AN IMPROVED PERCEPTUAL QUALITY METRIC FOR HIGHLY TO MODERATELY IMPAIRED AUDIO

BY

KUMAR DEEPAK SYAM KALLAKURI, B.E.

A thesis submitted to the Graduate School

in partial fulfillment of the requirements

for the degree

Master of Science in Electrical Engineering

New Mexico State University

Las Cruces, New Mexico

December 2004

"An Improved Perceptual Quality Metric for Highly to Moderately Impaired Audio," a thesis prepared by Kumar Deepak Syam Kallakuri in partial fulfillment of the requirements for the degree, Master of Science in Electrical Engineering, has been approved and accepted by the following:

_____

Linda Lacey
Dean of the Graduate School

_____

Charles D. Creusere
Chair of the Examination Committee

_____

Date

Committee in charge:

Dr. Charles D. Creusere, Chair

Dr. Phillip De Leon

Dr. Wolfgang Mueller

DEDICATION

This thesis is dedicated to my loving parents and my brother Yashu for their love and support, and all my teachers for their valuable guidance.

ACKNOWLEDGEMENTS

I gratefully acknowledge my graduate advisor, Dr. Charles Creusere, for his help and guidance throughout my master"s program. It is an honor to be working with him as a research assistant. This thesis would not have been possible without his guidance and help. I would also like to thank Dr. Phillip De Leon and Dr. Deva Borah for their great teaching and guidance.

VITA

Sep. 27, 1981	Born at Rajahmundry, Andhra Pradesh, India
1998	Intermediate Pubic Examination, Andhra Pradesh, India.
2002	Bachelor of Engineering (B.E) from Andhra University, Visakhapatnam, Andhra Pradesh, India.
Aug-Dec 2004	Graduate Research Assistant, Klipsh School of Electrical and Computer Engineering, New Mexico State University, Las Cruces, New Mexico.

PUBLICATIONS

Kumar Kallakuri, Charles D. Creusere, "An Improved Perceptual Quality Metric for Highly to Moderately Impaired Audio," Submitted to *IEEE ICASSP 2005*.

R. Vanam, K. Kallakuri, C. D. Creusere, "Scalable Objective Quality Metric for Evaluating Audio Impairment," Submitted to *IEEE DCC 2005*.

FIELD OF STUDY

Major Field: Electrical Engineering

Digital Signal Processing

ABSTRACT

AN IMPROVED PERCEPTUAL QUALITY METRIC FOR HIGH TO MODERATELY IMPAIRED AUDIO

BY

KUMAR DEEPAK SYAM KALLAKURI

Master of Science in Electrical Engineering

New Mexico State University

Las Cruces, New Mexico, 2004

Dr. Charles D. Creusere, Chair

A scalably compressed bitstream is one which can be decoded at different bitrates. MPEG-2 and MPEG-4 support fine-grained scalability through Bit Slice Arithmetic Coding (BSAC) and while these algorithms are typically tested with human subjective analysis, there also exist objective measures of perceived quality of audio. In previous work, we have found that the existing ITU recommendation BS.1387-1 for evaluating perceptual audio quality does not accurately reflect perceptual audio quality at low bitrates. Here, we develop a new metric which takes in the 11 model output variables (MOV) from BS.1387 and adds a new 12^{th} MOV, called energy equalization. With the newly designed weights and neural network, we evaluate and

compare this metric"s performance to that of BS.1387 at low and mid bitrates and show that the performance of the new metric is better in the mean square error sense.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

1 INTRODUCTION

1.1 The problem

Digital audio compression has been of great interest to researchers and industry alike for many years. In 1992, the International Organization for Standardization (ISO) designed Motion Picture Experts Group (MPEG) I standard for video and audio compression and transmission [1]. Later the more efficient MPEG-2 and MPEG-4 standards were developed. MPEG-4 provides for scalable audio compression—the ability to encode audio at one bitrate and then to decode it at any rate lesser than the encoded rate. Bit Slice Arithmetic coding (BSAC), a variant of the MPEG advanced audio coder (AAC), is of particular interest because it allows us to create many fine grained layers of scalability.

To evaluate audio compression algorithms, human subjective testing is typically performed. The time consuming nature of this procedure, however, makes it desirable to have an objective metric (i.e., a program that can be run in a computer) which can measure the perceptual quality of the audio. Many algorithms have been developed which addressed this problem, the most recent of which is a recommendation from the International Telecommunications Union (ITU): ITU-R BS.1387-1 [2]. This standard does not, however, predict perceptual quality accurately for low bitrate scalably compressed audio as is shown in [3]. Therefore we wish to develop a new metric which can improve upon the performance of BS.1387 at low bitrates while achieving similar performance at

high rates. For designing the desired metric, we incorporate the energy equalization technique [4], [5] which has been found to be useful for predicting quality of audio at low bitrates into the existing BS.1387.

1.2 Overview of Chapters

This research concentrates on analyzing and improving the existing ITU-R BS.1387. From [4] and [5] we note that the BS. 1387 does not perform well at low bitrates. Thus, we study the integration of energy equalization into BS. 1387 and evaluate its performance at low and mid bitrates relative to human subjective test data. Chapter 2 discusses the audio coding algorithms that are included in the MPEG-4 standard while Chapter 3 introduces various subjective testing procedures. Chapter 4 summarizes the ITU-R BS.1387 recommendation and describes the model output variables. Chapter 5 details the design of the new metric and the integration of the energy equalization into BS.1387. Next, Chapter 6 looks at the results of the different robustness tests and discusses the performance of the new metric at low and mid bitrates. Finally, conclusions and future work are presented in chapter 7.

2. MPEG AUDIO CODING ALGORITHMS

Three MPEG-4 audio codecs are used for validation of our new metric. Specifically, we compare BSAC with nonscalable transform weighted interleaved vector quantization (Twin VQ) and the nonscalable Advanced Audio Coder (AAC). Twin VQ is known to give good performance at low bitrates whereas AAC is known to perform better at high bitrates [4], [5]. We evaluate the performance of scalable BSAC relative to the nonscalable MPEG-4 codecs at low to mid bitrates.

All of the MPEG-4 audio compression algorithms contain a psychoacoustic model of the human ear, which use block-based Fast Fourier Transforms (FFT"s) to calculate signal masking parameters. A similar procedure is used in ITU-R BS.1387-1 for quality assessment. In our tests, we compare BSAC sequences encoded at 64 kb/s and decoded at 32 kb/s and 16 kb/s with Twin VQ encoded and decoded at 16 kb/s and with both Twin VQ and AAC encoded and decoded at 32 kb/s.

2.1 Introduction

The MPEG-4 audio coder, developed through many years of careful testing and evaluating represents the current state of the art. MPEG-4 allows for scalability in a variety of ways, including one general method where a core bit stream of low rate is generated by one kind of encoder and higher rate enhancement layers are generated by encoding the resulting error residual using a totally different algorithm [3]. Unfortunately, this approach cannot produce more than two or three layers of audio fidelity [3]. In this work, algorithms which support fine grained scalability are of particular interest. In MPEG, such fine grained scalability can only be achieved by

using Bit Slice Arithmetic Coding (BSAC), either with SSR (scalable sampling rate) of MPEG 4 or directly with the advanced audio coder (AAC) of MPEG 2. To achieve bit stream scalability, the SSR algorithm uses a uniform 4-band cosine modulated filter bank whose outputs are maximally decimated [3]. This results in four ¼ rate sample streams whose bandwidths are (0, 5.5 kHz), (5.5 kHz, 11kHz), (11 kHz, 16.5 kHz) and (16.5 kHz, 22 kHz), assuming that the original sampling rate is 44.1 kHz. Equal length modified discrete cosine transforms are encoded in a layered fashion using BSAC [3], [4], [5].

We focus in this thesis on three basic audio coding algorithms that are part of the MPEG-4 natural audio coding standard: scalable BSAC-AAC, nonscalable Twin VQ (Transform-Weighted Interleaved Vector Quantization), and nonscalable AAC. MPEG-4 AAC uses a Fast Fourier Transform (FFT) to psycho-acoustically analyze the input audio signal and extract the quantization parameters. Prior to quantization, the signal is processed by a modified discrete cosine transform (MDCT) having a 50% overlap and outputting either 1024 or 128 coefficients each time [3], [4], [5]. Twin VQ replaces AAC in some of our tests as it is known to provide superior performance at 16 kb/s and below [3]. As implemented in MPEG-4, Twin VQ uses a switched MDCT transforms like that of AAC, but applies it to the signal only after the input audio signal has passed through two stages of prediction that decorrelate the signal to the maximum extent possible [3]. After the prediction residual has been

transformed, the coefficients are fed into vector quantizer for encoding after being interleaved.

2.2 Audio Comparison

2.2.1 Subjective Comparison

Subjective testing is used for audio codec performance comparison. The disadvantage of subjective testing are that it is very time consuming, expensive and needs a large number of subjects to achieve accurate results. Previously, we have performed here subjective testing of MPEG 4 codecs discussed in section 2.1.

To evaluate the impact of scalability, we encode monoaural sequences at 64 kbits/sec (kb/s) and decoded them at 32 kb/s and 16 kb/s using MPEG AAC with BSAC. Using human subjective testing, these sequences are then compared to Twin VQ and AAC sequences encoded and decoded at 16 kb/s and 32 kb/s. We do such tests because the best performance of a scalable system is upper bounded by the best performance of the best nonscalable system available [3].

Comparison Category Rating (CCR) is a standard testing procedures suggested in ITU recommendation ITU-R BS.1116 [2]. CCR is used for subjective evaluation at low bitrates or of audio having large impairments. Since we are interested primarily in lower bitrate audio, we use a slightly different variation of CCR which is discussed in [16]. This variation of CCR is treated in greater detail in section 3.2. The fact that subjective testing cannot be used for real-time measurements along with other disadvantages make all the more plain the need for an objective quantification procedure for measuring perceptual audio quality.

2.2.2 Objective Quantification

Objective quantification allows for real-time measurement while being both less expensive and less time consuming than subjective testing. Many objective metrics for quantifying perceptual audio quality like the SNR (Signal to Noise ratio), do not work very well because they do not take the human perception of sound into account.

2.3 Past Work on Objective Metrics

2.3.1 DIX

The perceptual measurement method DIX (Disturbance Index) by Thiede and Kabot [7] is based on an auditory filter bank which allows higher temporal resolution compared to FFT-based approaches and a more precise understanding of temporal effects such as pre and post-masking. The fine structure of the temporal envelopes at each auditory level is preserved and is used to obtain additional information about the signals and the distortions introduced [2].

The center frequencies of individual filters are equally distributed over a perceptual pitch scale. In order to ensure that the chosen number of filters cover the whole range of frequencies without ripples in the overall frequency range, the top of the filter is rounded. Slope of the filter decreases exponentially over the Bark scale in order to model the masking thresholds. The threshold level of input filters decides the steepness of the filter slope and 40 filters cover the audible frequency range in DIX. Even though the filter bank algorithm is lower in complexity than filter banks

implemented as individual filters, it is much slower compared to block-based transforms like FFT [2].

DIX dynamically adapts the levels and spectra between the Signal Under test and the Reference signal for separating linear from non-linear distortions. The structure of the temporal envelope is evaluated at filter outputs in order to model the increased amount of masking caused by modulated and noise resembling maskers as compared to pure tones [2].

Various parameters like partial loudness of non-linear distortions, indicators for the amount of linear distortions and other measures for temporal effects are calculated by a comparison of internal representations of the Signal Under Test and the Reference signal. However, a good estimation of the basic audio quality can be achieved by using only two of the output parameters. Partial loudness of non-linear distortions along with one of the indicators for the amount of linear distortions is fed into a neural network to provide an estimate for the perceived basic audio quality of the Signal Under Test.

2.3.2 NMR

Noise to Masked- Ratio (NMR) method developed by Brandenburg [9] makes a measure of the level-difference between the masked threshold and the noise signal. Frequency content of the signal is analyzed by taking a DFT with a Hann window of 20 ms duration and the transform coefficients are mapped to a Bark scale (explained in [9]). The masked threshold is estimated for each band and the slope of the masked threshold is obtained using a worst-case approach, keeping in mind the fact that the

slopes are steeper for weak signals. Due to the fact that the absolute threshold is adapted to the resolution of input signal the NMR is robust to changes of reproduction level. This procedure has a pitch scale resolution of 1 Bark [2].

Masking flag rate which gives the percentage of frames with audible distortions, as well as the total and mean NMR (different ways of averaging the distortion between the error energy and masked threshold) are the most important output values of NMR [2].

2.3.3 OASE

Objective Audio Signal Evaluation (OASE) designed by Sporer [10] uses a filter bank with 241 filters for analyzing the input signals. The center frequencies are equally spaced with a distance of 0.1 Bark on the Bark scale. Each of the filters (which overlap one another) is adapted to the frequency response of a point on the basilar membrane. Similar to the NMR approach, the level dependencies of the filter slopes is included via a worst-case approach. Low-frequency-centered filters require calculation at the full sampling rate while the high-frequency-centered filters can be calculated at reduced sampling rates. After filtering, a model of the temporal effects of the human auditory system is calculated as done in Auditory Spectral Difference (ASD). Following this step, a reduction of the sampling rate in all the filters is realizable, which leads to a temporal resolution of 0.66 ms at a sampling rate of 48 kHz for the filter bank. Outputs from the matching filters of reference and Signal Under Test are compared with a probability of a detection function which uses the loudness of input signals to calculate the Just Noticeable Level Difference (JNLD).

Total probability of detection, derived from the probability of detection of each band is calculated as in (2.1) where $e[k,n]$ is given as $e[k,n] = \tilde{E}_{ref}[k,n] - \tilde{E}_{test}[k,n]$.

$$p_c[k,n] = 1 - 10^{\left(-a[k,n].e[k,n]^b\right)} \tag{2.1}$$

$$\tilde{E}[k,n] = 10 * \log_{10}(E[k,n]) \tag{2.2}$$

$$a[k,n] = \frac{10^{\frac{(\log 10(\log 10(2.0)))}{b}}}{s[k,n]} \tag{2.3}$$

As defined above probability of detection is calculated for both input channels and also for the center channel. Probability of detection in the center channel is the worst case of the probabilities of detection in the left and right channels. The number of steps above threshold is given by

$$q_c[k,n] = \frac{|INT(e[k,n])|}{s[k,n]} \tag{2.4}$$

$$Q_c[n] = \sum_{\forall k} q_c[k,n] \tag{2.5}$$

$$Q_{sum} = \sum_{\forall n} Q_c[n] \tag{2.6}$$

Equation 2.6 is useful in calculating the Average Distorted Block (ADB) MOV as discussed in Chapter 4.

Several ways for temporal averaging of the probability of detection and the steps above threshold are used [2]:

- the temporal average of the probability of detection

- the frequency of frames with a probability of detection above 0.5
- the maximum of a low pass filtered probability of detection
- the maximum of a low pass filtered probability of detection with forgetting
- the average number of steps above threshold for frames with a probability of detection above 0.5
- the average number of steps above threshold
- the maximum number of steps above threshold
- the average of the number of steps above the threshold of the 10% worst frames

2.3.4 Perceptual Audio Quality Measure (PAQM)

The basic idea of PAQM by Beerends and Stemerdink [11] is to subtract the internal representations (representations inside the head of the subject) of the reference and degraded signal and map the difference with a cognitive mapping to the subjectively perceived audio quality [2]. Transformation from the physical, external domain to the psycho-physical, internal domain is performed in four operations:

- a time frequency mapping which is done via a DFT with a Hann window of about 40 ms duration
- frequency warping using the Bark scale
- time-frequency spreading
- intensity warping (compression)

Using both time-frequency spreading and compression allows modeling of the masking behaviour of the human auditory system at and above the masked threshold.

Subjective results of the first MPEG audio codec evaluation are used in the optimization of compression. Difference in the internal representation is expressed in terms of the noise disturbance. In the latest PAQM version submitted to ITU-R, two cognitive effects were included in the mapping from the noise disturbance to the subjective quality [13], [2].

2.3.5 PERCEVAL

PERCEVAL (PERCeptual EVALuation) by Paillard et al. [14] models the transfer characteristics of the middle and inner ear to form an internal representation of the signal. A Hann window of 40 ms is typically applied on the input signal with a 50% overlap between successive windows to decompose the input into a time-frequency representation. The energy spectrum is multiplied by a frequency dependent function which models the effect of the ear canal and the middle ear. Attenuated spectral values are mapped from a frequency scale to a more linear pitch scale (linear with respect to both the physical properties and observed psycho-physical effects). To simulate the dispersion of energy along the basilar membrane, the transformed energy contents are convolved with a spreading function and the energies are expressed in decibels. An intrinsic frequency dependent energy is added to each pitch component to account for the absolute threshold of hearing [2].

In simulations of auditory masking experiments, a basilar membrane representation is formed for each stimulus. The information available for performing the task is given by the difference in stimulus representations. A representation of the

stimulus is the masker and the masker along with the test signal forms the other. Their difference represents the unmasked component of the signal. PERCEVAL calculates the probability of detecting this difference. The non-detection probability of the difference for each detector along the simulated basilar membrane is estimated using a sigmoidal probability function. With the assumption that the detectors are statistically independent, the global detection probability is calculated as the complement of the product of the individual non-detection probabilities [2].

For measuring the audio quality, PERCEVAL calculates the difference between the representations of the Reference Signal and the signal Under Test. By applying reasonable assumptions about higher level perceptual and cognitive processes, a number of parameters are calculated and mapped to an estimate of basic audio quality for the Signal Under Test. An optimized mapping is obtained minimizing the difference between the objective quality distribution and corresponding distribution of the mean subjective quality ratings for available data set [2].

2.3.6 POM

Perceptual Objective Measurement (POM) [15] quantifies a certain amount of degradation that may occur between a Reference signal and its "degraded" version. This is accomplished by comparing the internal basilar representations of both signals. The basilar representation models the different processes undergone by an audio signal when traveling through human ear. Thus, calculation of the internal

representation of an audio signal forms the primary stage of POM. The excitation pattern (in dB) spread over the basilar membrane models the firing rate of the neurons along the basilar membrane. This process of calculating the excitation pattern is called the artificial ear. Now that we have two internal representations of the signals, they can be compared against each other. POM determined whether or not the difference is audible –i.e., it is a detection process [2].

Like other metrics, POM also uses a 40 ms Hann-windowed DFT (with a 50% overlap between successive windows). The number of analysis basilar channels is 620. Remaining parts of auditory model are identical to that used in PAQM and PERCEVAL [2]. POM outputs the probability of detecting a distortion between the two compared signals, as well as a basilar distance that represents the perceptual gap between the two compared excitations [2].

2.3.7 The Toolbox Approach

Toolbox uses a three step approach to measure the perceived distance in audio quality of an audio test signal in relation to an audio Reference Signal. The method is based on well-known perceptual models which are used to describe the perceptual representation of the differences between the two audio signals. It also includes a weighting procedure for the perceived audio quality of a stereo test signal, taking into account the effect of both right and left channels.

Step 1 is based on the calculation of the specific loudness, calculated according to [6] using an 2048 point FFT, windowed with a 40 ms duration Hann window. The whole window is shifted by increments of 10 ms in addition to the

temporal pre- and post-masking which are applied as described in Zwicker [6]. Other perceptual parameters such as integrated loudness, partially masked loudness, sharpness and the amount of pre-echoes are calculated as a result of a pre-processing stage for the next steps [2].

The second step of toolbox includes weighting procedures which depend mainly on amount of the perceived difference in loudness and the variation of loudness in time [2].

The third step of toolbox includes the generation of a set of intermediate toolbox output values which are based on a statistical analysis of the values obtained in steps 1 and 2. The output of this analysis includes the mean, maximum and r.m.s values, as well as the standard deviation of the mean values. A weighted sum of these intermediate toolbox values is used for the final fitting of the perceptual distance between the Signal Under Test and the Reference Signal. These output data may be fitted to the Subjective Difference Grade, if necessary, using either a linear or higher order polynomial function [2].

2.4 ITU-R BS.1387

The ITU Radio communication Assembly, considering the various reasons cited in [2] proposed an objective metric which utilizes the objective metrics discussed in section 2.3. The ITU-R BS.1387-1 observes the input Reference Signal and the Signal Under Test, and extracts 11 features called MOVs (model output variables). These MOVs are mapped to the ODG (Objective Difference Grade) using

an artificial neural network. The artificial neural network has a hidden layer with three input nodes. The BS.1387 is treated in greater detail in chapter 4.

3. SUBJECTIVE TESTING

3.1 Introduction

Subjective testing forms a crucial part in many applications like audio codec evaluation, perceptual audio quality testing and testing audio compression systems as it allows us to quantify the compression performance of the codec relative to the retained human perceptual quality. There has been a lot of work done on MPEG-4 audio algorithms like AAC, BSAC and Twin VQ. Tests performed in [16a] cannot be used in the present context for to the following reasons: 1) only higher bitrates-greater than 64 kb/s were used in the BSAC comparisons, 2) the Comparison Category Rating (CCR) approach used in recommendation ITU-R BS.1116 for audio having large impairments was not followed, 3) the performance of TwinVQ was not directly evaluated against BSAC, 4) BSAC was not operated in scalable mode (i.e., the BSAC mode was not used to produce bitstreams at lower bitrates).

3.2 Comparison Category Rating as done in ITU-R BS.1116

Subjective tests for low bitrate audio (highly impaired audio) are based on Recommendation ITU-R BS.1116. These tests are carefully designed to come as close as possible to a reliable judgement of audio quality. However, one cannot expect these tests to fully reflect actual perception. The basic principle of this particular test can be briefly described as follows. The listener can select between

three sources “A,” “B” and “C”. The known Reference signal is always assigned to “A”. The hidden Reference Signal and the Signal Under Test are simultaneously available but are randomly assigned to “B” or “C” for each trial.

The listener is asked to rate the quality of “B” compared to “A” and “C” compared to “A” according to the continuous five-grade impairment scale. One of the sources, “B” or “C” should be indiscernible from “A”; the other may reveal impairments [2]. Any perceived difference between reference and test signal must be treated as impairment. Only one attribute, “Basic Audio Quality” is used. It is defined as the global attribute that includes any difference between the reference and Signal Under test [2].

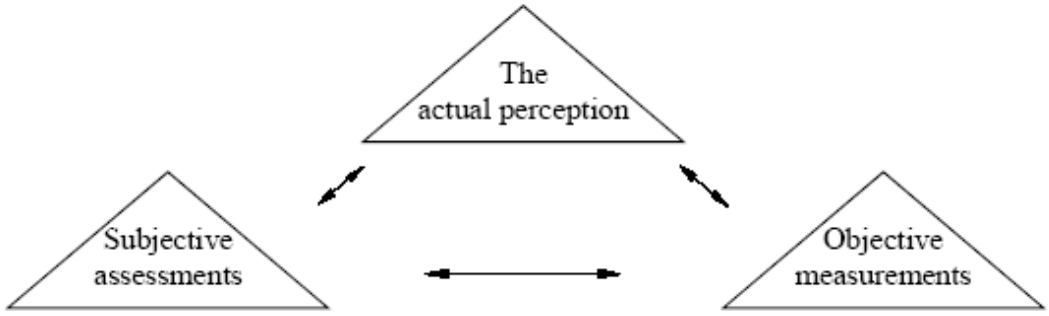

Fig. 3.1: Validation Model

The grading scale is treated as continuous with “anchors” borrowed from the ITU-R five-grade impairment scale given in Recommendation ITU-R BS.562 as shown in Fig. 3.2.

The analysis of the results from a subjective listening test is based on the Subjective Difference Grade (SDG) which is defined as

$$SDG = Grade_{\,Signal\ Under\ Test} - Grade_{\,Reference\ Signal}$$

The SDG values range from 0 to -4 where 0 corresponds to an imperceptible impairment and -4 to impairment judged as very annoying [2].

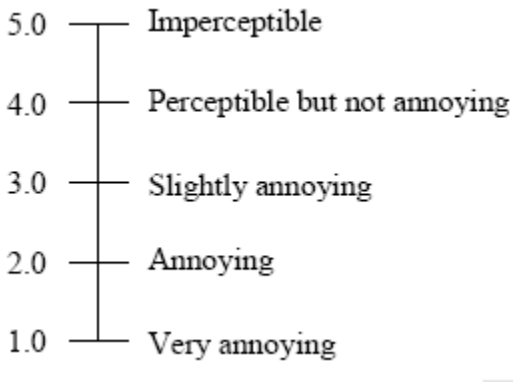

Fig. 3.2: The ITU-R five grade impairment scale

3.3 Testing procedure followed in this thesis

As our tests involve signals with large impairments, we use the Comparison Category Rating (CCR) approach as described in [16]. The test sequences used for the tests are the same sequences which were used in [4], [5]. For these tests, 21 subjects sat alone in a room and listened through headphones to a pre-recorded audio file being played back from a computer. Altering the playback of the test sequence was not allowed, but the subject had complete control over the volume. The test subject was presented with two sequences and asked to rate their quality relative to one another. The scoring metric followed is presented in Fig. 3.3. The seven test sequences used are from a broad variety of music types as shown in table 3.1. Two of them are from the MPEG-4 test set while the others are from various classical and popular music sources.

The subjects made 20 comparative evaluations. These evaluations are later used in our new metric design and analysis.

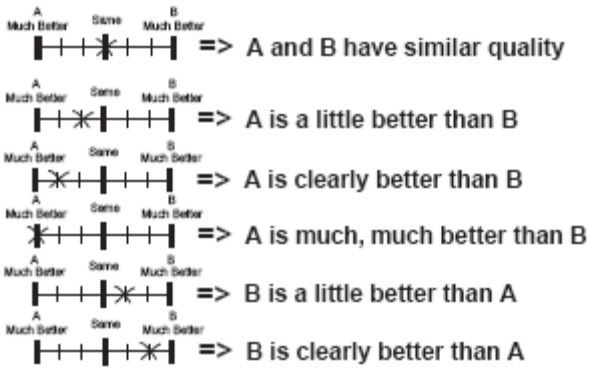

Fig. 3.3: Subjective scoring criteria

Table 3.1. Sequences used in subjective tests.

Sequence Name	Length (seconds)	Type
Pat Benetar	9	rock
Excaliber (movie)	24	classical
Harpsichord	16	MPEG 4 test sequence
Quartet	23	MPEG 4 test sequence
Ronnie James Deo	5	rock
Room with a view (movie)	15	opera
2001: Space Oddessy (movie)	17	classical

3.4 Results of Subjective testing

The nomenclature for the audio test sequences used is the same as used in [3], [4]. The label “bsac16” is used to represent a sequence that is first compressed at 64 kb/s using BSAC to generate a bitstream and is then decoded at 16 kb/s. Similarly, “bsac32” implies that encoding is done at 64 kb/s and then decoding is done at 32 kb/s. Sequences labeled “tvq16”, “tvq32” and “aac32” are encoded and decoded at 16 or 32 kb/s depending on the suffix using Twin VQ and nonscalable AAC,

respectively. The "filtered" label indicates that the input sequence is digitally filtered prior to 64 kb/s → 16 kb/s encoding/decoding (i.e., "bsac16") by a 31- tap linear phase lowpass filter having a cutoff frequency of approximately 6 Hz and stopband attenuation of about 22 dB. The effect of this filtering is that the encoder is forced to allocate more bits to the lower frequency components of the signal. The "total32" category simply summarizes the 32 kb/s results for Twin VQ and non-scalable AAC.

Table 3.2. Results of subjective testing.

Comparison	***Mean***	***Std. Deviation***	***99% Conf. Int.***
tvq16 / bsac16	2.2	0.64	±0.36
filtered / bsac16	1.17	0.92	± 0.52
tvq32 / bsac32	-0.17	0.98	± 0.55
aac32 / bsac32	0	0.71	± 0.4
total32 / bsac32	-0.08	0.85	± 0.4

Table 3.2 summarizes all the results of our subjective testing averaged over all the audio sequences and subjects. The mean score in the table shows how better the first algorithm sound compared to the second on the scale shown in Fig. 3.1. A "-3" indicates that the second algorithm is much, much better relative to the first while a "+3" indicates that the first is much, much better relative to second. Thus, a "0" in the Table 3.2 indicates that the two algorithms (when averaged over all test responses and all test sequences) are equivalent while a "3" indicates that the first is much much better than the second. The order in which the two algorithms are presented to the test subjects is randomized for each test entry and the results are permuted appropriately to generate these statistics. The third column contains the standard deviation as estimated from the subjective data. The fourth column contains the 99% confidence

interval– i.e., the interval that we are 99% certain contains the true mean assuming the underlying probability distribution is Gaussian.

From the table it can be seen that Twin VQ at 16 kb/s scores 2.2 on a scale of 3.0 relative to scalable BSAC, which indicates that the perceptual quality of Twin VQ is somewhere between much, much better and much better on the scale in Fig. 3.1. In contrast, the performance of BSAC at 32 kb/s is essentially the same as nonscalable algorithms- Twin VQ and AAC. Even with a 99% confidence interval of ±0.36, it is obvious that Twin VQ performs far better at 16 kb/s than scalable BSAC. Clearly, the scalability of BSAC is not good in perceptual sense, and there is a wide scope for improvement in scalable algorithms.

4. RECOMMENDATION ITU-R BS.1387-1

4.1. Introduction

Because of the time consuming nature of the human subjective testing, an objective that is capable of accurately predicting the subjective quality of audio is highly desirable. Existing measures like Signal to Noise ratio (SNR) and Total Harmonic distortion (TDR) do not reliably reflect the perceived quality of audio. During the 1990s, a committee was formed by the ITU to study this problem and make recommendations. Many algorithms were submitted for the recommendation and their best features were combined to form a final recommendation called the ITU-R BS.1387 [2]. The BS.1387 actually specifies two different quality metrics – the "basic" version which uses a short time fast Fourier transform (ST-FFT) and an "advanced" version which uses a filter bank model. We use the "basic" version for the comparative study which follows and hence we will discuss only it.

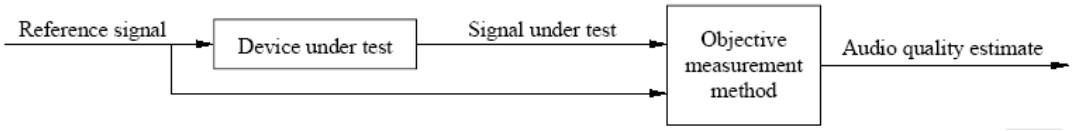

Fig. 4.1: Block diagram for making objective measurements

The objective measurement of the ITU-R BS.1387-1 measures audio quality and outputs a value which corresponds to the perceived audio quality. Fig. 4.1 shows the block diagram of the process of making objective measurements. The measurement method models the human auditory system for which different physiological and psycho-acoustical effects are modeled by the intermediate stages of

the model. In the "basic" version, the original and the reconstructed sequences are decomposed using ST-FFT and 11 features called the model output variables (MOVs) are extracted from them. These intermediate outputs are then used to characterize audio artifacts. The MOVs include parameters such as bandwidth of original and reconstructed audio sequences, noise loudness, noise to mask ratio and modulation differences [2] and, after calculation, they are fed into a neural network which determine the overall quality of the reconstructed audio signal–the Objective Difference Grade (ODG)–on a scale of 0 to -4 where 0 indicates that the reconstructed signal is of indistinguishable quality from the original signal whereas a value of -4 indicates that the reconstructed audio is highly degraded relative to the original. Subjectively, perceived quality is thus given as a measure of the ODG [2].

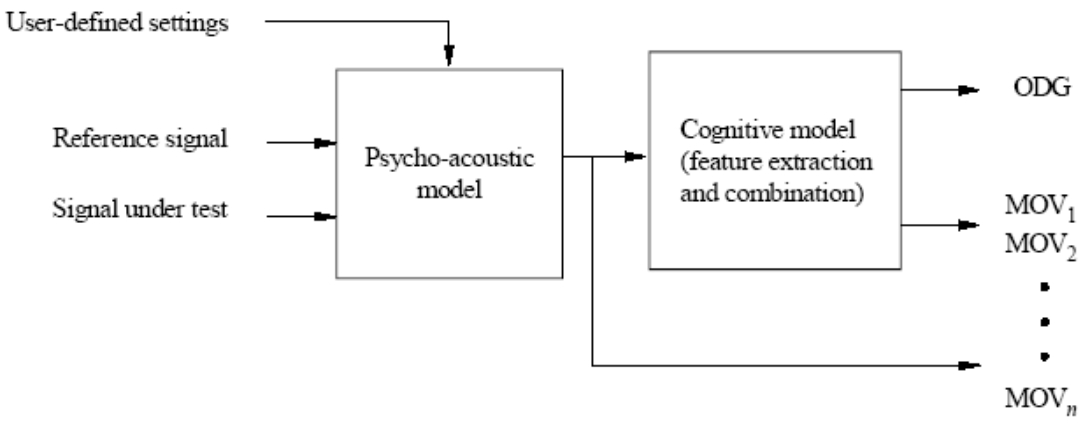

Fig. 4.2: Processing stages in ITU-R BS.1387-1

4.2 Psycho-acoustic Model

The "Basic" version uses a ST-FFT approach in the psycho-acoustic model, as shown in Fig. 4.2, to transform successive frames of the time-domain signal into a

basilar membrane representation. The frequency domain information is next mapped to the pitch domain which gives a pitch scale representation, the psycho-acoustic equivalent of frequency. Two different ways for calculating MOVs are used. The masked threshold concept is used for calculating some MOVs whereas some others use a *comparison of internal representations*- shown in Fig. 4.3. The first concept calculates a masked threshold using psycho-acoustical masking functions and the MOVs are based on the distance of the physical error signal to this masked threshold. In the comparison of internal representations method, the energies of both the Reference signal and the Signal Under Test (SUT) are spread to adjacent pitch regions to obtain excitation patterns. MOVs are based on a comparison of these excitation patterns. Excitation and the masked threshold as a function of time and frequency are the key outputs of the psycho-acoustic model.

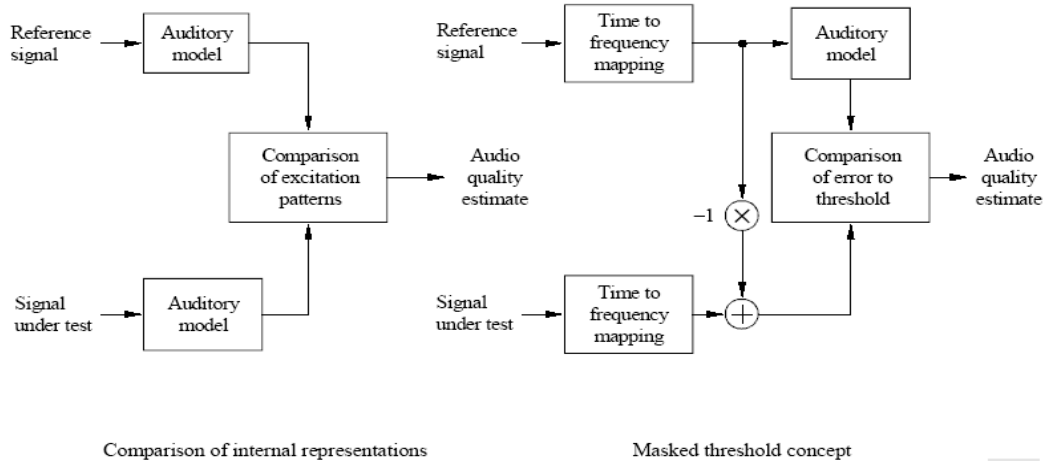

Fig. 4.3: Psycho-acoustic concepts used in different perceptual measurements schemes

4.3 Cognitive Model

The cognitive model condenses the information in the frames of sequences produced by the psycho-acoustic model. Important sources of information for making quality measurements are the differences between Reference Signal and the Signal Under Test in both the frequency and pitch domain. In frequency domain, the spectral bandwidths of both signals are measured as well as the harmonic structure of the error. Error measures in pitch domain are obtained from the envelope of the excitation modulation along with its magnitude.

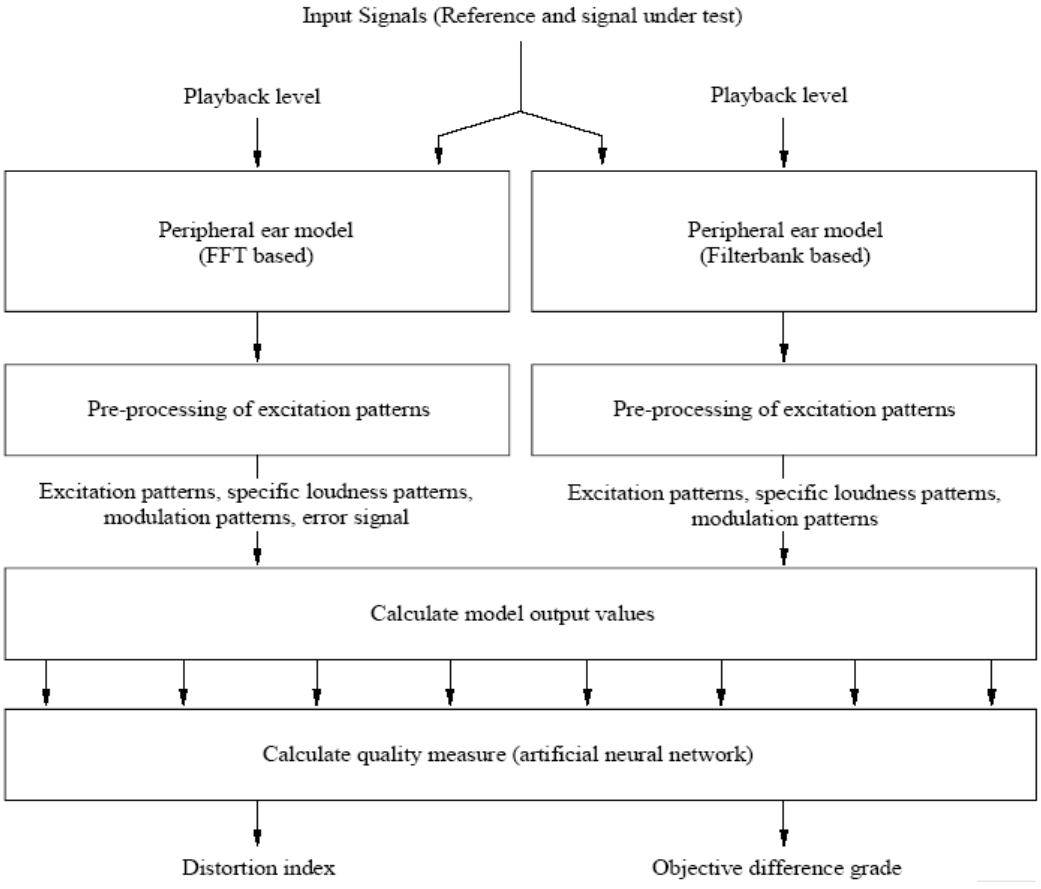

Fig. 4.4: Block diagram of the measurement scheme

As shown in Fig. 4.4, BS.1387-1 contains a peripheral ear model (FFT based in the "basic" version) and a pre-processing stage. The resulting output of the preprocess stage is used to calculate the psycho-acoustically based model output variables (MOVs), and a neural network maps the 11 MOVs to a single value –ODG.

4.4 Ear Model and Pre-processing

A block diagram showing an overview of the ear modeling and pre-processing stages is shown in Fig. 4.4. The input of the FFT-based ear model is a 48 kHz sampled and time aligned reference and a test signal, both cut into frames of 0.042 s (2048 samples) with a 50% overlap. Each frame is transformed to the frequency domain using a Hann window and an FFT. A weighting function modeling the outer and middle ear frequency response is applied to the spectral coefficients. By grouping the weighted spectral coefficients together into critical bands, transformation to pitch domain is performed to yield the energies of the frequency groups. To simulate the internal noise present in the auditory system, a frequency dependent noise is added. A level-dependent spreading function is used to model the spectral auditory filters in the frequency domain, and it is followed by a time domain spreading to model forward masking. These patterns at this stage of processing are referred to as "excitation patterns". Mask patterns are computed from the excitation patterns. The patterns are used to calculate the modulation patterns prior to the final time domain spreading.

The average levels of the Reference and the test Signals are adapted to each other in order to compensate for level differences and linear distortions. This is

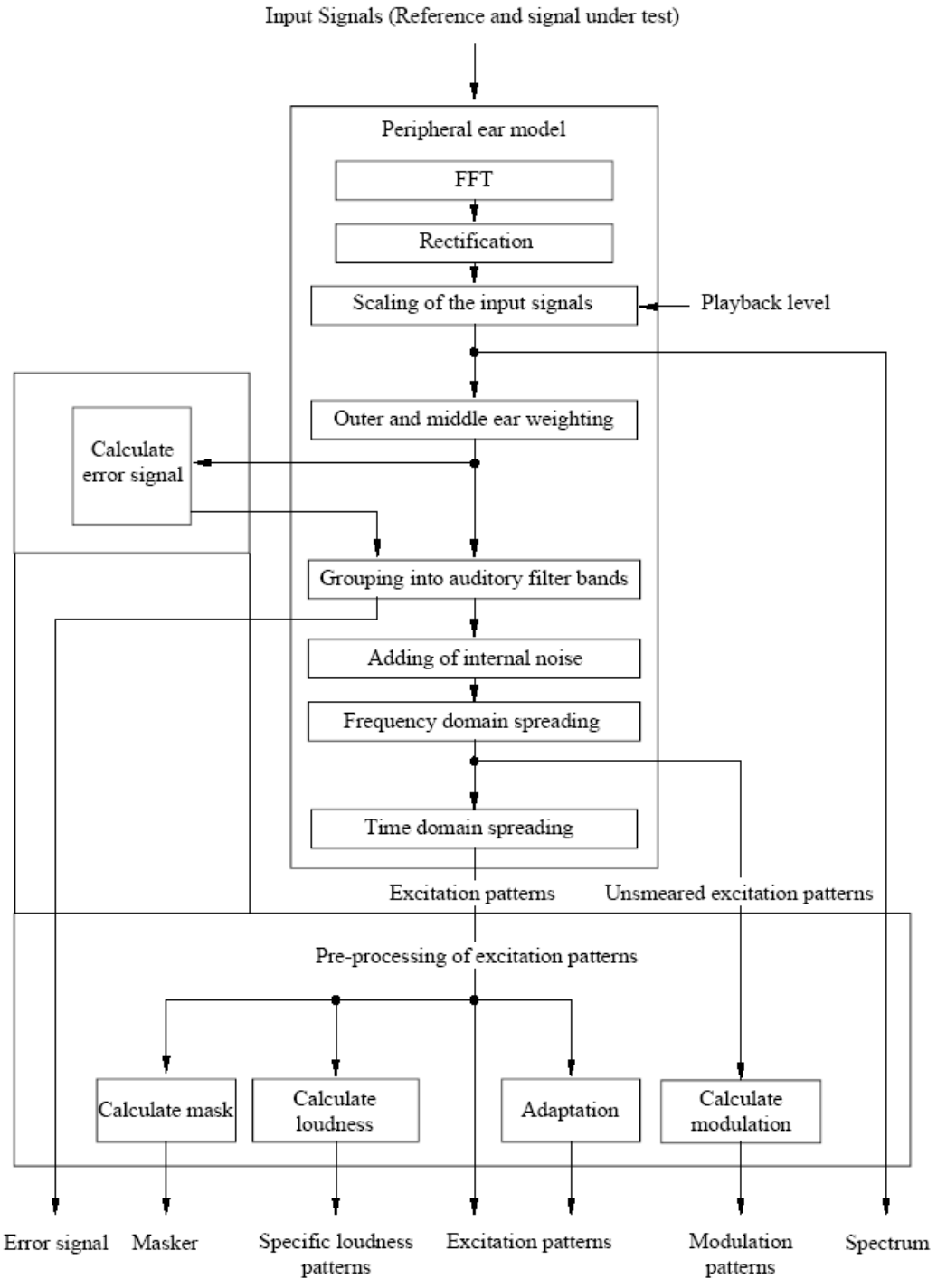

Fig. 4.5: Peripheral ear model and preprocessing of excitation patterns in FFT model

accomplished through the level and pattern adaptation, and the specific loudness patterns are then calculated. Next, the error signal is calculated in the frequency domain by taking the difference between the outer and middle ear filtered magnitude spectra of the reference and the test signal. The modulation differences in temporal envelopes of the Signal Under Test and the Reference Signal are measured by computing a local modulation difference for each filter channel.

4.6 Model Output Variables

Table 4.1: MOVs used in the BASIC version.

MOV	Purpose
$WinModDiff1_B$	Changes in modulation (related to roughness)
$AvgModDiff1_B$	
$AvgModDiff2_B$	
$RmsNoiseLoud_B$	Loudness of the distortion
$BandwidthRef_B$	Linear distortions (frequency response etc.)
$BandwidthTest_B$	
$RelDistFrames_B$	Frequency of audible distortions
$Total\ NMR_B$	Noise-to-mask ratio
$MFPD_B$	Detection probability
ADB_B	
EHS_B	Harmonic structure of the error

Table 4.1 lists all the 11 MOVs in the "basic" version. The MOV $WinModDiff1_B$ is the windowed average of the modulation difference calculated in the FFT-based ear model. Differences in the modulation of the temporal envelopes of the Reference Signal and the Test Signal are measured by computing modulation

differences for each frame as given below in equation 4.1. Mod_{Ref} and Mod_{test} signals are obtained from the psycho-acoustic model, and the average of local modulation differences over all filter channels yields the momentary modulation difference:

$$ModDiff\ [k,n] = w.\frac{|\ Mod_{test}[k,n] - Mod_{Ref}[k,n]\ |}{offset\ + Mod_{Ref}[k,n]} \tag{4.1}$$

$AvgModDiff1_B$ and $AvgModDiff2_B$ are the linear average of the modulation difference calculated from the FFT based ear model. The $AvgModDiff2_B$ and $AvgModDiff1_B$ differ only in the sense that the constants are chosen differently. The modulation Difference MOVs relate to the coarseness of the signal. These values, in general, increase in magnitude as the quality of the signal degrades.

The MOV $RmsNoiseLoud_B$ estimates the partial loudness of additive distortions in the presence of the masking Reference Signal. In particular, it is the squared average of the noise loudness calculated from the FFT-based ear model calculated as in equation 4.2.

$$NL[k,n] = \left(\frac{1}{S_{test}} * \frac{E_{thresh}}{E0}\right)^{0.23} * \left[\left(1 + \frac{\max(S_{test}.E_{test} - s_{ref}.E_{ref},0)}{E_{thresh} + s_{ref} * E_{ref} * \beta}\right)^{0.23} - 1\right] \tag{4.2}$$

E_{Thresh} is the internal noise function and s is calculated according:

$$s = ThreshFac_0 * Mod[k,n] + S_0 \tag{4.3}$$

The *Mod [k, n]* and E_{Thresh} are obtained from the outputs of the pre-processing stage of the model. The values of the momentary loudness are not taken into consideration until 50 ms after the overall loudness for either the left or the right

channel exceeds a value of N_{Thresh} = 0.1 sone for both the Reference and the Test Signals. Also, if the momentary loudness is below a certain threshold NL_{min}, it is set to zero. $RmsNoiseLoud_B$ is the squared average of the noise loudness calculated.

$BandwidthRef_B$ is the bandwidth of Reference Signal; similarly, $BandwidthTest_B$ is the bandwidth of the Test Signal. While calculating the bandwidths, frames with low energy at the beginning and at the end of the items are ignored. The bandwidth MOVs tell us about the linear distortions in the signal. For each frame, the local Bandwidth $BWw_{Ref}[n]$ and $Bw_{Test}[n]$ is calculated according to the Matlab code given in appendix. $BandwidthRef_B$ and $BandwidthTest_B$ are the linear averages of BwRef and BwTest respectively. Only frames with BwRef > 346 are taken into consideration for averaging.

For calculating $RelDistFrames_B$ the number of frames whose logarithms of the ratio P_{noise} to Modulation are above 1.5dB are counted. Frames with low energy at the beginning and at the end of the signals are ignored. The number of frames is calculated according to the equation in 4.4.

$$N = \max\left(10 * \log\left(\frac{P_{noise}[k,n]}{M[k,n]}\right)\right) \geq 1.5dB \qquad k \in [0, Z-1\} \tag{4.4}$$

$TotalNMR_B$ is calculated from the noise and masking values and provides information about the noise to masking ratio. The model output variable Total NMR_B is the linear average of the noise to mask ratio using equation 4.5 where N is total number of frames:

$$NMR_{ltotal}[n] = 10 * \log_{10} \frac{1}{N} \sum_{n} \left(\frac{1}{Z} \sum_{k=0}^{Z-1} \frac{Pnoise[k,n]}{M[k,n]} \right) \tag{4.5}$$

MFPD is the Maximum Filtered Probability of Detection. It models the phenomenon that distortions at the beginning of an excerpt of audio are less severe than at the end of the excerpt due to the fact that humans forget about the early distortion. For the present model, MFPD should be 1.0 since we are using the listening tests calibrated according to Recommendation ITU-R BS.1116.

ADB or Average Distorted Block is dependent upon the number of frames with a probability of detection of the binaural channel above 0.5. It is zero if the number of distorted frames is zero. For all valid frames Q_{bin} is calculated as given by equation 4.7 where q_{left} and q_{right} are given by equation 2.4. Q_{sum} is the total number of steps above threshold:

$$Q_{sum} = \sum_{\forall n} Q_{bin}[n] \tag{4.6}$$

$$q_{bin} = \max\left(q_{left}[k,n], q_{right}[k,n]\right) \tag{4.7}$$

The distortion of the average distorted block is then calculated as

- if $n_{distorted}$ is zero, then ADB = 0 (no audible distortion)
- if $n_{distorted} > 0$ and $Q_{sum} > 0$, then ADB = $\log_{10}((Qsum)/n_{distorted})$;
- if $n_{distorted} > 0$ and $Q_{sum} > 0$, then ADB = – 0.5;

Error Harmonic Structure (EHS_B) is a measure of the harmonic structure in the error. It is calculated by measuring the largest peak in the spectrum of the

autocorrelation function of the error vector (F_0) and the same error vector lagged by 256 samples (F_t) in our case of 48 kHz sampled signal. The equation is given by

$$C = \frac{\overrightarrow{F_0} * \overrightarrow{F_t}}{|\overrightarrow{F_0}| * |\overrightarrow{F_t}|} \tag{4.8}$$

The resulting correlations vector *C* is windowed with a normalized Hann window. After removal of the dc component by subtracting the mean, the power spectrum is calculated by taking an FFT. The maximum peak in the spectrum after the first valley (i.e., the point after which a point in spectrum is greater than the value prior to it) gives the dominant frequency in the auto correlation function. The average value of this maximum over the number of frames multiplied by 1000 gives the value of error harmonic structure MOV.

4.6 Artificial Neural Network

The MOVS obtained are mapped to a single ODG using an artificial neural network. The artificial neural network has one hidden layer and 3 input nodes. Activation function used in the neural network is an asymmetric sigmoid, i.e.,

$$Sig(x) = \frac{1}{1+e^{-x}} \tag{4.9}$$

The network uses *I* inputs and *J* nodes in hidden layer. Mapping is defined by a set of input scaling factors a_{min}[i], a set of input weights w_x[i], a set of output weights w_y[j] and output scaling factors b_{min} and $b_{max.}$The inputs are mapped to a distortion index by

$$DI = w_y + \sum_{j=0}^{J-1}\left(w_y[j] * sig\left(w_x[I,j] + \sum_{i=0}^{I-1} wx[i,j] * \frac{x[i] - a_{\min}[i]}{a_{\max}[i] - a_{\min}[i]} \right) \right) \tag{4.10}$$

which can be directly mapped to an perceived audio quality as the objective difference grade (ODG). The relation between the DI and the ODG by

$$\text{ODG} = b_{min} + (b_{max} - b_{min}) * \text{sig}(\text{DI}) \tag{4.11}$$

As was discussed earlier, the value of ODG obtained lies between 0 and –4, where a 0 means that the test signal is of equal perceptual quality as the Reference and a -4 means that the perceptual quality of the test signal is worse as compared to the Reference.

5. DESIGNING THE NEW METRIC

5.1 Energy Equalization

In previous research it was found that the perceived quality of audio signal is particularly degraded when isolated "island" of time-frequency energy are formed, particularly in the 2 kHz - 4 kHz region. These can be viewed as the lone time-frequency portion of the input signal remaining after those around them have dropped out due to quantization. Thus, if a reconstructed sequence has more isolated islands in the time-frequency plot than another reconstructed sequence of the same audio signal, then the former sequence should be rated worse than the latter by a quality metric [3]. Energy equalization is developed as such a metric to rate audio at low quality [4].

Energy equalization metric can be viewed as one of the most basic forms of transform coefficient quantization– retaining and encoding only those coefficients whose magnitudes are above a defined threshold. Our approach is to apply a truncation threshold T to the original audio spectrogram (i.e., ST-FFT) and to vary T until the truncated spectrogram has the same energy as the spectrogram after encoding and decoding at a lower bitrate. Thus, T is adjusted until the truncated version of the original spectrogram has the same island-like nature as the reconstructed signal"s spectrogram. In this way, T provides a measure of how island-like the spectrum of the reconstructed signal is. The energy of spectrogram of the reconstructed signal being evaluated is found using

$$e_k = \sum_{i=0}^{total\_blocks} \left(rec\_spec(i,j)_k\right)^2 \tag{5.1}$$

where **rec_spec** is a two dimensional matrix containing the spectrogram of the signal under test, k indexes the type of signal (bsac16, tvq16, bsac32, tvq32, filtered), i denotes time (0 – total_blocks where total_blocks is the number of temporal blocks in the decomposition). Next, we find **m_spec** – a modified spectrum obtained by applying the threshold T_{kn} for each codec k and each audio sequences n original spectrum **o_spec,**

$$\text{m\_spec(i, j)}_{\text{T}_{\text{kn}}} = \begin{cases} o\_spec, if \mid o\_spec(i,j) \mid \geq T_{kn} \\ 0, if \mid o\_spec(i,j) \mid < T_{kn} \end{cases} \tag{5.2}$$

The energy of above equation is then calculated as

$$e_{Tkn} = \sum_{i=0}^{total\_blocks} \left(m\_spec(i,j)_{Tkn} \right)^2 \tag{5.3}$$

and is then compared to the energy of the reconstructed signal as given by (5.1). An iterative procedure is followed over the threshold T_{kn} such that

$$\begin{aligned} e_{T_{kn}} < e_k &\Rightarrow T_{kn} = T_{kn} - \Delta \\ e_{T_{kn}} > e_k &\Rightarrow T_{kn} = T_{kn} + \Delta \end{aligned} \tag{5.4}$$

where Δ is the step size. The optimization procedure ends in finding the value of T_{kn} which results in equal energy, i.e. $e_{Tkn} = e_k$. An optimal solution can always be found since for any $T_{kn} \leq T_{pq}$ the energy $e_{Tkn} \geq e_{Tpq}$. This threshold T_{kn} can thus be used to predict the perceived quality of audio as discussed in [4], [5].

We have implemented energy equalization and verified its operation. We have had to make a couple of assumptions, however. If the energy of the reconstructed signal is greater than the energy of the original signal, then the threshold T_{kn} is assumed to be zero. Fig. 5.1 below verifies the performance of energy equalization as

a quality metric at low bitrates. We observe that the least squares fit for energy equalization is better than that for BS.1387. A detailed procedure for obtaining the plots and a discussion of the testing follows.

Differential subjective data are obtained resulting in a 20*1 vector. These values lie in the range of (0,3] and the threshold values T_{kn} are collected for the 20 differential signals. These thresholds are scaled to a range of (0,3]. These rescaled data are plotted and a line which fits the perceptual data against the obtained objective data of the energy equalization resulting in least mean square error is found in Fig. 5.1. The slope of the line fitting the energy equalization is 1.0683 as compared to the slope of line for BS.1387 which is 0.7229. Energy Equalization fares better with a mean squared error (MSE) (calculated as || (sub-(pinv(obj)*sub)*obj))|| of 3.05 as opposed to an MSE of 5.28 for BS.1387. The performance of energy equalization is as shown in [3], [5]. Now that we have verified the performance of energy equalization, we integrate it into the existing BS.1387 as a 12^{th} MOV. To do this, we need to redesign the neural network. Our task is now to train the neural network and obtain the set of optimal weights in the MSE sense.

5.2. Subjective Testing

The recommendation BS.1387 was designed to work on high quality audio signals, but it fails to work well for audio signals having large impairments [3], [5]. For reconstructed audio with low quality, the applicable recommendation is P.830, describing the Comparison Category Rating (CCR) system [4]. This testing

methodology is used to generate the subjective data used for our tests. The subjective testing has already been discussed in earlier chapters.

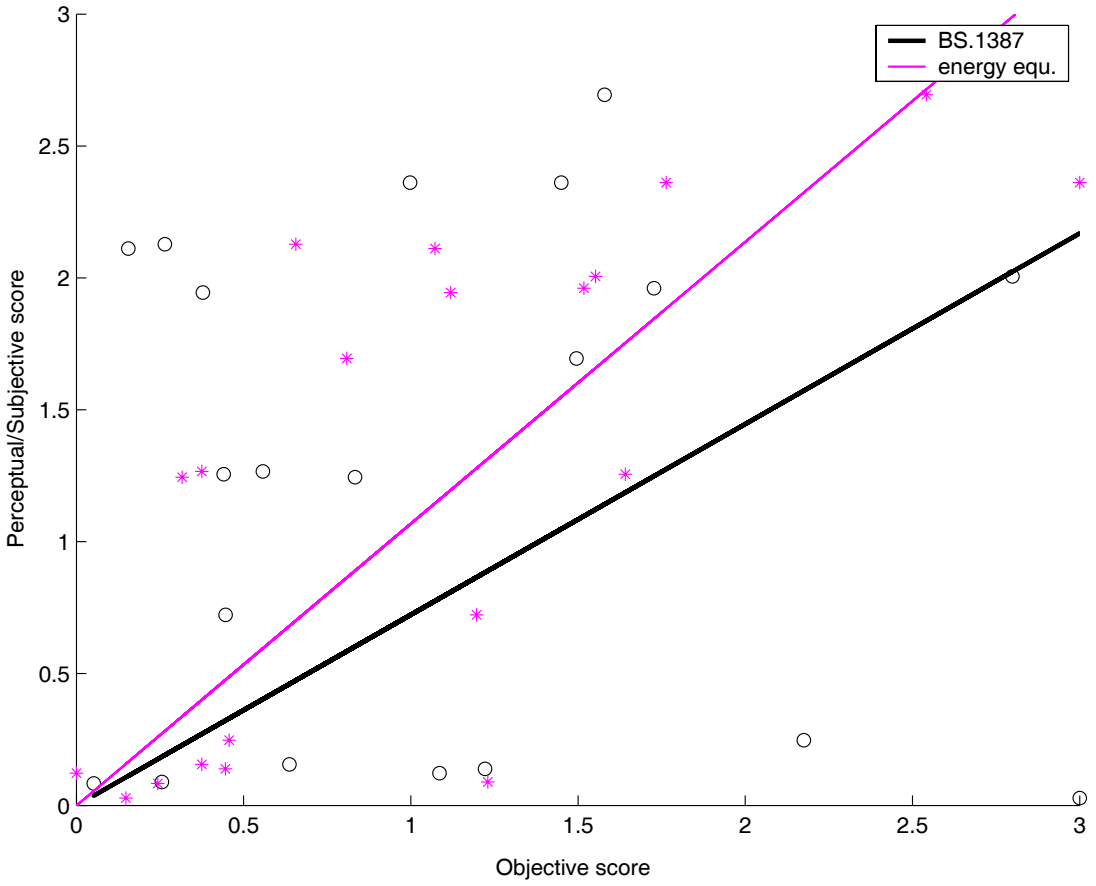

Fig 5.1: Plot showing the least squares fit of energy equalization and BS.1387.

5.3 Designing the Neural Network

The three layer neural network used in BS.1387 is relatively complex. Here, we use a simple one-layer neural network (i.e., a Perceptron network). Since the subjective comparisons are calculated in a differential manner, we take differences in the MOVs in our design and analysis. Specifically, a least squares optimal design

process is used in which the differential MOV vectors constitute the rows of the matrix **M** and a column vector **p** contains the results of the subjective data for the 20 test cases. Thus, **M** is a matrix of size 20*12. 14 test cases are decoded at a bitrate of 16 kb/s whereas 6 test cases are decoded at 32 kb/s. Audio from MPEG-4 codecs of bsac16, tvq16, filtered 16, bsac32, tvq32 types are used. Consequently, we must find the weight vector **w** which best solves the system of linear equations.

$$\mathbf{M}\,\mathbf{w} = \mathbf{p} \tag{5.5}$$

The least squares solution for **w** is given by

$$\mathbf{w}_{opt} = (\mathbf{M}^T\,\mathbf{M})^{-1}\,\mathbf{M}^T\,\mathbf{p} \tag{5.6}$$

and $\mathbf{w}_{opt}$ is thus the optimal in the least mean square error sense. The output from the neural network will now be

$$I = (\mathbf{w}_{opt}{}^T \,.\, \mathbf{m}) \tag{5.7}$$

where **m** is a 12*1 vector containing the 11 MOVs obtained from the BS.1387 and the threshold T_{kn} obtained from energy equalization as the 12th MOV. The above operation to get *I* simply performs an inner product between the weight vector and the model output variables vector.

The indicator *I* for all the audio sequences is obtained and the vector of results for the 20 differential test cases is calculated. $\mathbf{a} = [(I_{k1\ 1} - I_{k2\ 1}), (I_{k1\ 1} - I_{k3\ 1}), (I_{ki\ n} - I_{kn\ m})]^t$. In our case, $I_{k1\ 1}$ and $I_{k2\ 1}$ respectively correspond to scalable BSAC and non-scalable Twin VQ for the "benetar" sequence. The vector **a** is rescaled to a range of

(0,3]– the same range as the scaling used in human perceptual testing data for comparison purposes.

To determine the optimal linear fit for all the points, we solve the following equation. $\mathbf{a}\,x = \mathbf{p}$ **and** whose result is $x_{opt} = (\mathbf{a}^t\,\mathbf{a})^{-1}\,\mathbf{a}^t\,\mathbf{p}$.

This is the same procedure used to find the least squares fit for the energy equalization graph plotted in Fig. 5.1. In other words we are looking for a line through the origin which minimizes the MSE for the given points. Given vector **a**, which has the rescaled metric output values to (0,3], x is the slope of line passing through the origin and **p** is the vector of perceptual data for 20 test cases, we need to find the best slope of x, which minimizes the mean square error. x_{opt} is given by the product of what is also called as the pseudo-inverse of **a** and **p**.

6. PERFORMANCE EVALUATION

The performance of the new metric at low and mid bitrates is impressive compared to that of BS.1387. To evaluate the performance of BS.1387, the "basic" version of standard is coded in MATLAB. The outputs from the metric are then converted into a vector, each element of which is a differential comparison between ODG outputs of two reconstructed sequences: i.e., $\mathbf{a} = [(ODG_{k1\ 1} - ODG_{k2\ 1}), (ODG_{k1\ 1} - ODGk3_{\ 1}), \ldots, (ODG_{k1\ n} - ODG_{km\ n})]^t$ where ODG_{kn} is the Objective Difference Grade for one of the 10 audio sequences n, which has been encoded and decoded using algorithm k. Using these, the vector **a** can be determined.

To be more fair in the comparisons, we rescaled the vector to a range of [0, 3] as was done with energy equalization [3]. The rescaled ODG data is plotted in Fig. 6.1 along with the new metric and their respective least squares fit. It can be clearly seen that the new metric achieves greater predictive performance than the ITU-recommended BS.1387 with an overall MSE of 1.3614 – 75% lower than that of BS.1387 alone. Thus with the 12th MOV- energy equalization, we improve the performance of BS.1387 at low and mid bitrates.

As can been seen from Fig. 6.1, the slope of the line which fits the data of the new metric is 0.8742 with an overall MSE of 1.3614 as compared to the slope of the linear fit of the BS.1387 which is 0.7229 with a considerable higher MSE of 5.28. The linear fit of the new metric is very good given the inherent uncertainty in *any* subjective data.

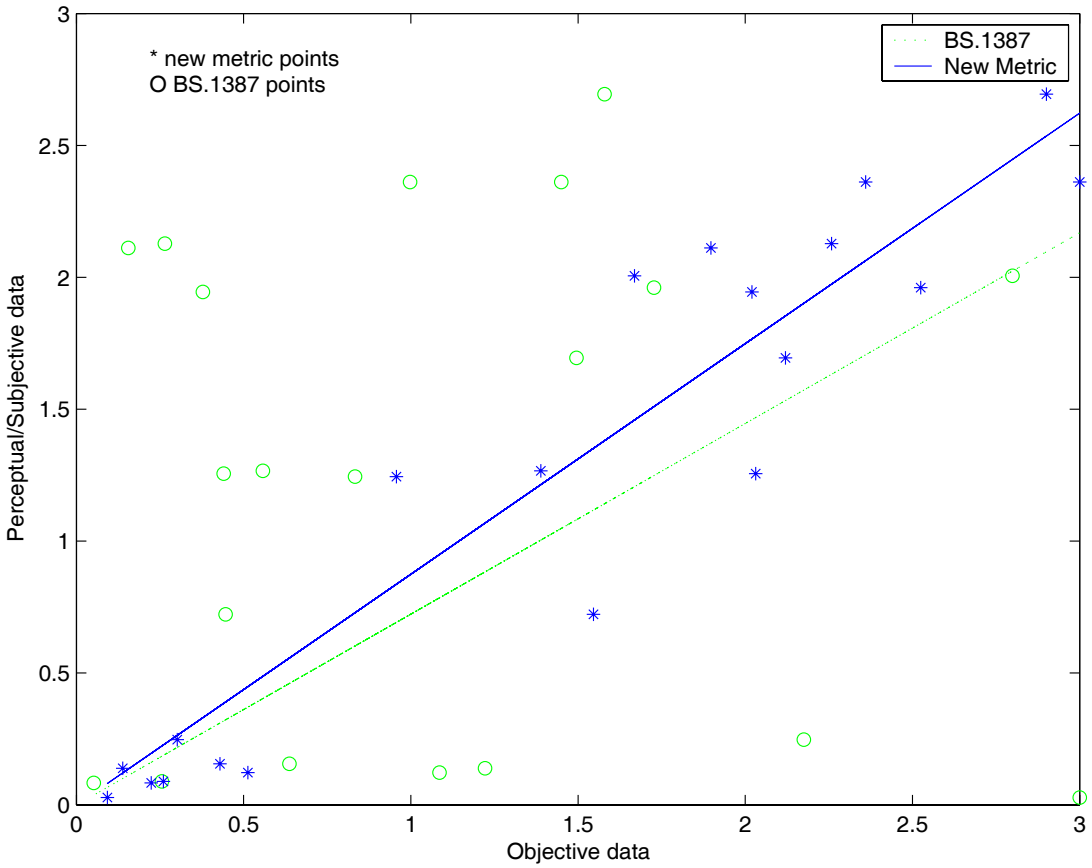

Fig. 6.1. Least squares fit of the objective data versus the subjective data for the new metric and BS.1387-1

As an additional test, we modified the original neural network in BS.1387 and replace in with a single layer neural network designed exactly as the new metric but without energy equalization. As can be seen from the Fig. 6.2, the overall MSE of the modified BS.1387 is approximately halved to 2.6 with a new slope of 0.8476. While performance has improved, using only the 11 MOVs defined in BS.1387 still results in an MSE twice as high as that achieved using energy equalization as well.

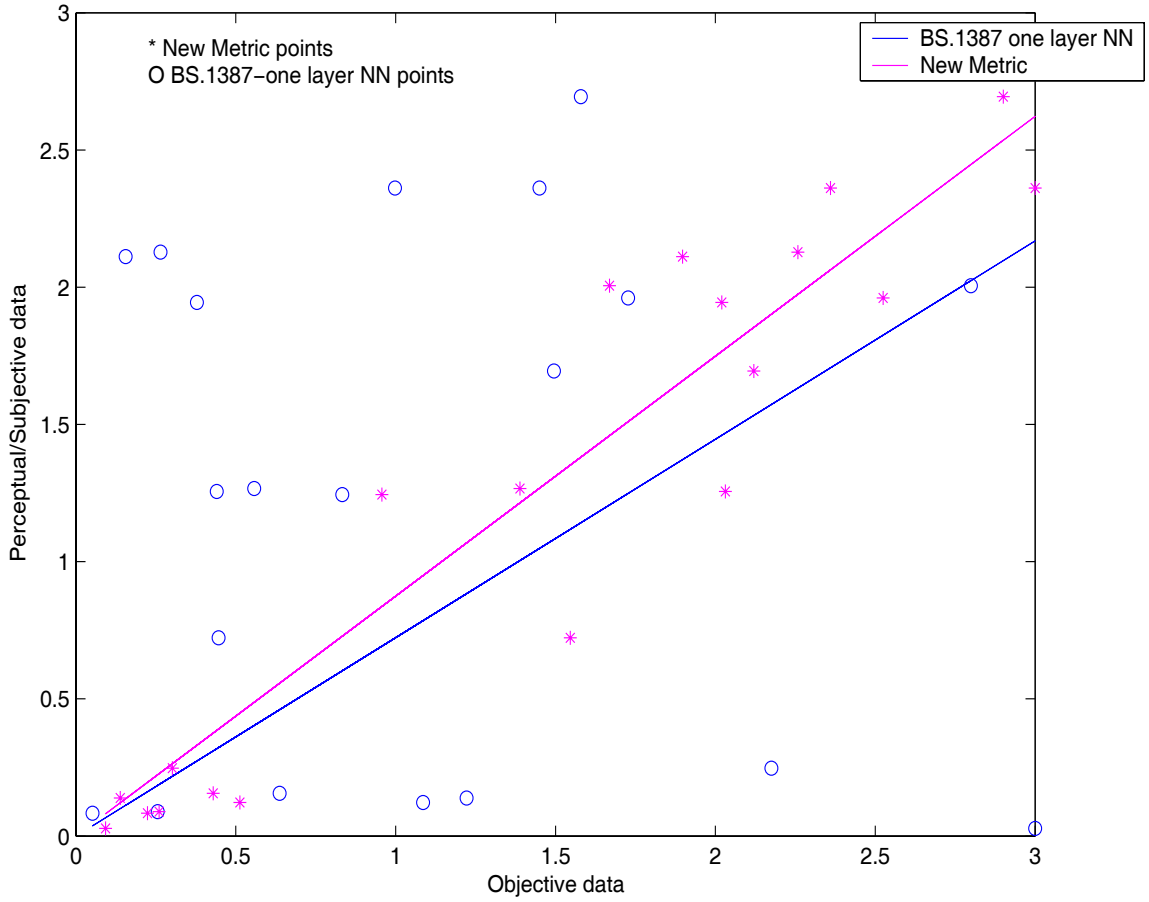

Fig. 6.2. Least squares fit of the objective data versus the subjective data for the new metric and BS.1387-1 modified to a one layer neural network.

The robustness of the new metric as well as that of BS.1387 with modified neural network are evaluated by successively eliminating each of the 20 test cases and then designing the optimal set of weights vector for the new system. This new weight vector is then applied to the test case not used in the optimization. From Fig. 6.3 it can be seen that the squared error for all the 20 test cases is less than 0.4, showing that the new metric is fairly insensitive to changes in training set. The squared error

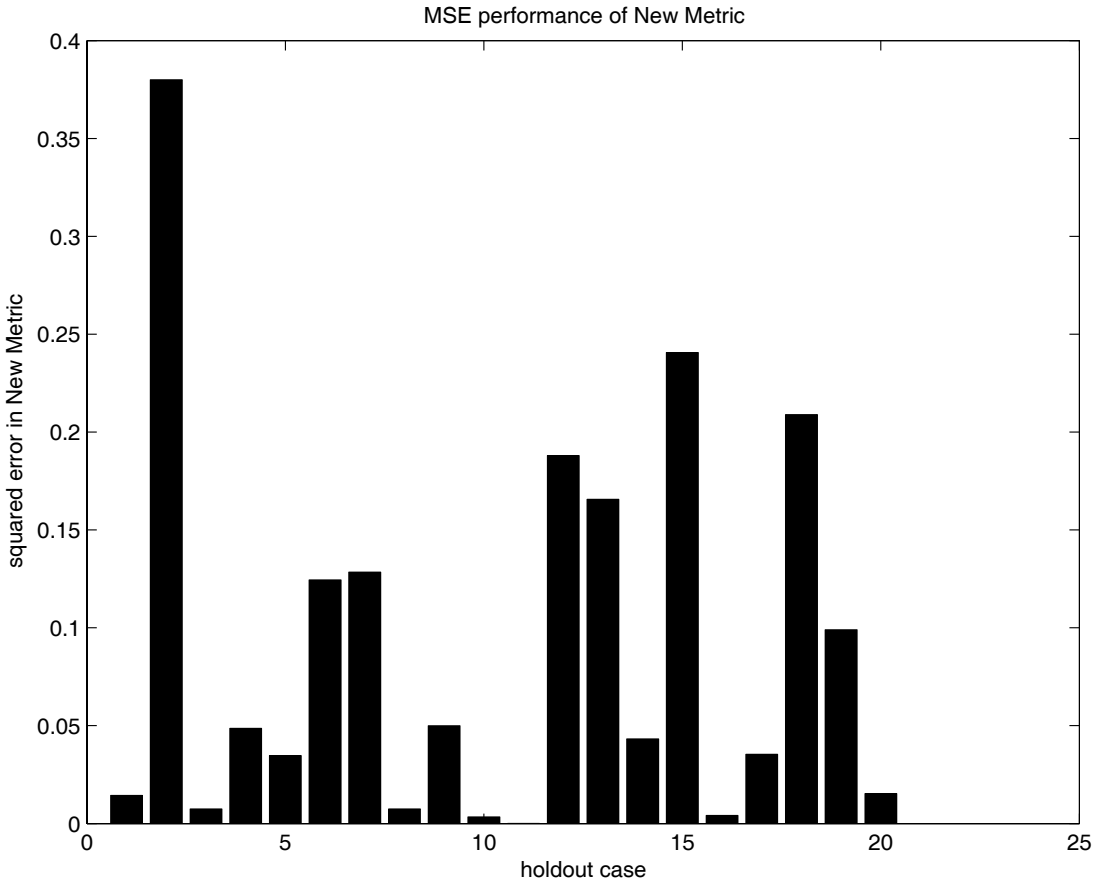

Fig. 6.3. Squared error in new metric when numbered case is not used in design.

for the modified BS.1387 is less than 0.5 for all the 20 test cases as can be seen from Fig. 6.4. We also looked at absolute change in slope of the line for successive elimination for the 20 test cases. It can be noted from Fig. 6.5 and Fig. 6.6 that the absolute change in slope is less than 0.35 for the new metric and BS.1387 modified. These tests suffice to show that the new metric is stable.

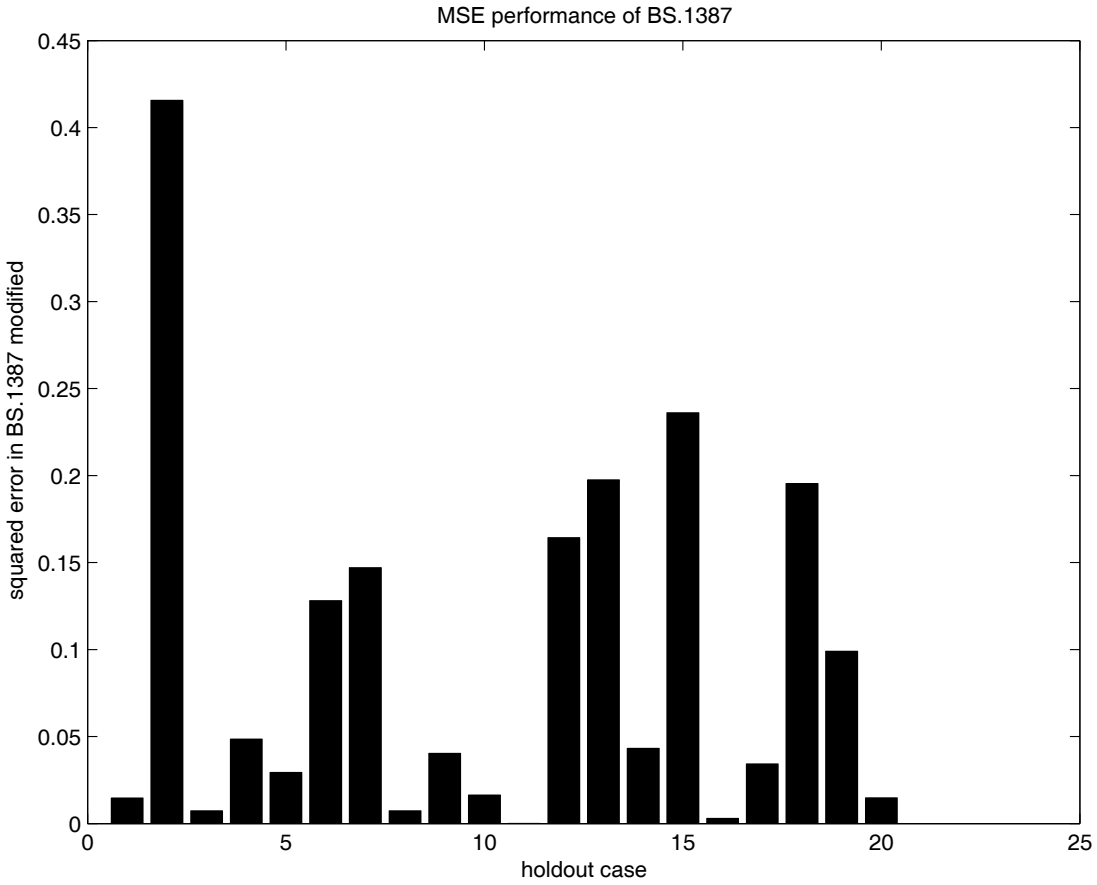

Fig. 6.4. Squared error in BS.1387 modified when numbered case is not used in design.

Table 6.1: Table comparing the mean squares error and slope of least squares fit.

Algorithm	MSE	Slope
New Metric	1.36	0.8742
BS.1387-1 modified	2.6	0.8476
BS.1387-1	5.28	0.7729

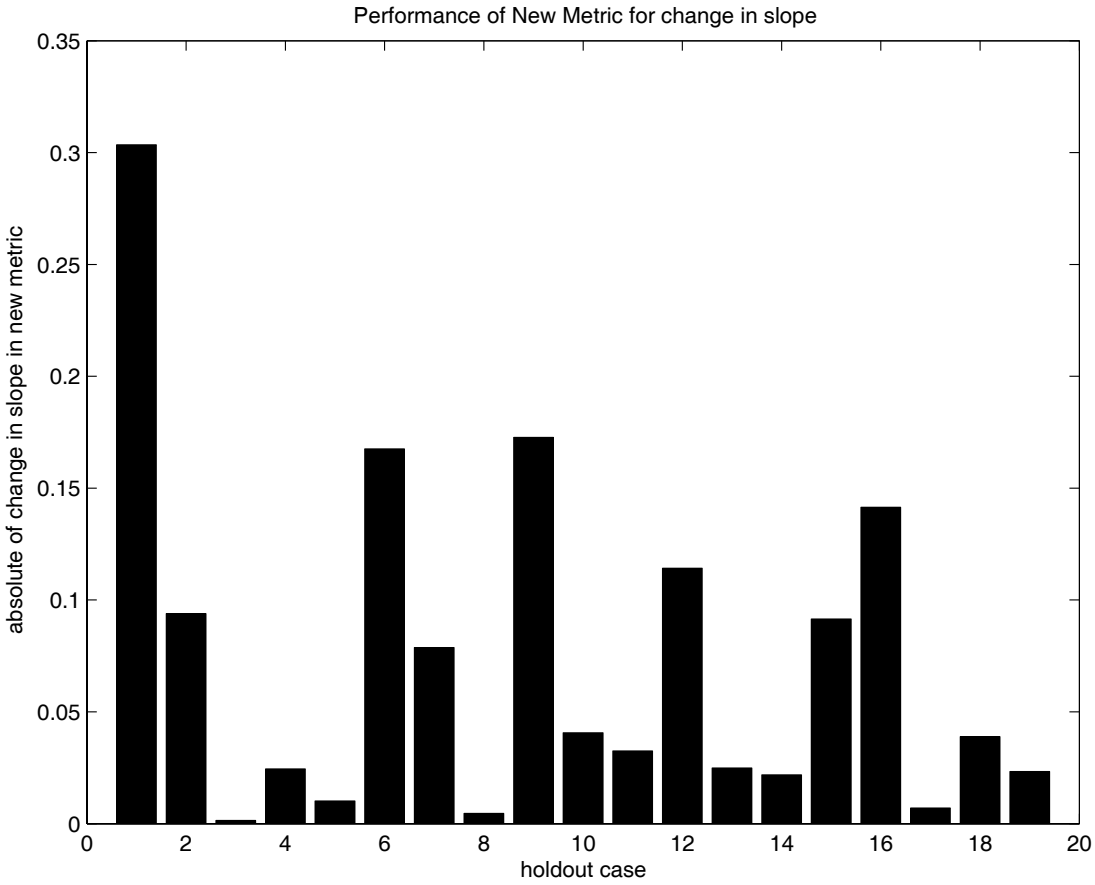

Fig. 6.5. Change of slope in new metric(BS.1387 + Energy Equalization) when numbered case is not used in design.

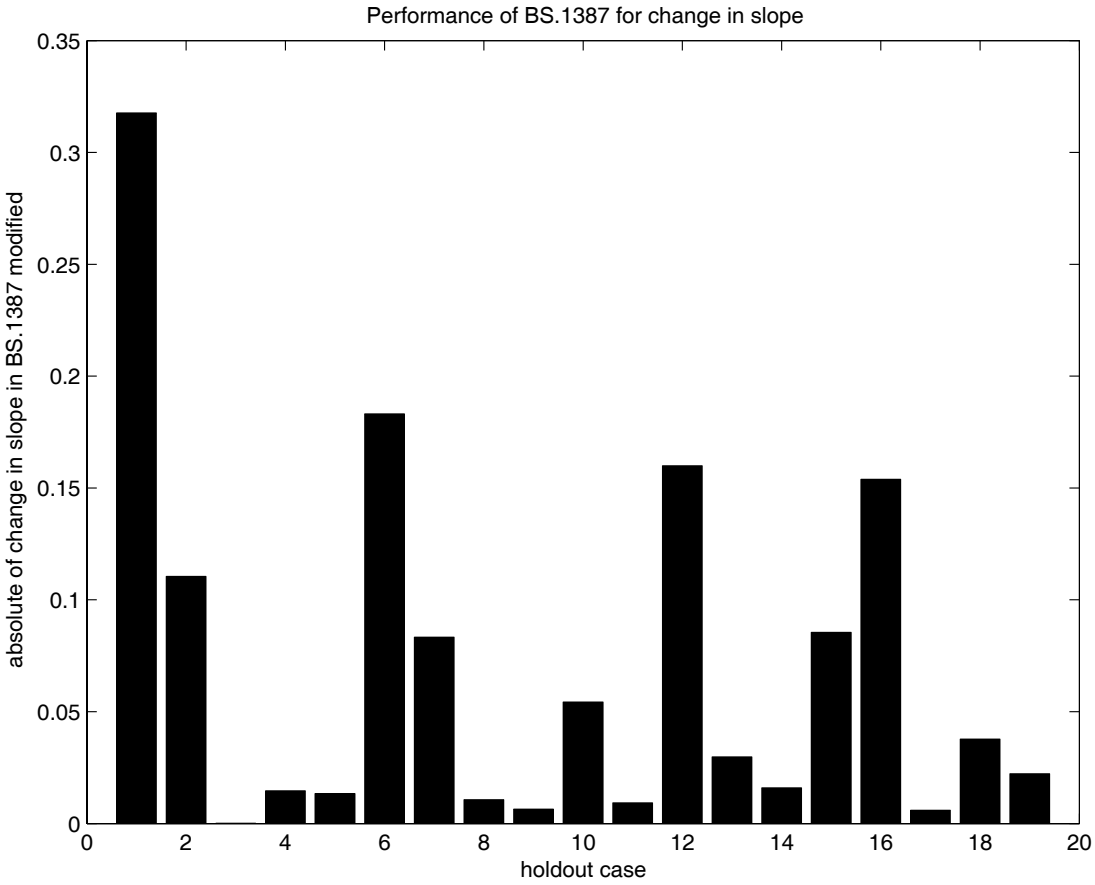

Fig. 6.6: Change of slope in BS.1387 modified for holdout case not used in design

6. CONCLUSION

We have shown in this thesis that the energy equalization technique [3] when added as a new MOV to the ITU-R BS.1387 and with a redesigned neural network can quantify perceptual distortion far more accurately than BS.1387 for audio having high to moderate impairment compared to the original. Furthermore, using a more complex neural network it may be possible to improve the performance of the energy equalization based algorithm presented here. More research and perceptual testing is needed to validate and optimize the metric for high rate/ low distortion audio.

APPENDIX

MATLAB CODE FOR BS.1387 IMPLEMENTATION

```
% Kumar Kallakuri

function [mov_table,ODG,DI] =
model(fin1,fin2,align,trunc)
% Description: This function Stereo--> Left & Right,
BS.1387 implementation
%
% Call Syntax: [output_variables] =
function_name(input_variables)
%
% Input Arguments:refernce,test file,align,trunc
%  Name: fin1,fin2,align,trunc
%  Type: vector,vector,scalar,scalar
%  Description: input signal
%
% Output Arguments:mov_table,ODG,DI
%  Name: mov_table
%  Type: vector
%  Description: output signal
%
% Creation Date: Feb,2004
% Last Modified: 24th Aug,2004
%
% Required subroutines:
%rcomm.m,ear_fft.m,preprocess.m,mov.m,mfpdb.m,pitchpat
%t.m,fband_lut,detpro
% b.m,energythresh.m,tables.m,network.m,audio_align.m,
%
% Notes: Works for stereo or mono and gives an ODG an
% DI also
%
% References:ITU-R BS.1387
%**************************************************
*******************

%------------------
% Check valid input
%------------------
if (nargin ~= 4)
   error("Error (function_name): must have 4 input
arguments.");
end;
```

```
%-----------
% Initialize
 global FS Z col Fres tref ttest Ethresh EPref EPtest
fr_st fr_end nthresh_fr Feref Fetest Fref Ftest Mref
Mtest
 Z=109;
%-----------

%-----
% Main
%-----

tic

%      [xwav FS] =
wavread("e:\Thesis\Codes\test_files\ref\grefcla");
%      ywav =
wavread("e:\Thesis\Codes\test_files\test\gcodcla");


if align==1
    [xwav ywav FS] = audio_align(fin1,fin2,10000);
elseif align==0
    [xwav FS] = wavread(fin1);
    ywav = wavread(fin2);
end
ch = size(ywav,2);

samp = ceil(10*FS/1024)*1024 + 1024;
if size(xwav,1)<samp
    xwav=[xwav; zeros(samp-length(xwav),ch)];
else
    if trunc==1
        xwav = xwav(1:samp,:);
    else
        nf = ceil(size(xwav,1)/1024);
        xzero = nf*1024 - size(xwav,1);
        xwav = [xwav; zeros(xzero,ch)];
    end
end

if size(ywav,1)<samp
    ywav=[ywav; zeros(samp-length(ywav),ch)];
else
```

```
    if trunc==1
        ywav = ywav(1:samp,:);
    else
        ywav = [ywav; zeros(xzero,ch)];
    end
end

xwav = 32768*xwav; ywav= 32768*ywav;

if ch==1
[mov_table,pL,qL] = recomm(xwav(:,1),ywav(:,1));
ADB = adb(pL,qL);
mov_table(5)=ADB;
elseif ch==2
[movL,pL,qL] = recomm(xwav(:,1),ywav(:,1));
[movR,pR,qR] = recomm(xwav(:,2),ywav(:,2));

if size(pL,2)>size(pR,2)
    pL = pR(:,1:size(pR,2));
    qL = qR(:,1:size(pR,2));
else
    pR = pL(:,1:size(pL,2));
    qR = qR(:,1:size(pR,2));
end

ADB = adb(max(pL,pR),max(qL,qR));
mov_table = (movL+movR)*0.5;
mov_table(5)=ADB;
end

% Artificial Neural Network
[ODG,DI] = nnetwork(mov_table);
toc
```

2. FUNCTION RECOMM.M

```
function [mov_table,p,q] = recomm(x1,y1);

global FS Z col Fres tref ttest Ethresh EPref EPtest
fr_st fr_end nthresh_fr Feref Fetest Fref Ftest Mref
Mtest
Z=109;

% modeling ear function->ear_fft
[tref,Fref,Feref E2ref Eref,Mref] = ear_fft(x1);
[ttest,Ftest,Fetest,E2test,Etest,Mtest] = ear_fft(y1);

if size(Fref,2)>size(Ftest,2)
    fr_end = size(Ftest,2);
    Eref = Eref(:,1:fr_end);Fref =
Fref(:,1:fr_end);E2ref = E2ref(:,1:fr_end);Feref =
Feref(:,1:fr_end);Mref = Mref(:,1:fr_end);
    tref = tref(:,1:fr_end);
else
    fr_end = size(Fref,2);
    Etest = Etest(:,1:fr_end);Ftest =
Ftest(:,1:fr_end);E2test = E2test(:,1:fr_end);Fetest =
Fetest(:,1:fr_end);Mtest = Mtest(:,1:fr_end);
    ttest = ttest(:,1:fr_end);
end
col = min(size(Fref,2),size(Ftest,2));
% Preprocessing function->preprocess.m
[EPref,EPtest,Edashref,Edashtest,Modref,Modtest,Pnoise
] = preprocess(Eref,E2ref,Etest,E2test,Feref,Fetest);

%Err = xref-xtest;
% MOV"s Out function-> mov.m
[WinModDiff1,AvgModDiff1,AvgModDiff2,RmsNL,TNMR,BWRef,
BWTest,RDF,EHS] =
mov(x1,y1,Fref,Ftest,Edashref,Mref,Mtest,Modref,Modtes
t,Pnoise);
%[p,q] = detprob(Eref,Etest);
[MFPD,p,q] = mfpdb(Eref,Etest);

format short g
mov_table =
[BWRef/Fres;BWTest/Fres;TNMR;WinModDiff1;0;EHS;AvgModD
iff1;AvgModDiff2;RmsNL;MFPD;RDF];
```

3. FUNCTION AUDIO_ALIGN.M

```
function [scomp,sc2,FS] = audio_align(fin1,fin2,N)
%
%  audio_align(filename1, filename2,N)
%
%  Takes in two WAV files and tries to precisely time
align them.  N
%  is the size of the correlation used
%fin1 = “e:\Thesis\Codes\Audio
files\quar”;fin2=“e:\Thesis\Codes\Audio
files\quarf”);N=10000;

[x,FS] = wavread(fin1);
y = wavread(fin2);

xt = x(30001:N+30000);
yt = y(30001:N+30000);

C = xcorr(xt,yt);

%plot(C,”r”)

[z,dex]=max(C”);

% s corresponds to shift of 2nd argument relative to
first

s = N - dex;

%  Shift 2nd sequence to properly align

if (s < 0)
   z = zeros(-s,1);
   scomp = [z;y];
else
    scomp = y(s+1:max(size(y)));
end

scomp = x(-s+1:max(size(x)));

if max(size(scomp))<max(size(y))
sc2 = y(1:max(size(scomp)));
else
    sc2=y;
end
```

```
%scomp = [scomp,scomp];
%sc2 = [sc2,sc2];

% wavwrite(scomp,44100,16,”sco.wav”);
% wavwrite(sc2,44100,16,”scbene_tbsac64.wav”);
```

4. FUNCTION EAR_FFT.M

```
%ear modeling
% Kumar Kallakuri
%Created:Feb,2004
%Last Modified: 17th June,2004

function [t,F,Fe,E2,E,M] = ear_fft(x);
global FS Z col Fres %fr_st fr_end

kt = 2047;N = 2048;
h = hann(2048)*sqrt(8/3);
% Time Processing--> getting frames from the input
signal
t = zeros(2048,ceil(FS*10/1024)); nf =
floor(length(x)/1024);
for n = 1:nf-1
    t(:,n) = x(1024*(n-1)+1:1024*(n-1)+kt+1);
end

    abs_sum = filter ([1 1 1 1 1],1,abs(t));
    t =
t(:,min(find(max(abs_sum)>200)):max(find(max(abs_sum)>
200)));
    fr_st = min(find(max(abs_sum)>200));
    fr_end = max(find(max(abs_sum)>200));
col = size(t,2);

hn = repmat(h,1,col);
% FFT ->mapping using Hann window and short term FT
tw = hn.*t;
F = zeros(N,col);
n = 1:col;
F(:,n) = abs(fft(tw(:,n),N));

% % calculating  fac
% x = 1:2048*10+1;
% swave = sin(2*pi*1019.5/FS*x);
% %plot([0:1],swave(1:102))
% n = 1;norm = zeros(1,430);fswave = zeros(1,2048);
%
% for i = 1:2048:20481-2048
% fswave = fft(swave(i:i+2047),2048);
% norm(n) = max(abs(fswave));
% n = n+1;
```

```
% end

%F = F*(10^(92/20)/max(norm));
F = F*(10^(92/20)/11361.301)/2048;
F = abs(F);

% Outer and middle ear
W = zeros(1024,1); Fres = FS/2048;
for k = 1:1024
    k1 = (k-1)*0.001*Fres;
    W(k) = -0.6*3.64*(k1)^-0.8 + 6.5*exp(-0.6*(k1-
3.3)^2)-0.001*(k1)^3.6;
end

Fe = zeros(N/2,col);
Fe = F(1:1024,:).*10.^(repmat(W,1,col)/20);  % check
this!

Fe = abs(Fe);

% finding pitch patterns
Pp = pitchpatt(Fe);

fband_lut
Pthresh = 10.^(0.4*0.364*(fc.*0.001).^-0.8);
Pp = Pp + repmat(Pthresh,1,col);

clear kt tw swave x; % Freeing up memory space

% Spreading
L = 10*log10(Pp);

Sl=27;
%Su(k,L(k,n)) = -24 - 230/fc(k) + 0.2*L(k,n);  % CHECK
THIS!

% Spreading for each frequency group, independently
res = 0.25; den_a1 = zeros(109,1);den_a2 =
zeros(109,1);Eline =zeros(109,1);Eline2=Eline;
E2=zeros(109,col);den_b1 =zeros(109,1);den_b2
=zeros(109,1);

for j = 1:Z

   mu1 = 1:j;
```

```
   den_b1(j,1) = sum(10.^((-res*(j+1-mu1)*Sl)*0.1));

   su_tilda = -24 - 230/fc(j);

   mu2 = j+1:Z;
   den_b2(j,1) = sum(10.^((res*(mu2-j-
1)*su_tilda)*0.1));
end

for n = 1:col
    for j = 1:Z

        mu1 = 1:j;
        den_a1(j,1) = sum(10.^((-res*(j+1-
mu1)*Sl)*0.1));

        su = -24 - 230/fc(j) + 0.2*L(j,n);
        mu2 = j+1:Z;
        den_a2(j,1) = sum(10.^((res*(mu2-j-
1)*su)*0.1));
    end
    for k = 1:Z
        num_tilda = zeros(109,1);num=num_tilda;
        for j = 1:Z
            if (k<j)
                num(j) = 10^(L(j,n)*0.1)*10.^((-
res*(j-k)*Sl)*0.1);
                num_tilda(j) = 10^((-res*(j-
k)*Sl)*0.1);

            elseif (k>=j)
                su = -24 - 230/fc(j) + 0.2*L(j,n);
                su_tilda = -24 - 230/fc(j);

                num(j) = 10^(L(j,n)*0.1)*
10.^((res*(k-j)*su)*0.1);
                num_tilda(j) = 10^((res*(k-
j)*su_tilda)*0.1);
            end
        end
        Eline_tilda = num_tilda./(den_b1 + den_b2);
        Eline(k,n) = sum((num./(den_a1 +
den_a2)).^0.4);
        NORMsp(k,1) = (sum(Eline_tilda.^0.4))^(1/0.4);
    end
```

```
        E2(:,n) = ((Eline(:,n)).^(1/0.4))./NORMsp;
%Unsmeared excitation patterns
end

clear Pp Sl Su den_a1 den_a2 den_b1 den_b2 Fsp Eline
Eline2 mu1 mu2 num num2

%Time domain Spreading
Tmin = 0.008; T100 = 0.030;
T = Tmin + 100*(T100-Tmin)./fc;
a = exp(-1024./(FS*T));
Ef = zeros(109,col);

Ef(:,1) = (1-a).*E2(:,1);
for n = 2:col;
Ef(:,n) = a.*Ef(:,n-1) + (1-a).*E2(:,n);
end
E = max(Ef,E2); % excitation patterns

res=0.25;k1= 12/res;
m = zeros(109,1);
m(1:k1)=3;
for i = k1+1:109;
m(i)=0.25*i*res;
end

mtmp = 10.^(m/10);
M = E./repmat(mtmp,1,col); %Mask Patterns
```

5. FUNCTION PREPROCESS.M

```
% Pre-processing
% Kumar Kallakuri
%Created:Feb,2004
%Last Modified: 17th June,2004

function
[EPref,EPtest,Edashref,Edashtest,Modref,Modtest,Pnoise
] = preprocess(Eref,E2ref,Etest,E2test,Feref,Fetest);
global FS Z col Fres Ethresh nthresh_fr

%Level and pattern adaptation
fband_lut
Tmin = 0.008; T100 = 0.050;stepsize=1024;
Pref = zeros(Z,col);Ptest = zeros(Z,col);
T = zeros(Z,1);a=T;
%k = 1:Z;
T = Tmin + 100*(T100-Tmin)./fc;

a = exp(-stepsize./(FS*T));
Ef = zeros(109,col);

Pref(:,1) = (1-a).*Eref(:,1);
Ptest(:,1) =(1-a).*Etest(:,1);
Levcorr =
(sum((Ptest(:,1).*Pref(:,1)).^0.5)/(sum(Ptest(:,1)+0.0
00000001)))^2;

if Levcorr > 1
    ELref(:,1) = Eref(:,1)/Levcorr;
    ELtest(:,1) = Etest(:,1);
else
    ELref(:,1) = Eref(:,1);
    ELtest(:,1) = Etest(:,1)*Levcorr;
end

for n = 2:col;
    Pref(:,n) = a.*Pref(:,n-1) + (1-a).*Eref(:,n);
    Ptest(:,n) = a.*Ptest(:,n-1) + (1-a).*Etest(:,n);
    Levcorr =
(sum((Ptest(:,n).*Pref(:,n)).^0.5)/sum(Ptest(:,n)+0.00
0000001 ))^2;

    if Levcorr > 1
```

```
        ELref(:,n) = Eref(:,n)/Levcorr;
        ELtest(:,n) = Etest(:,n);
    else
        ELref(:,n) = Eref(:,n);
        ELtest(:,n) = Etest(:,n)*Levcorr;
    end
end

for n = 1:col;
num = zeros(Z,1);den=zeros(Z,1);
    for i = 1:n
        num = num + (a.^i).*ELtest(:,n-
i+1).*ELref(:,n-i+1);
        den = den + (a.^i).*ELref(:,n-i+1).*ELref(:,n-
i+1);
    end

    for k = 1:Z
        if den(k)==0
            if num(k)==0
                if k>1
                    R_ref(k,n)=R_ref(k-1,n);
                    R_test(k,n)= R_test(k-1,n);
                else
                    R_ref(k,n)=1;
                    R_test(k,n)=1;
                end
            elseif num(k)>0
                R_test(k,n)=0;
                R_ref(k,n)=1;
            end
        elseif den(k)~=0
            R =num(k)/den(k);
            if R >=1
                R_test(k,n) = 1/R;
                R_ref(k,n) = 1;
            else
                R_test(k,n) = 1;
                R_ref(k,n) = R;
            end
        end
    end
end
M=8; M1= 0.5*M-1;M2 =
M*0.5;S_Rtest=zeros(109,1);S_Rref=zeros(109,1);
```

```
PattCorr_test =
zeros(109,col);PattCorr_ref=zeros(109,col);
for n=1:col
    for k = 1:109

        m1=M1;m2=M2;
        m1=min(m1,k);m2=min(m2,Z-k-1);
        M=m1+m2+1;

        S_Rtest(k) = sum(R_test(k-m1+1:k+m2+1,n));
        S_Rref(k) =  sum(R_ref(k-m1+1:k+m2+1,n));
    end
    if n==1
        PattCorr_test(:,n)= (1-a).*S_Rtest/M;
        PattCorr_ref(:,n) = (1-a).*S_Rref/M;
    else
        PattCorr_test(:,n)= a.*PattCorr_test(:,n-1) +
(1-a).*S_Rtest/M;
        PattCorr_ref(:,n) =  a.*PattCorr_ref(:,n-1) +
(1-a).*S_Rref/M;
    end
end

EPref = ELref.*PattCorr_ref;
EPtest = ELtest.*PattCorr_test;

%Modulation
Ederref = zeros(Z,col);Edashref=zeros(Z,col);Diff_ref
= zeros(Z,1);
Edashtest=zeros(Z,col);Edertest=zeros(Z,col);Diff_test
= zeros(Z,1);

Diff_ref = abs(E2ref(:,1).^0.3);
Diff_test = abs(E2test(:,1).^0.3);

Ederref(:,1)= (1-a).*Diff_ref*FS/stepsize;
Edashref(:,1) = (1-a).*E2ref(:,1).^0.3;

Edertest(:,1)= (1-a).*Diff_test*FS/stepsize;
Edashtest(:,1) = (1-a).*E2test(:,1).^0.3;

for n = 2:col
        Diff_ref = abs(E2ref(:,n).^0.3 - E2ref(:,n-
1).^0.3);
```

```
        Diff_test = abs(E2test(:,n).^0.3 - E2test(:,n-
1).^0.3);

        Ederref(:,n)=a.*Ederref(:,n-1) + (1-
a).*Diff_ref*FS/stepsize;
        Edashref(:,n) = a.*Edashref(:,n-1) + (1-
a).*E2ref(:,n).^0.3;

        Edertest(:,n)=a.*Edertest(:,n-1) + (1-
a).*Diff_test*FS/stepsize;
        Edashtest(:,n) = a.*Edashtest(:,n-1) + (1-
a).*E2test(:,n).^0.3;
end

Modref = Ederref./(1+(Edashref/0.3));
Modtest = Edertest./(1+(Edashtest/0.3));


%Loudness patterns
% implements equ(58)
Ethresh = 10.^(0.4*0.364*(fc*0.001).^-0.8);
Ethresh_rep = repmat(Ethresh,1,col);

k = 1:Z;
s = 10.^(0.1*(-2- (2.05*atan(fc/4000)) -
(0.75*atan((fc/1600).^2))));
s_rep = repmat(s,1,col);

Nref =
1.07664*((Ethresh_rep./(10000*s_rep+0.0000001)).^0.23)
.*((1-s_rep+(s_rep.*Eref./Ethresh_rep).^0.23-1));
Ntest =
1.07664*((Ethresh_rep./(10000*s_rep+0.0000001)).^0.23)
.*((1-s_rep+(s_rep.*Etest./Ethresh_rep).^0.23-1));

Nref = Nref.*(Nref>0); Ntest = Ntest.*(Ntest>0);
Ntotal_ref = 24/Z*sum(Nref);
Ntotal_test = 24/Z*sum(Ntest);

for n = 1:col
    if (Ntotal_ref(n)>0.1) & (Ntotal_test(n)>0.1)
        nthresh_fr = n;
        break
    end
end
```

```
% calculation of the error signal
Fnoise = (Feref(1:1024,:))-(Fetest(1:1024,:));

% finding Noise patterns
Pnoise = pitchpatt(Fnoise);
%Pnoise =  Pnoise*Fres;
```

6. PITCHPATT.M

```
%Pitch patterns
% Kumar Kallakuri
%Created:Feb,2004
%Last Modified: 14th June,2004

function [Pe] = pitchpatt(Fe);
global FS col
fband_lut
Fres = FS/2048;
Pe = zeros(109,col);

  for i = 1:109;
       Pe(i,:)=0;
        for k = 1:1024
            fup = (k-1+0.5)*Fres; flo = (k-1-
0.5)*Fres;

            if (flo>=fl(i)) & (fup<=fu(i))
                Pe(i,:) = Fe(k,:).^2 + Pe(i,:);
            elseif (flo<fl(i)) & (fup>fu(i))
                Pe(i,:) = (Fe(k,:).^2)*(fu(i)-
fl(i))/Fres  + Pe(i,:);
            elseif (flo<fl(i))&(fup>fl(i))
                Pe(i,:) = (Fe(k,:).^2)*(fup-
fl(i))/Fres + Pe(i,:);
            elseif (flo<fu(i)) & (fup>fu(i))
                Pe(i,:) = (Fe(k,:).^2)*(fu(i)-
flo)/Fres + Pe(i,:);
            else
                Pe(i,:) = Pe(i,:);
            end
        end
    Pe(i,:) = max(Pe(i,:),10^-12);
    end
```

7. MOV.M

```
% MOV”s calculation
% Kumar Kallakuri
%Created:Feb,2004
%Last Modified: 17th June,2004

function
[WinModDiff,AvgModDiff1,AvgModDiff2,RmsNL,TNMR,BWRef,B
WTest,RelDistFrames,EHS]=mov2(x1,y1,Fref,Ftest,Edashre
f,Mref,Mtest,Modref,Modtest,Pnoise);
global FS Z col tref ttest Feref Fetest Ethresh EPref
EPtest nthresh_fr
%WinModDiff
w = 1;offset=1;levWt=100;
Diff = abs(Modtest - Modref)./(offset + Modref);
ModDiff = 100/Z*sum(Diff);

% Windowed Average
L=4;N=length(ModDiff);WinX=0;
for n = L:N;
    WinX = (1/L*sum(ModDiff(n:-1:n-L+1).^0.5))^4 +
WinX;
end
WinModDiff = sqrt(1/(N-L+1)*WinX); % MOV--> WinModDiff

% AvgModDiff1B and AvgModDiff2B
negWt1= 1;negWt2= 0.1;offset1= 1;offset2= 0.01; levWt=
100;
fband_lut;
Ethresh = 10.^(0.4*0.364*(fc*0.001).^-0.8);
Ethresh_rep = repmat(Ethresh,1,col);

Etemp= Edashref + levWt*((Ethresh_rep).^0.3);
TempWt = sum(Edashref./Etemp,1);

w =
zeros(109,col);offset=1;levWt=100;Diff2=zeros(Z,col);
%w(find(Modtest>Modref))=1.0;w(find(Modtest<=Modref))=
negWt1;

Diff1 = abs(Modtest - Modref)./(1 + Modref);
ModDiff1 = 100/Z*sum(Diff1,1);

w = zeros(109,col);Diff2 =zeros(109,col);
```

```
w(find(Modtest>Modref))=1;w(find(Modtest<=Modref))=neg
Wt2;
Diff2 = w.*abs(Modtest - Modref)./(0.01 + Modref);

ModDiff2 =100/Z*sum(Diff2,1);

% Delay Averaging --> neglects first 0.5 sec of
measurement
davg = ceil(0.5/(1024/FS));
num2 = sum(TempWt(davg:end).*ModDiff1(davg:end));
den2 = sum(abs(TempWt(davg:end)));

num3 = sum(TempWt(davg:end).*ModDiff2(davg:end));
den3 = sum(TempWt(davg:end));

AvgModDiff1 = num2/den2;
AvgModDiff2 = num3/den3;

% RmsNoiseLoudB mov
Ethresh_rep = repmat(Ethresh,1,col);
E0=1;ThreshFac=0.15;S0=0.5;alpha=1.5;
sref = ThreshFac*Modref + S0;
stest = ThreshFac*Modtest + S0;

beta = exp(-alpha*(EPtest-
EPref)./(EPref+0.000000000001));

num = max(((stest.*EPtest)-(sref.*EPref)),0);
den = Ethresh_rep + (sref.*EPref.*beta);

NL_temp = ((1 + (num./den)).^0.23);
NL = ((Ethresh_rep./(E0*stest)).^0.23).*(NL_temp-1);

NLB = sum(NL(:,nthresh_fr+3:end),1)*24/Z;

NLB = max(NLB,0); % thresholding to zero

RmsNL = sqrt(sum(NLB.^2)/length(NLB));

%Total NMRB
NMR = 1/109*sum((Pnoise./Mref));
TNMR = 10*log10(1/col*sum(NMR));

%RelDistFramesB = Relative disturbed Frames mov
temp_rdf=10*log10(Pnoise./Mref);
```

```
Temp_r = max(temp_rdf,[],1);
%RelDistFrames = length(find(temp_rdf>-1.5))/col;
RelDistFrames = length(find(Temp_r>=1.5))/col;

%% BandwidthRefB and BandwidthTestB

FLevRef = 20*log10(Fref+.00000001); FLevTest =
20*log10(Ftest+0.00000001);
Lf = min(size(FLevTest,2),size(FLevRef,2));
BwRef = zeros(1,Lf); BwTest = zeros(1,Lf);
for i = 1:Lf
    ZeroThresh = max(FLevTest(922:1024,i));

    for k = 921:-1:1
        if FLevRef(k,i) >=10 + ZeroThresh
            BwRef(i) = k; break
        end
    end

    for k = BwRef:-1:1
        if FLevTest(k,i) >= 5 + ZeroThresh
            BwTest(i) = k;break
        end
    end
end
BWRef = mean(BwRef(find(BwRef>346)))*FS/2048;
BWTest = mean(BwTest(find(BwRef>346)))*FS/2048;

% EHSB
%Energy Threshold --> tests for frames with a energy
threshold


energy_ref = sum(tref(1025:2048,:).^2);
energy_test = sum(ttest(1025:2048,:).^2);
Ath = 8000*(max(max(tref))/32768)^2;

nfr=zeros(0,1);
for nt = 1:min(length(energy_ref),length(energy_test))
    %if (energy_ref(nt)>8000) & (energy_test(nt)>8000)
    if  (energy_ref(nt)>Ath) & (energy_test(nt)>Ath)
    nfr = [nfr nt];
    end
end
```

```
%Fnoise = log10(abs(Feref.^2))-log10(abs(Fetest.^2));
Fnoise = log10(abs(Fref.^2)) - log10(abs(Ftest.^2));

% for nfo =
1:min(length(energy_ref),length(energy_test))

% Energy thresholding
Fnn = Fnoise(:,[nfr]);
F0 =  reshape(Fnn,prod(size(Fnn)),1);

Ca = xcorr(circshift(F0,256),F0,256,"coeff");%
normalizes the peak to 1.0
C = Ca(257:512); % taking second half of values as we
want correlation of ft with fo

h = hann(256)*sqrt(8/3);

%C_hann = h.*C;
%C_hdc = C_hann - mean(C_hann);% Removing DC component
  C = C-mean(C);
  C_hdc = h.*C;
PowSpec = abs(fft(C_hdc,256))/256;
%Fig.,plot(PowSpec);

Powtemp = PowSpec(1:128);
dip=1;
for i = 1:127
    if (Powtemp(i+1)>Powtemp(i)) %& (Powtemp(i-
1)>Powtemp(i))
        dip = i; break
    end
end

EHS = 1000*max(PowSpec(dip:128))/size(Fnn,2);
```

8. TABLES.M

```
%I/P & O/P weight tables for Basic Version
% Kumar Kallakuri
%Created:Feb,2004
%Last Modified: 20th May,2004

% Scaling factors for i/p”s of the Basic Version
amin=[ 393.916656;361.965332;-24.0451;1.1107;-
0.2066;0.0743;1.1137;0.9503;0.029985;0.000101;0];
amax =
[921;881.1;16.21;107.14;2.88;13.93;63.26;1145.02;14.82
;1;1];

% Weights for input nodes of Basic version
Wx1=[-0.5027;4.3075;4.9842;0.0511;2.3216;-
5.3039;2.7310;0.6250;3.1029;-1.0515;-1.8047;-2.5183];
Wx2=[0.4363;3.2460;-2.2112;-1.7624;1.7900;-3.4523;-
6.1118;-1.3315;0.8713;-0.9399;-0.5036;0.6548];
Wx3=[1.2196;1.1237;-0.1921;4.3313;-0.7546;-
10.8150;1.5192;-5.9552;-5.9229;-0.1429;-0.6205;-
2.2072];
Wx = [Wx1 Wx2 Wx3];

% Weights for the output node of the Basic Version
Wy =[ -3.817048 4.107138 4.629582 -0.307594];

% Scaling factors for the output of the Basic Version
% ODG
bmin = -3.98; bmax = 0.22;
```

9. DETPROB.M

```
%Detection probablity -->called by mfpdb function
% Kumar Kallakuri
%Created:Feb,2004
%Last Modified: 25th April,2004

function [pc,qc] = detprob(Eref,Etest);
global col
L = zeros(109,col);
Etilda_ref = 10*log10(Eref);
Etilda_test = 10*log10(Etest);

L = 0.3*max(Etilda_ref,Etilda_test) + 0.7*Etilda_test;
s = zeros(109,col);
for n = 1:col
    for k = 1:109
        if L(k,n)>0
            sA = 5.95072*((6.39468)/L(k,n))^1.71332 +
9.01033*10^-11*L(k,n)^4;
            sB = 5.05622*10^-6*L(k,n)^3 -
0.00102438*L(k,n)^2 + 0.0550197*L(k,n) - 0.198719;
            s(k,n) = sA + sB;
        else
            s(k,n) = 10^30;
        end
    end
end
e = Etilda_ref - Etilda_test;
b = zeros(109,col);

b(find(e>0)) = 4; b(find(e<=0)) = 6;
a = (10.^(log10(log10(2.0))./b))./s;

pc = 1-(10.^((-a.*e.^b)));
qc = abs(double(int8(e)))./s;
```

PART B: ENERGY EQUALIZATION

ENERGY EQUALIZATION.M

```
% Created by Dr. Chuck Creusere
% modified by Kumar Kallakuri
% used for energy equalization

% clear all;clc;
 ref =
wavread(“e:\Thesis\Codes\test_files\ref\arefsna”);
 test =
wavread(“e:\Thesis\Codes\test_files\test\scodclv”);
%  function T = energy_equalization(fin1,fin2);
 tic
%  [ref test FS] = audio_align(fin1,fin2,10000);
% ref = wavread(fin1);
% test = wavread(fin2);

o_spec = abs(specgram(ref(:,1),1024,44100,1024,512));
rec_spec =
abs(specgram(test(:,1),1024,44100,1024,512));

e_k = mean(mean(rec_spec.^2))
e_T = mean(mean(o_spec.^2))
delta = 1;T=5;%max(max(o_spec))/2;

dir=1;tol=0.001;cnt=0;i=1;

while ((abs(e_k-e_T) > tol) & (cnt<10))
    i=i+1;
    if i>50 & i<100
        tol=.01;
    elseif i>100
        tol=.1;
    end

    ediff = abs(e_k - e_T);
    T = T + dir*delta;
    m_spec = o_spec.*(o_spec>T);
    e_T = mean(mean(m_spec.^2));

    if (abs(e_k - e_T) > ediff)
        cnt = cnt+1;
        dir = -dir;
```

```
        delta = delta/2;
    end
end
T,e_k,e_T

toc
```

PART C: NEW METRIC DESIGN

```
% Kumar Kallakuri
% Calculates the weights of the neural network and also
generates plots of
% changes in slope and squared error plots
% Generates plots of some of the results
% Aug 27th,2004


clear all;
load("e:\Thesis\Codes\mov_outputs.mat");
mov_tables;

A(1,:)=bene_tvq_bsac16";
A(2,:)=harp_bsac_tvq16";
A(3,:)=rjd_f_bsac16";
A(4,:)=exc_bsac_tvq32";
A(5,:)=quar_bsac_tvq16";

A(6,:)=room_bsac_16f";
A(7,:)=spo_tvq_bsac16";
A(8,:)=bene_tvq_bsac32";
A(9,:)=exc_bsac_tvq16";
A(10,:)=harp_f_bsac16";

A(11,:)=quar_tvq_bsac32";
A(12,:)=rjd_tvq_bsac16";
A(13,:)=spo_bsac_16f";
A(14,:)=room_bsac_tvq32";
A(15,:)=exc_bsac_16f";
A(16,:)=room_tvq_bsac16";

A(17,:)=harp_tvq_bsac32";
A(18,:)=bene_f_bsac16";
A(19,:)=spo_bsac_32o";
A(20,:)=quar_bsac_16f";


sub = [2.694444444;-2.127777778;0.027777778;0.088888889;-
1.961111111;-2.111111111;
1.944444444;-0.083333333;-2.361111111;1.266666667;-
0.247222222;2.005555556;
-1.244444444;0.138888889;-0.722222222;2.361111111;-
0.122222222;1.255555556;0.155555556;-1.694444444];
```

```
%%%%%%%%%%%%%%%%%%%%%%%%
% % code for generating the squared error plot and the
change in slope plot
tmp1=A;tmp2=sub;slo_old=0;slo_old2=0;
for i=1:20
    %%%%%%%%%%%%% plots slope change
    A=tmp1;sub=tmp2;
    if i>1
        sub(i-1,:)=[];A(i-1,:)=[];
    end
    X=pinv(A)*sub;

Igr=[2.5233;-1.9644;-0.0806;-0.2263;-2.1965;-
1.6507;1.7575;-0.1947;-2.0529;1.2082;
-0.2622;1.4517;-0.8324;-0.1207;-1.3452;2.6098;-
0.446;1.7672;-0.3737;-1.8447];

I_obj=[2.3386;-2.0655;-0.1107;0.1137;-2.2867;-
1.5726;1.6584;-0.1874;0;1.4376;
-0.0204;1.8467;-0.949;-0.2816;-1.0763;2.5317;-
0.4317;1.5017;-0.3015;-1.8361];

    if i==1
        slo = pinv(Igr)*sub;
        slop(i)=slo-slo_old;
        slo_old=slo;

        slo_obj = pinv(I_obj)*sub;
        slop_obj(i)=slo_obj-slo_old2;
        slo_old2=slo_obj;
    elseif i>1
        Igr(i)=[];
        slo = pinv(Igr)*sub;
        slop(i)=slo-slo_old;  % calculated difference in
slope

        I_obj(i)=[];
        slo_obj = pinv(I_obj)*sub;
        slop_obj(i)=slo_obj-slo_old2;  % calculated
difference in slope
    end

   %%%%%%%% plots squared error
```

```
   err(i) = (tmp2(i)-
(pinv(Igr)*sub)*(X”*tmp1(i,:)”))/sqrt(1+slo^2);  % err =
sub-a*obj
   err_itu(i) = (tmp2(i)-
(pinv(I_obj)*sub)*(X”*tmp1(i,:)”))/sqrt(1+slo_obj^2);
   clear Igr I_obj
end
bar(abs(slop(2:end)),”k”); xlabel(“holdout
case”);ylabel(“absolute of change in slope in new
metric”);
title(“Performance of New Metric for change in slope”);
Fig.,bar(abs(slop_obj(2:end)),”k”); xlabel(“holdout
case”);ylabel(“absolute of change in slope in BS.1387
modified”);
title(“Performance of BS.1387 for change in slope”);
Fig.,bar(err.^2,”k”); xlabel(“holdout
case”);ylabel(“squared error in New Metric”);
title(“MSE performance of New Metric”);
Fig.,bar(err_itu.^2,”k”); xlabel(“holdout
case”);ylabel(“squared error in BS.1387 modified”);
title(“MSE performance of BS.1387”);

%%%%%%%%%%%
% New Standard

% mov_table = bene_tvq16;
% Wt = pinv(A)*sub;
% I =  (Wt”*mov_table)

%%%%%%%%%%%
% BS.1387  with one layer NN

% mov_table = bene_tvq16(1:11);
% Wt = pinv(A(:,1:11))*sub;
% I =  (Wt”*mov_table)
```

PART E: PLOTS

COMP_BS1387.M

```
% The code plots the graph for obj Vs Sub data points and
plots the best LSE line.
clc;
%%%%%%%%%%%%%%%%%%%%%%
% obj data is the collection of results of BS.1387 over
the audio database.
%%%%%%%%%%%%%%%%%%%%%%
obj = [-0.3369;0.0565;0.64;-0.0545;0.3685;0.0331;-
0.0807;0.011;0.3093;-0.1189;
    -0.464;-0.5972;0.1777;-0.2607;0.0952;0.2129;0.2316;-
0.094;-0.1359;0.319];
sub = [2.6944;-2.1278;0.0278;0.0889;-1.9611;-
2.1111;1.9444;-0.0833;-2.3611;
1.2667;-0.2472;2.0056;-1.2444;0.1389;-0.7222;2.3611;-
0.1222;1.2556;0.1556;-1.6944];

obj = abs(obj);sub=abs(sub);
obj = obj/max(obj)*3; % scaled to make the largest value
3.0

slope = pinv(obj)*sub;
disp("MSE error for BS.1387")
norm(sub-slope*obj)

hold on
scatter(obj,sub,"k");
plot(obj,slope*obj,"k")
 xlabel("Objective score");ylabel("Perceptual/Subjective
score");

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Energy Equalization
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Tkn = [-13.5547;3.5;-0.79;6.5625;8.0938;5.7188;-5.9687;-
1.2968;16;-2;
       -2.4375;-8.28;1.6875;2.375;6.3828;-9.4063;0;-
8.7539;2;4.3125];

Tkn = abs(Tkn);
Tkn = Tkn/max(Tkn)*3;
slope2 = pinv(Tkn)*sub;
```

```
scatter(Tkn,sub,”*”,”m”);
plot(Tkn,slope2*Tkn,”m”)
AXIS([0 3 0 3])
disp(“MSE error for Energy Equalization”)
norm(sub-slope2*Tkn)
%%%%%%%%%%%%%%%%%%%%%%%%
% The New Standard with 12 features and the one layer
neural network
%%%%%%%%%%%%%%%%%%%%%%%%
Ikn=[2.5233;-1.9644;-0.0806;-0.2263;-2.1965;-
1.6507;1.7575;-0.1947;-2.0529;1.2082;
-0.2622;1.4517;-0.8324;-0.1207;-1.3452;2.6098;-
0.446;1.7672;-0.3737;-1.8447];

Ikn = abs(Ikn);
Ikn = Ikn/max(Ikn)*3;
slope3 = pinv(Ikn)*sub;
scatter(Ikn,sub,”*”,”m”);
plot(Ikn,slope3*Ikn,”m”)
AXIS([0 3 0 3])
disp(“MSE error for New Metric”)
norm(sub-slope3*Ikn)

%%%%%%%%%%%%%%%%%%%%%%%%%%
% The BS.1387 with the single layer neural network
results where the
% artificial neural network in the original standard is
replaced with the
% single layer network.
%%%%%%%%%%%%%%%%%%%%%%%%%%

I_obj=[2.3386;-2.0655;-0.1107;0.1137;-2.2867;-
1.5726;1.6584;-0.1874;0;1.4376;
-0.0204;1.8467;-0.949;-0.2816;-1.0763;2.5317;-
0.4317;1.5017;-0.3015;-1.8361];

I_obj = abs(I_obj);
I_obj = I_obj/max(I_obj)*3;
slope4 = pinv(I_obj)*sub;
scatter(I_obj,sub,”d”,”m”);
plot(I_obj,slope4*I_obj,”m”)
AXIS([0 3 0 3])
disp(“MSE error for BS.1387 with a single layer NN”)
norm(sub-slope4*I_obj)
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% New Standard with differentiated weights for low
bitrates and high
% bitrates
% The New Standard with 12 features and the one layer
neural network
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Ikn_LH=[2.3752;-2.0458;0.0305;0.0889;-2.2394;-
2.0317;1.9593;-0.0833;-2.3835;1.4085;
       -0.2472;2.0921;-0.9644;0.1389;-0.7730;2.3102;-
0.1222;1.4770;0.1556;-1.7074];

sub=abs(sub); Ikn_LH = abs(Ikn_LH);
Ikn_LH = Ikn_LH/max(Ikn_LH)*3;
slope3 = pinv(Ikn_LH)*sub;
scatter(Ikn_LH,sub,"*");
plot(Ikn_LH,slope3*Ikn_LH)
AXIS([0 3 0 3])
disp("MSE error for New Metric with differentiated
weights")
norm(sub-slope3*Ikn_LH)

%%%%%%%%%%%%%%%%%%%%
% LEGENDS

%legend("BS.1387","energy equ.")
%legend("BS.1387","New Metric","BS.1387-onelayer NN")
%legend("BS.1387","New Metric")
%legend("New Metric","BS.1387-onelayer NN")
legend("New Metric","New Metric-differentiated wt")
hold off
```

REFERENCES

[1] P. Noll, “MPEG digital audio coding,” *IEEE Signal Processing Magazine*, pp. 59-81, September 1997.

[2] *Methods for objective measurement of perceived audio quality*, Recommendation ITU-R BS.1387-1, 1998-2001, www.itu.int.

[3] C. D. Creusere, “An Analysis of perceptual artifacts in MPEG scalable audio coding,” *Proc. of Data compression Conference*, pp. 152-161, April 2002, Snowbird, UT.

[4] C. D. Creusere, “Quantifying Perceptual Distortion in Scalably Compressed MPEG audio,” *Proc. 37th Asilomar Conf. on Signals, Systems and Computers*, pp. 265-9, Nov. 2003, Pacific Grove, CA.

[5] C. D. Creusere, “Quantifying Perceptual Distortion in Scalably Compressed Audio,” accepted for IEEE journal publication.

[6] T. Painter and A. Spanias, "Perceptual coding of digital audio," *Proceedings of the IEEE*, Vol. 88, No. 4, April 2000.

[7] T. Thiede, and E. Kabot, “A New Perceptual Quality Measure for Bit Rate Reduced Audio,” *Contribution to the 100th AES Convention,* preprint4280, 1996, Copenhagen, Denmark.

[8] *Subjective performance assessment of telephone-band and wide-bandwidth digital codecs,* Recommendation ITU-R P.830, 1996, www.itu.int.

[9] K. Brandenburg, “Evaluation of Quality for Audio Encoding at Low Bit rates,” *Contribution to the 82nd AES Convention*, preprint 2433, 1987, London, United Kingdom.

[10] T. Sporer, “Objective Audio Signal Evaluation- applied psychoacoustics for modeling the perceived quality of digital audio,” *103rd AES- Convention,* preprint 4280, Oct. 1997, New York, USA.

[11] J. G. Beerends and J. Stemerdink, “A perceptual audio quality measure based on a psycho-acoustical representation,” *J. Audio Eng. Soc.,* Vol. 40, p. 963-978, Dec. 1992.

[12] J. G. Beerends, Van den Brink, W. A. C and B. Rodger, “The role of informational masking and perceptual streaming in the measurement of music codec

quality," *Contribution to the 100th AES Convention*, preprint 4176, May 1996, Copenhagen, Denmark.

[13] B. Paillard, P. Mabilleau, S. Morisette and J. Soumagne, "Perceval: Perceptual evaluation of the quality of audio signals," *J. Audio Eng. Soc.,* Vol. 40, p.21-31, 1992.

[14] C. Colomes, M. Lever, J.B. Rault, Y. F Dehery, "A perceptual model applied to audio bit-rate reduction," *J. Audio Eng. Soc.,* Vol. 43, p. 233-240, April 1995.

[15] E. Zwicker and R. Feldtkeller, *Das Ohr als Nachrichtenempfänger.* Stuggart: Hirzel Verlag, Federal Republic of Germany.

[16] *Report on the MPEG-4 Audio Version 2 Verification Test: N3075*, International Organization for Standardization, ISO/IEC JTC1/SC29/WG11, December 1999.