     Procedure for random pulses.
C
C        Obtain position of first pulse.
C
         CALL RANDU(IX, IY, YFL)
         MEAN(1) = 50 + 100*YFL
C
C        Obtain half width at half height for current pulse and
C        calculate st. dev. from hwhh.
C        Obtain pulse height, with respect to rms noise level.
C
         DO WHILE (PNUM.LT.15 .AND. MEAN(PNUM+1).LT.1024)

            PNUM = PNUM + 1
            CALL RANDU(IX, IY, YFL)
            HWHH(PNUM) = 5.0 + 15.0*YFL

            CALL RANDU(IX, IY, YFL)
            AMPL(PNUM) = (1.+YFL)*0.5*AMAX*RMS
C
C           Generate the point values.
C
            MIN_POINT = MAX(1, NINT(MEAN(PNUM))-100)
            MAX_POINT = MIN(1024, NINT(MEAN(PNUM))+100)
```

```
          DO J = MIN_POINT, MAX_POINT

             IF (SHAPE.EQ.1) THEN
C
C            Gaussian pulses.
C
                SDEV = HWHH(PNUM)/SQRT(2.*LOG(2.))
                SIG(J) = SIG(J)+AMPL(PNUM)*EXP(-(J-MEAN(PNUM))**2/
     -           (2*SDEV**2))

             ELSE IF (SHAPE.EQ.2) THEN
C
C            Lorentzian pulses.
C
               SIG(J) = SIG(J)+AMPL(PNUM)/
     -           (1+((J-MEAN(PNUM))/HWHH(PNUM))**2)

             END IF

          END DO
C
C         Obtain the position of the next pulse.
C
          CALL RANDU (IX, IY, YFL)
          MEAN(PNUM+1) = MEAN(PNUM) + 50 + 100*YFL

       END DO

      ELSE IF (SEQ_TYPE.EQ.2) THEN
C
C     Generation of uniform pulses.
C
         DO PNUM = 1, 10

            MEAN(PNUM) = 51 + (PNUM-1)*100
            HWHH(PNUM) = 5.0 + (PNUM-1)*15.0/9.0
            AMPL(PNUM) = AMAX
            SDEV = HWHH(PNUM)/SQRT(2.*LOG(2.))

            DO J =  (PNUM-1)*100+1, PNUM*100

               IF (SHAPE.EQ.1) THEN
                  SIG(J) = AMAX*EXP(-(J-MEAN(PNUM))**2./(2.*SDEV**2.))
               ELSE
                  SIG(J) = AMAX/(1.+((J-MEAN(PNUM))/HWHH(PNUM))**2.)
               END IF

            END DO

         END DO
         PNUM = 10

      END IF

      IF (SHAPE.EQ.1) THEN
          TYPE *, 'Signal generation (Gaussian) completed.'
      ELSE IF (SHAPE.EQ.2) THEN
          TYPE *, 'Signal generation (Lorentzian) completed'
      END IF
C
```

```
C     Calculate SNR for all pulses.
C
      DO I = 1, PNUM
           SSUM = 0
           NSUM = 0
           DO J = NINT(MEAN(I)-2*HWHH(I)), NINT(MEAN(I)+2*HWHH(I))
                IF ((J.GE.1).AND.(J.LE.1024)) THEN
                   SSUM = SIG(J)**2 + SSUM
                   NSUM = NOI(J)**2 + NSUM
                END IF
           END DO
           SNR(I) = SSUM/NSUM
      END DO
      RETURN
      END
```

```
C*******************************************************************************
C
C       SUBROUTINE RANDU
C
C       VAX-11 FORTRAN SOURCE FILENAME:          SIGNAL_GEN.FOR
C
C       DEPARTMENT OF EECE                       KANSAS STATE UNIVERSITY
C
C
C*******************************************************************************
C
C       CALLING SEQUENCE:  CALL RANDU (IX, IY, YFL)
C
C       PURPOSE:  This routine generates a number which has a value evenly
C       distributed between 0 and 1.
C
C       ROUTINE(S) CALLED OR ACCESSED BY THIS ROUTINE:
C         None.
C
C       ARGUMENTS REQUIRED:
C
C         IX              input     integer
C                         The seed for the random number generation.
C
C         IY              output    integer
C                         The random number that is generated, before being
C                         normalized.
C
C         YFL             output    integer
C                         IY normalized to a value between 0 and 1.
C
C*******************************************************************************

      SUBROUTINE RANDU (IX, IY, YFL)
C
C--   Random number generation routine.  Value of YFL will be evenly
C--   distributed between 0 and 1.
C
      IY = IX*65539
      IF (IY) 5, 6, 6
    5 IY = IY + 2147483647 + 1
    6 YFL = IY*0.4656613E-9
      RETURN
      END
```

```
C***********************************************************************
C
C       SUBROUTINE FILTER
C
C       VAX-11 FORTRAN SOURCE FILENAME:          FILTER.FOR
C
C       DEPARTMENT OF EECE                       KANSAS STATE UNIVERSITY
C
C       AUTHOR:  Damon Mick
C
C       DATE CREATED:  July 1987 (Final version)
C
C***********************************************************************
C
C       CALLING SEQUENCE:  CALL FILTER(SIG, OUTP)
C
C       PURPOSE:  This routine computes the result of filtering the data
C       sequence SIG with a filter matched to a Lorentzian pulse shape.
C
C       ROUTINE(S) CALLED OR ACCESSED BY THIS ROUTINE:
C         FFT
C
C       ARGUMENTS REQUIRED:
C
C         SIG              input      real
C                          The array of the data to be filtered.
C
C         OUTP             output     real
C                          The array containing the output of the filter.
C
C***********************************************************************

      SUBROUTINE FILTER (SIG, OUTP)

      REAL     SIG(1024), OUTP(1024), FIL(128), SDEV
      COMPLEX  H(256), X(256), Y(256)
      INTEGER  J, INV

C
C     Generation of filter response.
C
      DO  J = 1, 128
         FIL(J) = 1/(1 + ((J-63)/10.0)**2)
      END DO
C
C     Initialize the filter output array.
C
      DO J = 1, 1024
         OUTP(J) = 0
      END DO
C
C     Transfer filter response to complex H array and initialize
C     second half of H array.
C
      DO  J = 1, 128
         H(J) = CMPLX(FIL(J), 0.0)
         H(J+128) = (0, 0)
      END DO
C
C     Convolve filter sequence with signal sequence by overlap-add.
```

```
C     Segments of signal sequence are placed in complex X array.
C
      INV=0
      CALL FFT (H, 256, INV)

      DO Q = 0, 1024/128-1
C
C     Do the Qth segment.
C
         DO  J = 1, 128
            K = J+Q*128
            X(J) = CMPLX(SIG(K), 0.0)
            X(J+128) = (0,0)
         END DO

         INV=0
         CALL FFT(X, 256, INV)
C
C        X and H arrays contain the DFTs of the data segment and the filter
C        response, respectively.
C
         DO J = 1, 256
            Y(J) = X(J)*H(J)
         END DO
C
C        Take inverse transform of Y.
C
         INV=1
         CALL FFT(Y, 256, INV)

         DO J = 1, 256
            K = J+Q*128
            OUTP(K) = OUTP(K)+REAL(Y(J))
         END DO

      END DO

      RETURN
      END
```

```
C*****************************************************************************
C
C       SUBROUTINE DETECT
C
C       VAX-11 FORTRAN SOURCE FILENAME:         DETECT.FOR
C
C       DEPARTMENT OF EECE                      KANSAS STATE UNIVERSITY
C
C       AUTHOR:  Damon Mick
C
C       DATE CREATED:  July 1987 (Final version)
C
C*****************************************************************************
C
C       CALLING SEQUENCE:  DETECT(OUTP, DET, RMS, ALPHA, BETA, LEVEL,
C       SAMPLE_INTERVAL)
C
C       PURPOSE:  This routine performs a Wald testing procedure on the output
C       of the matched filter.
C
C       ROUTINE(S) CALLED OR ACCESSED BY THIS ROUTINE:
C         None.
C
C       ARGUMENTS REQUIRED:
C
C         OUTP               input      real
C                            The array containing the matched filter output.
C
C         DET                output     integer
C                            The array containing the detector output.
C
C         RMS                input      real
C                            The rms noise level at the matched filter output.
C
C         ALPHA              input      real
C                            The theoretical value for the false-alarm probability.
C
C         BETA               input      real
C                            The theoretical value for the false-dismissal
C                            probability.
C
C         LEVEL              input      real
C                            The assumed value for the signal level.
C
C         SAMPLE_INTERVAL input      integer
C                            The spacing between samples.
C
C*****************************************************************************

      SUBROUTINE DETECT(OUTP, DET, RMS, ALPHA, BETA, LEVEL,
     -  SAMPLE_INTERVAL)

      IMPLICIT NONE

      REAL      OUTP(1024), RMS, ALPHA, BETA, LEVEL, INIT_BIAS_0,
     &          INIT_BIAS_1, BIAS_0, BIAS_1, SUM, EXPT_VALUE
      INTEGER   DET(1024), SAMPLE_INTERVAL, SAMPLE_NO, DETECT_FLG,
     &          I, J, K, MAX_SAMPLE_LEN

      MAX_SAMPLE_LEN = 15
```

```
C
C     Determination of threshold levels.
C
      INIT_BIAS_0 = RMS*LOG(BETA/(1.-ALPHA))/LEVEL+0.5*LEVEL*RMS
      INIT_BIAS_1 = RMS*LOG((1.-BETA)/ALPHA)/LEVEL+0.5*LEVEL*RMS
C
C     Procedure for first decision.
C
      I = 1
      DETECT_FLG = 0
      SAMPLE_NO = 1
      BIAS_0 = INIT_BIAS_0
      BIAS_1 = INIT_BIAS_1
      SUM = 0.

      DO WHILE (DETECT_FLG.EQ.0)

         SUM = SUM + OUTP(I)
C
C        Decisions made.
C
         IF (SUM.LE.BIAS_0) THEN
C
C           No signal present.
C
            DET(I) = 0
            DETECT_FLG = 1

         ELSE IF (SUM.GE.BIAS_1) THEN
C
C           Signal present.
C
            DO K = I-SAMPLE_INTERVAL+1, I
                DET(K) = 1
            END DO
            DETECT_FLG = 1

         ELSE
C
C        No decisions made; update thresholds and sample count.
C
            DET(I) = 0
            SAMPLE_NO = SAMPLE_NO+1
            BIAS_0 = BIAS_0 + 0.5*RMS*LEVEL
            BIAS_1 = BIAS_1 + 0.5*RMS*LEVEL

         END IF

         I = I+SAMPLE_INTERVAL

      END DO

      DO WHILE (I.LE.1024)

         DETECT_FLG = 0
         SAMPLE_NO = 1
         BIAS_0 = INIT_BIAS_0
         BIAS_1 = INIT_BIAS_1
         SUM = 0.
```

```
      DO WHILE ((DETECT_FLG.EQ.0).AND.(I.LE.1024))
         SUM = SUM+OUTP(I)

         IF (SUM.LE.BIAS_0) THEN

            DETECT_FLG = 1
            DO J = I-SAMPLE_INTERVAL*SAMPLE_NO+1, I
               DET(J) = 0
            END DO
            PRINT*, DET(I), SAMPLE_NO

         ELSE IF (SUM.GE.BIAS_1) THEN

            DETECT_FLG = 1
            DO K = I-SAMPLE_INTERVAL*SAMPLE_NO+1, I
               DET(K) = 1
            END DO
            PRINT*, DET(I), SAMPLE_NO

         ELSE
C
C        No decision made.
C
            IF (SAMPLE_NO.LT.MAX_SAMPLE_LEN) THEN
C
C        Continue sampling; update thresholds and sample count.
C
               SAMPLE_NO = SAMPLE_NO+1
               BIAS_0 = BIAS_0 + 0.5*LEVEL*RMS
               BIAS_1 = BIAS_1 + 0.5*LEVEL*RMS
            ELSE
C
C        Maximum sample length reached; make a decision based on a
C        Neyman-Pearson test for sample size MAX_SAMPLE_LEN.
C
               DETECT_FLG = 1
               IF (SUM.GE.(3.8*SQRT(FLOAT(MAX_SAMPLE_LEN))*RMS)) THEN
                  DO J = I-SAMPLE_INTERVAL*SAMPLE_NO+1, I
                     DET(J) = 1
                  END DO
               ELSE
                  DO J = I-SAMPLE_INTERVAL*SAMPLE_NO+1, I
                     DET(J) = 0
                  END DO
               END IF
               PRINT*, DET(I)
            END IF

         END IF
C
C        Location of next sample.
C
         I = I+SAMPLE_INTERVAL

      END DO

    END DO

    RETURN
    END
```

```
C*************************************************************************
C
C       SUBROUTINE PLOT
C
C       VAX-11 FORTRAN SOURCE FILENAME:          PLOT.FOR
C
C       DEPARTMENT OF EECE                       KANSAS STATE UNIVERSITY
C
C       AUTHOR:  Damon Mick
C
C       DATE CREATED:  July 1987 (Final version)
C
C*************************************************************************
C
C       CALLING SEQUENCE:  CALL PLOT(Y_DATA, DEV_NUMBER, PLOT_NUMBER,
C       MEAN, SNR)
C
C       PURPOSE:  This routine plots data sequences representing either
C       signal, signal plus noise, matched filter output, or detector
C       output, as chosen in the main program.
C
C       ROUTINE(S) CALLED OR ACCESSED BY THIS ROUTINE:
C
C         GETUTX
C         PAXIS
C         PCLOSP
C         PCLRSC
C         PFIXNM
C         PFLTNM
C         PINIT
C         PLINE
C         PORIG
C         PPLOT
C         PPLOTR
C         PSCALE
C         PSLPEN
C
C       ARGUMENTS REQUIRED:
C
C         Y_DATA            input     real
C                           Array containing the data to be plotted.
C
C         DEV_NUMBER        input     integer
C                           Indicates the plotting device.
C                             1:     7475 Plotter
C                             2:     4014 Display
C
C         PLOT_NUMBER       input     integer
C                           Indicates the data which is being plotted.
C                             1:     Signal alone
C                             2:     Signal plus noise
C                             3:     Matched filter output
C                             4:     Detector output
C
C         MEAN              input     real
C                           Array containing the center positions for each
C                           pulse.
C
C         SNR               input     real
C                           Array containing the signal-to-noise ratios for
```

```
C                              each pulse.
C
C****************************************************************************
      SUBROUTINE PLOT(Y_DATA, DEV_NUMBER, PLOT_NUMBER, MEAN, SNR)

      IMPLICIT NONE

      REAL      X_DATA(1024), Y_DATA(1024), MEAN(*), SNR(*)
      INTEGER   DEV_NUMBER, PLOT_NUMBER, IGOT, I

      REAL      X, Y, FIRSTX, DELTAX, DIVLNX, FIRSTY, DELTAY,
     -          DIVLNY, FIRDEL(4), SCALE

      CHARACTER*20  STR, UNITS
      CHARACTER*8   INARY

      DO I = 1, 1024
         X_DATA(I) = I
      END DO

      IF (DEV_NUMBER.EQ.1) THEN
          DEV_NUMBER = 7475
          SCALE = 1.0
      ELSE
          DEV_NUMBER = 4014
          SCALE = 1.52
      END IF
C
C--  Initialize plotting device; set origin.
C
      CALL PINIT(DEV_NUMBER,' ', SCALE, 'A')
      X = 6.3
      Y = 5.5
      CALL PORIG(X,Y)
C
C--   Scale axes.
C
      CALL PSCALE (X_DATA, 1024, 18.0, FIRSTX, DELTAX, DIVLNX)
      CALL PSCALE (Y_DATA, 1024, 11.0, FIRSTY, DELTAY, DIVLNY)
C
C--   Draw X axis.
C
      CALL PSLPEN(1)
      CALL PAXIS(0.0, 0.0, 'Raman Shift', 'cm(t-1t)', 220, 2201,
     -  18., 0., FIRSTX, DELTAX, DIVLNX)
C
C--   Draw Y axis.
C
      IF (PLOT_NUMBER.NE.4) THEN
         STR = 'Amplitude'
      ELSE
         STR = 'Detector Decision'
      END IF
      CALL PAXIS(0.0, 0.0, STR, ' ', 120, 1201, 11., 90.,
     -  FIRSTY, DELTAY, DIVLNY)
C
C--   Plot data.
C
      FIRDEL(1) = FIRSTX
```

```
      FIRDEL(2) = DELTAX
      FIRDEL(3) = FIRSTY
      FIRDEL(4) = DELTAY

      CALL PSLPEN(2)
      CALL PLINE (X_DATA, Y_DATA, 1024, FIRDEL, 0, ' ', DIVLNX, DIVLNY)
C
C--   Plot the signal-to-noise ratios at their respective pulse position.
C
      IF (PLOT_NUMBER.NE.1) THEN
         DO I = 1, 15
            IF (SNR(I).NE.0.) THEN
C
C--            Calculation of horizontal position.
C
               X = (MEAN(I)/1200.)*18.
C
C--            Select vertical position according to whether pulse number
C--            is even or odd.
C
               IF (2*(I/2).NE.I) THEN
                  Y = 11.7
               ELSE
                  Y = 11.2
               ENDIF

               CALL PPLOT(X, Y, 0)
               CALL PFLTNM(SNR(I), 5, 2)
C
C--            Show position of original pulse.
C
               Y = 1.0
               CALL PPLOT(X, Y, 0)
               CALL PFIXNM(I, 2)
               CALL PPLOT(X, 0.0, 0)
               CALL PPLOTR(0.0, 3.0, 1)
            END IF
         END DO
      ENDIF
C
C--   Wait for carriage return to clear the screen.
C
      CALL GETUTX(1, ' ', 1, INARY, IGOT)
      CALL PCLRSC
      CALL PCLOSP

      RETURN
      END
```

```
C*****************************************************************************
C
C       SUBROUTINE FFT
C
C       VAX-11 FORTRAN SOURCE FILENAME:          FFT.FOR
C
C       DEPARTMENT OF EECE                       KANSAS STATE UNIVERSITY
C
C       REFERENCE:  Nasir Ahmed and T. Natarajan, Discrete-Time Signals and
C       Systems, Reston Publishing Company, 1983.  pp. 160-161.
C
C*****************************************************************************
C
C       CALLING SEQUENCE:  CALL FFT(X, N, INV)
C
C       PURPOSE:  This routine performs a forward or inverse decimation-in-
C       frequency FFT on the sequence X.
C
C       ROUTINE(S) CALLED OR ACCESSED BY THIS ROUTINE:
C         None.
C
C       ARGUMENTS REQUIRED:
C
C         X                  complex
C                            The array containing the sequence on which the FFT
C                            is to be performed; also the array in which the result
C                            is returned.
C
C         N                  integer
C                            The length of the array X.
C
C         INV                integer
C                            Indicates whether a forward (0) or inverse (1) trans-
C                            form is to be performed.
C
C*****************************************************************************

      SUBROUTINE FFT(X, N, INV)
         COMPLEX X(N), W, T
         ITER = 0
         IREM = N
10       IREM = IREM/2
         IF (IREM.EQ.0) GO TO 20
         ITER = ITER+1
         GO TO 10
20       CONTINUE
         S = -1
         IF (INV.EQ.1) S=1
         NXP2=N
         DO 50 IT=1, ITER
            NXP=NXP2
            NXP2=NXP/2
            WPWR = 3.141592/FLOAT(NXP2)
            DO 40 M = 1, NXP2
               ARG = FLOAT(M-1)*WPWR
               W = CMPLX(COS(ARG), S*SIN(ARG))
               DO 40 MXP = NXP, N, NXP
                  J1 = MXP-NXP+M
                  J2 = J1+NXP2
```

```
                  T = X(J1)-X(J2)
                  X(J1) = X(J1)+X(J2)
40                X(J2) = T*W
50      CONTINUE
        N2=N/2
        N1=N-1
        J=1
        DO 65 I=1, N1
           IF (I.GE.J) GO TO 55
           T = X(J)
           X(J) = X(I)
           X(I) = T
55         K = N2
60         IF (K.GE.J) GO TO 65
           J = J-K
           K = K/2
           GO TO 60
65      J = J+K
        IF (INV.EQ.1) GO TO 75
        DO 70 I = 1, N
70      X(I) = X(I)/FLOAT(N)
75      CONTINUE
        RETURN
     END
```

THE DETECTION OF RAMAN SPECTRAL PEAKS BY USE OF MATCHED FILTERING AND WALD (SEQUENTIAL) TESTING

by

DAMON M. MICK

B.S., Kansas State University, 1984

AN ABSTRACT OF A THESIS

submitted in partial fulfillment of the

requirements for the degree

MASTER OF SCIENCE

Department of Electrical and Computer Engineering

KANSAS STATE UNIVERSITY
Manhattan, Kansas

1987

The objective was to develop a method for the automatic detection of signals resembling those encountered in Raman spectroscopy. First a matched filter is used to enhance the spectral peaks. This is followed by a detector, which was developed through the application of statistical decision theory.

Decision theory involves the use of some decision criterion to decide between two or more alternative hypotheses. The Neyman-Pearson criterion, and a modification known as Wald or sequential testing, were chosen for study because they require relatively little knowledge of the signal; in particular the a priori probabilities for the signal do not need to be known. The Wald test, which uses a variable number of samples to make each decision, was finally chosen for experimentation.

For the matched filter the impulse respone matches that of a Lorentzian peak of typical width. In addition to the matched filter and detector, it was found that it was necessary to fit and remove a trend from the output of the filter if the threshold comparison procedure of the detector was to work best.

A program was written to generate signal sequences and perform the filtering and the detection procedure. In the detector there are some parameters that need to be chosen by the observer. These are the theoretical probabilities of false detection and false dismissal, and also what is termed the assumed signal level. Tests were made on several sequences of signals to determine how well the system detects signals, and also to determine what effects the detector parameters have. The results from tests on four sequences are presented. For some of the sequences several tests were made with different sets of parameter values.

The results show that it is possible to detect peaks that are as low as 0.5 or lower in signal-to-noise ratio. However, in some cases peaks with signal-to-noise ratios as high as 1.5 to 2.0 were impossible to detect, mainly because of the effect of the trend removal. Detection at low signal-to-noise ratios requires accepting an increased likelihood of false detections. This is affected primarily by the choice of the theoretical false-detection probability; the choice of the false-dismissal probability and the assumed signal level did not appear to have a consistent effect on the actual probability of detection or the probability of false detection.

In conclusion it is suggested that the system may be a useful addition to the methods of peak enhancement that are now employed, although it has some limitations in that an estimate of the noise statistics must be available, and also the performance may depend on the particular form of the spectra.