REAL-WORLD EVALUATION OF MOBILE PHONE SPEECH ENHANCEMENT ALGORITHMS

By

WILLIAM THOMAS O'ROURKE

A THESIS PRESENTED TO THE GRADUATE SCHOOL
OF THE UNIVERSITY OF FLORIDA IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF
MASTER OF SCIENCE

UNIVERSITY OF FLORIDA

2002

ACKNOWLEDGMENTS

I am immensely indebted to Dr. John Harris for giving me the opportunity to work on this project. The direction, guidance and encouragement I received from him gave me the confidence needed to achieve. He is the epitome of an academic advisor and his contributions to this thesis cannot be overstated. I would like to extend this gratitude to his wife, Mika. For it was her support at their home that was equally important in allowing him to be have such a significant effect on my research.

I would like to express my sincere thanks to Dr. Jose Principe and Dr. Purvis Bedenbaugh for agreeing to be on my thesis committee and giving me guidance in the process of completing my research.

I would like to thank my parents, Tom and Maureen O'Rourke, my sisters, Maureen, Colleen, Kathy, Elizabeth, Bridget, Marie and Dawn, and my brothers Bobby and Tommy, for the love, concern and support they gave me in all my endeavors.

Finally, I would also like to thank friends and fellow lab mates Dr. Marc Boillot of Motorola iDEN Division, Mark Skowronski, Adnan Sabuwala and Kaustubh Kale for their help and contributions to my research.

TABLE OF CONTENTS

Abstract of Thesis Presented to the Graduate School
of the University of Florida in Partial Fulfillment of the
Requirements for the Degree of Master of Science

REAL-WORLD EVALUATION OF MOBILE PHONE SPEECH
ENHANCEMENT ALGORITHMS

By

William Thomas O'Rourke

December 2002

Chairman: John G. Harris
Major Department: Electrical and Computer Engineering

This work evaluates the performance of two new classes of automatic speech enhancement algorithms by modelling listening tests closer to real-world environments. Results from earlier listening tests show that the warped bandwidth expansion algorithm increases perceptual loudness and the energy redistribution voiced/unvoiced algorithm increases intelligibility of the speech without adding additional power to the speech signal.

This thesis presents results from listening tests conducted with a model of real-world environments and provides a platform for cellular phone based listening tests. Both algorithms are combined on a frame basis to increase intelligibility and loudness. The speech signals are encoded and decoded to model effects of cellular phone vocoders. Three typical environmental noises (pink, babble and car) are used to test the algorithms' performance to noise. Perceptual techniques are used to calculate the signal to noise ratio (SNR). A speaker EQ model is used to emulate the frequency limits of cellular phone speakers. Finally, cellular phone based listening tests are

developed using the Java 2 Micro Edition platform for Motorola iDEN Java enabled cellular phones.

The listening tests resulted in a 4.8% intelligibility increase at -5dB SNR and a 4dB perceptual loudness increase for the combined algorithm. The cellular phone based listening tests will provide an ideal listening test environment once the Java environment is equipped with streaming audio abilities and the algorithms are implemented on the phone.

CHAPTER 1
INTRODUCTION

The use of cellular phones is on an increase all over the world and naturally it is becoming more common to see people using their phones in high noise environments. These environments may include driving in cars, socializing in loud gatherings or working in factories. To deal with the noise, cellular phone users will often press the phone to their head and turn up the volume to the maximum which many times is still not enough to understand the speech. Cellular phone manufacturers could use more powerful speakers or higher current drivers, both increasing cost and battery size. Algorithms that increase intelligibility and overall loudness will help lower battery usage and ease user strain when using the phone.

This thesis studies the implementation and evaluation of a new class of algorithm for cellular phone use. The energy redistribution algorithm [22], described in Section 1.1.1, is an effort to increase overall intelligibility of speech. Section 1.1.2 describes the bandwidth expansion algorithm [4], used to increase perceptual loudness of speech. The aim of implementing these algorithms, is either (1) to enhance the speech for noisy environments or (2) to maintain he quality of the speech at a lower signal power in order to extend battery life.

This thesis primarily addresses the testing of these algorithms to ensure real-world applicability. These tests include controlled environment laboratory testing on PCs and real-world environment testing on cellular phones. The PC testing first tests the performance of the two algorithms without real-world considerations. Next, the PC testing is modified to better model real-world environments. Finally, the listening tests are implemented on a cellular phone to evaluate real-world performance.

1.1 Background

This thesis is a final requirement for the fulfilment of work done for iDEN Division of Motorola. The proposed work attempted to increase intelligibility and perceptual loudness without incurring additional power cost. The work required algorithms that were feasible for real-time implementations and would not affect the naturalness of the speech. It assumed that standard noise reduction techniques would be performed prior to the application of these algorithms and that the received speech could be assumed to be clean. In support of this work, the extended laboratory testing and cellular phone based listening tests were required. The idea was to enable the complete evaluation of the two algorithms which resulted from this research.

1.1.1 Energy Redistribution

The energy redistribution voiced/unvoiced (ERVU) algorithm [22] has been shown to increase the intelligibility of speech. The algorithm was developed based on psychoacoustics. First, the power of the unvoiced speech is crucial for intelligibility. Second, the power of the voiced regions can be attenuated up to a certain point without affecting the intelligibility and naturalness of the speech. Voiced speech is generated anytime glottal excitation is used to make the sound. Voiced signals typically have higher power than unvoiced signals. Additionally, most of the signal power lies in the lower frequencies for voiced speech.

Energy redistribution is performed in three basic steps. First, the voiced and unvoiced regions are determined through the spectral flatness measurement (SFM) [11, 22], shown in Equation 1.1, on individual windows of speech.

$$SFM = \frac{\left(\prod_{k=0}^{N-1} |X_k|\right)^{\frac{1}{N}}}{\frac{1}{N}\sum_{k=0}^{N-1} |X_k|} \tag{1.1}$$

where N is the window length and X_k is the Discrete Fourier Transform (DFT) of the window.

Second, the SFM is compared to two thresholds T_1 and T_2. These thresholds are determined based on statistical classification of the SFM on voiced and unvoiced speech, shown in Figure 1.1. The values for $T_1$and T_2 were set to 0.36 and 0.47 respectively. The decision cases can be seen in Equation 1.2.

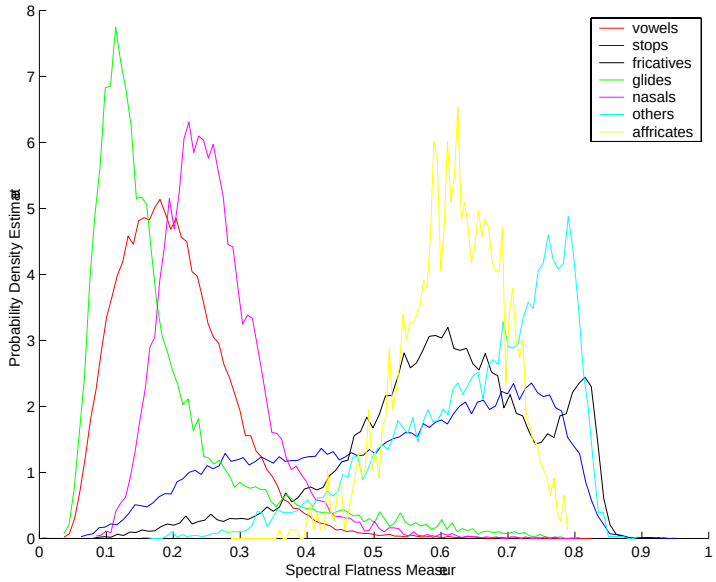

Figure 1.1: Discrimination of phonemes by the SFM

$$Decision = \begin{cases} Voiced & \text{for } \; SFM < T_1 \\ Unvoiced & \text{for } \; SFM > T_2 \\ Previous\ Decision & \text{otherwise} \end{cases} \tag{1.2}$$

Next, the boosting level for the voiced and unvoiced regions must be determined. The boosting level is a gain factor that will be applied to the window. For unvoiced windows, the boosting will be greater than 1 and for voiced windows, it must be less than 1. For windows that fall between both thresholds, the boosting level will remain the same. Boosting levels were determined by evaluating various sentences obtained

from the TIMIT database [27]. The levels were adjusted until naturalness was lost and then set to the previous level. The resulting levels were set to be 0.55 for voiced speech and 4 for unvoiced. In order to smooth the transitions (going from voiced to unvoiced windows or vice-versa), the boosting level is adjusted linearly in the first 10 milliseconds of the window. An example of original speech utterance "six" is plotted with the modified version in Figure 1.2. The SFM technique was chosen over two

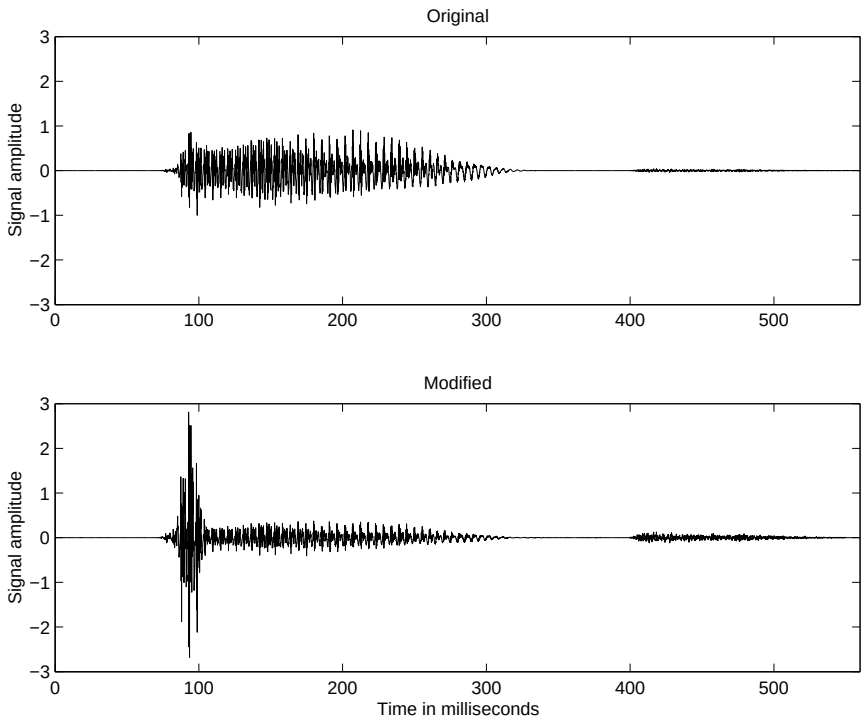

Figure 1.2: Result of applying ERVU to the utterance "six"

other techniques discussed by Reinke [22]. These techniques use a measure of spectral transition to dictate enhancement. The points of high spectral transition are important for retaining vowel perception in co-articulation. Similar results were obtained in intelligibility tests. The SFM technique is also less computationally complex than the other methods.

The results of tests conducted by Reinke [22] have shown an increase of intelligibility close to 5 percent at 0 dB SNR. Results indicate the performance of the ERVU algorithm decreased when the original speech was corrupted with noise. This

shortfall is the result of using SFM. The added noise fills the nulls between formants typically associated with voiced speech. This increases the geometric mean (numerator of Equation 1.1) of the spectrum significantly. The resulting SFM is increased and leads to a misclassification of voiced speech. However, this thesis examines the intelligibility of clean speech for the sender's side with noise on the receiver's side.

1.1.2 Bandwidth Expansion

Bandwidth expansion [4] utilizes a warped filter to increase perceptual loudness of vowels in clean speech. Like the ERVU, bandwidth expansion uses motivation from a psychoacoustics perspective. The underlying principle is that loudness increases when critical bands are exceeded [3]. Loudness refers to the level of perceived intensity of signal loudness and can be measured both objectively and subjectively. Objective measurements can be made using the ISO-532B standard (Zwicker method) [10]. The human auditory system acts as if there is a dedicated band-pass filter around all frequencies that can be detected by the humans. Within this band, perceptual loudness is dominated by the frequencies with the strongest intensity. Various tests have been performed by Zwicker and Fastl [29] to measure these bands. The underlying idea is that when energy within a band is fixed, the loudness remains constant. However, once the bandwidth is exceeded (the energy is spread over more than one critical band) there will be an increase in the perceived loudness.

The bandwidth expansion algorithm uses this idea of spreading the spectral energy of a speech signal over more critical bands. The regions of interest, vowels, are found using voice activity detection described by Motorola Corporation [19]. Speech enhancement is performed in three steps. First, a vocal tract model is estimated using linear prediction coefficients (LPC) $\bar{\mathbf{a}}$, calculated using the Levinson-Durbin recursion [1]. The excitation is then found using the inverse filter $\mathrm{A}(z)$, an FIR filter whose coefficients are $\bar{\mathbf{a}}$. Then, the signal is evaluated off the unit circle in the z-domain. Evaluation off the unit circle is done by first selecting the radius (r) at which

the signal will be evaluated, then the signal is passed through an IIR filter A($\tilde{z}$) shown in Equation 1.3. Figure 1.3 shows that the pole displacement widens the bandwidth of the formants in the z-domain.

$$A(\tilde{z})|_{\tilde{z}=re^{j\omega}} = \sum_{k=0}^{P} a_k r^{-k} e^{-jwk} \tag{1.3}$$

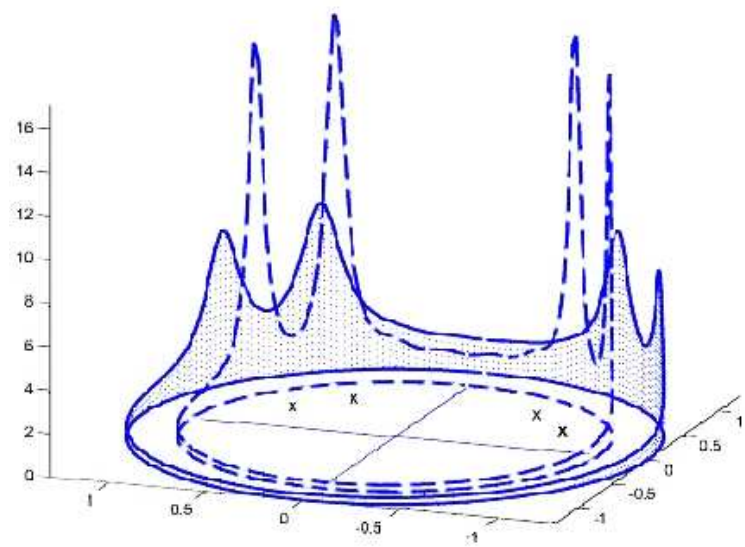

Figure 1.3: Visualization of Pole Displacement.

Since critical bands are not of equal bandwidth, expansion by evaluation along the circle of radius r will not be optimal. For this reason, a warping technique is used. This warping technique is performed like the LPC bandwidth widening, however, it works on the critical band scale. The idea is to expand the bandwidth on a scale closer to that of the human auditory system. The fixed bandwidth LPC pole displacement method is modified by applying the same technique to a warped implementation. The Warped LPC filter (WLPC) is implemented by replacing the unit-delay of A($\tilde{z}$) with an all-pass filter shown in Equation 1.4. The warping filter provides an additional term α called the warping factor. The range of values for

α are -1 to 1. When α is 0, no warping takes effect, for $\alpha > 0$ high frequencies are compressed and low frequencies expanded and for $\alpha < 0$ low frequencies are compressed and high frequencies expanded.

$$\tilde{z}^{-1} = \frac{z^{-1} - \alpha}{1 - \alpha z^{-1}} \tag{1.4}$$

After the WLPC analysis is done, the excitation is passed through a warped IIR (WIRR) that results in the warped bandwidth enhanced speech. An additional radius term γ is added the WIIR so that the resulting spectral slope remains the same. The new algorithm for bandwidth expansion can be seen in Equation 1.5.

$$H(z) = \frac{A(\tilde{z}/r)}{A(\tilde{z}/\gamma)} \tag{1.5}$$

The final architecture used for bandwidth expansion can be seen in Figure 1.4. Boillot found values for of γ and α equal to 0.35 and 0.4 respectively, the algorithm performed the best in subjective loudness tests. The value of r was set between 0.4 and 0.8 as a function of tonality [4].

1.1.3 Combined Algorithm

In order to evaluate the effects of both ERVU and bandwidth expansion, the two algorithms have been combined on a window by window basis. The combined algorithm increases intelligibility and perceptual loudness gain without increasing the signal power.

1.2 Listening Tests

In order to verify the real-world effects of speech enhancement algorithms, some form of subjective listening tests must be performed. Three different levels of listening tests were performed for this purpose. First, controlled tests were administered in the laboratory. These tests were performed using clean speech from the TIMIT

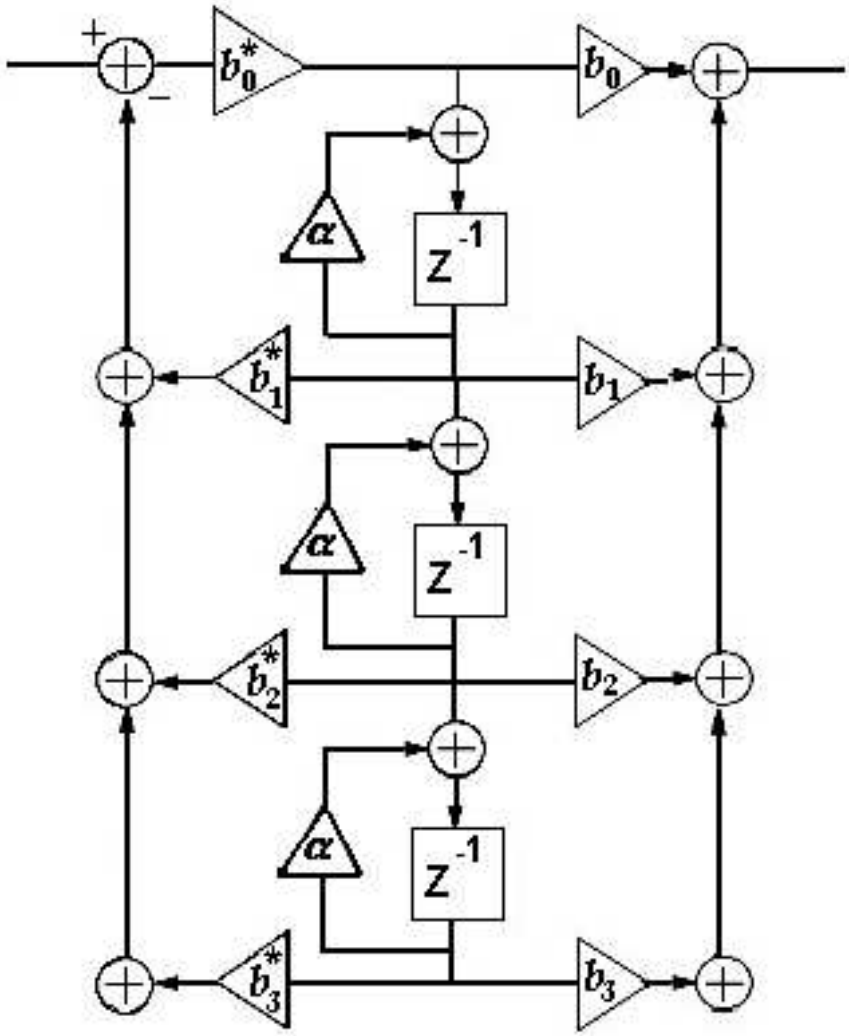

Figure 1.4: Realization of the Warped Bandwidth Expansion Filter.

and TI-46 databases. The purpose of these tests was to obtain preliminary results and make necessary adjustments to the algorithms. Second, the tests are modified to better model the real-world environment. These models include the introduction of environmental noise and modelling of the iDEN cellular phone speaker. Finally, the tests are conducted using the actual phone. This is enabled by the recent incorporation of the Java 2 Micro-Edition virtual machine on certain iDEN phones. The three tests are discussed along with their results in Chapters 2, 3 and 4 respectively.

1.3 Chapter Summary

The outline for the remainder of this thesis is as follows:

Chapter 2: PC based listening tests. This chapter will describe the basic set-up of the laboratory listening tests with emphasis on clean speech and noise free environment. It will also present results obtained in past research using this set-up.

Chapter 3: Expanded PC based listening tests. In this chapter, the basic listening tests described in Chapter 2 will be modified to better approximate real-world conditions. For this test, background noise consisting of pink, car and babble noise is added. Additionally, a model of the frequency response of the Motorola i90c phone is used to better simulate cellular phone use. The results will show that the algorithms both improve intelligibility by 4.8 percent at -5 dB SNR and result in a perceptual loudness gain of 4 dB.

Chapter 4: Java implementation of listening tests. This chapter will describe the implementation of listening tests for Motorola Java enabled cellular phones. It will also describe the support applications developed to manage and evaluate listening tests.

Chapter 5: Conclusions and future work. This chapter will discuss the results of the listening tests, shortcomings of the algorithms and future work on the subject of speech enhancement and real-world testing.

CHAPTER 2
PC BASED LISTENING TESTS

The first step in evaluating the effects of speech enhancement algorithms is some form of listening test. The results of these tests can be used to tune, modify or discard algorithms based on initial performance. For the purpose of this thesis, the evaluation of the algorithms starts with a simplified listening test. These tests are written in MATLAB and were originally developed by CNEL member Mark Skowronski. The tests have been modified through time to accommodate changes to the listening tests. This chapter will describe the tests as they were when Boillot [4] tested the bandwidth expansion algorithm. Intelligibility, loudness and acceptability tests were conducted. The Energy Redistribution Voiced/Unvoiced (ERVU) algorithm, originally tested by Reinke [22], was not initially tested in this listening test environment. However, the results of ERVU intelligibility tests will be discussed in Section 2.1.2.

2.1 Intelligibility Test

The purpose of communications is to get the message across as clearly as possible. Algorithms that attempt to increase intelligibility or perceptual loudness require testing to verify their applicability. Intelligibility testing methods come in many forms. Some of these methods include the Diagnostic Rhyme Test (DRT), the Modified Rhyme Test (MRT), and the Phonetically Balanced word lists (PB). The method used in this experiment is a variant of the DRT [28, 4].

The intelligibility tests were conducted using speech samples from the TI-46 database [26] sampled at 10 kHz. Sets I, II and III, from Table 2.1 were used. These sets were originally used by Junqua [12] to test the effect of the Lombard effect on

speech intelligibility. The Lombard effect is the way people speak differently when in a noisy environment. They try to compensate for the noise by speaking louder, slower, more clearly and with more stress and emphasis. The individual sets are considered to be easily confusable and provide a good vocabulary for testing intelligibility.

The MATLAB GUI used to conduct the test is shown in Figure 2.1. For each utterance, an alternate utterance was selected from the same set and presented as a confusable choice. The GUI allowed only one choice and did not limit the number of times the utterance was played. Each utterance had an equal chance of being selected. The order of selection was randomized so that the listener had no knowledge of which utterance was correct. Half the utterances presented were left in their original form and the other half enhanced with the bandwidth expansion algorithm.

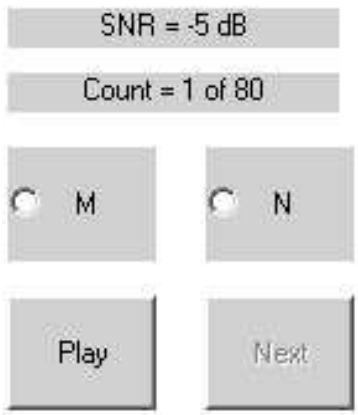

Figure 2.1: Intelligibility test GUI

2.1.1 Bandwidth Expansion Results

All though the bandwidth expansion algorithm only attempts to increase the loudness of speech, it still must be tested to ensure that there is no decrease in intelligibility. A total of 60 utterances be presented to each listener with added Gaussian noise at 0 dB SNR. The test resulted in an overall decrease in intelligibility of 0.3% $\pm$ 3.1% at a 95% confidence interval. These results showed that the bandwidth expansion algorithm had no measurable effect on intelligibility [4]. These results

were as expected since bandwidth expansion only modifies the voiced phonemes and unvoiced phonemes predominantly define intelligibility.

Table 2.1: Vocabulary of words used for Intelligibility Test

I	f, s, x, yes
II	a, eight, h, k
III	b, c, d, e, g, p, t, v, z, three
IV	m, n

2.1.2 ERVU Results

Reinke initially used the same form of intelligibility test as Boillot to test his ERVU algorithm and a high-pass filter. He later used the DRT to test other intelligibility enhancement algorithms [22]. The tests used sets I, II , III and IV from Table 2.1. Gaussian noise was added to the utterances at 0 dB and -10 dB SNR levels. He reported a 3% increase in intelligibility at 0 dB SNR and 5.5% at -10 dB SNR. His test was administered to 25 listeners, whose ages ranged from 21 to 37.

2.2 Perceptual Loudness Test

Loudness is the human perception of intensity of sound [8]. It is a function of sound intensity, frequency and quality. The loudness of single sinusoids can be measured using the *equal loudness contours* [29]. This measure, using the units phon, in only valid for narrow band signals. For this reason, Boillot used ISO 532B analysis[10]. Subjective measures are performed using a loudness test.

2.2.1 Objective Loudness

Speech is a wide band signal and perceptual loudness cannot be measured using the *equal loudness contours*. Instead, a model first developed by Zwicker [6], ISO 532B was used to perform objective measures. ISO 532B uses concepts from auditory filter models and the critical band concept [7]. Additionally, ISO 532B follows the model

discussed by Baer, Moore and Glasberg [2], which was a revision of Zwicker's model and compensated for low frequencies and levels. Boillot's test on TIMIT sentences showed a 2.1 dB gain for vowels using ISO 532B analysis.

2.2.2 Subjective Loudness

To verify the perceptual loudness increase of the bandwidth expansion algorithm, an apparent gain measurement would be required. Until now, there were no known tests for quantifying speech loudness. For this purpose, subjective loudness tests were developed. The subjective loudness tests were performed in MATLAB and used utterances from the TI-46 database shown in Table 2.2. The utterances were sampled at 10 kHz and the α term of Equation 1.4 was set to 0.5. The listener was presented with two utterances of the same word, one being the original and the other being the enhanced. The MATLAB GUI used can be seen in Figure 2.2. The listener would play each sound and then make a judgment on which one sounded louder.

Table 2.2: Vocabulary of words used for Loudness Test

V1	zero, one, two, four, five, six, seven, nine
V2	enter, erase, help, repeat, right, rubout
V4	a, eight, h, k
V5	b, c, d, e, g, p, t, v, z, three
V5	m, n

A total of 80 words was presented to each listener. Boillot used 15% of these words to perform a screening evaluation. The listener had no knowledge of which was the modified or the enhanced word and the order was randomly selected. First, both words are normalized to equal power and then the enhanced version would be scaled randomly between 0 dB and 5 dB in 0.5 dB increments. This provided the perceptual gain of the enhanced algorithm. The scaling point which produced a 50% selection rate marks this perceptual gain crossover point. The bandwidth expansion

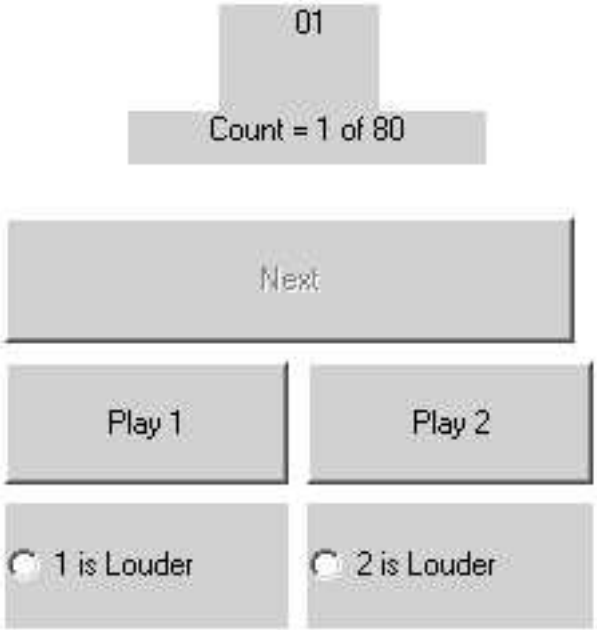

Figure 2.2: Loudness test GUI

algorithm resulted in an approximate 2dB crossover point. This is the point at which the listener is guessing, hence, 50% the enhanced version was chosen. These results can be seen in Figure 2.3, where the results are shown in solid lines.

The screening process ensured that the data collection was accurate. For this portion of the tests neither word was modified but one was scaled. It tested the hearing resolution of the listener and, at the same time, ensured the listener was paying attention and not suffering from fatigue. It was important to verify the level at which the human auditory system could perceive a change in loudness before the algorithm could be considered effective. From Figure 2.3, the screening results (indicated by dashed lines) are equal at 50% at 0dB and diverge as expected.

2.3 Acceptability Test

The goal of the speech enhancement algorithms is to increase the intelligibility and perceptual loudness of speech without deteriorating the naturalness of the speech. Boillot found that the loudness of vowels increased monotonically until the spectrum

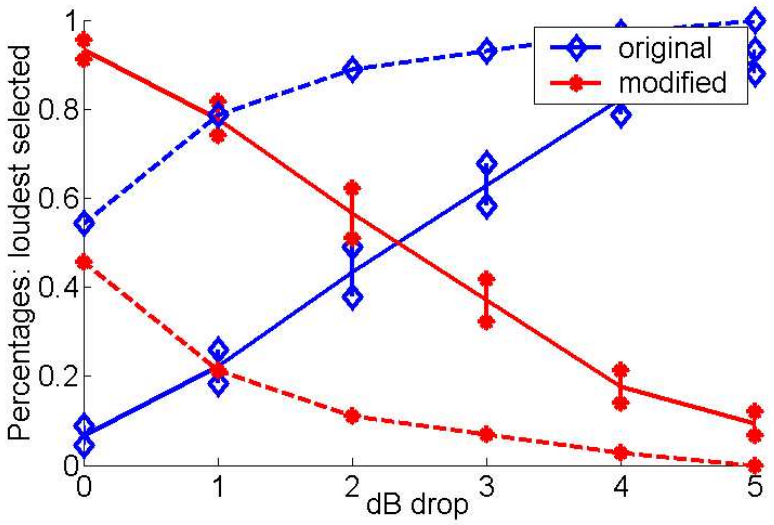

Figure 2.3: The Bandwidth Expansion results for all listeners on the Loudness test. Vertical bars indicate 95% confidence intervals [4].

was flat. This lead to an obvious distortion of the speech. To ensure that the warped bandwidth expansion algorithm did not effect the quality of speech, Boillot included an acceptability, or quality, test [4]. This test used a number rating system to quantify the overall impression. Boillot used paired comparison tests to evaluate acceptability. Additionally, the test included a subjective loudness assessment. The test used an original and modified (enhanced) version of the same sentence taken from the TIMIT database. A total of 20 phonetically balanced sentences sampled at 16 kHz were used. The original and the modified sentences were scaled to have equal power on a frame-by-frame basis. The listeners would play each sentence pair and then subjectively rank each one. The listeners gave a mark of excellent, good, or fair for each sentence. The number one through three corresponded to each mark respectively. Listeners were directed to score relatively. For example, even if both sentences sounded excellent, they should still try to determine which of the two was better and give that sentence the excellent mark and the other a mark of good. The loudness assessment just asked the listener to determine which sentence sounded louder overall.

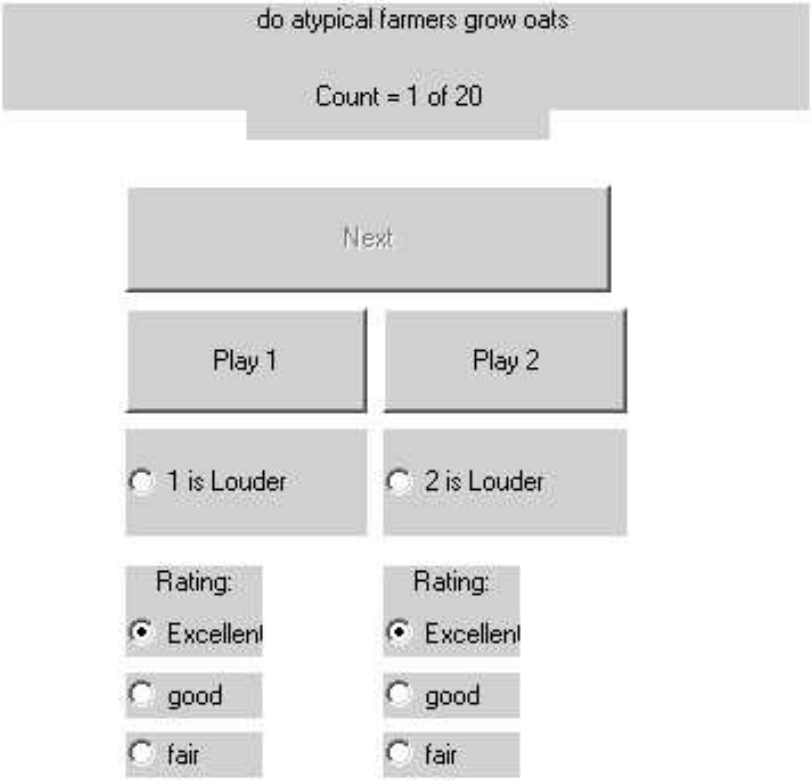

Figure 2.4: Acceptability test GUI

Boillot reported a quality rating of 1.56 for original and 1.47 for modified speech. The modified sentence was selected louder 90% of the time. These results in indicate that the overall quality is not affected. The loudness assessment results provide a preview to how the algorithm would perform on sentences instead of the single word utterances used in loudness and intelligibility tests.

CHAPTER 3
EXPANDED PC BASED LISTENING TESTS

The listening tests discussed in Chapter 2 did not implement typical real-world effects on speech. In this chapter, we will expand these tests to more closely model real-world environments. This is done in three steps. First, to simulate the effects of vocoders used on cellular phones, the speech is vocoded then devocoded. Then, noise sources are chosen based on real-world environments. Finally, the Audio EQ Model for the Motorola I85c cellular phone is implemented to simulate the cellular phone speaker frequency response. Additionally, the bandwidth expansion and ERVU algorithms are combined in an attempt to increase perceptual loudness and intelligibility. It is this combination algorithm that will be used in this chapter. The resulting signals modified by this algorithm will be referred to as the "enhanced" or "modified" signal in this chapter. For these tests, Sony MDR-CD-10 headphones were used.

3.1 Motrola™ VSELP Vocoder

The Motorola iDEN i90c phone uses the Vector Sum Excited Linear Prediction (VSELP) Vocoder [17] to provide encoding and decoding of speech for transmission. The purpose of the encoding on the send side is to compress speech to limit transmission bandwidth. On the receive side, the vocoder then decodes the compressed speech. The result of encoding and decoding is degradation of the speech. This degradation includes an already limited frequency range and loss of naturalness. To simulate this degradation, Motorola has provided a C program to emulate the encoding and decoding of speech. This allows the listening tests to better model the sound of speech delivered by the phone. The C program uses the Advanced Multi-Band

Table 3.1: Description and Samples of Noise Sources.

Pink Noise acquired by sampling high-quality analog noise generator (Wandel & Goltermann). Exhibits equal energy per 1/3 octave.
Babble Noise acquired by recording samples from 1/2" B&K condensor microphone onto digital audio tape (DAT). The source of this babble is 100 people speaking in a canteen. The room radius is over two meters; therefore, individual voices are slightly audible. The sound level during the recording process was 88 dBA.
Volvo 340 noise acquired by recording samples from 1/2" B&K condensor microphone onto digital audio tape (DAT). This recording was made at 120 km/h, in 4th gear, on an asphalt road, in rainy conditions.

Excitation (AMBE) vocoder model which is also used in iDEN phones. To do this, the speech must first be re-sampled at a rate of 8 kHZ (the sampling rate used on the phones). This furthers the real-world model by bandwidth-limiting the speech.

3.2 Noise Sources

The listening tests described in Chapter 2 exclusively used white Gaussian noise but, there is a larger variety of noise types we experience in our everyday life and few of them are Gaussian. Some of these noises are car, cocktail party (babble), machine and pink noise. It would be ideal to test the performance of our algorithm with all possible noise sources. However, in order to prevent listener fatigue and to allow multiple SNR level testing, the noise sources were limited to car, babble, and pink noises. These noise sources were obtained from Rice University Signal Processing Information Base [23]. The noise samples are sampled at 19.98 kHz and available in both MATLAB .MAT format and .WAV format. Table 3.1 describes the noises and provides samples.

3.2.1 SNR Calculation

There are several methods for calculating the SNR of a signal. The classic form is shown in Equation 3.1. Where $x[n]$ is the clean speech signal, $v[n]$ is the additive noise and N is the number of samples for the signal.

$$SNR = 10 \times \log_{10} \left(\frac{\sum_{n=0}^{N-1} (x[n])^2}{\sum_{n=0}^{N-1} (v[n])^2} \right) \tag{3.1}$$

This form of SNR is usually taken over the entire signal length. Unfortunately, this form will not effectively measure the perceptual significance of noise to human hearing. Humans are limited in their frequency range of hearing. Human hearing typically ranges from 20 Hz to upwards of 20 kHz. Obviously, a high power signal at frequencies above and below this range will not effect the perceptual SNR from the listeners standpoint. Additionally, there is a sharp drop-off in the intensity of sound above 4 kHz. This drop-off is apparent in the *equal loudness contours* shown in Figure 3.1. For this reason, an alternate approach to the method used in Equation 3.1 is needled.

3.2.2 Segmental SNR

Classic SNR calculations carry little perceptual significance, since they will tend to be higher as the ratio of voiced to unvoiced speech increases. A better calculation can be obtained by considering the SNR values for frames of speech. Segmental SNR (SNR_{seg}), shown in Equation 3.2, uses a frame based average of the standard SNR in Equation 3.1. The basis of the SNR_{seg} calculation is that it takes into account the short duration of unvoiced speech. If the SNR_{seg} equation is rearranged, we see that it is a geometric mean of the windowed speech signal SNR. This is seen in Equation 3.3.

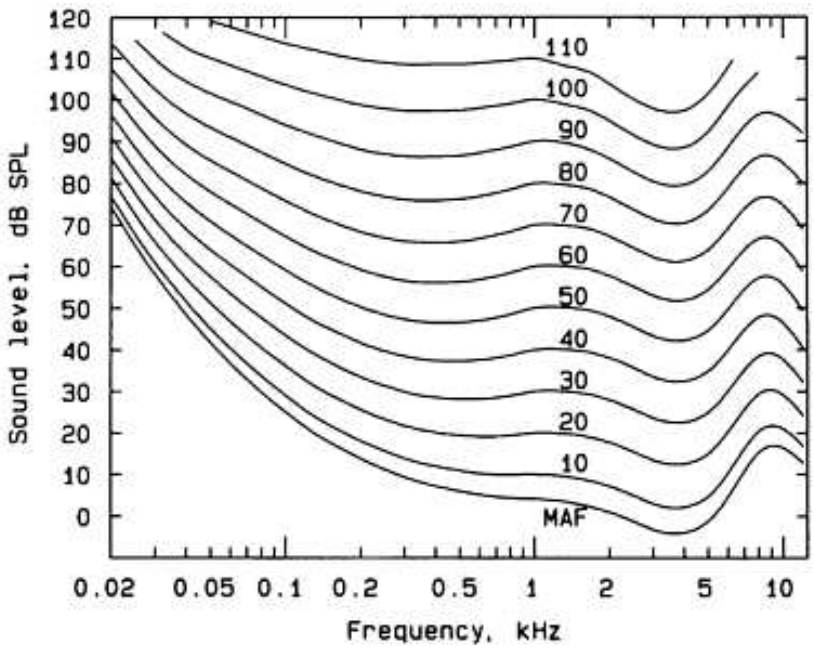

Figure 3.1: Equal Loudness Contours

$$SNR_{seg} = \frac{1}{L}\sum_{k=0}^{M-1} 10 \times \log_{10}\left(\frac{\sum_{n=L\times k}^{L\times(k+1)-1}(x[n])^2}{\sum_{n=L\times k}^{L\times(k+1)-1}(v[n])^2}\right) \tag{3.2}$$

$$SNR_{seg} = 10 \times \log_{10}\left(\prod_{k=0}^{M-1}\left(\frac{\sum_{n=L\times k}^{L\times(k+1)-1}(x[n])^2}{\sum_{n=L\times k}^{L\times(k+1)-1}(v[n])^2}\right)^{\frac{1}{L}}\right) \tag{3.3}$$

The windowed based SNRs are usually limited to upper and lower values. This is used for instances where the widowed SNR is significantly low or high. In these cases the extremes would dominate the measurement. An example of this is if the signal power goes to zero the windowed SNR would be set to -10dB instead of $-\infty$. Likewise, if the noise signal goes to zero, the widowed SNR is set to 45dB instead of ∞.

The primary problem with using SNR_{seg} to calculate the noise gain required, is that it calls for an iterative process. That is, since the calculation is non-linear, there is no closed form solution like standard SNR. The calculation must be repeated for

several noise gain values before the desired SNR level is achieved. For this reason, we needed another approach to calculate perceptual SNR.

3.2.3 A-Weighting

Another approach to effectively adding noise at a specific perceptual SNR level is A-Weighting [20]. A-Weighting coefficients are used to filter both the speech signal and the noise. These temporary signals are then used to calculate the required gain for the noise source in order to achieve a specific perceptual SNR. The frequency response of the A-Weighting filter is shown in solid red in Figure 3.2. The figure also shows the results of averaging the inverse of the *equal loudness contours* from Figure 3.1 in dashed blue.

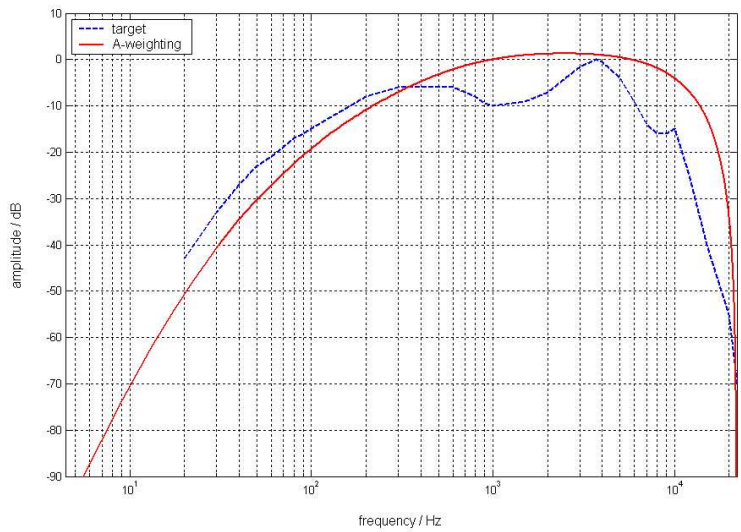

Figure 3.2: A-Weighting Filter Magnitude Response.

Namba and Miura [20] found that A-Weighting was ideal for calculating the perceptual SNR of narrow-band noise. Though speech and many noise types are considered wide-band signals, the use of A-Weighting is still a better approximation than the classic calculation. Additionally, the time required for the listening tests should be relatively short and more complex calculations, such as ISO 532b, are not practical

for the listening tests. For these reasons, A-weighting was used for the calculation of the SNR level for the listening test discussed in this chapter.

3.2.4 Choosing the SNR levels

If we were to test the full range of a listener's hearing in noisy environments, the peak increase in intelligibility could be found. This would require adjusting the SNR from the point where the listener was achieving 100% accuracy to where he or she is merely guessing every time on the intelligibility test. Figure 3.3 shows the expected results if the SNR was adjusted from a very low SNR level to a very high level. This figure shows an expected performance and values are provided for clarity only. At a particular SNR level, the maximum percent increase would be achieved. Unfortunately, this would require that the test duration be extremely long. And, given the three noise sources, it would most definitely lead to listener fatigue. This required some pretesting to establish what levels to use. We decided that the desired accuracy for un-enhanced speech should be somewhere close to 75%. This number, being half way between the upper and lower limits, would leave room for inexperienced and experienced listers.

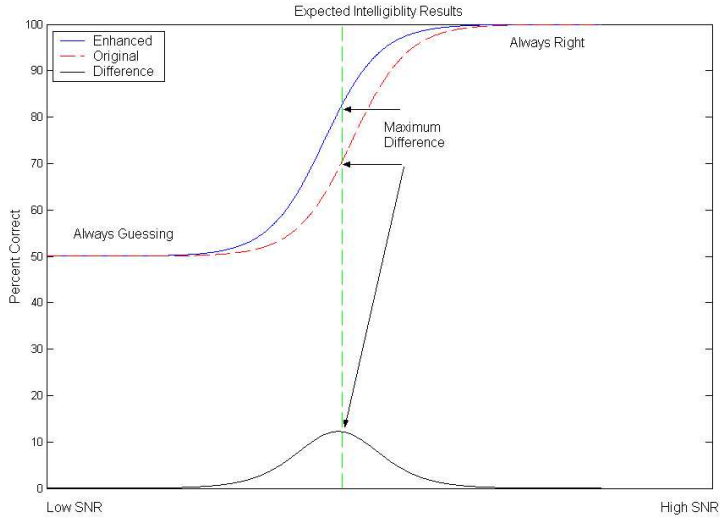

Figure 3.3: Conceptual Intelligibility Test Results for a Single Listener.

The procedure for finding these SNR levels involved a preliminary intelligibility test. Four listeners were used for these tests. First the SNR was set to 5dB and then an adaptive algorithm was used. For each noise source, a total of 80 utterances were presented. After 20 utterances, the percent correct was calculated. If the percent correct was higher than 75%, the SNR was lowered by 1dB. If it was lower than 75%, the SNR was raised. After each additional five utterances, the percent correct was again calculated. If the percent correct was still approaching 75%, then the SNR was not changed. However, if it was moving away or unchanged the SNR was adjusted. Table 3.2 shows the results of the preliminary intelligibility tests. The listener's language and SNR dB level which resulted in 75% correct for the respective noise source. Based on these results, SNR levels of -5dB and 5dB were chosen.

Table 3.2: Results of Preliminary Intelligibility Test.

Listener	Native Language	Babble	Car	Pink
I	English	1.7dB	-9.4dB	-0.8dB
II	English	1.0dB	-2.2dB	0.0dB
III	Hindi	2.2dB	-3.2dB	-3.5dB
IV	Chinese	3.5dB	2.5dB	2.7dB

3.3 Audio EQ Filter

Cellular phone speakers are designed with size and cost in mind. They require compact size and limited cost in order to be used in cellular phones. Additionally, the Federal Communications Commission (FCC) puts constraints on the speaker peak output sound pressure level (SPL). Because of these design constraints the speakers have limited frequency range. Motorola provided the frequency response for the iDEN i85 phone speaker (assumed to be linear), shown in Figure 3.4. The MATLAB function $firls.m$, from the Signal Processing Toolbox, was chosen to design the filter.

The *firls.m* function uses a least-squares (LS) approach to derive the FIR filter coefficients for the Audio EQ model [14]. The FIR filter is linear-phase and therefore does not distort the speech signal. The purpose of this model was to mimic the

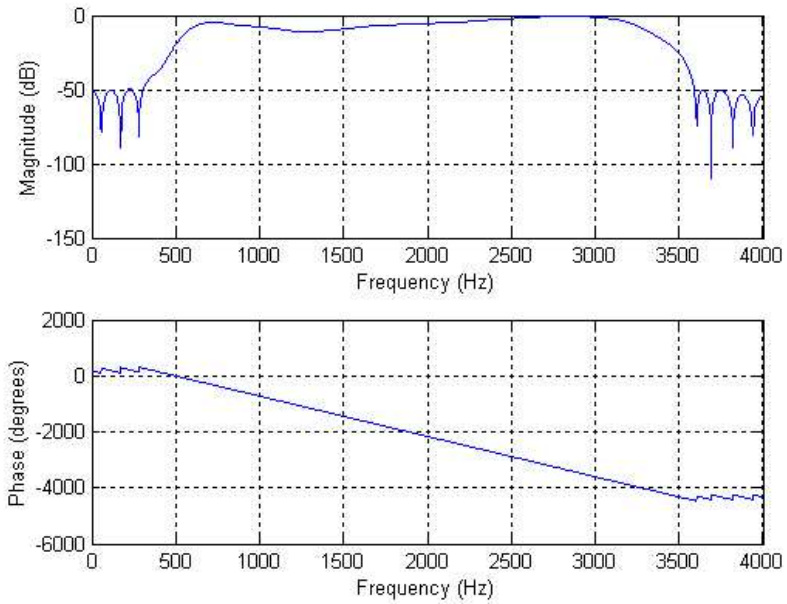

Figure 3.4: Phase and Frequency Response for the Speaker EQ Model.

cellular phone environment. This model was used to filter the speech signals after they were enhanced with the combined algorithm. Since we know that environmental noise is not limited to the same bandwidth as cellular phone speech, this was performed before the SNR calculation and only the speech signal was filtered.

3.4 Listener Demographics and Test Results

Demographics A total of 22 listers were tested in both Loudness and Intelligibility tests. A total of six listeners were native English speakers. Nineteen of the listeners were male and three were female. Five of the listeners were considered experienced (taken multiple listening tests) listeners. The listeners ranged from 22 to 42 years of age. The average test time was 22 minutes and 38 seconds.

Listening Test Flow Diagrams Figures‘3.5 and 3.6 show the flow diagrams for the loudness and intelligibility tests respectively.

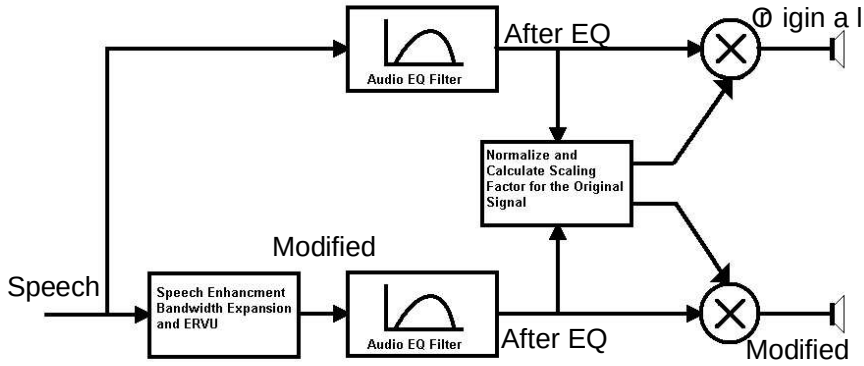

Figure 3.5: Signal Flow Diagram for the Loudness Tests.

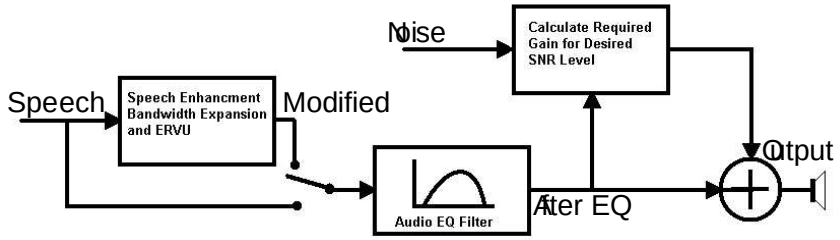

Figure 3.6: Signal Flow Diagram for the Intelligibility Tests.

Results for Loudness Tests The average perceptual loudness gain was 4dB. This result is apparent in the crossover plot, Figure 3.7. The screening process, shown by the dotted lines, indicates that the results are accurate and that the users were paying attention. Of the 22 listeners results only four fell below the 4dB crossover and none of these fell below 2dB. Table 3.3 shows the total results for the loudness tests. These results are higher than earlier tests conducted by Boillot. This may be attributed to the application of the Audio EQ model. The 2.5dB gain was for formant expansion on speech sampled at 16 kHz. The 4dB gain was achieved using the combined algorithm on vocoded speech sampled at 8 kHZ. The Audio EQ model has a peak around the

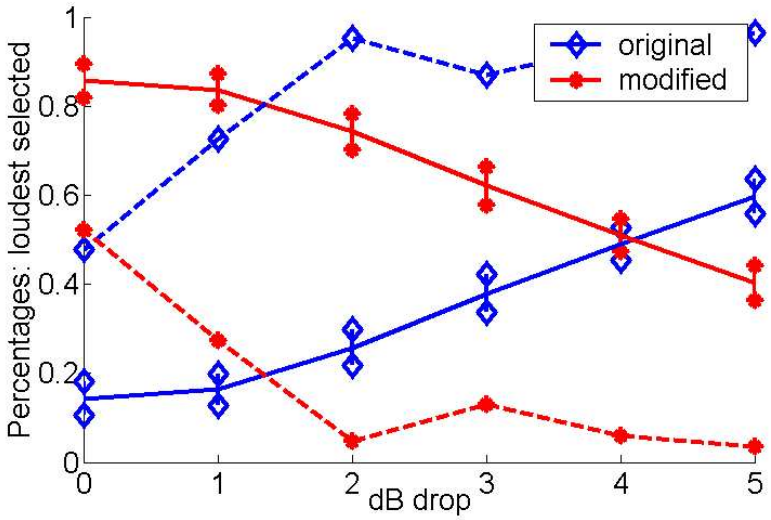

Figure 3.7: The results for all listeners on the Expanded Loudness test. Vertical bars indicate 95% confidence intervals [4].

2-3 kHz range which also corresponds to the highest sensitivity on the ISO-226 equal loudness curves [9].

Results for Intelligibility Tests The Intelligibility tests resulted in an increase of 4.8% at -5dB SNR for enhanced speech over all noise types and confusable sets. This is a minimum increase and we expect that the maximum increase would be larger. At a 5db SNR level the tests resulted in less than 1% decrease in intelligibility. The 95% confidence intervals are shown for overall results, in Tables 3.4 and 3.5. These tables

Table 3.3: Results for Subjective Loudness Tests.

Scaling of Modified	Times Selected Original	Times Selected Enhanced	Percent Enhanced Selected
-5dB	138	93	40
-4dB	105	109	51
-3dB	66	108	62
-2dB	64	185	68
-1dB	41	210	84
0dB	36	215	86

Table 3.4: Intelligibility Test results for 5dB SNR.

	Alg.	All	I	II	III	IV
Overall	O	**83.56 ± 5.94**	89.06	92.36	79.93	69.00
	E	**82.70 ± 4.69**	82.24	86.31	81.96	81.18
Car	O	88.35	91.67	90.35	86.46	34.21
	E	88.35	91.23	88.85	85.15	82.89
Babble	O	84.92	93.86	91.05	78.59	45.61
	E	81.65	82.46	92.11	79.21	65.79
Pink	O	78.20	56.00	90.79	72.02	60.53
	E	77.40	65.44	51.05	83.50	75.69

Table 3.5: Intelligibility Test results for -5dB SNR.

	Alg.	All	I	II	III	IV
Overall	O	**66.49 ± 3.98**	66.57	81.83	64.31	53.73
	E	**71.31 ± 4.92**	68.45	73.25	74.59	61.26
Car	O	73.08	83.86	74.91	64.31	31.58
	E	78.47	81.14	59.74	82.22	39.47
Babble	O	62.48	55.26	78.20	67.79	14.47
	E	72.17	70.18	66.18	74.65	44.91
Pink	O	64.13	50.88	79.91	61.00	43.86
	E	65.09	46.05	75.12	69.49	50.38

also show results for all three noise sources vs. all four confusable sets in Table 2.1 and for the original (O) and enhanced (E) signals.

3.5 A Note on ERVU

Though the SFM technique was used for voiced/unvoiced decision in testing for its lower computational complexity, there is an issue of precision when using a fixed-point Digital Signal Processor (DSP). The geometric mean is taken on the DFT values of a frame of speech. We know that these values are less than one. The result of multiplying all the values in one frame would result in a number less than the precision of the Motorola 56000 DSP used in the iDEN phones. For example $0.9^{160} = 4.8 \cdot 10^{-8}$ when compared to the DSPs precision of $2^{-15} = 3.11 \cdot 10^{-5}$. More elaborate

calculations using logarithms produce a solution but require higher computational complexity. We propose an alternate method for voice/unvoiced decision using a peak autocorrelation ratio technique.

$$r[k] = \frac{1}{N}\sum_{n=0}^{N-1} x[n] \cdot x^*[n-k], \ \texttt{where} \ N > 0 \tag{3.4}$$

Equation 3.4 shows the biased autocorrelation function for lag k. Autocorrelation, is commonly used in pitch recognition [1] systems. Pitch, the rate at which the glottal opens and closes, is inherent to voiced speech. The peaks in the autocorrelation function take on voiced speech are separated by the periodicity of the pitch. For unvoiced speech the autocorrelation function resembles something close to an impulse response. This is due to the characteristic of unvoiced speech being close to stationary white Gaussian noise.

$$ratio = \frac{\max\limits_{m \in \triangle m \rightarrow N-1} r[m]}{r[0]}, \ \texttt{where} \ N > 0 \ \texttt{and} \ \frac{f_s}{\triangle m} = \ \texttt{maximum pitch.} \tag{3.5}$$

Instead of calculating the pitch, we consider the ratio of the signal power, equivalent to $r[0]$, and the maximum value of the autocorrelation function from lag $\triangle m$ to $N-1$. $\triangle m$ is chosen to remove any spreading of the impulse around lag zero and to ignore unrealistic pitch values. The peak autocorrelation ratio technique results in a 6.2% voiced/unvoiced classification error as compared to a 3.8% error using the SFM technique. However, the performance of SFM decreases as the SNR decreases. This can be alleviated using the peak autocorrelation ratio technique. This technique is considered very robust to noise [1] for pitch detection.

CHAPTER 4
JAVA IMPLEMENTATION OF LISTENING TESTS

In Chapter 3 we discussed methods for making the listening tests discussed in Chapter 2 more relevant to future real-world operation of the algorithms. These enhancements included real-world noise, vocoder effects and modelling the speaker EQ cures of the cellular phones. However, these listening tests results are still somewhat artificial since the user is actually listening on a headset and not on a cellular phone. Clearly, if we could move our whole listening test environment to the phone, then the tests could be run in the true environment such as riding in a car on a highway or using the phone in a crowded social gathering. This chapter discusses the efforts of implementing the listening tests on the Javaphone –Java enabled cellular phone–, the interface between the PC and phone and the database management. The listening tests conducted in Chapter 3 gave promising results towards real-world performance. To finish the evaluation, we must be able to quantify the performance in the true listening environment. Real-world testing can be performed using the Javaphone in a natural environment.

4.1 J2ME and J2SE

The Java 2 Micro Edition (J2ME) is a software development kit (SDK) commonly used in mobile devices. Development in J2ME is limited in several ways as compared to the Java 2 Standard Edition (J2SE). First, the limited memory space, described in section 4.3, requires efficient coding. Second, the limited class set reduces functionality and sometimes requires additional coding. Unlike J2SE, development of J2ME applications requires the use of a configuration and profile. The Connected Limited

Device Configuration describes (CLDC) the API for a certain family of devices which includes the Motorola iDEN series Java enabled phones. The Mobile Information Device Profile (MIDP) sits on top of the configuration and targets a specific class of devices [15]. All Java applications written for the phone are an extension of the MIDlet class. A MIDlet is a MIDP application. Programming in J2ME is performed by utilizing the classes within MIDP and CLDC APIs. The MIDlet is developed specific to the devices it will run on and, in this case, the Motorola Java enabled iDEN phones.

4.2 Why J2ME?

Java is an ever-evolving software development tool. Motorola has incorporated the Java 2 Micro Edition (J2ME) virtual machine, described in Section 4.1, in certain iDEN model phones. Some advantages of Java are portability (write once, run anywhere), thorough documentation and extended networking ability. Though J2ME is limited in these advantages, it still provides an excellent environment for the development of applications on cellular phones. The following is a list of some of the abilities J2ME has.

1. Communication with a PC via the serial port.
2. Communication with web resources via the internet.
3. A database storage system.
4. Basic GUI operation.
5. Control of vocoded speech files.
6. Image rendering and animation.
7. Multi-threaded operation.

The development of the listening tests on the iDEN phones incorporates the abilities listed in items 1, 2, 3, 4 and 5, in the list above.

4.2.1 Developing in J2ME

The procedure for developing J2ME code is as follows:

1. The code is written within the limitation of the device it designed to run on.
2. It is compiled using any standard Java compiler and J2ME libraries.
3. It is preverified using a device-specific preverifier.
4. The code is tested on an emulator for bugs.
5. Any problems encountered in the emulation are debugged.
6. The code is recompiled, packaged into a MIDlet suite, converted to Java Archive (JAR) files and uploaded to the phone.

The uploading of MIDlets is performed using the Java Application Loader (JAL) utility. When the phone is connected to the PC's serial port, the JAL utility can be used to view, delete, and upload MIDlets. If any bugs are discovered after running the MIDlet, the code must be debugged and steps two through six are repeated. The preverification and emulation software may not always catch problems that will occur once the MIDlets are executed on the phone. Multiple MIDlets can be uploaded to the phone. If these MIDlets are required to share resources, they must be part of the same suite. The multiple MIDlets are packaged into a MIDlet suite and then compiled and converted to JAR files to be uploaded to the phone. See Appendix A for an explanation of all classes and methods created to implement the listening tests.

4.2.2 J2ME Constraints

Constraints on font, screen colors, screen size and image rendering need to be carefully considered when writing J2ME code for a specific device. The basic user

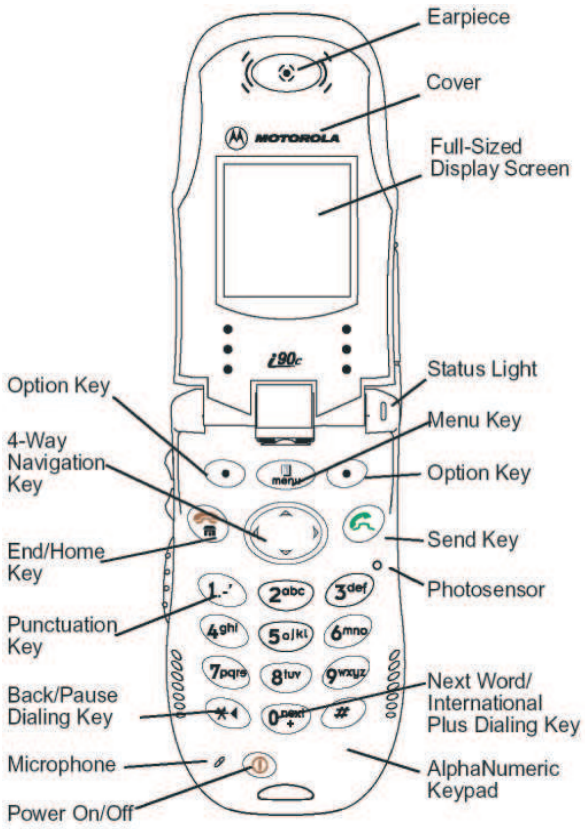

Figure 4.1: Motorola i90c iDEN phone

display object used in J2ME is the *Display* class. This class contains all methods for bringing displayable information to the foreground of the display screen. These methods can control any displayable objects. The *Screen* object is a subclass of the *Display* class, hence it inherits displayable properties. The *Screen* class and its subclasses are used in all code written for the listening tests. Another class, the *Canvas* object is used for drawing on the screen and was not utilized in the listening tests. Three subclasses of *Screen* are *Form*, *List* and *TextBox*. These classes allow for user input through the device keypad and information display on the device screen. A MIDlet is typically written to navigate through different screens based on user inputs.

4.3 Motorola iDEN Series Phones

The Javaphone has three user input devices [21]. They include a alpha-numeric keypad, similar to standard touch-tone telephones, a 4-way navigation key and two option keys. Through the keypad all numbers, letters and punctuation can be entered by sequentially pressing keys (multi-tapping.). The 4-way navigation key can be is used to move though menus, lists, radio-buttons and choicegroups. The two option keys are used as control to select from menus, lists, radio-buttons, choice-groups or to program defined options. The Motorola i90c iDEN phone is shown in Figure 4.1. The phone has 3 types of memory dedicated to the Java VM [16]. Data memory (256k Bytes) is used to store application data, such as image files. Program memory (320k Bytes) refers to the memory used to install applications (MIDlets). Heap memory (256k Bytes) refers to the Random Access Memory (RAM) available to run a Java application.

4.4 Listening Test Setup

Three separate listening tests run on the phone. They include loudness, intelligibility and acceptability, similar to those used in Chapter 2. These tests can be run using the *ListenT* MIDlet on the phone which is part of the *ListenM* package. The user is asked to enter the information including; name, age, native language and date. Next, the user selects between one of the three tests. The basic flowchart for the *ListenT* MIDlet is shown in Figure 4.2. The three listening test flowcharts can be seen in Figures 4.4, 4.5 and 4.6.

4.5 *ListenT* MIDlet

The *ListenT* MIDlet is an extension of class *MIDlet* and implements the interface *CommandListener*. The implementation of *CommandListener* allows the MIDlet to monitor commands received through the phone's option keys. The commands are then interpreted based on the current screen, command selected and index selected,

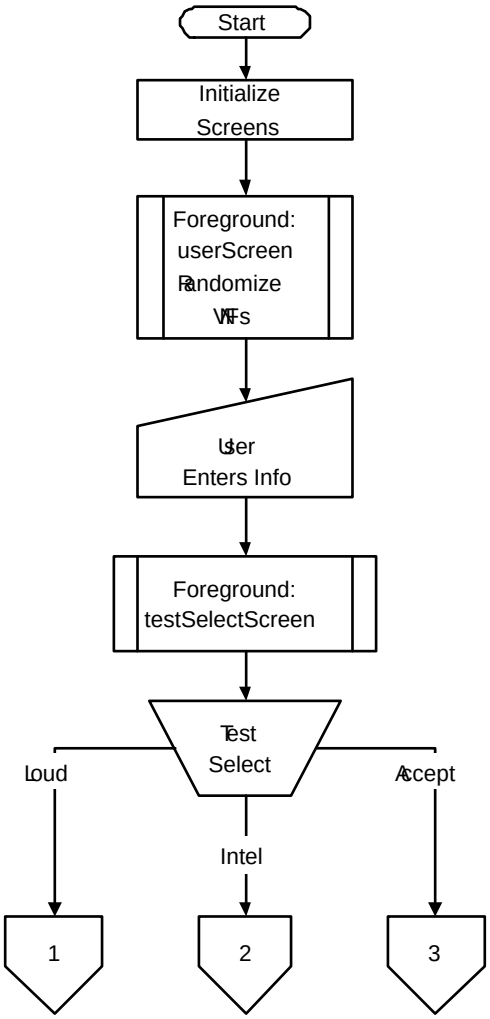

Figure 4.2: Flowchart for Listening Tests *ListenT.java*

if any. Initially, when the MIDlet is run, it first executes its constructor. This sets up any classes or variables that are initially needed to execute the MIDlet properly. In *ListenT* the constructor creates three *ListenDB* databases (explained in section 4.6) for each of the three listening tests, the *ansBuffer* buffer and the *RandObj* object– a random number generator which extends the ability of the *Random* class. It then initializes the screens *testScreen*, *userScreen* and *doneScreen* by calling their initialization methods. Next, it calls the *MIDlet.startApp* method (this is always the case with any MIDlet). Within this method the *userScreen* is set as the current display and the listening test is ready to start and the *randomizeDir* method is called. The *randomizeDir* method takes all Voice Note Files (VNF) (described in

Figure 4.3: Javaphone Listening Test GUI.

Section 4.7) and randomizes the order so that every time a test is taken the order of utterances changes. Three separate VNF directories are created for the loudness test, intelligibility test and acceptability test. To do this a naming scheme was used on the VNFs. Each VNF name begins with a two letter designator followed by an "_". These designators are "lt", "it" and "at" for loudness test, intelligibility test and acceptability test respectively. From this point on the user navigates through different screens based on the input received.

Once information is entered and the command "Begin" is called by pressing the left option key. The *CommandListener* method *commandAction* method then compares the command to the current screen. It stores the user information in $ansBuffer$ and

sets the display to *testScreen*. The user then selects one of the three test types and presses the command "Select" using the left option key. Again, the *commandAction* method compares the command to the current screen. It then initializes the corresponding listening test screen. The next three subsections will give detailed program execution based on the users test selection.

4.5.1 Loudness Test

The loudness test flowchart is shown Figure 4.4 for a reference. If the user selection was "Loudness Test" from the *testScreen*, the *CommandListener* method *commandAction* will initialize the loudness test screen, set the counter to zero and generate a random sequence, based on the length of the test (in this case 20). The sequence will consist of the numbers zero or one and will be used to determine in which order the enhanced and un-enhanced utterances will be played. The *loudScreen* screen is set as the current display. The user then selects one of the utterances, not knowing which is enhanced, and plays the sound by pressing "Play" using the left option key. The method *playSample* of class *ListenT* is passed the utterance selected. This method utilizes method *VoiceNote.play* to play the utterance. *VoiceNote* is a Java package provided by Motorola that allows the playing, recording and management of sound files in .vcf and .vnf format.

Once both sounds have been played, the user then selects which sounded louder by pressing "Select" using the right option key. The *commandAction* method verifies that both sounds have been played and then compares the selection to the random sequence element at the number of the counter. If they are equivalent, then "right" is stored in the *ansBuffer*. Otherwise, "wrong" is stored. Next, the counter is checked to see if 20 utterances have been evaluated. If it is not reached, the display remains *loudScreen* and the test is continued.

Once the test is complete (the counter reaches 20), the test results are stored to the database using the *ltAnsDB* object. Database storage will be discussed in

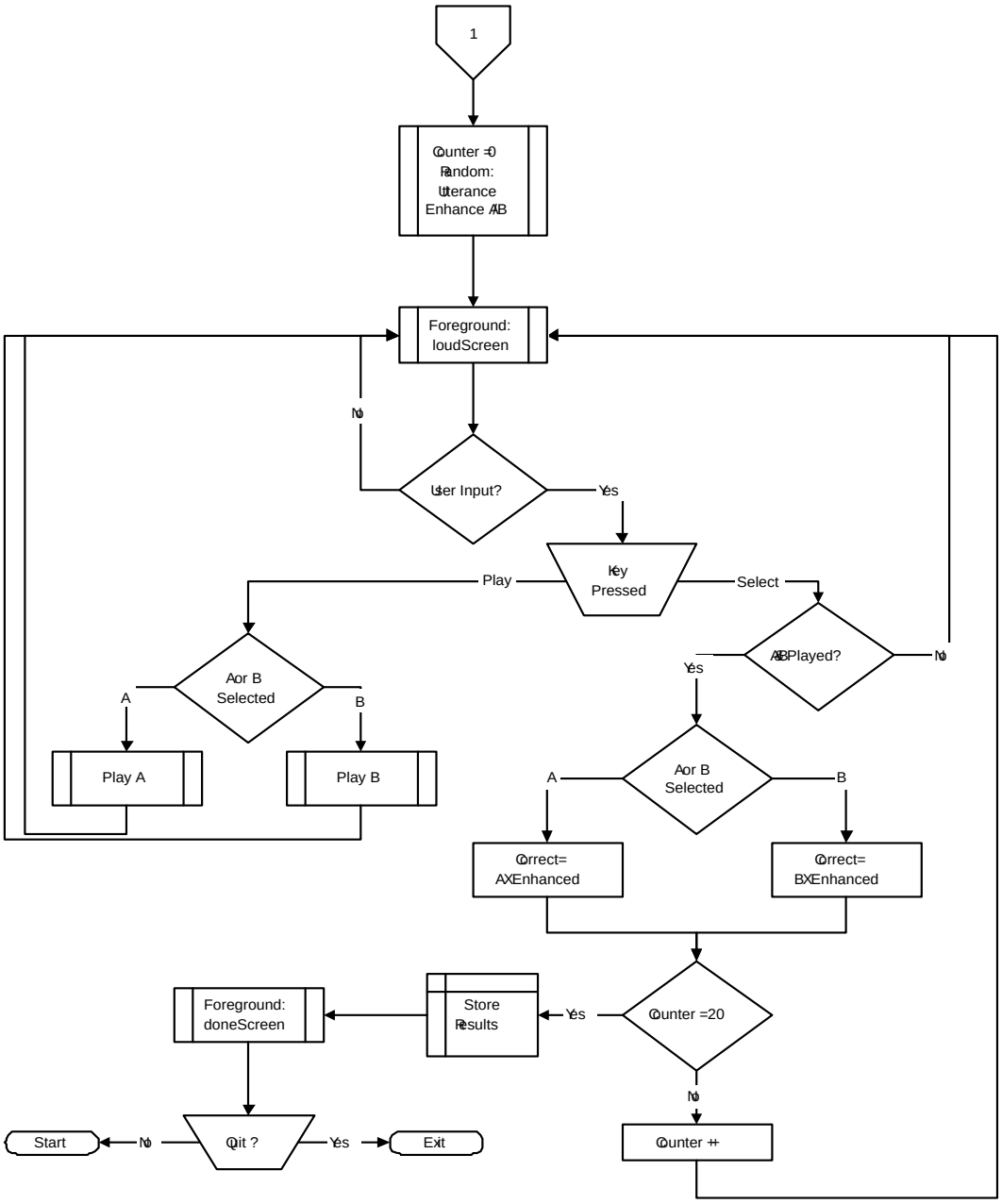

Figure 4.4: Flowchart for Loudness Test subprogram.

Subsection 4.5.4. After the data is stored, the display is set *doneScreen*. From here, the user can choose to take another test or exit the MIDlet. Figure 4.3 shows the GUI for the listening tests on the phone discussed in the next three sections.

4.5.2 Intelligibility Test

The intelligibility test flowchart is shown Figure 4.5 for a reference. If the user selection was "Intelligibility Test" from the *testScreen*, the *CommandListener* method *commandAction* will initialize the intelligibility test screen, set the counter to zero and generate a random sequence, based on the length of the test (in this case 20e). The sequence will consist of the numbers zero and one and will be used to determine the order that the correct and incorrect choices will be displayed. The *intelScreen* screen is then as the current display. The user then plays the sound by pressing "Play" using the left option key. The method *playSample* of class *ListenT* is passed the utterance selected.

Once the sound has been played, the user then selects which utterance was heard by pressing "Select" using the right option key. The *commandAction* method verifies that the sound has been played and then compares the selection to the random sequence element at the number of the counter. If they are equivalent, then "right_[utterance]_[algorithm]" is stored in the $ansBuffer$. Otherwise, "wrong_ [utterance]_[algorithm]" is stored. Next, the counter is checked to see if 20 utterances have been evaluated. If it is not reached, the display remains *intelScreen* and the test is continued.

Once the test is complete (the counter reaches 20), the test results are stored to the database using the *itAnsDB* object. Database storage will be discussed in Subsection 4.5.4. After the data is stored, the display is set *doneScreen*. From here, the user can choose to take another test or exit the MIDlet.

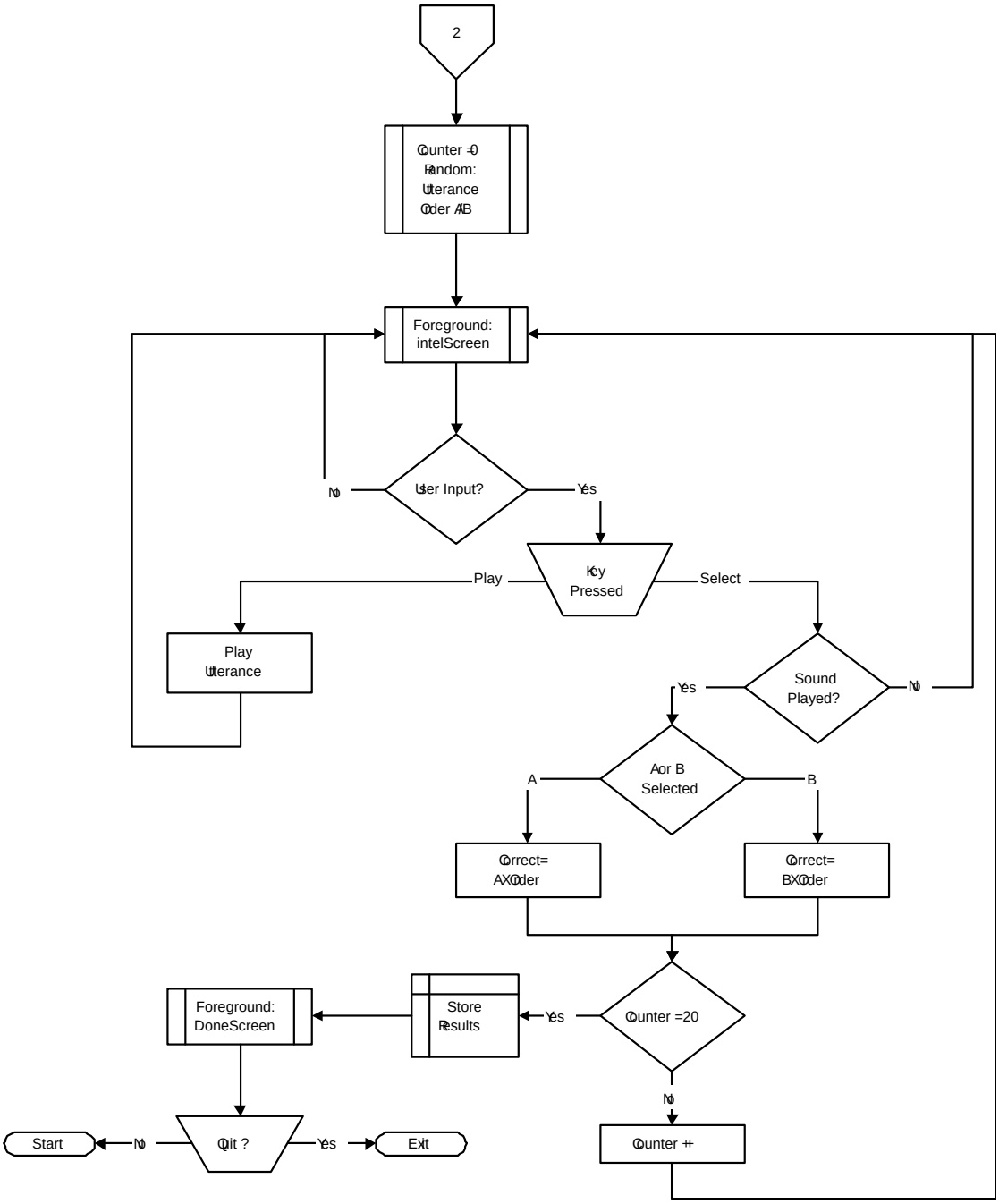

Figure 4.5: Flowchart for Intelligibility Test subprogram.

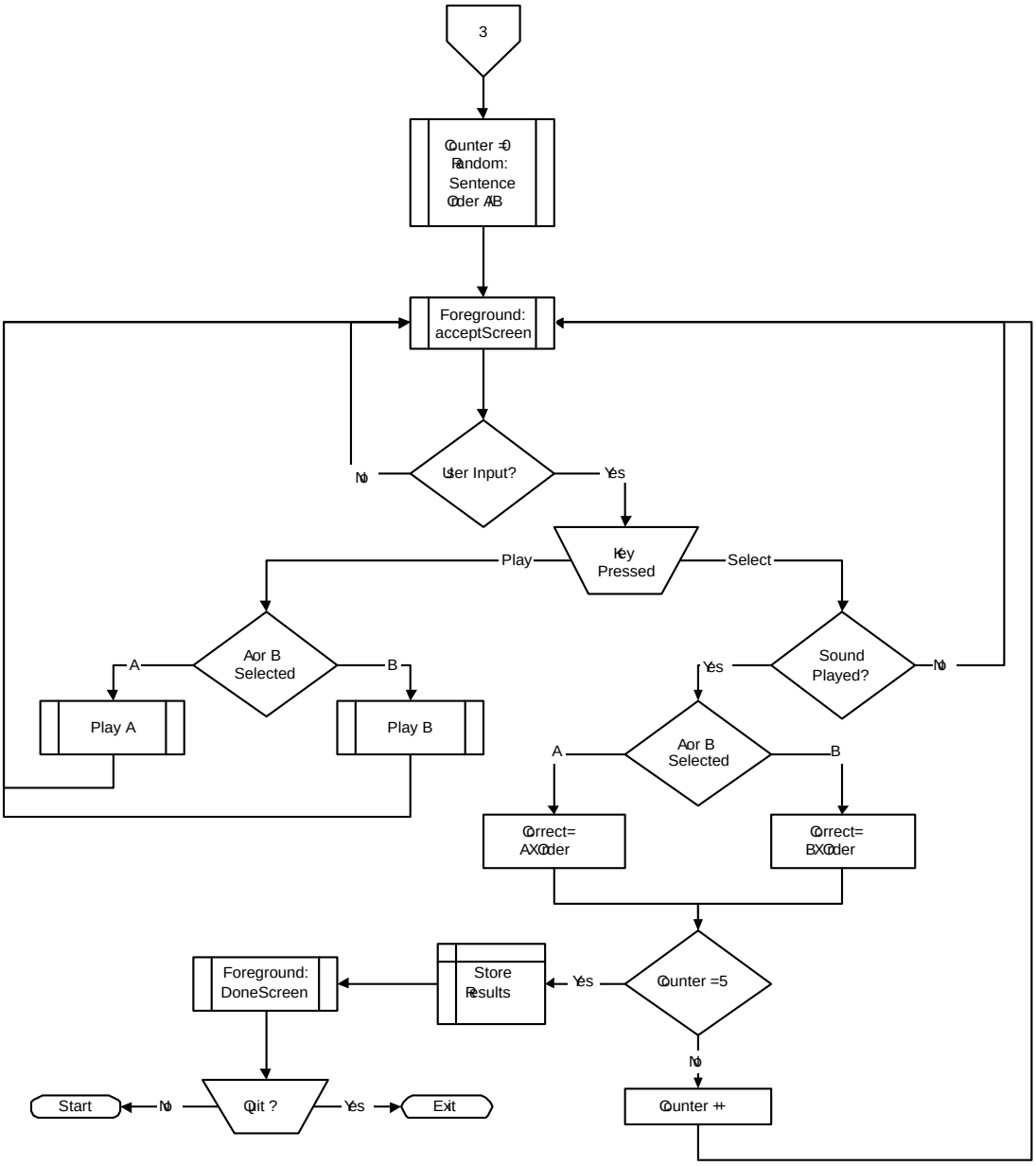

Figure 4.6: Flowchart for Acceptability Test subprogram

4.5.3 Acceptability Test

The acceptability test flowchart is shown Figure 4.6 for a reference. If the user selection was "Acceptability Test" from the *testScreen*, the *CommandListener* method *commandAction* will initialize the acceptability test screen and set the counter to zero. The *acceptScreen* screen is then set as the current display. The user then plays the sound by pressing "Play" using the right option key. The method *playSample* of class *ListenT* is passed the utterance selected. Once the sound has been played, the user then rates the quality by selecting "Excellent", "Good", "Fair" or "Poor" and pressing "Select" using the right option key. The *commandAction* method verifies that both the sound has been played and then compares the selection to the random sequence element at the number of the counter. Then "sent#_[algorithm]_[quality rating]" is stored in the *ansBuffer*. Next, the counter is checked to see if ten utterances have been evaluated. If it is not reached, the display remains *acceptScreen* and the test is continued.

Once the test is complete (the counter reaches ten), the test results are stored to the database using the *atAnsDB* object. Database storage will be discussed in Subsection 4.5.4. After the data is stored, the display is set *doneScreen*. From here, the user can choose to take another test or exit the MIDlet.

4.5.4 Database Storage

The storage of answers is performed using the *RecordStore* class and methods. The *ltAnsDB*, *itAnsDB* and *atAnsDB* objects are instances of *ListenDB*, which were created in the MIDlet constructor, and control storage of data to a record using the method *ListenDB.addTaskRecord*. The user information type and answers are stored sequentially using the delimiter ":?:" in a *String* object. The data is then stored in a *RecordStore* by the passing the string to the method *addTaskRecord*.

4.5.5 *RandObj* Class

The *RandObj* class is an extension of the *Random* class and allows the generation of random numbers and sequences. The *Random* class is a pseudo random number generator capable of providing a random number uniformly distributed between $\pm 2^{15}$. The method *Date.getTime* is used to generate a seed which is then passed to the method *Random.setSeed* within the RandObj object. Method *getRandNum* of the *RandObj* class are called from *ListenT* to generate these numbers.

4.6 *ListenDB* Class

The *ListenDB* class provides database management ability to both *ListenT* and *DbUpload* MIDlets. The two main control methods of *ListenDB*, *open* and *close*, allow access to a *RecordStore* database by checking to see if the database exists, verifying it is open, opening it and closing it. Four functional methods, *addTaskRecord*, *deleteTaskRecord*, *getRecordsByID* and *enumerateTaskRecord*, allow the addition of records, deletion of records, browsing of records and organization of records, respectively.

Note: The *ListenDB* class is not a MIDlet. It is only a class that adds functionality to other classes. It is not executable and becomes a inner-class when instantiated inside a MIDlet. The use of a separate class conserves memory. Since it is used by multiple MIDlets, it will not occupy memory in each MIDlet.

4.7 *Voicenotes* MIDlet

The *Voicenotes* MIDlet ,which implements the *VoiceNote* Class, allows the recording, playback, renaming and deleting of voice samples recorded on or off the phone. The only direct access to sound playback and recording is through *VoiceNotes*. The sound files are stored as vocoded data. The flowchart can be seen in Figure 4.7. Two other applications are written to support uploading and downloading sound files to and from a PC. The *simpleWriteVNF* MIDlet is run on the phone while the

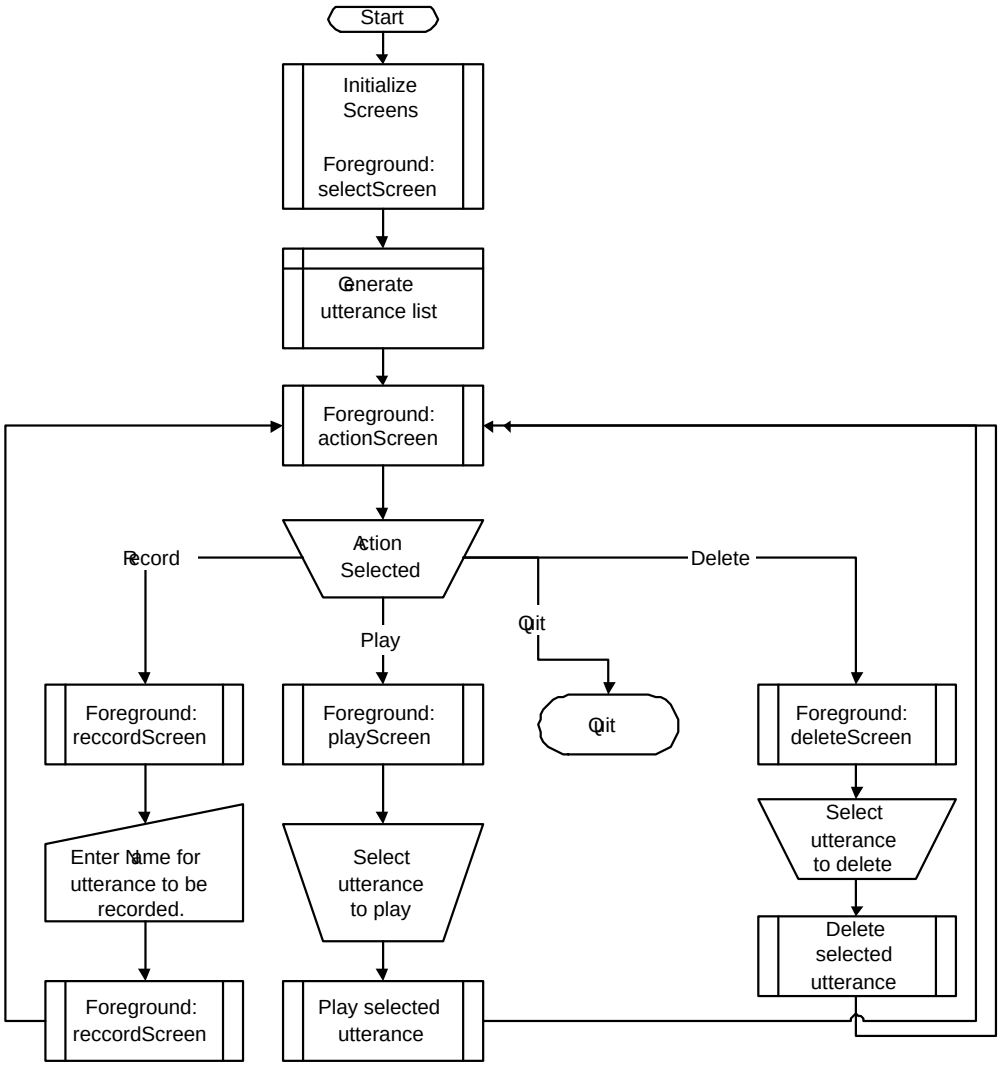

Figure 4.7: Flowchart for *VoiceNotes* MIDlet

simpleRead J2SE application is run on the PC. These two programs, provided by Motorola, utilize the Java™Communications API (*commapi*) package [24].

4.8 Voice Note Collection

Voice Note Files (VNFs) collected on the phone and down loaded to the PC as described in Section 4.7. These files are then de-vocoded using a PC application called "Voice Note Recorder" provided by Motorola. The de-vocoded file can then be read by MATLAB and processed. At this point, any enhancement or modification of the sound sample can be done. The next step would be to vocode the sound sample using the "Voice Note Recorder" utility and upload it to the phone. Additionally, PC recorded sound files can be vocoded and uploaded to the phone. Since, the PC recorded samples are only vocoded once, therefore, the quality of the sample is better than that of the samples recovered from the phone. The new VNFs are uploaded to the phone using the JAL utility. From there, they can be played or deleted using the *VoiceNotes* MIDlet. Figure 4.8 shows the GUI and program flow for the *Voicenotes* MIDlet.

4.9 *DbUpload* MIDlet

The listening tests results are stored as records on the phone. To recover these records, the *DbUpload* MIDlet is used. This MIDlet is part of the *ListenM* package, which allows it access to records stored in *recAnsDb* by the *ListenT* MIDlet. The flowchart for this MIDlet is shown in Figure reffig:dbflow. This program requires that the phone be connected to the PC's serial port for proper execution. The user is first prompted to connect the phone to the PC and run the PC based application *DbDownload*, explained in Section 4.10. Next, the user chooses which of the three test types to download. When finished, the user is asked if the downloaded files are to be deleted. Finally, the MIDlet is ended. Figure 4.10 show the GUI for the *DbUpload* MIDlet.

Figure 4.8: Javaphone Voicenotes GUI.

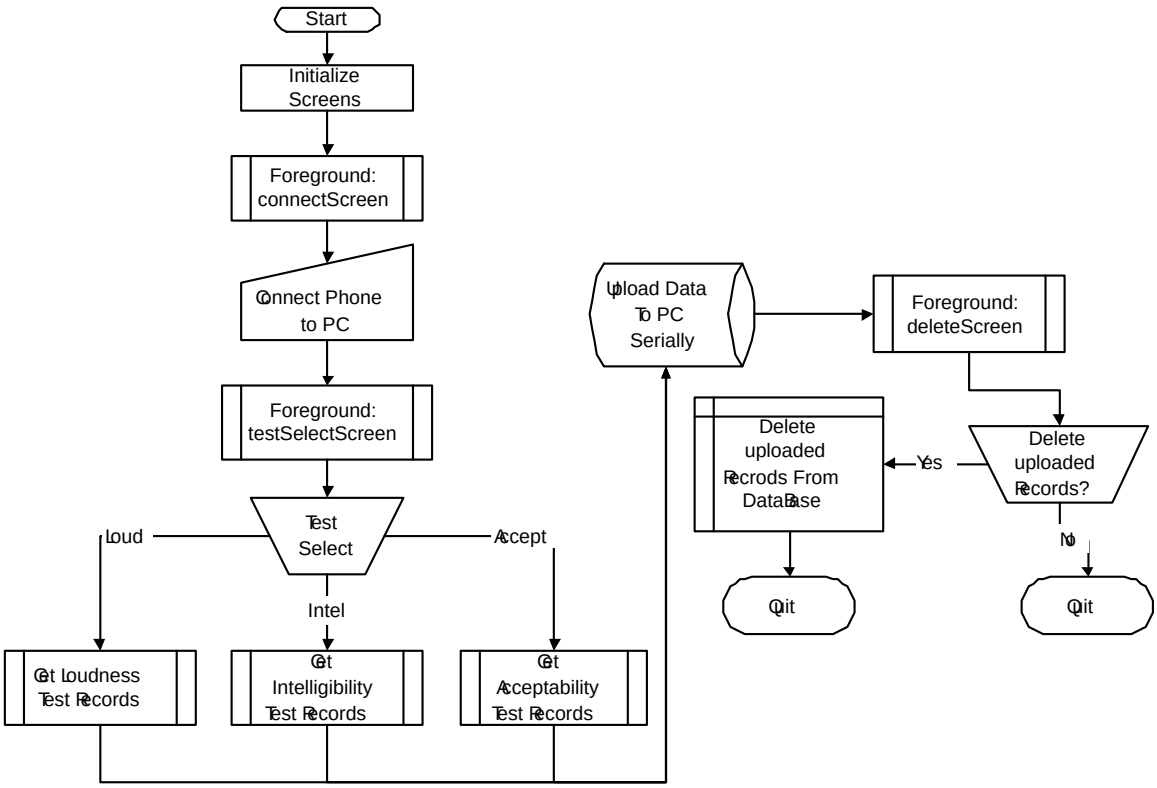

Figure 4.9: Flowchart for *DbUpload* MIDlet

4.10 PC Coding Using J2SE

The J2SE application *DbDownload* is used concurrently with the *DbUpload* MIDlet on the phone. For this application, the OBDC utility was used in Windows to connect an MS Access database to the application. The MS access database and the JDBC connection must share the same file name. Initially, the application connects to the database using JDBC (Java Database Connectivity). The application window can be seen in its initial mode confirming connection to the database, in reffig:dbdnfig. This window, created from class *Frame*, has three functional buttons; Exit, Connect to Phone, and Add Records. It also has three text-boxes to indicate the test type, number of records to be add and an edit comment box to added to the records. Additionally, it has a status display that indicates what step the MIDlet is in the download and database storage process. This display confirms step-by-step procedures and indicates when problems occur. Once the phone is connected to the serial port and

Figure 4.10: Javaphone DbUpload GUI.

DbUpload is run, the "Connect to Javaphone" button is pressed. The application will indicate the number of records and the test type that was downloaded. The user can then choose to add a comment to the records or leave it blank. The button "Add Records" is pushed and the system adds the records to the database using standard SQL commands in sub-class *AddRecords*. These results can then be analyzed using MS Access.

Figure 4.11: GUI for *DbDownload* Application Running on PC

CHAPTER 5
CONCLUSION

The goal of this thesis was to evaluate the real-world performance of the Energy Redistribution Voived/Unvoiced (ERVU) and Warped Bandwidth Expansion algorithms. Earlier testing resulted in increased intelligibility and increased perceptual loudness for these algorithms respectively. The algorithms were combined to concurrently enhance both intelligibility and perceptual loudness. Environmental noise, vocoding/devocoding effects and cellular phone speaker characteristics were incorporated in laboratory testing to mimic the cellular phone listening environment. PC based listening tests were performed to quantify the performance of the combined algorithm. To overcome the limits of laboratory testing, cellular phone based listening tests were developed in J2ME to provide a platform for testing algorithms in real-world environments. This will provide concrete results and help determine if the algorithms will be implemented fleet-wide.

The listening tests resulted in a 4.5% increase in intelligibility at -5dB SNR and 4dB perceptual loudness gain. These results show that the combined algorithm will provide increased performance without any added power to the speech signal. This provides sufficient motivation towards implementation of the enhancement algorithms on cellular phones.

The applications developed for the phones based listening tests allow the evaluation of vocoded speech. There are two short-comings which must be resolved before the tests will provide any conclusive results. First, the Java Virtual Machine (JVM) must have access to controlling streaming audio. At this time it only has direct control over prerecorded vocoded speech (Voice Notes). This may require separate class

development and a more elaborate interface between the JVM and the cellular phone DSP. At this time, speech enhancement can only be performed before the speech is vocoded. This is the reverse order of which the implementation will process speech. This will lead to inconclusive results since the effect of the encoding process by the vocoder on enhanced speech is unknown. When the decision is made by the Motorola iDEN Group to implement the algorithms on the cellular phone DSP and the interface between the JVM and DSP are completed, the J2ME code can be appropriately modified.

Future work should consider extensive cellular phone based testing once the proper implementations are made. This may lead to a reevaluation of the algorithms and parameters. It will be these tests that will provide the true performance of the algorithms. Additionally, wireless implementation of the communication between the phone and PC will require less overhead on the phone and make test alteration simpler. Making changes to tests from the PC side such as modifying questions, test length and speech samples could be possible. The use of *User Datagram Protocol* (UDP) [5] – an internet communications protocol that involves the sending of information packets – and streaming audio may help expedite any desired changes.

The J2ME listening test may also provide a useful byproduct that allows the evaluation of speech coders used in the cellular phone industry. Like algorithm evaluation, the optimal testing environment for vocoders is on the cellular phone itself. The ability to evaluate and quantify performance of vocoders on the phone could lead to a more timely determination of implementation.

APPENDIX A
J2ME CLASSES AND METHODS

The following sections will present the classes, subclasses and methods developed in the J2ME environment. All methods created will be discussed. Methods inherited through extension of a class or implementation of an interface will not be discussed. For a description of these methods see [25, 18, 13]. Additionally, objects defined in a class that are already part of the J2SE or J2ME SDKs will not be discussed. Source code may be obtained by sending an email request to William O'Rourke at worourke@cnel.ufl.edu.

A.1 The ListenT Class

The ListenT class extends the MIDlet Class and implements the CommandListener interface. See Table A.1.

The RandObj class RandObj is a subclass of the ListenT class that extends the Random class. It generates pseudo random integers from $\pm 2^{31}$. A seed is set by calling Random.setseed(long seed) and passing it the current system date measured from the epoch. See Table A.2.

A.2 The DbUpload

The DbUpload class extends the MIDlet class and implements the CommandListener interface. See Table A.3.

A.3 VoiceNotes2

The VoiceNotes2 class extends the MIDlet class and implements the CommandListener and VoicenotesEnglish interfaces. See Table A.4.

Table A.1: Methods for ListenT.class

Method	Arguments	Returns	Description
getVoiceNoteList		String[]	Returns all voice notes on the phone.
initUserScreen			Initializes the User Information Screen.
initTestScreen			Initializes the Test Select Screen
initLoudTestScreen			Initializes the Loudness Test Screen
initInteligTestScreen			Initializes the Intelligibility Test Screen
initAcceptTestScreen			Initializes the Acceptability Test Screen
initDoneScreen			Initializes the Exit Screen
setUpIntel			Sets up the Intelligibility screen for proper display.
playSample	String		Plays a voice note by name.
playSample	int, int		Plays a voice note randomly by ID.
randomizeDir	String[]	String[]	Randomizes voice note directory.

Table A.2: Subclass RandObj.class of ListenT.class

Method	Arguments	Returns	Description
getRandNum	int	int	Returns the next pseudo random integer limited to $\pm 2^{numBits-1}$.
getRandNum		int	Returns the next pseudo random integer.

Table A.3: Methods for DbUpload.class

Method	Arguments	Description
initConnectScreen		Initializes the PC Connection Screen.
initTestListScreen		Initializes the Test Select Screen
initDeleteRecordsScreen		Initializes the Delete Option Screen
initDeleteOKScreen		Initializes the Verify Delete Screen
initDoneScreen		Initializes the Exit Screen
sendRecords	int	Gathers records by type and calls sendRecord.
deleteSentRecords	int	Deletes a Listening Test Records by type
sendRecord	String	Deletes a specific record

Table A.4: Methods for VoiceNotes2.class.

Method	Arguments	Description
initCommandScreen		Initializes the Command Select Screen.
initRecordInfoScreen		Initializes voice note info Screen.
initRecordScreen		Initializes the voice note recorder Screen.
initList		Initializes the voice note list Screen.
initDeleteScreen		Initializes the voice note Delete Screen.
initRenameScreen		Initializes the voice note rename Screen.
initDoneScreen		Initializes the Exit Screen.
record	String	Records a voice note.
play		Plays a voice note.
delete		Deletes a voice note.
rename	String, String	Renames a specific voice note.

APPENDIX B
J2SE CLASSES AND METHODS

The following sections will present the classes, subclasses and methods developed in the J2SE environment in the same manner as Appendix A.

B.1 DbDownload

The DbDownload.class extends the JFrame class and has subclasses shown in Figure B.1. This class creates and instance of itself which is shown in the figure as DbDownload$1. This class creates the JDBC connection, instantiates ScrollingPanel and ControlPanel objects. No methods were required to be added to this class.

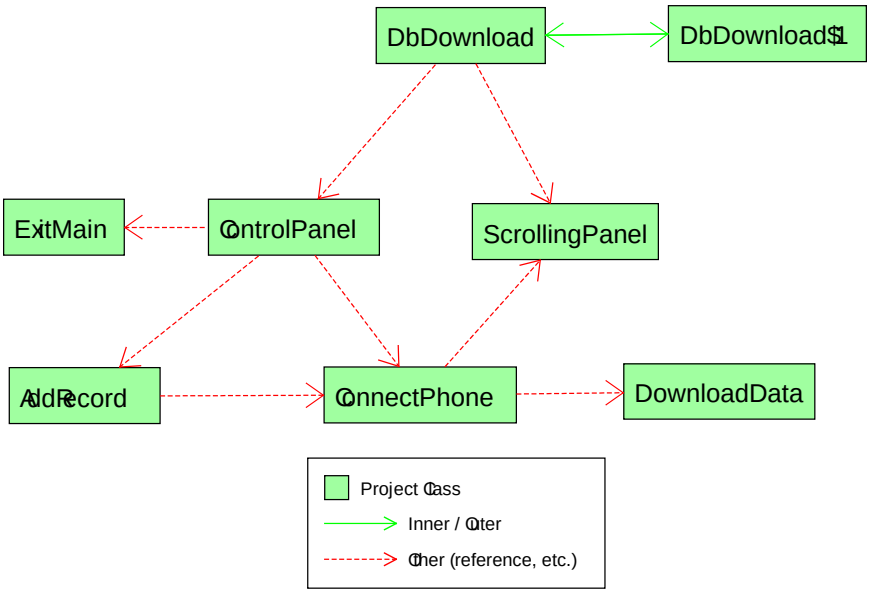

Figure B.1: The DbDownload class and its subclasses

Table B.1: Methods for ConnectPhone.class

Method	Arguments	Description
sortData	String[]	Sorts received string using delimiter ":?:".

B.2 ControlPanel and ScrollingPanel

The ControlPanel and ScrollingPanel classes extend the JPanel class. These class have no added methods. The ControlPanel class has instantiates ExitMain and ConnectPhone objects. The ExitMain class has no added methods.

B.3 ConnectPhone

The ConnectPhone class implements the ActionListener interface. This class instantiates an DownloadData object. DownloadData implements Runnable and SerialPortEventListener interfaces. See Table B.2 and B.1.

Table B.2: Methods for DownloadData.class

Method	Arguments	Description
getPhoneData		Sets up serial port input stream.
run		Starts Thread for reading data.
stop		Stops Thread for reading data.
serialEvent	SerialPortEvent	Listens for events on the serial port.
phoneConnect		Captures the serial port.

REFERENCES

[1] A. Acero X. Huang and H. Hon. *Spoken Language Processing.* Prentice-Hall PTR, Upper Saddle River, NJ, 2001.

[2] T. Baer B.R. Glasberg and B.C. Moore. Revision of Zwicker's Loudness Model. *Acustica*, 82:335–445, 1996.

[3] M.A. Boillot. A Psychoacoustic Approach to the Loudness Enhancement of Speech Part I: Formant Expansion. Submitted to *International Conference on Acoustic, Speech and Signal Processing,* Hong Kong 2003.

[4] M.A. Boillot. *A Warped Filter Implementation for the Loudness Enhancement of Speech.* PhD Dissertation, University Of Florida, 2002.

[5] H. Deitel and P. Deitel. *Java^{TM}How to Program.* Prentice-Hall Inc., Upper Saddle River, NJ, 1999.

[6] H. Fastl E. Zwicker and C. Dallmayr. Basic-program for Calculating the Loudness of Sounds from their 1/3-oct. Band Spectra According to ISO 532 B. *Acustica*, 55:63–67, 1984.

[7] H. Fletcher and W. J. Munson. Loudness, its Definition, Measurement, and Calculation. *Journal of the Acoustical Society of America*, 5:82–108, 1933.

[8] W. Hartmann. *Signals, Sound and Sensation.* Springer, New York, NY, 1998.

[9] The International Organization for Standardization. *ISO 226. Acoustics–Normal Equal Loudness Contours.* ISO, Geneva, Switzerland 1987.

[10] The International Organization for Standardization. *ISO 532 B. 1975. Method B for Calculating the Loudness of a Complex Tone That Has Been Analyzed in Terms of One-third Octave Bands.* ISO, Geneva, Switzerland 1975.

[11] J. Johnston. Transform Ccoding of Audio Signals Using Perceptual Noise Criteria. *IEEE Journal on Selected Areas in Communications*, 6(2):314–323, 1988.

[12] J. C. Junqua. The Influence of Psychoacoustic and Psycholinguistic Factors on Listener Judgments of Intelligibility of Normal and Lombard Speech. In *Proceedings of International Conference on Acoustic, Speech, and Signal Processing*, volume 1, pages 361 – 364, Causal Productions Pty Ltd., Toronto, Canada 1991.

[13] M. Kroll and S. Haunstein. *Java^{TM}2 Micro Edition Application Development.* Sams Publishing, Indianapolis, IN, 2002.

[14] S. Mitra. *Digital Signal Processing: A Computer-Based Approach* Second Edition. McGraww-Hill, New York, NY, 2001.

[15] M. Morrison. *Sams Teach Yourself Wireless Java with J2ME in 21 Days.* Sams, Indianapolis, IN, 2001.

[16] Motorola Corporation. J2ME^{TM}Technology for Wireless Communication Devices. http://developers.motorola.com/developers/j2me, Accessed July 2002.

[17] Motorola Corporation. *Motorola Protocol Manual 68P81129E15-B VSELP 4200 BPS Voice Coding Algorithm for iDEN.* iDEN Division, Plantation, FL, 1997.

[18] Motorola Corporation. *Motorola SDK Components Guide for the J2ME Platform.* Austin, TX, 2000.

[19] Motorola Corporation. *Stand Alone Voice Activity Detector High-Level and Low-Level Design Document.* iDEN Division, Plantation, FL, 1999.

[20] S. Namba and H. Miura. Advantages and Disadvantages of A-Wweighted Sound Pressure Level in Relation to Subjective Impression of Environmental Noises. *Noise Control Engineering Journal*, 33:107–115, 1989.

[21] Nextel iDEN. *i90c Phone Users Guide.* Nextel Communications Inc., Reston, VA, 2001.

[22] T.L. Reinke. *Automatic Speech Intelligibility Enhancement.* Master's Thesis, University Of Florida, 2001.

[23] Rice University. Signal Processing Information Base. http://spib.rice.edu/spib/select_noise.html, Accessed June 2002.

[24] Sun Microsystems Inc. Communications API. http://java.sun.com/products/javacomm/, Accessed April 2002.

[25] Sun Microsystems Inc. Java^{TM}2 Standard Edition API. http://java.sun.com/j2se/1.4/docs/api/index.html, Accessed September 2001.

[26] Texas Instruments. TI 46-Word Speaker-Dependent Isolated Word Corpus (cd-rom), 1991.

[27] Texas Instruments. DARPA TIMIT Acoustic-Phonetic Continuous Speech Corpus (cd-rom), 1991.

[28] W. D. Voiers. Ch. 34 *Diagnostic Evaluation of Speech Intelligibility.* Dowden, Hutchinson, and Ross Inc., New York, NY, 1977.

[29] E. Zwicker and H. Fastl. *Psychoacoustics: Facts and Models* 2nd Edition. Springer Verlag, New York, NY, 1999.

BIOGRAPHICAL SKETCH

William O'Rourke was born on November 5, 1970, in Buffalo, NY. At the age of five his family moved to Boca Raton, Florida. After high school, he entered the US Navy. During that time, he spent five years stationed on two ships forward deployed to the US Seventh Fleet in Japan. He travelled extensively through Asia meeting new people and learning new customs, a priceless experience. At the age of 27, he decided to leave the Navy and pursue a degree in electrical engineering at the University of Florida.