Hearing Research, 47 (1990) 103–138
Elsevier

HEARES 01396

Derivation of auditory filter shapes from notched-noise data

Brian R. Glasberg and Brian C.J. Moore

Department of Experimental Psychology, University of Cambridge, Downing Street, Cambridge, U.K.

(Received 14 September 1989; accepted 27 February 1990)

A well established method for estimating the shape of the auditory filter is based on the measurement of the threshold of a sinusoidal signal in a notched-noise masker, as a function of notch width. To measure the asymmetry of the filter, the notch has to be placed both symmetrically and asymmetrically about the signal frequency. In previous work several simplifying assumptions and approximations were made in deriving auditory filter shapes from the data. In this paper we describe modifications to the fitting procedure which allow more accurate derivations. These include: 1) taking into account changes in filter bandwidth with centre frequency when allowing for the effects of off-frequency listening; 2) correcting for the non-flat frequency response of the earphone; 3) correcting for the transmission characteristics of the outer and middle ear; 4) limiting the amount by which the centre frequency of the filter can shift in order to maximise the signal-to-masker ratio. In many cases, these modifications result in only small changes to the derived filter shape. However, at very high and very low centre frequencies and for hearing-impaired subjects the differences can be substantial. It is also shown that filter shapes derived from data where the notch is always placed symmetrically about the signal frequency can be seriously in error when the underlying filter is markedly asymmetric. New formulae are suggested describing the variation of the auditory filter with frequency and level. The implications of the results for the calculation of excitation patterns are discussed and a modified procedure is proposed. The appendix lists FORTRAN computer programs for deriving auditory filter shapes from notched-noise data and for calculating excitation patterns. The first program can readily be modified so as to derive auditory filter shapes from data obtained with other types of maskers, such as rippled noise.

Frequency selectivity; Auditory filter; Masking; Excitation pattern; Power-spectrum model

Introduction

The peripheral auditory system is usually conceived as containing a bank of filters, the auditory filters, which are often assumed to be approximately linear. The shape of the auditory filter at a given centre frequency can be thought of as a weighting function which is applied to the power spectrum of a sound to determine the effective magnitude of the output of the filter. Methods for deriving the shape of the auditory filter are generally based on the assumptions of the power-spectrum model of masking, which states that the power of a signal at masked threshold, P_s, is given by:

$$P_s = K\int_{-\infty}^{\infty} N(f)W(f)df, \quad (1)$$

where f is frequency, N(f) represents the long-term power spectrum of the masker and W(f) is the weighting function describing the shape of the auditory filter. K is a constant for a given subject, type of masker and centre frequency; it equals the signal-to-masker ratio at the output of the filter required to achieve threshold. The assumptions of the model, and the conditions under which they are valid, have been reviewed elsewhere (Patterson and Moore, 1986; Moore and Glasberg, 1987).

In this paper we are concerned with a method of estimating the auditory filter shape which has been used quite widely, namely, the measurement of the threshold for a sinusoidal signal in a notched-noise masker (e.g. Patterson, 1976; Weber, 1977; Patterson and Nimmo-Smith, 1980; Moore and Glasberg, 1983; Glasberg and Moore, 1986; Dubno and Dirks, 1989; Rosen and Stock, 1989). The signal frequency is fixed, and the signal threshold is measured as a function of notch width. To estimate the asymmetry of the auditory filter, conditions are included in which the notch is

Correspondence to: Brian R. Glasberg, Department of Experimental Psychology, University of Cambridge, Downing Street, Cambridge CB2 3EB, U.K.

positioned both symmetrically and asymmetrically about the signal frequency. In practice, the noise is always composed of two bandpass noises, one above and one below the signal frequency.

The fitting procedure assumes that the auditory filter shape can be approximated by a simple mathematical expression with a small number of free parameters. For the purposes of this paper, each side of the auditory filter was assumed to have the form of the roex (p, r) filter described by Patterson et al. (1982):

$$W(g) = (1 - r)(1 + pg)\exp(-pg) + r, \qquad (2)$$

where g is the normalised deviation from the centre of the filter (deviation from centre frequency divided by centre frequency), p is a parameter determining the slope of the filter skirts and r is a parameter which flattens the filter at frequencies remote from the centre frequency, thereby placing a dynamic range limitation on the filter. The value of p was allowed to differ for the upper and lower halves of the auditory filter; the upper and lower p values are called p_u and p_l, respectively. The value of r was assumed to be the same for the two sides of the filter. The equivalent rectangular bandwidth (ERB) of this filter, when r is small and when p_u and p_l are > 3 (as is normally the case) is approximately equal to $2f_c/p_u + 2f_c/p_l$, where f_c is the centre frequency.

The fitting procedure works in the following way. Eqn. 1 is rewritten in terms of the variable g, and the expression for W is substituted. The integral can then be solved analytically; for details see Patterson et al. (1982), Glasberg et al. (1984) and Patterson and Moore (1986). The integral is evaluated over limits corresponding to the edges of the noise bands (Glasberg and Moore, 1986); these limits must be shifted appropriately for each condition (each notch width). Starting values of p_u and p_l and r are assumed, and the equation is used to predict the threshold for each condition, for notches placed both symmetrically and asymmetrically about the signal frequency.

It is assumed that, for each notch width, the observer makes use of the single auditory filter giving the highest signal-to-masker ratio. This filter is not always centred exactly at the signal frequency, i.e., it is assumed that off-frequency listening occurs. The process of choosing the optimum centre frequency is sometimes described in terms of shifting a single filter, and sometimes as selecting one filter from an array of filters. The latter is probably a more realistic characterisation, but the two are equivalent as far as the fitting procedure is concerned and, for convenience we will talk of a single filter shifting. The filter selected will often have a different centre frequency for each notch width. However, since the shift in centre frequency is generally small (as discussed in more detail later), it is usually assumed that the shape of the filter does not vary significantly when it is shifted.

A least-squares minimisation procedure is used to find the values of p_u, p_l and r which minimise the mean-squared deviation between the data and the fitted values. The minimisation is done with the thresholds expressed in decibels. The values of p_u, p_l and r obtained in this way define the shape of the derived filter.

In previous work deriving filter shapes with this method a number of simplifying assumptions and approximations were made. In this paper we describe several modifications to the fitting procedure which avoid these simplifications and allow more accurate derivations. Illustrations are given of the effect of each modification on the derived filter shapes. Later in the paper we discuss the implications of the results for filter shapes published in the past and for the derivation of excitation patterns. Finally, the Appendix gives listings of two FORTRAN computer programs, one for deriving auditory filter shapes from notched-noise data using the modified procedure and the other for calculating excitation patterns.

Modifications to the fitting procedure

Allowing for the variation of filter bandwidth with centre frequency

As mentioned earlier, in previous work (both our own and, as far as we are aware, that of others) the filter shape and bandwidth were assumed not to vary when the centre frequency of the filter was shifted so as to maximise the signal-to-masker ratio. This is a reasonable assumption when the shift is small. However, when the filters are markedly asymmetric (which occurs at high

sound levels) or when the notch is positioned very asymmetrically about the signal frequency, the shift can sometimes be rather large (greater than 20% of the centre frequency). In this case, it is more realistic to assume that the bandwidth becomes smaller when the filter shifts down in centre frequency, and larger when it shifts up in centre frequency.

In order to make quantitative allowance for this effect, it is necessary to make some assumptions about how the filter bandwidth varies with centre frequency. For medium centre frequencies (1–5 kHz), there is general agreement on the form of this function (Zwicker, 1961; Scharf, 1970; Moore and Glasberg, 1983), but at very low and very high centre frequencies there is considerable uncertainty. This leads to a circular problem: to derive filter shapes at low or high centre frequencies it is necessary to make an assumption about how the bandwidth varies with centre frequency; however, the function relating bandwidth to centre frequency cannot be determined unless the filter shapes have been derived.

In practice the problem can be overcome by an iterative process. An initial assumption is made about the form of the function relating bandwidth to centre frequency; we will refer to this function as the ERB function. The ERB function is used in the derivation of filter shapes and bandwidths; it is assumed that the values of p_u and p_l vary with frequency as 1/ERB. The new bandwidth values are then used to modify the ERB function, and the whole process is repeated until the bandwidth values no longer change.

We have applied this method using several sets of data, including the data summarised in Moore and Glasberg (1983), some recent data of Dubno and Dirks (1989) at medium centre frequencies, the data of Moore et al. (1990) at low centre frequencies (0.1–0.8 kHz) and the data of Shailer et al. (1990) at high centre frequencies (8 and 10 kHz). For the data at high centre frequencies, the derived filters were hardly affected by the form of the ERB function, since the filters were sharply tuned (relative to the centre frequency), and the shifts associated with off-frequency listening were very small. For the low-frequency data, the effects of the assumed ERB function were larger. The ERB function on which the process converged was essentially the same as the function suggested by Moore and Glasberg (1983) for frequencies up to 5 kHz, but the ERB values at 8 and 10 kHz were

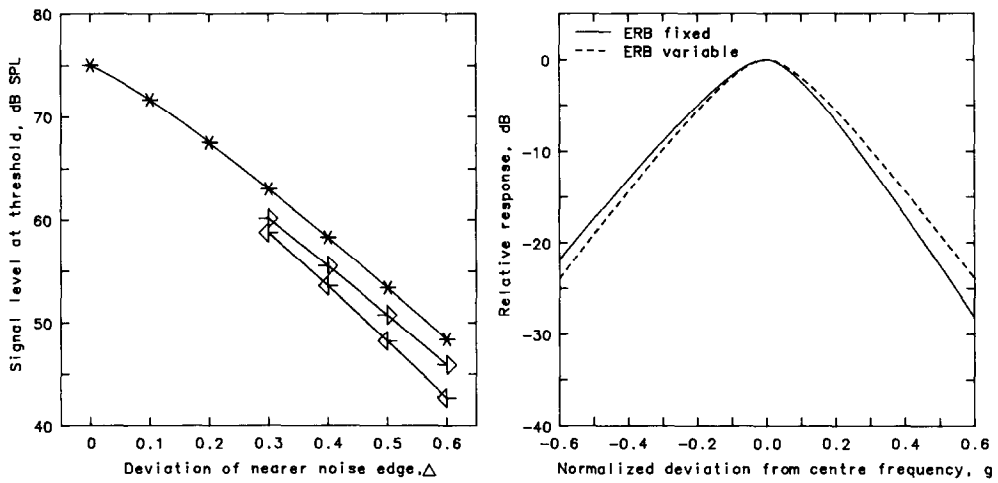

Fig. 1. The panel on the left shows 'ideal thresholds' obtained using a simulated symmetric auditory filter. Thresholds are plotted as a function of the deviation of the nearer edge of the notch in the noise from the centre frequency, f_c, divided by f_c; this will be denoted by Δ. The asterisks indicate conditions where the notch was symmetrically placed about f_c. Right-pointing arrows indicate conditions where the upper edge of the notch was 0.2 units farther from f_c than the lower edge. Left-pointing arrows indicate mirror-image conditions. The right panel shows auditory filters derived from the data assuming that the ERB of the filter does not vary when its centre frequency shifts (solid line) or that it does vary (dashed line).

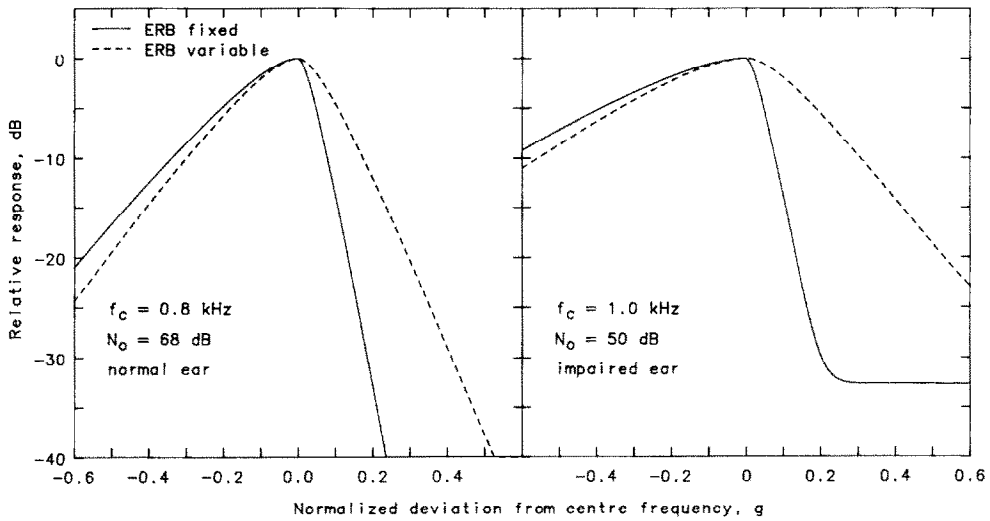

Fig. 2. Auditory filters derived assuming that the ERB of the filter does not vary when its centre frequency shifts (solid line) or that it does vary (dashed line). For the left panel, the data were obtained using a normally hearing subject, a 0.8-kHz signal, and a noise spectrum level (N_0) of 68 dB. For the right panel, the subject had a moderate, flat cochlear hearing loss; $f_c = 1.0$ kHz and $N_0 = 50$ dB.

smaller than predicted by that function. Later in this paper we suggest a modified ERB function which gives a good fit to the data over the whole frequency range tested (0.1–10 kHz). That modified function was used in the fitting procedure given in the Appendix, and it was also used in the examples that follow.

Fig. 1 illustrates the effects of changing the fitting procedure to allow for changes in bandwidth when the filter shifts. In the left panel are shown some 'ideal' data, derived from a simulated auditory filter. The asterisks indicate 'thresholds' when the notch was placed symmetrically about f_c. Right-pointing arrows indicate conditions where the upper edge of the notch was 0.2 units farther from f_c than the lower edge. Left-pointing arrows indicate conditions where the lower edge of the notch was 0.2 units farther from f_c than the upper edge. The left-pointing arrows lie below the right pointing arrows. Previously, this would have been taken as an indication that the filter was asymmetric with a shallower lower skirt. In fact, the filter used to produce these 'ideal' data was perfectly symmetric. The lines in the left panel show the thresholds fitted to the data using the modified procedure. The right panel shows the filters derived from the data using the 'old' procedure (assuming that the bandwidth does not change when the filter shifts: solid line) and the modified procedure (dashed line). The filter derived using the old method is asymmetric, while that derived with the new method is symmetric; the latter is the correct characterisation of the filter used to generate the data.

Fig. 2 shows two examples of filter shapes derived from real data. For the left panel, the subject had normal hearing, the centre frequency was 0.8 kHz and the noise spectrum level was 68 dB (data from Moore et al., 1990). The filter derived with the old procedure (solid line) is highly asymmetric, with a very steep upper skirt. The filter derived with the new procedure (dashed line) is asymmetric, but less so, and the upper skirt is not nearly so sharp. The right panel shows a more extreme example, for a subject with a moderate, flat, cochlear hearing loss, a centre frequency of 1 kHz and a noise spectrum level of 50 dB (data from Glasberg and Moore, 1986). The filter derived with the old procedure has a shallow lower skirt and a very steep upper skirt, steeper indeed, than would be found for normally hearing subjects. This was troublesome, since it seemed implausible that a cochlear hearing impairment would lead to a steeper-than-normal slope. The

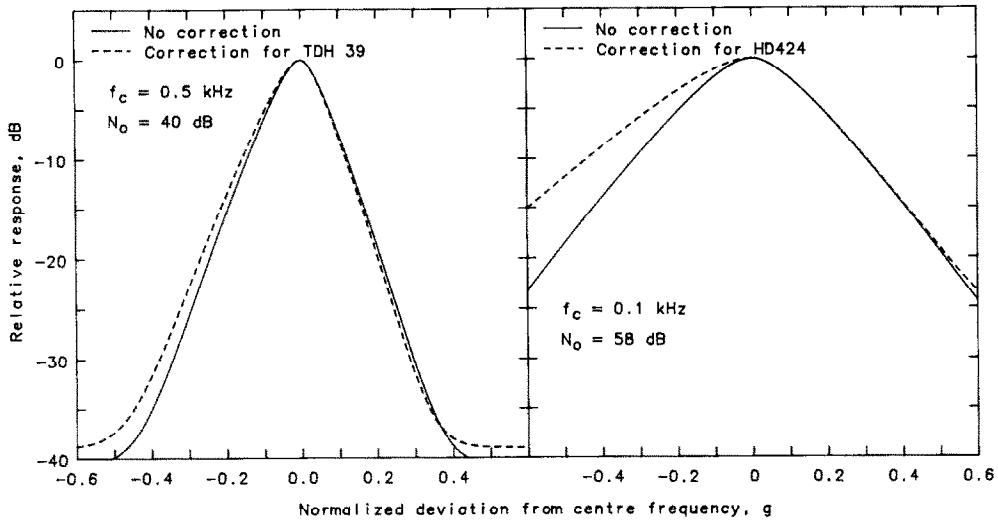

Fig. 3. Effects on the derived auditory filter shapes of correcting for the non-flat frequency response of the earphones used. Filters derived without any correction are shown by solid lines, and those with corrections by dashed lines. All data were obtained using normally hearing subjects. For the left panel, the data were obtained using a TDH 39 earphone; $f_c = 0.5$ kHz and $N_0 = 40$ dB. For the right panel, the data were obtained using a Sennheiser HD424 earphone; $f_c = 0.1$ kHz and $N_0 = 58$ dB.

filter derived with the new method is much less asymmetric, and the upper skirt is shallower than normal, as would be expected.

The two examples given above represent rather extreme cases. When the filter is reasonably symmetric, as it usually is at moderate sound levels, then the differences between the old and new procedures are smaller. It should be noted that when the filter is highly asymmetric, as in the examples given in Fig. 2, then the steeper skirt is not very well defined by the notched-noise data. As a rule of thumb, when the p value for one side of the filter is more than twice the p value for the other side, then the greater p value is poorly defined.

Allowing for changes in noise spectrum occurring before auditory filtering

When the spectrum of the notched noise is flat within its passbands, it may be treated as a constant in Eqn. 1, and the signal threshold can be predicted by integrating the equation for the filter shape (Eqn. 2) over the frequency range covered by the noise. This has formed the basis of earlier methods for estimating the auditory filter shape, all of which made use of an analytic integral (e.g. Patterson, 1974; Patterson et al., 1982; Glasberg and Moore, 1986). Unfortunately, a flat spectrum of the electrical signal delivered to the earphone or loudspeaker does not guarantee a flat spectrum at the inputs of the auditory filters.

The auditory filters presumably have their basis in the cochlea, and/or in some higher part of the auditory system. The cochlea is the first stage in the auditory system where spectral analysis takes place. However, the spectra of stimuli reaching the cochlea will differ from the spectra of the electrical signals applied to the acoustic transducers, for two reasons: firstly, the transducers used always have imperfect frequency responses; secondly, the efficiency of sound transmission through the outer and middle ear varies markedly with frequency. Theoretically, at least, it seems sensible to think of the stimuli as being passed through a fixed, broad filter prior to analysis in the auditory filters. If we wish to derive the shapes of the auditory filters, then we need to take into account this prior, fixed filtering. * To predict the signal threshold, it is necessary to determine the transfer function of the

* If one's goal was to characterise the frequency selectivity of the whole auditory system, then one might want to consider the spectral changes produced by the outer and middle ear as part of the overall filtering process. However, this could give a misleading indication of the extent to which the frequency components of a complex sound were resolved in the auditory system.

initial broad filter, to calculate from it the spectra of the stimuli at the input to the cochlea for each condition, and then to evaluate the integral in Eqn. 1 by numerical methods; this modification to the fitting procedure is used in the remainder of this paper.

We start by considering the effects of the imperfect frequency responses of the transducers used. Two examples of the effect of this are given in Fig. 3. The left panel shows auditory filters derived from notched-noise data for a centre frequency of 0.5 kHz and a noise spectrum level of 40 dB (previously unpublished data supplied by R.D. Patterson). The solid line shows the filter shape derived when no correction was made for the non-uniform frequency response of the TDH 39 earphone used. The dashed line shows the filter shape obtained with a correction for the frequency response of the earphone; the correction was based on the response of the free-field equaliser for the TDH 39 earphone proposed by Fastl and Zwicker (1983). The effect of the correction in this case is relatively small, the main change being to make the low-frequency side of the filter slightly less steep. Similar small changes were found for other data sets at this centre frequency. The changes were also small for other centre frequencies up to 4 kHz. We may conclude that filter shapes derived from data obtained with TDH 39 earphones over the frequency range 0.5 to 4 kHz are only slightly affected by the non-flat frequency response of those earphones.

The right panel gives an example for a very low centre frequency, 0.1 kHz, where the effect of the correction is somewhat larger. The noise spectrum level was 58 dB (data from Moore et al., 1990). Again, the solid line shows the filter shape derived without any correction, and the dashed line shows the filter shape obtained after correcting for the fact that the response of the Sennheiser HD 424 earphone fell gradually below 100 Hz (Moore et al., 1990). The correction has the effect of making the low-frequency slope considerably less steep. As a result, the ERB of the derived filter increases from 31 to 38 Hz. In general, the effects of correcting for the frequency response of the earphone are largest in frequency regions where the slope of that frequency response approaches the slope of the shallower side of the auditory filter. This applies for normal subjects only at very low centre frequencies, but it may be true at other frequencies for hearing-impaired subjects, since such subjects tend to have broader filters than normal. It should be noted that some experimenters working at very low centre frequencies have pre-filtered the electrical stimuli so as to flatten the acoustic responses as measured in the ear canal (e.g. Rosen and Stock, 1989).

We consider next how to deal with the non-flat transmission characteristics of the outer and middle ear. Unfortunately, these characteristics are not well known in humans, so it is necessary to estimate them in an indirect way. One possibility is based on the idea that, at least in young, normally hearing listeners, the transducers within the cochlea are equally sensitive at all audible frequencies and the variation of absolute threshold with frequency reflects a frequency-dependent attenuation resulting from the transfer function of the outer and middle ear (see Dallos, 1973, and Pickles, 1988, for reviews). According to Dallos (1973), part of the sensitivity loss at low frequencies can be attributed to changes in the cochlear input impedance resulting from the action of the helicotrema (see also, Nedzelnitsky, 1980, and Lynch et al., 1982). However, this can still be regarded as a frequency-dependent attenuation applied before auditory filtering. Thus the shape of the absolute threshold curve can be used to 'correct' for the transmission characteristics of the outer and middle ear.

In this paper, we use two 'corrections' of this type. One is based on the shape of the minimum audible field (MAF) curve published in ISO recommendation R. 226; this correction, the MAF correction, would be applicable to stimuli delivered in a free field or using earphones with a free-field response (or a free-field equaliser; Fastl and Zwicker, 1983). A second correction is based on minimum audible pressures (MAP) at the eardrum as estimated by Killion (1978) for frequencies up to 8 kHz, and by Green et al. (1987) and Shailer et al. (1990) for frequencies from 8 to 16 kHz. This correction, the MAP correction, would be applicable to sound systems designed to produce a flat frequency response at the eardrum, such as the Etymotic Research ER2 insert earphone (Killion, 1984).

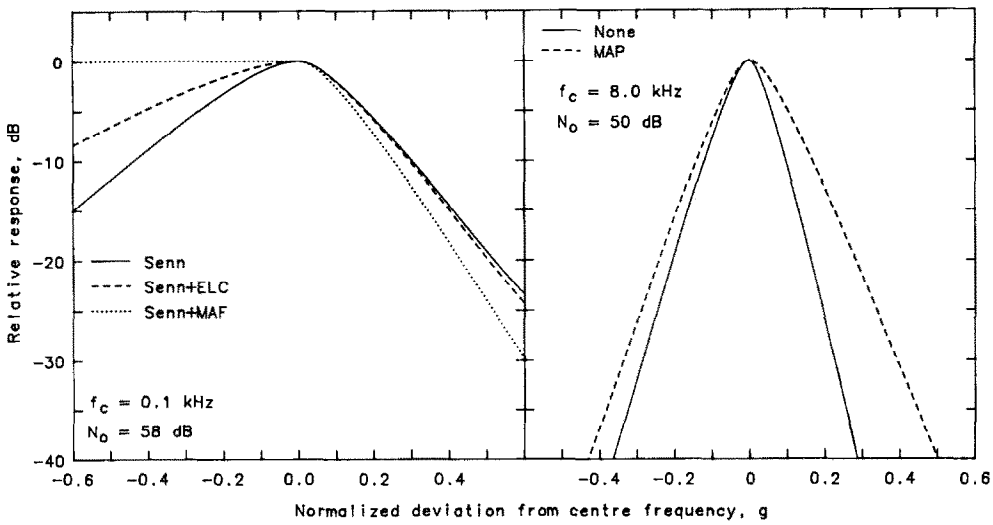

Fig. 4. Like Fig. 3, but showing the effect of applying different corrections for the transmission characteristics of the outer and middle ear. See text for details. For the left panel, $f_c = 0.1$ kHz and $N_0 = 58$ dB. For the right panel, $f_c = 8.0$ kHz and $N_0 = 50$ dB.

It seems likely, however, that part of the variation of absolute threshold with frequency arises in other ways. For frequencies below 1 kHz, the behavioural absolute threshold appears to change more rapidly with frequency than would be predicted from the transfer function of the outer and middle ear, both in cat (Lynch et al., 1982) and in humans (Zwislocki, 1975). A possible explanation is that internal noise in the cochlea is greater at low frequencies than at high (Soderquist and Lindsey, 1972; Nedzelnitsky, 1980). This would raise absolute thresholds at low frequencies, but would have little effect on the perception of low frequencies at high sound levels. This could explain why equal-loudness contours at high sound levels are flatter than the absolute threshold curve. The equal-loudness contours would be almost unaffected by the internal noise, and so their shape would largely reflect the transfer characteristic of the outer and middle ear. For low frequencies, then, it would seem more appropriate to use a correction based on the equal-loudness contours at high levels. In this paper we use a correction, the ELC correction, based on the 100-phon equal-loudness contour as specified in ISO recommendation R. 226. It should be noted that, for centre frequencies of 1 kHz and above, the 100-phon equal-loudness contour and the MAF curve are essentially parallel. Thus the MAF correction and the ELC correction become equivalent.

In summary, the fitting procedure was modified to allow filter shapes to be derived for cases where the noise spectra were subject to a frequency-dependent attenuation before reaching the auditory filter. Three different types of correction have been evaluated: MAF, MAP and ELC. Each may be appropriate in different situations. In all cases, an additional correction for the frequency response of the sound delivery system was included when appropriate. Note that when these corrections are applied, the signal level is also corrected by the appropriate amount.

Examples of the effects of these corrections are given in Fig. 4. The left panel shows auditory filters derived from the data of a normally hearing subject for a centre frequency of 0.1 kHz and a noise spectrum level of 58 dB (data from Moore et al., 1990). The solid curve (Senn) shows the filter derived when the only correction applied was for the frequency response of the Sennheiser HD424 earphone. The dashed curve shows the filter obtained with the ELC correction. The filter is less steep on the low-frequency side in this case. The dotted curve shows the filter obtained with the MAF correction. The filter is now completely flat on the low-frequency side, but is a little steeper on

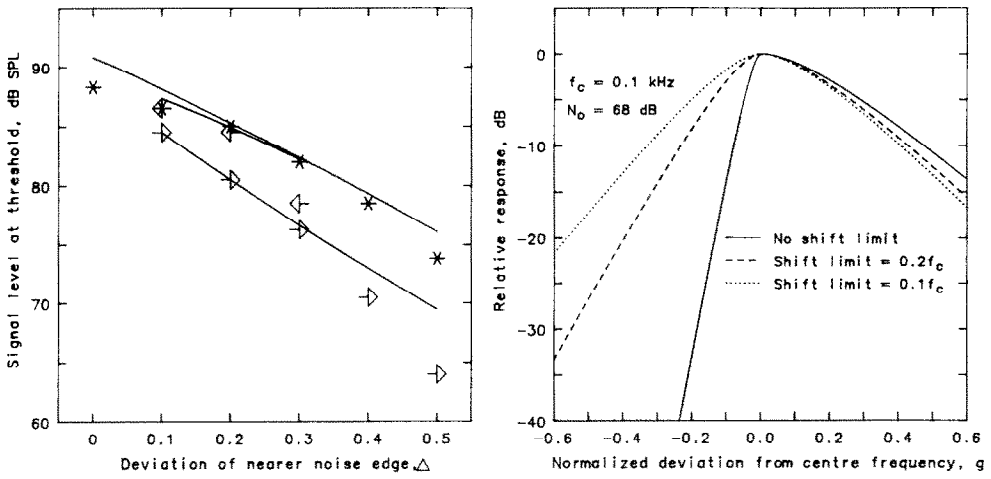

Fig. 5. The left panel shows data obtained from a normally hearing subject with $f_c = 0.1$ kHz and $N_0 = 68$ dB. The format is the same as for Fig. 1. The right panel shows auditory filter shapes derived with different limits placed on the amount by which the centre frequency of the filter was allowed to shift in the fitting procedure.

the high-frequency side. Such completely flat filters only occurred for a few subjects, and only when the MAF correction was used. When the ELC correction was used, the results were generally less variable across subjects.

The right panel of Fig. 4 shows filter shapes derived from the mean data of three subjects for a centre frequency of 8 kHz and a noise spectrum level of 50 dB (data from Shailer et al., 1990). The Etymotic Research ER2 insert earphone used is intended to produce a flat frequency response at the eardrum. The solid curve shows the filter derived with no correction applied. The dashed curve shows the filter derived using the MAP correction. The latter filter is markedly broader, particularly on the high-frequency side. The same pattern was observed for the data of individual subjects.

In summary, it appears reasonable to correct for the frequency-dependent attenuation which occurs as a result of the transmission of sound through the outer and middle ear to the cochlea. At low centre frequencies, the most appropriate estimate of the required attenuation function appears to be the ELC correction. The effect of applying this correction is to broaden somewhat the low-frequency skirt of the derived filter. At medium to high centre frequencies, either the MAF or the MAP correction should be used, depending on how the stimuli are generated and specified. Application of the MAP correction at high frequencies results in a reduction of the sharpness of the high-frequency skirt of the derived filter. At medium frequencies the corrections have relatively little effect.

Restricting the extent of off-frequency listening

As mentioned earlier, for each notch width the fitting procedure shifts the filter so as to find the centre frequency giving the highest signal-to-masker ratio. Usually, the shifts are small; expressed as a proportion of the centre frequency, they rarely exceed 0.2 and are usually less than 0.1. Occasionally, however, it is possible to find two almost equally good fits to the data, one involving small degrees of shift, and the other involving much larger ones. In such cases, the fit involving the large shifts often gives filters which appear unusual in some way. This suggests that extreme degrees of off-frequency listening do not in fact occur, and the procedure is wrong in assuming that they do.

An example is given in Fig. 5. The data in the left panel were obtained from a normally hearing subject at a centre frequency of 0.1 kHz and a noise spectrum level of 68 dB (data from Moore et al., 1990). The filter derived from these data without any restriction on the degree of off-frequency listening is shown as the solid line in the right panel (this filter was derived using the

ELC correction described above). The lower slope is much steeper than the upper slope. However, the data for six other normally hearing subjects gave filters which were roughly symmetric or had steeper high-frequency skirts than low-frequency skirts. Inspection of the output of the fitting procedure showed that for the first subject extreme degrees of off-frequency listening had been assumed in fitting the data; shifts up to $-0.43f_c$ were used (i.e., the filter centre frequency was assumed to be shifted to 57 Hz). For the other subjects, the shifts were generally less than $0.2f_c$. Thus, the unusual filter shape found for this subject may have been a consequence of assuming too great a degree of off-frequency listening in the fitting procedure. The dashed and dotted lines show the filters obtained when the shift was limited to maximum values of $0.2f_c$ and $0.1f_c$. The filter becomes more nearly symmetrical as the shift limit is decreased and its shape becomes more like that for the other subjects.

The lines in the left panel are the thresholds fitted to the data with a shift limit of $0.1f_c$. Clearly, the fit is not perfect, but it is reasonable given the fact that all behavioural data have an inherent variability. The root-mean-square deviation of the data from the fitted values was 1.28 dB without any restriction on the degree of shifting, and 1.36 and 1.35 dB with the shift limited to $0.2f_c$ and $0.1f_c$, respectively. These differences are rather small, indicating that the goodness of fit is not greatly affected by restricting the degree of off-frequency listening. We have found that this is generally the case.

We feel that the filter shown by the dotted line is likely to be a more accurate representation of the auditory filter of this subject. It seems unlikely that shifts in centre frequency as large as $-0.43f_c$ would be used. At low centre frequencies, the efficiency of the detection process following the auditory filter decreases markedly with decreasing frequency (Moore et al., 1990). This would offset any advantages which might be gained from large downward shifts of the filter. It seems reasonable therefore, to incorporate a shift limit in the fitting procedure.

It is not entirely clear what the value of the limit should be, but a limit between $0.1f_c$ and $0.2f_c$ seems reasonable. For most data sets a limit of 0.2 has no effect, since shifts larger than 0.2 would not occur anyway. We analysed 189 sets of notched-noise data for centre frequencies from 0.2 kHz to 4.0 kHz without using any shift limit; only seven cases were found where the shift exceeded $0.2f_c$, and these were all for hearing-impaired subjects with abnormally broad filters. The large shifts were always negative (i.e. towards lower centre frequencies). The derived filters in these cases tended to be very asymmetric, with steep lower skirts, often steeper than would be expected for normally hearing subjects. Restricting the shift to $0.1f_c$ or $0.2f_c$ had the effect of reducing the slopes of the low-frequency sides of the filters, without significantly changing the goodness of fit; the 0.1 limit reduced the slopes more than the 0.2 limit, but the differences were not large. With the shift limits, the derived filters were never sharper than would be found for normally hearing subjects at the same centre frequencies and noise levels.

In summary, placing a limit on the amount by which the centre frequency of the filter is allowed to shift in the fitting procedure has little effect on the derived filter shape in the great majority of cases, but it has the advantage of avoiding anomalous fits, particularly at low centre frequencies and for hearing-impaired subjects. A shift limit between $0.1f_c$ and $0.2f_c$ seems reasonable.

Derivation of filter shapes from data using only symmetric notches

In Patterson's (1976) first study deriving auditory filter shapes from notched-noise data, he used notches which were symmetrically placed about the signal frequency. He presented evidence that, at the moderate noise level he used, the auditory filter was roughly symmetric on a linear frequency scale. Therefore, it was reasonable to assume that off-frequency listening could be neglected, and that the subject always used a filter centred close to the signal frequency. Since that time, several other studies have been published which used only symmetric notched noises. However, these studies have not always been restricted to normal listeners, or to moderate noise levels (e.g. Dubno and Dirks, 1989; Rosen and Stock, 1989). For impaired listeners, or at high noise levels, the auditory filter

may be markedly asymmetric. An example of the effect of such asymmetry is given in Fig. 6.

The left panel shows data obtained from a normally hearing subject for a signal frequency of 0.1 kHz and a noise spectrum level of 58 dB (data from Moore et al., 1990). The filter derived from the data, including thresholds for asymmetric notches and using the ELC correction, is shown as the solid line in the right panel of Fig. 6. It is markedly asymmetric, with a shallow lower branch. The ERB is 28 Hz. The dashed curve shows the filter shape obtained when the data were restricted to conditions where the notch was symmetrically placed around the signal frequency, and the filter was assumed to remain centred at the signal frequency. No correction was applied to allow for any frequency dependent attenuation arising before the auditory filter. This is the way that symmetric notched-noise data have traditionally been analysed. In this case the derived filter is forced to be symmetric; the steepness of the lower branch is considerably overestimated. The ERB is 21 Hz. In general, we have found that the traditional method for analysing symmetric notched-noise data can lead to filters which differ from those derived from more complete data sets whenever the latter are moderately asymmetric. At medium to high centre frequencies, the slope of the shallower branch is slightly overestimated and the slope of the steeper branch is considerably underestimated, by up to a factor of two.

Even when the only data available are those for symmetric notched noises, it is possible to apply corrections such as the ELC or MAP correction, and to allow the filter to be asymmetric. The dotted line in Fig. 6 shows the filter shape derived from the symmetric conditions with our standard procedure (as listed in the Appendix); the ELC correction was used and a shift limit of $0.15f_c$ was applied. The derived filter shape is actually quite similar to that obtained with the full data set and shows a similar degree of asymmetry. The asymmetry arises because, after the correction has been applied, the noise spectrum level is different for frequencies which are equal amounts above and below f_c. We recommend strongly, however, against trying to derive the asymmetry of the auditory filter from symmetric notched-noise data. Although it can sometimes work reasonably well (as in our example), the shape is usually not well defined by the data, and it is possible for the fitting procedure to give filter shapes which differ considerably from those which would be obtained with a more complete data set.

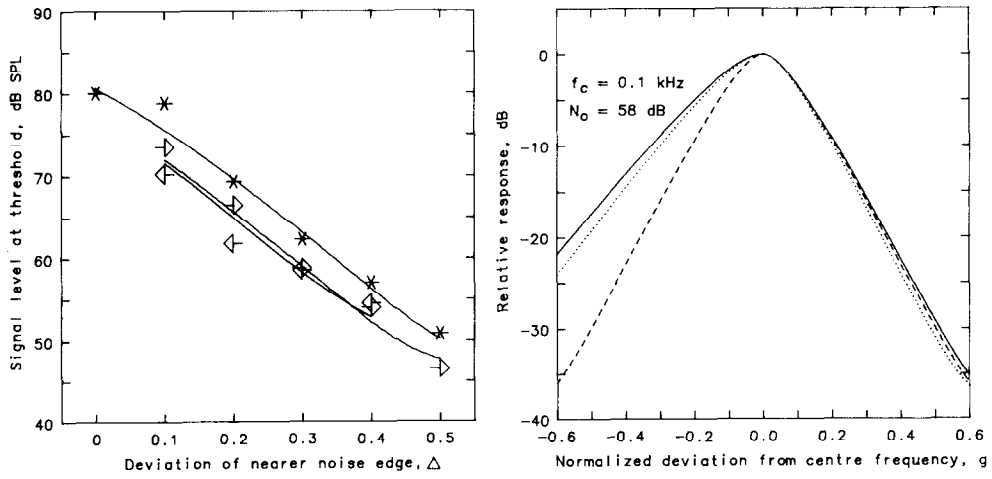

Fig. 6. The left panel shows data obtained from a normally hearing subject with $f_c = 2$ kHz and $N_0 = 50$ dB. The format is the same as for Fig. 1. The right panel shows auditory filter shapes derived from the complete data set (solid line) and with the data restricted to conditions where the notch was symmetrically placed about f_c (dashed and dotted lines). For the dashed line, no correction was applied, the filter was assumed to be symmetric, and no off-frequency listening was assumed. For the dotted line the standard fitting procedure was used.

A revised equation describing the variation of the ERB with centre frequency

In 1983 we published a summary of studies estimating the shape of the auditory filter using the notched-noise method (Moore and Glasberg, 1983). All of the data were obtained from normally hearing subjects using notches placed symmetrically about the signal frequency. Thus, these studies provided no information about the asymmetry of the auditory filter. However, the data were mostly obtained at moderate sound levels, and, as mentioned above, other data show that the auditory filter is roughly symmetric at these levels (Patterson, 1976; Moore and Glasberg, 1987). The ERBs of the filters derived from these data are shown as asterisks in Fig. 7. The dashed line shows the equation fitted to the data in 1983. The open squares show ERBs obtained in a more recent study by Dubno and Dirks (1989) also using symmetric notched-noise maskers and a moderate noise level; the data were fitted with the roex (p, r) model, as were the earlier data. The ERBs obtained by Dubno and Dirks correspond very well with the ERBs obtained earlier and with the 1983 equation.

The open circles in Fig. 7 show ERBs derived from the data of Moore et al. (1990), which include thresholds for notches placed asymmetrically about the signal frequency. The filters were derived using the modifications described above, with the ELC correction and a shift limit of $0.2f_c$. The noise spectrum levels used were 58, 55, 52 and 49 dB for centre frequencies of 0.1, 0.2, 0.4 and 0.8 kHz, respectively. The ERBs for the two lowest centre frequencies lie very close to the values found earlier. However, the ERBs at 400 and 800 Hz are somewhat above the earlier values. This can be attributed to the relatively high noise levels; the bandwidth of the auditory filter tends to increase with increasing level (Weber, 1977; Moore and Glasberg, 1987), and the data summarised in Moore and Glasberg (1983) were all obtained at moderate noise levels.

The solid circles show ERBs estimated by Shailer et al. (1990) using the MAP correction and a shift limit of $0.2f_c$. The noise spectrum level was 35 dB at 8 kHz and 50 dB at 10 kHz. These ERBs

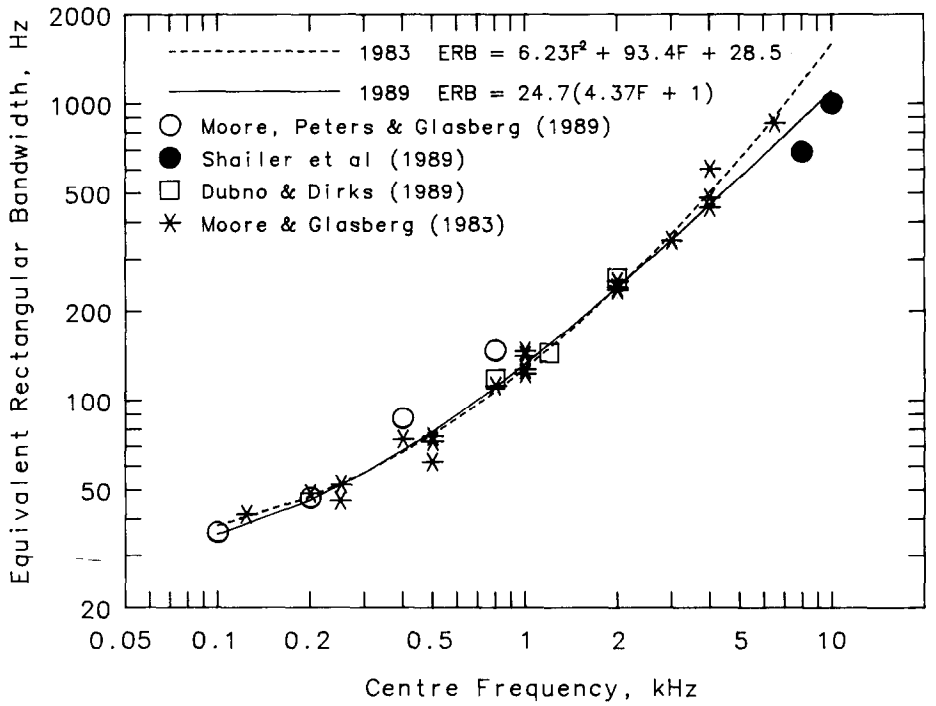

Fig. 7. Summary of data relating the ERB of the auditory filter to centre frequency.

fall somewhat below the values given by the 1983 equation, although that equation was only meant to be applicable for centre frequencies up to 6.5 kHz. The solid line in Fig. 7 provides a good fit to the ERB values over the whole frequency range tested. It is described by the following equation:

$$\text{ERB} = 24.7(4.37\text{F} + 1), \quad (3)$$

where F is frequency in kHz. This equation is a modification of one originally suggested by Greenwood (1961) to describe the variation of the critical bandwidth with centre frequency. He based it on the assumption that each critical bandwidth corresponds to a constant distance along the basilar membrane. Although the constants in Eqn. 3 differ from those given by Greenwood, the form of the equation is the same as his. This is consistent with the idea that each ERB corresponds to a constant distance along the basilar membrane (Moore, 1986).

It is sometimes useful to scale frequency in terms of units of the ERB, giving a scale comparable to the Bark scale, which is based on the 'classical' critical bandwidth values (Zwicker and Terhardt, 1980). An equation relating number of ERBs to frequency may be obtained by integrating the reciprocal of the ERB function. The result is:

$$\text{Number of ERBs, E} = [1000/(24.7 \times 4.37)] \log_e(4.37\text{F} + 1)$$

$$= 21.4 \log_{10}(4.37\text{F} + 1), \quad (4)$$

where the constant of integration has been chosen so as to make E = 0 when F = 0. Again, F is in kHz.

The variation of the auditory filter shape with level

Moore and Glasberg (1987) presented a summary of measurements of the auditory filter shape using maskers with notches placed asymmetrically about the signal frequency. They concluded that the lower skirt of the filter becomes less sharp with increasing level, while the higher skirt becomes slightly steeper. We have re-analysed the data from the studies summarised in that paper, but using the modified fitting procedure, and we have also examined the data presented in Moore et al. (1989) and Shailer et al. (1989). The reanalysis leads to the following conclusions:

1) The auditory filter for a centre frequency of 1 kHz is roughly symmetric on a linear frequency scale when the level of the noise is approximately 51 dB/ERB. This corresponds to a noise spectrum level of about 30 dB. The auditory filters at other centre frequencies are approximately symmetric when the input levels to the filters are equivalent to the level of 51 dB/ERB at 1 kHz. These equivalent input noise levels can be calculated using the ELC correction for centre frequencies up to about 1 kHz, and the MAF or the MAP correction (depending upon the way in which the stimuli are delivered and specified) for higher centre frequencies.

2) The low-frequency skirt of the auditory filter becomes less sharp with increasing level. The variation can be described in terms of the parameter p_l. Let X denote the equivalent input noise level in dB/ERB (as defined above), and let $p_{l(X)}$ denote the value of p_l at level X. Then:

$$p_{l(X)} = p_{l(51)} - 0.38(p_{l(51)}/p_{l(51.1k)})(X - 51), \quad (5)$$

where $p_{l(51)}$ is the value of p at that centre frequency for an equivalent noise level of 51 dB/ERB and $p_{l(51.1k)}$ denotes the value of p_l at 1 kHz for a noise level of 51 dB/ERB (a spectrum level of 30 dB). The value of $p_{l(51)}$ can be calculated from Eqn. 3, remembering that $p_{l(51)} = 4f_c/\text{ERB}$. Note that Eqn. 5 differs from the one suggested in Moore and Glasberg (1987; their Eqn. 8). Eqn. 5 is the same as Eqn. 8 of Moore and Glasberg for a centre frequency of 1 kHz, but it assumes that at other centre frequencies the values of p_l change by the same ratio for a given change in level as is found at 1 kHz. This gives a better fit to the more recent data obtained at very low and high centre frequencies.

3) Changes in slope of the high-frequency skirt of the filter with level are less consistent. At medium centre frequencies (1–4 kHz) there is a trend for the slope to increase slightly with increasing level, but at low centre frequencies (Moore et al., 1990) there is no clear trend with level, and

the filters at high centre frequencies (Shailer et al. 1990) show a slight decrease in slope with increasing level. Overall, changes in the high-frequency slope with level tend to be rather small, and for most purposes they can be ignored. That is what we do in the revised program for calculating excitation patterns which is given in the Appendix.

Implications of the results for the derivation of excitation patterns

We have defined the excitation pattern as the output of each auditory filter as a function of filter centre frequency (Moore and Glasberg, 1983, 1987). A procedure for calculating excitation patterns based on this definition was described in Moore and Glasberg (1987). The changes to the ERB function (Eqn. 3) and to the equation describing changes in filter shape with level (Eqn. 4) have implications for the shapes of excitation patterns calculated using this definition. Also, the ELC 'correction' for the transmission characteristics of the outer and middle ear gives somewhat different results from the MAF 'correction' suggested earlier (Moore and Glasberg, 1987). Some examples of the effects of these changes are given below.

The new ERB function gives more realistic excitation patterns for pure tones and narrow bands of noise at high levels. This is illustrated in Fig. 8 which shows excitation patterns for a 1-kHz sinusoid at a level of 90 dB, calculated by the 'old' method (Moore and Glasberg, 1987; solid line) and the 'new' method (dashed line). With the old method, the calculated excitation pattern curves upwards at very high centre frequencies, an effect which is not seen in masked audiograms for narrowband noises (Zwicker, 1975). This happens because the old ERB function gives values which are too large at high centre frequencies. The new ERB function gives excitation patterns which correspond rather well to the shapes of masked audiograms, and which do not show an upturn at high centre frequencies.

The excitation patterns in Fig. 8 also differ slightly on the low-frequency side. The reason for

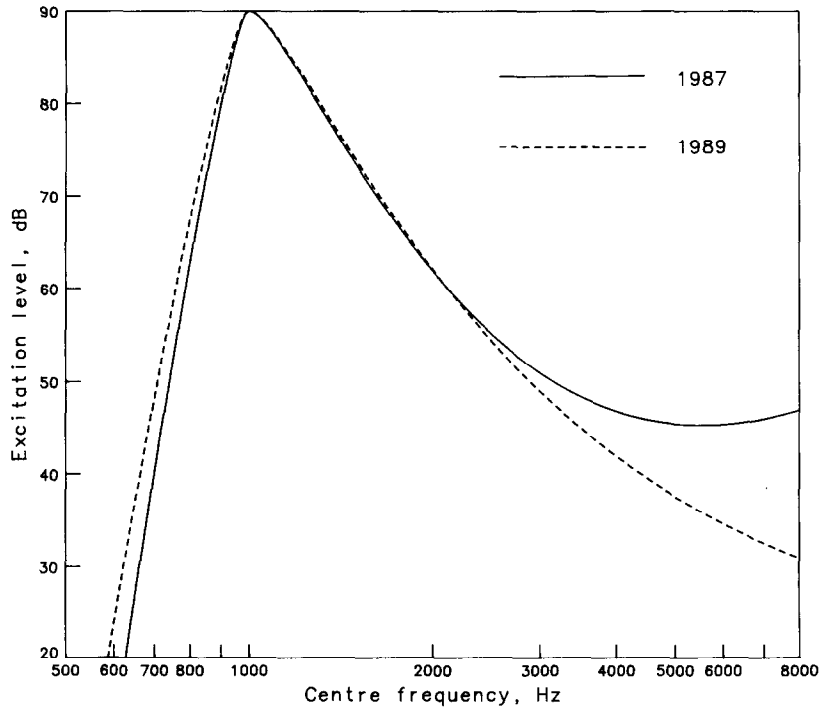

Fig. 8. Excitation patterns calculated with the 'old' (solid line) and 'new' (dashed line) procedures for a 1-kHz sinusoid with a level of 90 dB SPL. See text for details.

this is that the old method assumed that the high-frequency skirt of the auditory filter gets slightly sharper with increasing level, whereas the new method assumes that the high-frequency skirt does not vary with level. The low-frequency side of the excitation pattern is determined by the high-frequency side of the auditory filter.

In our earlier paper (Moore and Glasberg, 1987), we assumed, for simplicity, that the transmission characteristics of the outer and middle ear were reflected in the variation of absolute threshold with frequency. As discussed earlier, we now feel that, for frequencies up to about 1 kHz, the shape of the equal-loudness contours at high levels gives a better estimate of the frequency-dependent attenuation applied in the transmission of sound to the cochlea. Above 1 kHz, a correction based on the shape of the absolute threshold curve is probably appropriate.

Fig. 9 shows excitation patterns for a synthetic vowel, /u/, one calculated with the MAF correction and one with the ELC correction. The essential features of both excitation patterns are the same, but the one calculated with the ELC correction gives somewhat higher overall excitation levels below about 800 Hz. The program for calculating excitation patterns given in the Appendix allows the use either of the MAF or the ELC correction, or some other user-defined correction function.

General Discussion and Conclusions

In this paper we have described several modifications to the procedure for deriving auditory filter shapes from notched-noise data. These include: 1) taking into account changes in filter bandwidth with centre frequency when allowing for the effects of off-frequency listening; 2) correcting for the non-flat frequency response of the sound delivery system; 3) correcting for the transmission characteristics of the outer and middle ear; 4) limiting the amount by which the centre frequency of the filter can shift in order to maximise the signal-to-masker ratio. In many cases, these modifications result in only small changes to the derived filter shape. However, at very high and

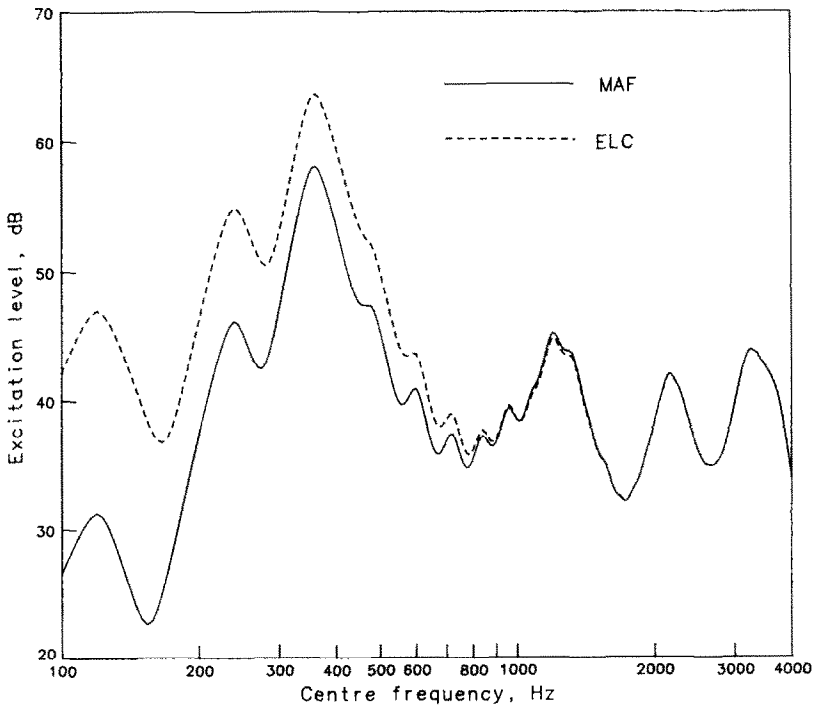

Fig. 9. Excitation patterns for a synthetic vowel, /u/, calculated using either the MAF correction (solid line) or the MAP correction (dashed line).

low centre frequencies and for hearing-impaired subjects the differences can be substantial. We have also shown that filter shapes derived using data restricted to symmetric-notch conditions can be seriously in error when the underlying filter is asymmetric.

Using the modified fitting procedure, we have reanalysed several earlier sets of data, and also analysed some recent data obtained for very low and high centre frequencies. These analyses have led to a revised formula describing the variation of the ERB as a function of frequency, and a new equation describing how the slope of the low-frequency skirt of the auditory filter changes with level. The implications of these results for the calculation of excitation patterns are described, and a revised program for calculating excitation patterns is given in the Appendix.

It should be emphasised that individual differences in auditory filter shape can be marked, particularly when hearing-impaired subjects are used. The equations and procedures given for calculating excitation patterns are applicable to the average results of normally hearing subjects.

Finally, it should be noted that the new procedure for deriving auditory filter shapes is not restricted to data obtained using notched-noise maskers. In principle, *any* masker spectrum can be used, provided it can be specified and substituted in Eqn. 1. The procedure could be applied, for example, to the analysis of data obtained using rippled noise (e.g. Houtgast, 1977). This greater flexibility arises because the integral in Eqn. 1 is evaluated by numerical methods. In practice, of course, the noise spectrum should be chosen to limit off-frequency listening and to provide sufficient information to define the filter shape adequately. The new procedure could also be readily adapted to filter shapes other than the roex (p, r) filter assumed in this paper. Indeed, any equation for the filter shape could be used, even one that did not have an analytic integral.

Acknowledgments

This work was supported by the Medical Research Council (U.K.). We thank Roy Patterson, Bob Peters and Mike Shailer for making their raw data available for analysis. We are grateful to Numerical Recipes Software for permission to reproduce some subroutines from Numerical Recipes: The Art of Scientific Computing (Press et al., 1986). We thank Tom Baer, Roy Patterson, Stuart Rosen, Michael Shailer and Michael Stone for helpful comments on an earlier version of this paper.

Appendix

The appendix gives listings of two computer programs. The first is for deriving auditory filter shapes from notched-noise data. It includes all of the modifications to the fitting procedure described in the text. It assumes that the data are stored in a file with a specific format; an example of such a file is also given. The name of that file is requested by the program. Provision is made for applying a correction to allow for any frequency-dependent attenuation applied to the stimuli before auditory filtering. The data specifying the correction are stored in a file (a table of values) whose name is given in the data file. The program performs a smooth polynomial interpolation between the values given in the table. Three example tables are given, containing the MAF, MAP and ELC corrections. Users may supply their own corrections in tabular form, for example, to allow for the frequency response of the transducer used. If no correction file is specified, then the fitting procedure makes use of an analytic integral. This runs approximately seven times faster than the numerical integration used when a correction file is specified.

The second program is for calculating the excitation pattern of a signal from its power spectrum. It also requests the name of a correction file, which may be one of the MAF, MAP or ELC corrections, or a user-specified correction. The input spectrum is specified as a line spectrum, which may be either harmonic or non-harmonic. The excitation pattern for a noise can easily be calculated by approximating the noise spectrum as a series of components spaced at, say, 10-Hz intervals. It is easy to modify the program to read the stimulus spectrum from a file; this may be helpful when the spectrum is very complex.

Both programs contain subroutines taken from the book Numerical Recipes: The Art of Scientific

Computing, published by Cambridge University Press. The copyright to these subroutines is owned by Numerical Recipes Software, and they are reproduced with permission. Machine readable forms of these and other subroutines (for IBM PC and Apple Macintosh users) may be obtained from Cambridge University Press. Distribution tapes for workstations and multi-user mainframes can be obtained from Numerical Recipes Software, PO Box 243, Cambridge, MA 02238, U.S.A.

References

Dallos, P. (1973) The Auditory Periphery: Biophysics and Physiology. Academic Press, New York.

Dubno, J.R. and Dirks, D.D. (1989) Auditory filter characteristics and consonant recognition for hearing-impaired listeners. J. Acoust. Soc. Am. 85, 1666–1675.

Fastl, H. and Zwicker, E. (1983) A free-field equalizer for the TDH 39 earphones. J. Acoust. Soc. Am. 73, 312–314.

Glasberg, B.R. and Moore, B.C.J. (1986) Auditory filter shapes in subjects with unilateral and bilateral cochlear impairments. J. Acoust. Soc. Am. 79, 1020–1033.

Green, D.M., Kidd, G. and Stevens, K.N. (1987) High-frequency audiometric assessment of a young adult population. J. Acoust. Soc. Am. 81, 485–494.

Greenwood, D.D. (1961) Critical bandwidth and the frequency coordinates of the basilar membrane. J. Acoust. Soc. Am. 33, 1344–1356.

Houtgast, T. (1977) Auditory-filter characteristics derived from direct-masking data and pulsation-threshold data with a rippled-noise masker. J. Acoust. Soc. Am. 62, 409–415.

Killion, M. C. (1978) Revised estimate of minimum audible pressure. Where is the 'missing 6 dB'? J. Acoust. Soc. Am. 63, 1501–1508.

Killion, M. C. (1984) New insert earphones for audiometry. Hearing Instruments 35, 45–46.

Lynch, T.J., Nedzelnitsky, V. and Peake, W.T. (1982) Input impedance of the cochlea in cat. J. Acoust. Soc. Am. 72, 108–130.

Moore, B.C.J. (1986) Parallels between frequency selectivity measured psychophysically and in cochlear mechanics. Scand. Audiol. Suppl. 25,139–152.

Moore, B.C.J. and Glasberg, B.R. (1983) Suggested formulae for calculating auditory-filter bandwidths and excitation patterns. J. Acoust. Soc. Am. 74,750–753.

Moore, B.C.J. and Glasberg, B.R. (1987) Formulae describing frequency selectivity as a function of frequency and level, and their use in calculating excitation patterns. Hear. Res. 28, 209–225.

Moore, B.C.J., Peters, R.W. and Glasberg, B.R. (1990) Auditory filter shapes at low center frequencies. J. Acoust. Soc. Am. (in press).

Nedzelnitsky, V. (1980) Sound pressures in the basal turn of the cat cochlea. J. Acoust. Soc. Am. 68, 1676–1689.

Patterson, R.D. (1976) Auditory filter shapes derived with noise stimuli. J. Acoust. Soc. Am. 59, 640–654.

Patterson, R.D. (1974) Auditory filter shape. J. Acoust. Soc. Am. 55, 802–809.

Patterson, R.D. and Moore, B.C.J. (1986) Auditory filters and excitation patterns as representations of frequency resolution. In: B.C.J. Moore (Ed.), Frequency Selectivity in Hearing, Academic Press, London. pp. 123–177.

Patterson, R.D. and Nimmo-Smith, I. (1980) Off-frequency listening and auditory-filter asymmetry. J. Acoust. Soc. Am. 67, 229–245.

Patterson, R.D., Nimmo-Smith, I., Weber, D.L. and Milroy, R. (1982) The deterioration of hearing with age: Frequency selectivity, the critical ratio, the audiogram, and speech threshold. J. Acoust. Soc. Am. 72, 1788–1803.

Pickles, J.O. (1988) An Introduction to the Physiology of Hearing. Academic Press, London.

Press, W.H., Flannery, B.R., Teukolsky, S.A. and Vetterling, W.T. (1986) Numerical Recipes: The Art of Scientific Computing. Cambridge University Press, Cambridge, U.K.

Rosen, S. and Stock, D. (1989) Auditory filter bandwidths as a function of level at low frequencies (125 Hz–1 kHz): A preliminary report. Speech, Hearing and Language: Work in Progress UCL, 3, Department of Phonetics and Linguistics, University College London. pp. 205–215

Scharf, B. (1970) Critical bands. In: J.V. Tobias (Ed.), Foundations of Modern Auditory Theory, Vol. I., Academic Press, London. pp. 159–202.

Shailer, M.J., Moore, B.C.J., Glasberg, B.R., Watson, N. and Harris, S. (1990) Auditory filter shapes at 8 and 10 kHz. J. Acoust. Soc. Am. (in press).

Soderquist, D.R. and Lindsey, J.W. (1972) Physiological noise as a masker of low frequencies: the cardiac cycle. J. Acoust. Soc. Am. 52, 1216–1220.

Weber, D.L. (1977) Growth of masking and the auditory filter. J. Acoust. Soc. Am. 62, 424–429.

Zwicker, E. (1961) Subdivision of the audible range into critical bands (Frequenzgruppen). J. Acoust. Soc. Am. 33, 248.

Zwicker, E. (1975) Scaling. In: W.D. Keidel and W.D. Neff (Eds.), Handbook of Sensory Physiology, Vol. 5/3, Springer-Verlag, Berlin. pp. 401–448.

Zwicker, E. and Terhardt, E. (1980) Analytical expressions for critical- band rate and critical bandwidth as a function of frequency. J. Acoust. Soc. Am. 68, 1523–1525.

Zwislocki, J.J. (1975) The role of the external and middle ear in sound transmission. In: D.B. Tower (Ed.), The Nervous System, Vol. 3: Human Communication and its Disorders. Raven Press, New York.

```
*** Program for deriving auditory filter shapes from notched-noise ***
********** data assuming filters with a roex(p,r) shape ************
* The language used is FORTRAN 77
* The program, subroutines and functions are as follows:
*       Main program
*     subroutine openfiles()
*     subroutine input(title,corr_file,No)
*     character*(*) function noblanks(text)
*     subroutine calc_abs(adjust,cf_corr,corr_file)
*     real*8 function thr_corr(freqHz,n_thr,corr_hz,corr_dB)
*     subroutine locate(xx,n,x,j)
*     subroutine polint(xa,ya,n,x,y,dy)
*     subroutine simplex(p,y,mp,np,ndim,ftol,funk,iter)
*     real*8 function sscalc(par)
*     real*8 function brent(ax,bx,cx,f,tol,xmin)
*     real*8 function nscalc(ns_shift)
*     real*8 function p_erb(shift1)
*     real*8 function filter_wt(pg,r_lin)
*     real*8 function thresh(ns_shift)
*     subroutine qromb(a,b,ss,p,r_lin,hz1,hz2)
*     subroutine trapzd(a,b,s,n,p,r_lin,hz1,hz2)

* The include file, used in compilation, is given after the end of the program.
*-------------------------------------------------------------------------
*  The input file should have the following format:
*  1. Title
*  2. The number of data points (npts), how often the progress of the
*     fitting should be printed, e.g every ten iterations (n_print), and the
*     maximum allowed value of pl or pu (pmax)
*  3. The noise spectrum level in dB (No), the centre frequency in Hz (cf),
*     the noise bandwidth as a proportion of cf (width), and the name of the
*     correction file (corr_file)
*  4. Start values for the parameters & maximum shift: pl,pu,r,max_shift
*  5. The threshold values in dB
*  6. The lower & upper edges of the notch for each data point, in the same
*     order as the threshold values
*  See below for an example data file
*-------------------------------------------------------------------------
* main program
      implicit undefined     ! Every variable used must be defined.
      include 'roex3.h'  ! See separate listing of roex3.h.
      integer k,j,iter,mp,np
      real*8 ftol,st_par(ndim)
      parameter(np = ndim,mp = np+1,ftol = 1.0d-2)
      real*8 params(mp,np),ssq(mp),sscalc,temp(np)
      real*8 No,adjust,sscheck,cf_corr
      real*8 erb,st_diff
      character title*60,corr_file*40
      external sscalc

      call openfiles()
10    call input(title,corr_file,No)  ! Routine for getting input data.
      adjust = No+10.0d0*log10(cf) ! Adjustment to get correct value of K.
      cferb = c1 * (c2 * cf/1000.0d0 + 1.0d0) ! Centre freq in erb units.
      call calc_abs(adjust,cf_corr,corr_file)   ! Set up correction values.
      sscheck = 5001.0d0 ! Constants determining whether simplex repeats
```

```
      ssq(1) = 5000.0d0  ! with starting values of parameters at previous
      st_diff = 1.1d0    ! best values.
      j = 1

* Label 30 is the return point for second and subsequent entries to simplex.
30    sscount = 0 ! Counts iterations in simplex; restarted on entering.
      st_par(1) = pl ! Starting values of parameters for next entry to
      st_par(2) = pu ! simplex.
      st_par(3) = r
* Decide whether to repeat simplex.
      if(ssq(j).lt.sscheck-0.1.and.ssq(j).gt.0.5d0) then
        sscheck = ssq(j)
           if(sscheck.lt.5000.0d0.and.n_print.gt.0)
     &              write(6,"(' restart ssq = ',f9.3)") ssq(j)
* Set starting vertices of simplex.
        do 50 j = 1,mp
        do 40 k = 1,np
        params(j,k) = st_par(k)
        if(j.eq.k+1) params(j,k) = params(j,k) / st_diff
        temp(k) = params(j,k)
40      continue
* Calculate the sum of the squared deviations of the data from the fitted
* values (ssq) for each vertex of the simplex.  See later for function sscalc.
        ssq(j) = sscalc(temp)
50      continue
* Perform multidimensional minimization using the downhill simplex method.
        call simplex(params,ssq,mp,np,ndim,ftol,sscalc,iter)
        st_diff = st_diff * 2.0d0
        j = 1
        do 55 k = 2,mp
        if(ssq(k).lt.ssq(j)) j = k ! Pick up the index of the lowest ssq.
55      continue
        pl  =  abs(params(j,1)) ! Set parameters to values giving the
        pu  =  abs(params(j,2)) ! lowest ssq.
        r   = -abs(params(j,3))
        goto 30                 ! Check whether to re-enter simplex.
      else                      ! End of minimization.
        write(8,"('   sum of squares =',f8.1,/)")ssq(j) ! Write results.
        erb = 2./pl+2./pu       ! Calculate ERB.
        write(8,"(' pl =',f5.1,' pu =',f5.1,' r =',f7.1,' k(dB) =',
     &    f5.1,' erb =',f6.3,' ss =',f6.1)")pl,pu,r,xk,erb,ssq(j)
        write(8,"(' level (cf + noise) = ',f6.1,/)")adjust
        write(8,"(6x,'lower     upper     data      calc      resid',
     &              '     shift     gain(db)')")
        call sscalc(st_par) ! Calculate fitted values from final pl, pu, r
        do 60 notch = 1,npts
* Return data and fitted values to initial form (before correction).
        data(notch) = data(notch)+adjust + cf_corr
        calcdb(notch) = calcdb(notch)+adjust + cf_corr
        write(8,9030)el(notch),eu(notch),data(notch),calcdb(notch),
     &              diff(notch),shift(notch),gain(notch)
9030    format(1x,5f10.2,2f10.2)
60      continue
        write(8,"(10x,'------------------------',/)")
        goto 10  ! Check if there is another set of data to process.
      endif
      end
```

```
*-----------------------------------------------------------------------
* Opens the data and results file; for an example datafile see below.
      subroutine openfiles()
      implicit undefined
      character*40 filename

      write(6,"(' data file name? ',$)")
      read (5,"(a)") filename
      open (7,file=filename)
      write(6,"(' file name for results? ',$)")
      read (5,"(a)") filename
      open (8,file=filename)
      end
*-----------------------------------------------------------------------
* This subroutine reads the data from the input file.
      subroutine input(title,corr_file,No)
      implicit undefined
      include 'roex3.h'
      real*8 No
      character title*(*)
      character corr_file*(*)
      character noblanks*40

      read(7,*)title
      read(7,"(2i6,f6.0)")npts,n_print,pmax
      if(npts.gt.nmax) stop ! Use a large number (e.g. 999) to signal end.
      if(pmax.eq.0.0d0) pmax = 50.0d0 ! Default upper limit for p.
      read(7,"(3f10.0,a)")No,cf,width,corr_file
      corr_file = noblanks(corr_file) ! Takes out blanks.
      if(corr_file.eq.'') corr_file = 'none' ! Default correction file.
      read(7,"(5f10.0)")pl,pu,r,max_shift ! Start parameters and max shift.
* If no value is specified for max_shift, default is 0.15.  If zero shift is
* desired, put a very small value for max_shift (e.g. 0.0001) in data file.
      if(max_shift.eq.0.0d0) max_shift = 0.15d0
      read(7,*)(data(notch),notch = 1,npts)
      read(7,*)(el(notch),eu(notch),notch = 1,npts)
      if(corr_file.eq.'none') then  ! If no correction file is specified
        numeric = .false. ! function thresh will use analytic integration.
        write(8,"(' roex3n : ( analytical integration)')")
        write(8,"(' No threshold correction used')")
      else
        numeric = .true.
        write(8,"(' roex3n : ( numerical integration)')")
        write(8,"(' Correction file : ',a)")corr_file
      endif
      write(8,"(' Maximum allowed shift = ',f6.2,/)") max_shift
      write(8,"(1x,a)")title
      if(n_print.gt.0) then
        write(6,"(' roex3n ')")
        write(6,"(' correction file : ',a,/)")corr_file
        write(6,*)title
      endif
      end
*-----------------------------------------------------------------------
```

```
* This function removes any blanks or tabs from the string of text.
      character*(*) function noblanks(text)
      implicit undefined
      integer i,j
      character text*(*)

      noblanks = ''
      j = 1
      do 10 i = 1,40
        if(text(i:i).eq.' '.or.text(i:i).eq.'\t') goto 10
        noblanks(j:j) = text(i:i)
        j = j + 1
10    continue
      end
*-----------------------------------------------------------------------
* This subroutine uses the values in the correction file to determine
* correction values at 10-Hz spacing from 0 to 15000 Hz and stores them in
* array lin_corr as linear intensities.
      subroutine calc_abs(adjust,cf_corr,corr_file)
      implicit undefined
      include 'roex3.h'
      integer i,n_thr,ioval,max_corr
      parameter(max_corr=100)
      real*8 adjust,cf_corr,thr_corr,hz,corr_hz(max_corr),corr_dB(max_corr)
      character corr_file*(*)

      if(.not.numeric) then   ! No correction file used.
        cf_corr = 0.0d0       ! No correction applied to the signal level.
        do 10 notch = 1,npts  ! The thresholds are adjusted by
          data(notch) = data(notch) - adjust  ! subtracting 10log(cf)+No.
10      continue
        return
      else              ! Threshold correction used.
        open (10,file=corr_file,err=50,status='old',iostat=ioval)
        read (10,"(i10)") n_thr
        if(n_thr.gt.max_corr) then
          write(6,"(' too many points in file ',a)")corr_file
          stop
        endif
        read (10,*)(corr_hz(i),i = 1,n_thr)
        read (10,*)(corr_dB(i),i = 1,n_thr)
        close (10)
* Calculate correction for the specified centre frequency.
        cf_corr = thr_corr(cf,n_thr,corr_hz,corr_dB)
        do 30 notch = 1,npts
          data(notch) = data(notch) - adjust - cf_corr
30      continue
* Convert index values to frequencies in Hz, calculate threshold correction
* at each frequency in linear intensity, and place results in array lin_corr.
        do 40 i = 1,max_corr_data
          hz = real(i-1) * 10.0d0
          lin_corr(i) =10.0d0 ** (-thr_corr(hz,n_thr,corr_hz,corr_dB)/10.0d0)
40      continue
      endif
      return
50    stop (' error opening threshold correction file ')
      end
```

```
*-----------------------------------------------------------------------
* This function returns a threshold correction in dB for a given frequency
* (freqHz) in Hz using polynomial interpolation with 3 points. See Numerical
* Recipes p. 80-83.
      real*8 function thr_corr(freqHz,n_thr,corr_hz,corr_dB)
      implicit undefined
      integer n_thr,readfile,index
      real*8 corr_hz(*),corr_dB(*),freqHz,dy

        call locate(corr_hz,n_thr,freqHz,index)
        if(index.eq.0) then
          thr_corr = corr_dB(1)
        else if(index.ge.n_thr) then
          thr_corr = corr_dB(n_thr)
        else
          call polint(corr_hz(index),corr_dB(index),3,freqHz,thr_corr,dy)
        endif
      end
*-----------------------------------------------------------------
* This subroutine picks the numbers over which the interpolation is to be
* performed.  See Numerical Recipes p. 89-90.
      subroutine locate(xx,n,x,j)
      implicit undefined
      integer ju,jl,jm,j,n
      real*8 xx(n),x

      jl = 0
      ju = n + 1
10    if(ju-jl.gt.1) then
        jm = (ju + jl)/2
        if((xx(n).gt.xx(1)).eqv.(x.gt.xx(jm))) then
          jl = jm
        else
          ju = jm
        endif
      goto 10
      endif
      j = jl
      end
*-----------------------------------------------------------
```

```
* Subroutine polint performs polynomial interpolation.  See Numerical
* Recipes p. 80-83.
      subroutine polint(xa,ya,n,x,y,dy)
      implicit  undefined
      integer nmax,i,m,ns,n
      parameter (nmax=10)
      real*8 x,y,dy,xa(n),ya(n),c(nmax),d(nmax)
      real*8 dif,dift,ho,hp,w,den

      ns=1
      dif=abs(x-xa(1))
      do 11 i=1,n
        dift=abs(x-xa(i))
        if(dift.lt.dif) then
          ns=i
          dif=dift
        endif
        c(i)=ya(i)
        d(i)=ya(i)
11    continue
      y=ya(ns)
      ns=ns-1
      do 13 m=1,n-1
        do 12 i=1,n-m
          ho=xa(i)-x
          hp=xa(i+m)-x
          w=c(i+1)-d(i)
          den=ho-hp
          if(den.eq.0.0d0) pause 'polint : den = 0.0 '
          den=w/den
          d(i)=hp*den
          c(i)=ho*den
12      continue
        if(2*ns.lt.n-m) then
          dy=c(ns+1)
        else
          dy=d(ns)
          ns=ns-1
        endif
        y=y+dy
13    continue
      end
*---------------------------------------------------------------------
* This is the simplex subroutine which performs multidimensional
* minimization.  See Numerical Recipes, p. 289-293.
      subroutine simplex(p,y,mp,np,ndim,ftol,funk,iter)
      implicit undefined
      integer mp,np,ndim,iter,nmax,itmax
      integer i,j,mpts,ilo,ihi,inhi
      real*8 ftol,alpha,beta,gamma,funk,rtol,ypr,yprr
      parameter (nmax=20,alpha=1.0d0,beta=0.5d0,gamma=2.0d0,itmax=500)
      real*8 p(mp,*),y(mp),pr(nmax),prr(nmax),pbar(nmax)

      mpts=ndim+1
      iter=0
1     ilo=1
      if(y(1).gt.y(2)) then
```

```
        ihi=1
        inhi=2
      else
        ihi=2
        inhi=1
      endif
      do 11 i=1,mpts
        if(y(i).lt.y(ilo)) ilo=i
        if(y(i).gt.y(ihi)) then
          inhi=ihi
          ihi=i
        else if(y(i).gt.y(inhi)) then
           if(i.ne.ihi) inhi=i
        endif
11    continue
      rtol=2.*abs(y(ihi)-y(ilo))/(abs(y(ihi))+abs(y(ilo)))
      if (rtol.lt.ftol) return
      if(iter.eq.itmax) print*,' simplex exceeding maximum iterations'
      iter=iter+1
      do 12 j=1,ndim
        pbar(j)=0.0d0
12    continue
      do 14 i=1,mpts
        if(i.ne.ihi) then
          do 13 j=1,ndim
            pbar(j)=pbar(j)+p(i,j)
13        continue
        endif
14    continue
      do 15 j=1,ndim
        pbar(j)=pbar(j)/ndim
        pr(j)=(1.0d0+alpha)*pbar(j)-alpha*p(ihi,j)
15    continue
      ypr=funk(pr)
      if(ypr.le.y(ilo)) then
        do 16 j=1,ndim
          prr(j)=gamma*pr(j)+(1.-gamma)*pbar(j)
16      continue
          yprr=funk(prr)
          if(yprr.lt.y(ilo)) then
            do 17 j=1,ndim
              p(ihi,j)=prr(j)
17        continue
          y(ihi)=yprr
        else
          do 18 j=1,ndim

            p(ihi,j)=pr(j)
18        continue
          y(ihi)=ypr
        endif
      else if(ypr.ge.y(inhi))then
        if(ypr.lt.y(ihi)) then
          do 19 j=1,ndim
            p(ihi,j)=pr(j)
19        continue
          y(ihi)=ypr
```

```
        endif
        do 21 j=1,ndim
          prr(j)=beta*p(ihi,j)+(1.-beta)*pbar(j)
21      continue
        yprr=funk(prr)
          if(yprr.lt.y(ihi)) then
          do 22 j=1,ndim
            p(ihi,j)=prr(j)
22        continue
          y(ihi)=yprr
        else
          do 24 i=1,mpts
            if(i.ne.ilo)then
            do 23 j=1,ndim
              pr(j)=0.5*(p(i,j)+p(ilo,j))
              p(i,j)=pr(j)
23          continue
            y(i)=funk(pr)
            endif
24        continue
        endif
      else
        do 25 j=1,ndim
          p(ihi,j)=pr(j)
25      continue
        y(ihi)=ypr
      endif
      goto 1
      end
*--------------------------------------------------------------------------
* This function returns the total sum of squares difference between the
* threshold data and the calculated values using parameters held in par (as
* determined by the starting values, or, subsequently, by the simplex
* subroutine).
      real*8 function sscalc(par)
      implicit undefined
      include 'roex3.h'
      real*8 par(ndim),ns,nscalc,sum,thresh
      real*8 tol1,neg_el,pos_eu,brent
      parameter(tol1 = 1.0d-3) ! Determines when function brent stops.
      external nscalc

      sscount = sscount+1     ! Increment each time sscalc is entered.
      pl =  abs(par(1)) ! Extract the parameters.
      pu =  abs(par(2))
      r  = -abs(par(3))
      sum = 0.0d0
      do 10 notch = 1,npts
        if(el(notch).eq.0.0d0.and.eu(notch).eq.0.0d0) then
          pr_erb = 1.0 !! needed so thresh uses the right (un)shifted erb
          ns = thresh(0.0d0)
          shift(notch) = 0.0d0
        else
* Find the lowest noise-to-signal ratio (ns) and the shift at which it
* occurs for each notch width.
          neg_el = -el(notch)               ! Set lower limit for shift.
            if(neg_el .lt. -max_shift) neg_el = -max_shift
```

```
            pos_eu = eu(notch)                ! Set upper limit for shift.
            if(pos_eu .gt. max_shift)  pos_eu =  max_shift
          ns = brent(neg_el,0.0d0,pos_eu,nscalc,tol1,shift(notch))
        endif
        calcdb(notch) = 10.*log10(ns)
        pr_erb = 1.0 !! needed so thresh uses the right (un)shifted erb
        gain(notch) = 10.0d0*log10(thresh(0.0d0))-calcdb(notch)
        sum = sum-calcdb(notch)+data(notch)
10    continue
      xk = sum/dble(float(npts))
      sscalc = 0.0d0
      do 20 notch = 1,npts            ! Slide the calculated thresholds so
      calcdb(notch) = calcdb(notch)+xk ! they have the same mean as the data,
      diff(notch) = data(notch)-calcdb(notch)   ! to get an estimate of K.
      sscalc = sscalc + diff(notch)**2
20    continue
* This is a safe way to set limits on the parameters.
      if(pl.gt.pmax.or.pu.gt.pmax)   sscalc = sscalc + 1000.0d0
      if(pl.lt.0.1d0.or.pu.lt.0.1d0) sscalc = sscalc + 1000.0d0
        if(n_print.gt.0) then
          if(mod(sscount,n_print).eq.0)
     &        write(6,9000)sscount,pl,pu,r,sscalc
        endif
9000  format(1x,i4,' pl =',f7.3,'  pu =',f7.3,'  r = ',f9.2,' sscalc =',f9.3)
      end
*-------------------------------------------------------------
* This is the subroutine brent which performs a one-dimensional minimization
* of the noise-to-signal ratio.  This is used for finding the optimum shift
* for each notch width. See Numerical Recipes, p. 283-286.
      real*8 function brent(ax,bx,cx,f,tol,xmin)
      implicit undefined
      integer itmax,iter
      real*8 ax,bx,cx,f,tol,xmin
      real*8 cgold,zeps,a,b,v,x,e,fx,fu,fv,fw,xm
      real*8 tol1,tol2,w,q,p,r,etemp,d,u
      parameter (itmax = 100,cgold = 0.381966d0,zeps = 1.0d-10)

      a = min(ax,cx)
      b = max(ax,cx)
      v = bx
      w = v
      x = v
      e = 0.0d0
      fx = f(x)
      fv = fx
      fw = fx
      do 11 iter = 1,itmax
        xm = 0.5 * (a + b)
        tol1 = tol * abs(x) + zeps
        tol2 = 2.0d0 * tol1
        if(abs(x-xm).le.(tol2-0.5d0 * (b-a))) goto 3
        if(abs(e).gt.tol1) then
        r = (x-w) * (fx-fv)
        q = (x-v) * (fx-fw)
        p = (x-v) * q-(x-w) * r
        q = 2.0d0 * (q-r)
        if(q.gt.0.0d0) p = -p
```

```
        q = abs(q)
        etemp = e
        e = d
        if(abs(p).ge.abs(0.5d0 * q * etemp).or.p.le.q * (a-x)
     $                                .or.p.ge.q * (b-x)) goto 1
        d = p/q
        u = x + d
        if(u-a.lt.tol2.or.b-u.lt.tol2) d = sign(tol1,xm-x)
        goto 2
        endif
1       if(x.ge.xm) then
          e = a-x
        else
          e = b-x
        endif
        d = cgold * e
2       if(abs(d).ge.tol1) then
          u = x + d
        else
          u = x + sign(tol1,d)
        endif
        fu = f(u)
        if(fu.le.fx) then
          if(u.ge.x) then
            a = x
          else
            b = x
          endif
          v = w
          fv = fw
          w = x
          fw = fx
          x = u
          fx = fu
        else
        if(u.lt.x) then
          a = u
        else
          b = u
        endif
        if(fu.le.fw.or.w.eq.x) then
          v = w
          fv = fw
          w = u
          fw = fu
        else if(fu.le.fv.or.v.eq.x.or.v.eq.w) then
          v = u
          fv = fu
        endif
      endif
11    continue
*     print*,' brent exceeded maximum iterations'
3     xmin = x
      brent = fx
      end
*-------------------------------------------------------------------------
```

```
* This function calculates the noise-to-signal ratio for a filter shifted by
* ns_shift from the on-frequency filter.
      real*8 function nscalc(ns_shift)
      implicit undefined
      include 'roex3.h'
      real*8 r_lin,sig_int,push,plsh,filter_wt
      real*8 ns_shift,thresh,off_shift,p_erb

      r_lin = 10.**(r/10.) ! Convert r to linear intensity.
      if(ns_shift .lt. -max_shift) ns_shift = -max_shift
      if(ns_shift .gt.  max_shift) ns_shift =  max_shift
* The shift passed in is relative to the on-frequency filter but needs to be
* measured from the shifted (off-frequency) filter. pr_erb is the ERB of the
* off-frequency filter as a proportion of the ERB of the on-freq filter.
* sig_int is the weighting applied by the shifted filter at the signal
* frequency.
      pr_erb = p_erb(ns_shift)
      off_shift = ns_shift / pr_erb ! Shift rel. to off-frequency filter.
      if(off_shift.eq.0.0d0) then
        sig_int = 1.0d0
      else if(off_shift.lt.0.0d0) then
        push = pu*(-off_shift)
        if(push.gt.maxexp) push = maxexp
        sig_int = filter_wt(push,r_lin)
      else if(off_shift.gt.0.0d0) then
        plsh = pl*off_shift
        if(plsh.gt.maxexp) plsh = maxexp
        sig_int = filter_wt(plsh,r_lin)
      endif
* Calulate the noise-to-signal ratio at the output of the shifted filter.
      nscalc = thresh(ns_shift)/sig_int
        if(nscalc.le.0.0d0) then          ! Just in case = 0.0.
          write(6,"(' nscalc .le.  zero !! ')")
          nscalc = exp(-maxexp) ! Set to lowest positive number.
        endif
      end
*-----------------------------------------------------------------------
* This function returns the ERB of the off-frequency shifted filter as a
* proportion of the ERB of the on-frequency filter.
      real*8 function p_erb(shift1)
      implicit undefined
      include 'roex3.h'
      real*8 shift1, sh_freq

      sh_freq = (cf*(1.0d0 + shift1))/1000.0d0 ! Freq of shifted filter, kHz
      p_erb = (c1 * (c2 * sh_freq + 1.0d0))/cferb
      end
*-----------------------------------------------------------------------
* Equation of the filter weighting function: pg = p * g
       real*8 function filter_wt(pg,r_lin)
      implicit undefined
      real*8 pg,r_lin
      filter_wt = (1.0d0 - r_lin) * (1.0d0 + pg) * exp(-pg) + r_lin
      end
*-----------------------------------------------------------------------
```

```
* This function returns the amount of noise passing through the filter using
* either numerical integration (if a threshold correction file was specified)
* or analytic integration (if no correction file was specified).
      real*8 function thresh(ns_shift)
      implicit undefined
      include 'roex3.h'
      real*8 width1,r_lin,cu,cl,hz11,hz12,hzu1,hzu2
      real*8 ns_shift,gl,gu,sum_low,sum_hih
      real*8 pgush,pglsh,pucu,plcl,thresh_l,thresh_u,thrint_l,thrint_u

      if(ns_shift.eq.0.0d0) pr_erb = 1.0d0  ! Unshifted filter.
      r_lin = 10.0d0**(r/10.0d0)    ! r_lin in linear intensity units.
      hz11 = (1.0d0 - el(notch)-width)*cf ! Lower band outer edge freq, Hz.
      if(hz11.lt.0.0d0) hz11 = 0.0d0
      hz12 = (1.0d0 - el(notch)) * cf ! Lower band inner edge freq, Hz.

      hzu1 = (1.0d0 + eu(notch)) * cf     ! Upper band inner edge freq, Hz.
      hzu2 = hzu1 + width * cf      ! Upper band outer edge freq, Hz.
      gu = (eu(notch) - ns_shift) / pr_erb      ! Upper band inner edge and
      cu = gu + width / pr_erb            ! outer edge in terms of g.
      gl = (el(notch) + ns_shift) / pr_erb      ! Same for lower band.
      width1 = width / pr_erb       !
      if(el(notch) + width1.gt.1.0d0) width1 = 1.0d0-el(notch)
      cl = gl + width1
      if(numeric) then        ! Threshold correction required.
*  Integrate over equally spaced 'g' values.
        call qromb(gl,cl,sum_low,pl,r_lin,hz12,hz11) ! Lower noise band.
        call qromb(gu,cu,sum_hih,pu,r_lin,hzu1,hzu2) ! Upper noise band.
        thresh = sum_hih + sum_low  ! Use qromb result.
      else
*  Analytic integration.
        pgush = pu * gu
        pglsh = pl * gl
        if(pgush.gt.maxexp) pgush = maxexp
        if(pglsh.gt.maxexp) pglsh = maxexp
        pucu = pu*cu
        plcl = pl*cl
        if(plcl.gt.maxexp) plcl = maxexp
        if(pucu.gt.maxexp) pucu = maxexp
        thrint_l = (r_lin-1.0d0)*(2.0d0 + plcl)*exp(-plcl)/pl + r_lin*cl
        thresh_l = (r_lin-1.0d0)*(2.0d0 + pglsh)*exp(-pglsh)/pl + r_lin*gl
        thrint_l = thrint_l - thresh_l
        thrint_u = (r_lin-1.0d0)*(2.0d0 + pucu)*exp(-pucu)/pu + r_lin*cu
        thresh_u = (r_lin-1.0d0)*(2.0d0 + pgush)*exp(-pgush)/pu + r_lin*gu
        thrint_u = thrint_u - thresh_u
        thresh = thrint_l + thrint_u
      endif
        if(thresh.le.0.0d0) then          ! Why you should use double
          write(6,"(' thresh le 0.0d0')") ! precision variables.
          thresh = exp(-maxexp) ! Set to lowest positive number.
        endif
      end
*-------------------------------------------------------------------------
```

```
* Subroutine for performing numerical integration by Romberg's method.
* See Numerical Recipes, p. 114-115.
* This routine has variables p,r_lin,hz1 and hz2 passed through it to trapzd.
      subroutine qromb(a,b,ss,p,r_lin,hz1,hz2)
      implicit undefined
      integer jmax,j,jmaxp,k,km
      real*8 a,b,ss,dss,p,r_lin,hz1,hz2,eps
      parameter (eps=1.0d-1, jmax=10, jmaxp=jmax+1,k=5,km=k-1)
      real*8 s(jmaxp),h(jmaxp)

      h(1) = 1.0d0
      do 11 j = 1,jmax
        call trapzd(a,b,s(j),j,p,r_lin,hz1,hz2)
        if(j.ge.k) then
          call polint(h(j-km),s(j-km),k,0.0d0,ss,dss)
          if(abs(dss).lt.eps*abs(ss)) return
        endif
        s(j+1)=s(j)
        h(j+1)=0.25 * h(j)
11    continue
      pause ' qromb: Too many steps'
      end
*--------------------------------------------------------------------
* This routine decreases the fineness of spacing of points in the numerical
* integration until further changes make no significant difference.
* See Numerical Recipes, p. 111.
* This version of the routine calls function filter_wt(pg,wlin) by name.
* It also calculates the threshold correction to be applied to each
* point before doing the integration.
      subroutine trapzd(a,b,s,n,p,r_lin,hz1,hz2)
      implicit undefined
      include 'roex3.h'
      integer n,it,j,index1,index2
      real*8 filter_wt,a,b,s,sum,x,tnm,del,p,pg,pgb,r_lin
      real*8 delhz,xhz,hz1,hz2
      character dummy

      if(n.eq.1) then
        pg  = p * a
        pgb = p * b
        xhz   = hz1
        delhz = hz2
        index1 = ifix((xhz+5.0d0)/10.0d0) + 1  ! Pick a frequency index for
        index2 = ifix((delhz+5.0d0)/10.0d0) + 1 ! the threshold correction.
        if(index1.gt.max_corr_data) index1 = max_corr_data
        if(index2.gt.max_corr_data) index2 = max_corr_data
        s = 0.5*(b-a)*(filter_wt(pg,r_lin)*lin_corr(index1)
     $                 + filter_wt(pgb,r_lin)*lin_corr(index2))
        it = 1
      else
        tnm = it
        del = (b - a)/tnm
        delhz = (hz2 - hz1)/tnm
        x = a + 0.5 * del
        xhz = hz1 + 0.5 * delhz
        sum = 0
        do 11 j = 1,it
```

```
          index1 = ifix((xhz+5.0d0)/10.0d0) + 1
          if(index1.gt.max_corr_data) index1 = max_corr_data
          pg = p * x
          sum = sum + filter_wt(pg,r_lin) * lin_corr(index1)
          x = x + del
          xhz = xhz + delhz
11      continue
        s = 0.5 * (s + (b - a) * sum/tnm)
        it = 2 * it
        endif
      end
**********************End of filter fitting program************************



* The include file roex3.h, used in compilation, is :-
      integer nmax,ndim,notch,npts,n_print,sscount
      integer max_corr_data
      parameter (ndim = 3,nmax = 100,max_corr_data=1501)
      real*8 el(nmax),eu(nmax),data(nmax),calcdb(nmax),gain(nmax),diff(nmax)
      real*8 lin_corr(max_corr_data),xk,width
      real*8 shift(nmax)
      real*8 pl,pu,r,pmax,max_shift
      real*8 cf,cferb,pr_erb
      real*8 c1,c2,maxexp
      parameter (c1 = 24.673, c2 = 4.368, maxexp = 700.0d0)
* Maxexp should be set to something near the largest number a negative
* exponential can have on the system being used; e.g. exp(-700.0) is close
* to 10**(-304) and exp(-182) is close to 10**(-78).
      logical numeric
      common el,eu,data,calcdb,gain,diff,xk,width,notch,
     &       npts,n_print,sscount
      common /tc/ lin_corr,numeric
      common /sh/ shift,max_shift
      common /param/ pl,pu,r,pmax
      common /erbscale/ cf,cferb,pr_erb
```

```
* Example correction files are given below.  The first line gives the number, n,* of
points in the file.  Subsequent lines give n frequencies (Hz) and n
* corresponding levels in dB.
*--------------------------------
*Correction file ELC
*--------------------------------
46
  20. ,  25. ,  30. ,  35. ,  40. ,  45. ,  50. ,  55. ,  60. ,  70. ,
  80. ,  90. , 100. ,  125. ,  150. ,  177. , 200. , 250. , 300. , 350. ,
 400. , 450. , 500. ,  550. ,  600. ,  700. , 800. , 900. ,1000. ,1500. ,
2000. ,2500. ,2828. , 3000. , 3500. , 4000. ,4500. ,5000. ,5500. ,6000. ,
7000. ,8000. ,9000. ,10000. ,12748. ,15000.
  31.8,  26.0,  21.7,  18.8,  17.2,  15.4,  14.0,  12.6,  11.6,  10.6,
   9.2,   8.2,   7.7,   6.7,   5.3,   4.6,   3.9,   2.9,   2.7,   2.3,
   2.2,   2.3,  2.5,   2.7,   2.9,   3.4,   3.9,   3.9,   3.9,   2.7,
   0.9,  -1.3,  -2.5,  -3.2,  -4.4,  -4.1,  -2.5,  -0.5,   2.0,   5.0,
  10.2,  15.0,  17.0,  15.5,  11.0,  22.0
*----------------------------------
*Correction file MAF
*----------------------------------
46
  20.0,  25.0,  30.0,  35.0,  40.0,  45.0,  50.0,  55.0,  60.0,  70.0,
  80.0,  90.0, 100.0,  125.0,  150.0,  177.0, 200.0, 250.0, 300.0,  350.0,
 400.0, 450.0, 500.0,  550.0,  600.0,  700.0, 800.0, 900.0,1000.0, 1500.0,
2000.0,2500.0,2828.0, 3000.0, 3500.0, 4000.0,4500.0,5000.0,5500.00,6000.0,
7000.0,8000.0,9000.0,10000.0,12748.0,15000.0
  73.4,  65.2,  57.9,  52.7,  48.0,  45.0,  41.9,  39.3,  36.8,  33.0,
  29.7, 27.1,  25.0,  22.0,  18.2,  16.0,  14.0,  11.4,   9.2,   8.0,
   6.9,  6.2,   5.7,   5.1,   5.0,   5.0,   4.4,   4.3,   3.9,   2.7,
   0.9, -1.3,  -2.5,  -3.2,  -4.4,  -4.1,  -2.5,  -0.5,   2.0   5.0,
  10.2, 15.0,  17.0,  15.5,  11.0,  22.0
*----------------------------------
*Correction file MAP
*----------------------------------
14
 125., 250., 500., 1000., 1500., 2000., 3000.,
4000.,6000.,8000.,10000.,12000.,14000.,16000.
30., 19., 12.0,  9.0, 11.0, 16.0, 16.0,
14., 14.,  9.9, 24.7, 32.7, 44.1, 63.7
*----------------------------------
* Example input file
*----------------------------------
' The Boss 800 Hz 77 dB overall'
15,1
49,800,0.4,elc
20,20,-20
76.2,73.3,66.3,56.5,49.5,41.0,34.8
54.5,47.3,39.5,32.8
42.0,34.2,31.8,25.2
0.0,0.0,0.1,0.1,0.2,0.2,0.3,0.3,0.4,0.4,0.5,0.5,0.6,0.6
0.3,0.5,0.4,0.6,0.5,0.7,0.6,0.8
0.5,0.3,0.6,0.4,0.7,0.5,0.8,0.6
finish
999
```

```
*-------------------------------------------------------
* Results obtained from the example input file:-
*-------------------------------------------------------
roex3n : ( numerical integration)
Correction file : elc
Maximum allowed shift =   0.15
 The Boss 800 Hz 77 dB overall
  sum of squares =    33.0
pl = 20.2 pu = 33.9 r =  -56.8 k(dB) =-48.0 erb = 0.158 ss =  33.0
level (cf + noise) =   78.0
     lower     upper     data      calc      resid     shift     gain(db)
      0.00      0.00     76.20     78.31     -2.11      0.00      0.00
      0.10      0.10     73.30     71.50      1.80      0.01      0.20
      0.20      0.20     66.30     63.83      2.47      0.03      0.72
      0.30      0.30     56.50     56.09      0.41      0.04      1.10
      0.40      0.40     49.50     48.45      1.05      0.05      1.18
      0.50      0.50     41.00     40.88      0.12      0.05      0.92
      0.60      0.60     34.80     33.83      0.97      0.03      0.39
      0.30      0.50     54.50     55.21     -0.71      0.11      1.94
      0.40      0.60     47.30     48.14     -0.84      0.08      1.47
      0.50      0.70     39.50     40.85     -1.35      0.06      0.96
      0.60      0.80     32.80     33.93     -1.13      0.03      0.38
      0.50      0.30     42.00     43.20     -1.20      0.00      0.01
      0.60      0.40     34.20     34.54     -0.34      0.01      0.04
      0.70      0.50     31.80     29.05      2.75      0.01      0.03
      0.80      0.60     25.20     27.09     -1.89      0.00      0.00
```

```
************* Program for Calculating Excitation Patterns *************

* The language used is FORTRAN 77.
* Declare variables.

      implicit undefined      ! every variable used must be defined

* Copies of the next 2 lines are in functions erbtofq and fqtoerb
      real*8 c1,c2,c3
      parameter (c1 = 24.673, c2 = 4.368, c3 = 2302.6/(c1 * c2))
      integer i,npoints,comp
      integer N_COMP,N_ERBDB,MAX_THR,n_thr
      parameter (N_COMP=200,N_ERBDB=400,MAX_THR=100)
* Increase N_COMP for > 200 components.
* Increase MAX_THR for > 100 data points in the correction file.
      real*8 compfq(N_COMP),compdB(N_COMP),compint(N_COMP),comperb(N_COMP)
      real*8 lowfrq,highfq,intensity,erbdB(N_ERBDB),thr_corr
      real*8 fqtoerb,erbtofq,estart,eend,E,estep,frqhz,frqkhz
      real*8 erb,p51,p51_1k,p,g,freq,ELdB,thrdiff,corr_1k
      real*8 corr_hz(MAX_THR),corr_dB(MAX_THR)
      parameter ( p51_1k = 4.0 * 1000.0/(c1*(c2 + 1.0)) )
      character*40 filename, corr_file

* Request a file name where the results will be stored.
      write(6,"(' file name for results ',$)")
      read(5,"(a)") filename
      open(8,file=filename)

* Request the name of a file containing the 'correction' data.
* See the examples following the program for filter fitting.

      write(6,"(' correction file [none for none, elc default] ',$)")
      read(5,"(a)")corr_file
      if(corr_file.eq.'') corr_file = 'elc'
      if(corr_file.ne.'none') then
        call read_thr(n_thr,corr_hz,corr_dB,corr_file)! Read correction file.
        corr_1k = thr_corr(1000.0d0,n_thr,corr_hz,corr_dB)! Relative to 1 k
      else
        corr_1k = 0.0
      endif
      print*,'corr_1k = ',corr_1k
      write(6,"(1x,a)")corr_file

* The erb spacing (estep) refers to the separation of adjacent filters used in
* the program. If spacing finer then 0.1 is required, increase the value of the
* constant N_ERBDB from 400 to at least 1 + (36 + 1 - 3)/estep.

      write(6,"(' Erb spacing [default 0.1] ',$)")
      read (5,"(f5.0)") estep
      if(estep.eq.0) estep = 0.1
      write(6,"(1x,f6.2)") estep

* Request the frequencies of the components in Hz. Set up an array, compfq,
* holding the component frequencies, and request the level of each component
* in dB. Store the levels in array compdB.

      call input(npoints,compfq,compdB)
```

```
* The level of each component is corrected to allow for the difference
* between the effective level at 1 kHz and that at the frequency
* of the component caused by the sound system and/or the outer/middle ear.
* The correction, in dB, is obtained by polynomial interpolation.
* The corrected level is stored in array compdB.

      do 20 i = 1,npoints
      if(corr_file.ne.'none') then
        thrdiff = thr_corr(compfq(i),n_thr,corr_hz,corr_dB) - corr_1k
      else
        thrdiff = 0.0
      endif
      compdB(i) = compdB(i)-thrdiff

* For each component, convert the level in dB to linear intensity and store the
* values in array compint. Convert the frequency to the E value and store in
* array comperb. The function fqtoerb (frequency to E value) is defined later.

      compint(i) = 10.0**(compdB(i)/10.0)
      comperb(i) = fqtoerb(compfq(i))
20    continue

* The equations in the text are valid for E values between about 3 and 36.
* For each value of E (in steps of estep) the total intensity in each ERB
* is calculated: the intensities of components within +/- 0.5ERB of that E
* value are summed.  A level of -100 dB (intensity=1e-10) is assigned when no
* components fall within +/- 0.5ERB.  The intensity sum is converted to dB
* and held in array erbdB.  The function erbtofq (E value to frequency) is
* defined later in the program.

      estart = 3.0    ! 0.1 kHz = 3.4 ERBs on this scale.
      eend = 36.0 ! 10 kHz = 35.3 ERBs on this scale.
      i = 0

      do 40 E = estart,eend,estep
      i = i+1
      if(i.gt.N_ERBDB) then
        write(6,"(' increase the value of N_ERBDB and recompile')")
        stop
      endif
      lowfrq = erbtofq(E-0.5)
      highfq = erbtofq(E+0.5)
      intensity = 0.0
      do 30 comp = 1,npoints
      if(compfq(comp).lt.lowfrq) goto 30
      if(compfq(comp).gt.highfq) goto 35
      intensity = intensity+compint(comp)
30    continue
35    if(intensity.lt.1e-10) intensity = 1e-10
      erbdB(i) = 10.0 * log10(intensity)
40    continue

* The excitation pattern is determined by calculating the total output from
* each auditory filter as a function of filter centre frequency (in estep
* steps). When the filter is centred above the frequency of the component
* the p value to be used for each filter is calculated from Eqn. 5 in the text.
```

```
* The value of X used when calculating the filter output for a given component
* is the total level in the ERB around that component, as held in array erbdB.
* When the filter is centred below the frequency of the component the p value
* is simply 4fc/erb. The final excitation levels are stored in array ELdB. The
* corresponding E values are stored in array E, and frequency values in array
* freq.

      do 70 E = estart,eend,estep
      frqhz = erbtofq(E)
      frqkhz = frqhz/1000.0
      erb = c1 * (c2 * frqkhz + 1.0)
      p51 = 4.0 * frqhz/erb
      intensity = 0.0
      do 50 comp = 1,npoints
      i = (comperb(comp)-estart)/estep + 1.5
      g = (compfq(comp)-frqhz)/frqhz
* To prevent floating point errors in line 777 (when taking exp(-p*g)), g
* is not allowed to be greater than 2.
      if(g.gt.2.0) goto 60
      if(g.lt.0.0) p = p51 - 0.380 * (p51/p51_1k) * (erbdB(i)-51.0)
      if(g.ge.0.0) p = p51    ! upper skirt doesn't change with level
      g = abs(g)
777   intensity = intensity+(1.0+p*g)*exp(-p*g)*compint(comp)
50    continue
60    if (intensity.lt.1e-10) intensity = 1e-10
      freq = erbtofq(E)
      ELdb = 10.0 * log10(intensity)
      write(8,9030) freq,E,ELdB
9030  format(1x,3f10.2)
70    continue
      close(8)
      end
*-------------------------------------------------------------------------
* This is the subroutine for getting the frequencies and levels of the
* components.
      subroutine input(npoints,compfq,compdB)
      implicit undefined
      integer npoints,i
      real*8 compfq(*),compdB(*),fund,first,last
      character*1 char

10    write(6,9000)
9000  format(' Harmonic [h] or Inharmonic [i] Complex Tone ? ',$)
      read(5,*)char
      if (char.eq.'h') then   ! Harmonic.
        write(6,"(' fund, first, last all in Hz :- ',$)")
        read(5,"(3f10.0)") fund,first,last
        npoints = ifix((last-first)/fund + 1.5)
        do 20 i = 1,npoints
          compfq(i) = first + (i-1) * fund
20      continue
      else if (char.eq.'i') then ! Inharmonic.
        write(6,"(' How many components are required ?',$)")
        read (5,*) npoints
        do 30 i = 1,npoints
          write(6,"(' Component',i3,' Freq. (Hz)? ',$)")i
          read (5,*) compfq(i)
```

```
30      continue
      else
        goto 10
      endif
      do 40 i = 1,npoints
      write(6,"(' Component freq. ',f7.1,' Hz. level (dB)? ',$)") compfq(i)
      read(5,"(f5.0)")compdB(i)
      write(6,"(1x,i3,2f8.1)")i,compfq(i),compdB(i)
40    continue
      end
*-------------------------------------------------------------------------
* Function fqtoerb converts a frequency in Hz to the corresponding E value.
      real*8 function fqtoerb (fq)
      implicit undefined
      real*8 c1,c2,c3
      parameter (c1 = 24.673, c2 = 4.368, c3 = 2302.6/(c1 * c2))
      real*8 fq,fqkHz
      fqkHz = fq/1000
      fqtoerb = c3 * log10(c2 * fqkHz + 1.0)
      end
*-------------------------------------------------------------------------
* Function erbtofq converts the E value to its corresponding frequency in Hz.
      real*8 function erbtofq(E)
      implicit undefined
      real*8 c1,c2,c3,E
      parameter (c1 = 24.673, c2 = 4.368, c3 = 2302.6/(c1 * c2))
      erbtofq = 1000.0 * (10.0 ** (E/c3) - 1.0) / c2
      end
*-------------------------------------------------------------------------
* This is the subroutine for reading the correction file.
      subroutine read_thr(n_thr,corr_hz,corr_dB,corr_file)
      implicit undefined
      integer n_thr,i,ioval
      real*8 corr_hz(*),corr_dB(*)
      character*40 filename,corr_file*(*)

      open (10,file=corr_file,err=10,status='old',iostat=ioval)
      read (10,"(i10)") n_thr
      if(n_thr.gt.100) then
        write(6,"(' more than 100 points in file ',a)")corr_file
        stop
      endif
      read (10,*)(corr_hz(i),i = 1,n_thr)
      read (10,*)(corr_dB(i),i = 1,n_thr)
      close (10)
      return
10    write(6,"(' error opening threshold correction file ',a)")corr_file
      stop
      end
*-------------------------------------------------------------------------
* Function thr_corr, and subroutines polint and locate are the same as in the
* filter fitting program, so they are not reproduced here.
```