Reduction of Low Frequency Loudspeaker Distortion Using an Adaptive, Driver Independent Digital Signal Processing System

Frank Jania & Taylor Sherman
Senior Thesis
09.May.97
Dept. of Electrical Engineering
The Cooper Union
New York, NY
Prof. Fred Fontaine
An adaptive digital filtering system is developed for low-frequency audio transducers to reduce nonlinear transducer responses in an effort to eliminate audible harmonic distortion. A Spectrum PC/C5X digital signal processing (DSP) board is used to implement the filtering algorithm, and a piezoelectric accelerometer is used to measure the transducer’s movement. The filter is based on the least-mean-square adaptive algorithm. MATLAB simulation shows at least 90% reduction in distortion, but processor speed limits the overall effectiveness of the system.
TABLE OF CONTENTS

1 INTRODUCTION ........................................................................................................1-1
  1.1 LOUDSPEAKERS VS. OTHER HI-FI EQUIPMENT .................................................1-1
  1.2 DIGITAL SIGNAL PROCESSING .......................................................................1-2
  1.3 APPROACH ........................................................................................................1-3
  1.4 HARDWARE .........................................................................................................1-3
  1.5 OVERVIEW .........................................................................................................1-4

2 LOUDSPEAKER DESIGN ISSUES ............................................................................2-1
  2.1 SOURCES OF DISTORTION ............................................................................2-2
  2.2 ANALOG CONTROL SYSTEMS ......................................................................2.3
    2.2.1 LOUDSPEAKER ENCLOSURE DESIGN ....................................................2-4
    2.2.2 CONSEQUENCES OF ANALOG SYSTEMS ............................................2-5

3 DIGITAL FILTERS ....................................................................................................3-1

4 ADAPTIVE FILTERS ...............................................................................................4-1
  4.1 ADAPTIVE FILTER LAYOUT .........................................................................4-2
  4.2 LMS ALGORITHM ..........................................................................................4-2
  4.3 APPLICATIONS OF ADAPTIVE FILTERS ....................................................4-4

5 SYSTEM SETUP AND MEASUREMENTS ...............................................................5-1
  5.1 LOUDSPEAKER AND AMPLIFIER ................................................................5-1
  5.2 ACCELEROMETER ........................................................................................5.2
  5.3 MEASUREMENTS .............................................................................................5-3
5.4 ADDITIONAL EQUIPMENT

6 DSP SYSTEM
   6.1 TEXAS INSTRUMENTS TMS320C50
   6.2 SPECTRUM PC/C5x BOARD
   6.3 SPECTRUM AM/D16SA DAUGHTER MODULE
   6.4 C50 ASSEMBLY LANGUAGE
      6.4.1 CIRCULAR BUFFERING
      6.4.2 AMELIA

7 SIMULATION
   7.1 MATLAB SETUP
   7.2 ANALYSIS

8 RESULTS
   8.1 CHANGES IN THE SYSTEM
   8.2 RESULTS

9 CONCLUSION
   9.1 WHAT WORKED AND WHAT DIDN’T
   9.2 CONCLUSIONS

REFERENCES

APPENDIX A: MATLAB SIMULATION
APPENDIX B: ASSEMBLY PROGRAM
APPENDIX C: EQUIPMENT LIST
Adaptive Distortion Correction

Figures

2.1 Dynamic Loudspeaker Cross-section ........................................2-1
2.2 Magnetic Fields In Driver, Cross-section .................................2-2
2.3 Frequency Response of Driver ..................................................2-4
3.1 Fourth Order FIR Digital Filter .................................................3-1
4.1 Adaptive FIR Digital Filter .......................................................4-2
5.1 System Block Diagram ...........................................................5-1
5.2 Quartz Crystal Cut Types .........................................................5-2
5.3 Compression Mode Accelerometer ..............................................5-3
5.4 Shear Mode Accelerometer .......................................................5-3
5.5 Power Spectral Density of Signal Generator Output, 15Hz .......5-4
5.6 Power Spectral Density of Loudspeaker Output, 15Hz ............5-5
5.7 100Hz Low Pass Filter Circuit Diagram ....................................5-5
5.8 Low Pass Filter Frequency Response .......................................5-6
5.9 Accelerometer Amplifier Circuit ..............................................5-6
5.10 Accelerometer Amplifier Frequency Response, Gain = 3 ......5-7
5.11 Accelerometer Amplifier Frequency Response, Gain = 70 ......5-7
6.1 Play Through Using DSP, Fs=1kHz, No Filtering ......................6-3
7.1 Time Domain Response, Before Convergence ............................7-3
7.2 Time Domain Response, After Convergence ..............................7-3
7.3 Spectrum of Distorted Signal vs. Input ....................................7-4
7.4 Spectrum of Filter Output vs. Input ........................................7-4
8.1 Final System Configuration ......................................................8-3
8.2 Input and Output of System, 15Hz ...........................................8-5
8.3 PSD of System Output, Filters Inactive, 15Hz ..........................8-5
8.4 PSD of System Output, Filters Active, 15Hz .............................8-5
8.5 Input and Output of System, 30Hz ..........................................8-6
8.6 PSD of System Output, Filters Inactive, 30Hz .........................8-6
8.7 PSD of System Output, Filters Active, 30Hz ............................8-6

Tables

6.1 Circular Buffering .....................................................................6-6
6.2 AMELIA Register Mapping ....................................................6-6
I. INTRODUCTION

The state of the art for music and sound reproduction has advanced continuously since the introduction of recording and broadcasting techniques around the turn of the century. Over the years, sound reproduction has continued to become more life-like, to the point where today certain systems have been known to fool very well trained ears into thinking that a reproduction was the real thing. However, these systems are still rare and enormously expensive. As further technologies become available, there is no doubt they will be utilized in the never-ending effort to bring more realism to sound reproduction. One critical problem in sound reproduction has always been, and still is today, nonlinear loudspeaker effects at low frequencies.

1.1 LOUDSPEAKERS VERSUS OTHER HI-FI EQUIPMENT

As mentioned above, music and sound reproduction today is very advanced. New technologies in electronics and digital systems have led to storage and playback systems with 100dB dynamic ranges, 110dB Signal-to-Noise ratios, and harmonic distortion on the order of $10^{-5}$ percent. That is, except for loudspeakers. Of course, loudspeakers have the most difficult job of the playback system, as they have to convert one type of energy (electricity) to another (sound), and do it at very high power levels. Even if one ignores all of the directional, phase, and magnitude problems associated with loudspeaker designing, one problem remains: very few low frequency drivers are capable of linear operation at even moderate power levels. A glance at Stereo Review or Stereophile will show that most loudspeaker systems which would fit in a home exhibit harmonic distortion in excess of 10% at 95dB and frequencies below 100Hz -- a perfectly reasonable
Several methods for eliminating nonlinear loudspeaker responses have been proposed and implemented over the past decade or so. These include push-pull driver arrangements, improved magnetic field and materials design, and motion-based analog control systems. However, these systems all have severe limitations. Push-pull designs eliminate only odd-order harmonic distortion, and require multiple redundant drivers. Also, despite improvements in driver design and routing of the magnetic field, even the most modern and expensive drivers exhibit nonlinear responses. Finally, several analog accelerometer-based control systems have been marketed by Velodyne and others which reduce distortion very successfully. However, these must be integrated systems, where the driver, amplifier, enclosure, and feedback transfer function must all be designed carefully and integrated together. This leaves no room for the consumer to choose a specific amplifier or speaker. As yet, no systems exist which are capable of providing significant distortion reduction on existing loudspeaker systems.

1.2 DIGITAL SIGNAL PROCESSING

Currently, very few audio playback systems utilize digital processes in relation to the loudspeakers. Quite the contrary, passive analog filters are still commonly used to send the proper frequencies to the respective drivers. As mentioned above, the only systems which attempt to actively reduce loudspeaker distortion are analog. Digital systems are used elsewhere in the audio chain, so it seems obvious that they be applied here.

The most promising form of digital signal processing for this project is an adaptive filter. Adaptive digital filters alter their own structure to satisfy certain goals. In this way, an adaptive
system could continuously pre-filter the signal before it reaches the loudspeaker, effectively removing distortion before it’s added.

1.3 **APPROACH**

To analyze the usefulness of a particular adaptive system, a MATLAB simulation can be conducted. MATLAB is a very powerful tool for simulating signal processing systems. Signals and filters are represented as vectors and matrices in MATLAB, which greatly simplifies their implementation. MATLAB is capable of simulating analog, digital, linear, and nonlinear systems. For this project, a model of a low-frequency loudspeaker was designed in MATLAB, and various adaptive schemes are simulated to judge their effectiveness.

Once a particular design is accepted, the system is implemented using the loudspeaker itself. The filter operations are performed using a digital signal processor (DSP), which is designed for executing these operations quickly and efficiently. The processing power involved should require only a single medium-power DSP, various models of which are currently available for less than a hundred dollars.

1.4 **HARDWARE**

The full system implementation requires a loudspeaker, amplifier, digital-to-analog and analog-to-digital converters, DSP system, music source, and an accelerometer. The accelerometer measures the loudspeaker’s output, and the DSP system receives this, as well as the input from the music source (a CD player), as input. The output of the DSP system is sent through the amplifier to the loudspeaker. The accelerometer, a Kistler K-SHEAR 8614A500M1, is very small and
lightweight, and mounted on the cone of the loudspeaker itself.

The DSP used for this project is a Texas Instruments TMS320C50, a 40MHz processor capable of performing 20 million instructions per second (MIPS). This chip is mounted on a Spectrum PC/C5x ISA-compatible card, which allows programming and debugging of the system through a 486DX4/120 PC. A Spectrum daughter module (DM) mounted on the board performs the necessary analog-digital signal conversions.

1.5 OVERVIEW

This paper explores the topics described above. Specifically, it details the design and implementation of a adaptive digital system which should reduce the distortion output from a low-frequency loudspeaker system. Chapter 2 begins by addressing in detail the design of loudspeaker drivers, what causes this distortion, and methods which attempt to reduce it.

Chapters 3 and 4 discuss the digital aspects of the system, starting with digital filtering in general and then describing adaptive filters. Chapter 4 also goes into depth on the LMS algorithm, the particular adaptive algorithm used for this project. The actual location and implementation of these filters in the overall system is described in Chapter 5. Chapter 5 discusses the details of the external hardware present in the system, from the accelerometer to various additional pieces of equipment.

The DSP system is described in Chapter 6. The TMS320C50, the ISA board, and the daughter module are all discussed, as well as programming methods specific to these systems. The MATLAB simulation is detailed in Chapter 7, and then the overall results of the system are presented and explained in Chapter 8.
The conclusion discusses successes, problems, their causes, and possible changes made to future systems. Based on the outcome of this project, it appears that the reduction of loudspeaker distortion is a perfect application for adaptive digital filters. With technology currently available, it should be possible to bring distortion levels from loudspeaker down an order of magnitude or more, and maybe eventually down to the same level as that of other parts of audio playback systems.
2. **LOUDSPEAKER DESIGN ISSUES**

In more than two thirds of the loudspeaker designs on the market, low frequency information is reproduced by a conic, electromagnetic transducer. These transducers, commonly known as dynamic loudspeakers, consist of a signal-driven coil, attached to a cone, which is placed in a permanent magnetic field. The current in the coil, which travels perpendicular to the magnetic field, causes the coil, and in turn the cone, to move in a way proportional to the input signal:

\[
F = I \times B
\]

(2.1)

where \(F\) is the induced force, \(I\) is the electric current in the voice coil, and \(B\) is the magnetic flux through the gap between the top plate and the pole piece, illustrated in Figure 2.1. The movement of the cone creates pressure waves which radiate as sound. To keep the cone moving linearly in one dimension, two elements, the surround and the spider, support the cone and attach it to the driver’s chassis.

If the loudspeaker were projecting into a small enclosed space, the pressure level in the space would be proportional to the cone’s position. However, most listening areas are of a volume much greater than the volume displaced by the cone’s movements. In these large spaces, the pressure waves created by the cone are proportional to the acceleration of the cone. That the sound waves are not proportional to velocity can...
be shown by the following example: if the cone moved with a constant velocity, the resultant pressure wave incident on a stationary observer would be an impulse from the initiation of the movement. Constant velocity movement creates pressure zones only on the boundaries of the object in motion. For waves to be created, the object must accelerate.

Since the cone’s acceleration is proportional to the forces acting upon it, this means that the sound waves created are directly proportional to the input current. However, the motion of the cone is linearly proportional to the input signal only if certain ideal conditions are met. First, the magnetic field through which the voice coil moves must be uniform. Second, the surround and spider must be infinitely rigid in two dimensions, and infinitely compliant in the other. Of course, the world is not ideal and these conditions often do not hold.

2.1 SOURCES OF DISTORTION

Non-ideal conditions in a dynamic loudspeaker lead to response nonlinearities and therefore distortion. First, the magnetic field in the region where the voice coil travels, the magnetic gap, is not uniformly distributed. The presence of the cone prevents creating a symmetrical magnet array; therefore the coil experiences different magnetic fields traveling forwards than backwards. The field is also attenuated when the coil travels either direction from center. The latter effect alters both the positive and negative pulls on the coil, and leads to even-order harmonic dis-
The former affects positive and negative motion differently, thereby causing odd-order harmonic distortion.

The surround and spider suffer from similar problems. For both, although some finite resistance is acceptable and simply lowers efficiency, their resistance increases as the cone travels further from the rest position. Also, inconsistencies in construction lead to different resistances for forwards and backwards motion. Like the magnetic field’s inconsistency, the former leads to odd-order harmonic distortion and the latter to even-order distortion. For a typical, “high-fidelity” low frequency system, the sum of these distortions may exceed twenty percent at reasonable frequencies and sound pressure levels (for instance 90dB at 30Hz).

### 2.2 Analog Control Systems

Several methods have been implemented to reduce this distortion, such as T-shaped or more complex pole piece geometries. This helps reduce asymmetry in the magnetic field, but the region of uniform flux density is still of some finite length, thus limiting the power output for linear operation. To reduce extraneous cone movement, many amplifier manufacturers have built their amplifiers to have very low output impedances, citing the ratio of driver impedance to amplifier output impedance as the “damping factor”, but the finite resistance introduced by cables and passive filters actually negates this effect. Some systems, such as Yamaha’s Active Servo Technology, utilize negative output impedance for increased cone control. However, these must be dedicated systems where the amplifier and loudspeaker are sold as a unit.

The most successful distortion-reducing systems on the market use an accelerometer to provide motional feedback. Velodyne, Infinity, and Genesis use this method in some of their
products, but, again, they are all expensive, dedicated systems. The output of the accelerometer is fed back through an analog filter, then subtracted from the original input signal, then sent to the loudspeaker’s amplifier. The feedback filter must be chosen carefully to ensure stability and provide useful results. This requires knowledge of the transfer functions of the loudspeaker and the amplifier.

### 2.2.1 Loudspeaker Enclosure Design

A loudspeaker’s transfer function, especially including the above mentioned nonlinearities is complex and difficult to determine. All low frequency drivers must be placed in an enclosure to operate properly. This is because sound, at low frequencies, radiates omnidirectionally. Any cone-based loudspeaker can be considered a dipole radiator, where the pressure waves created by the front and back areas of the cone are 180° out of phase. At high frequencies, where these waves are directional, they don’t interfere with one another. However, at low frequencies these waves both travel in all directions and therefore interfere with each other. Since the distance between dipole sources (the thickness of the cone) is much less than a wavelength, these waves are almost entirely 180° out of phase wherever they encounter one another. Therefore, a loudspeaker with no enclosure is terribly inefficient at low frequencies (usually those below 100 or
To circumvent this problem, various methods have been employed. Ideally, the driver would be mounted on an infinite planar baffle, preventing the rear wave from interfering with the forward wave. In the 1950’s, this idea was realized by mounting drivers in the wall of a home or theater. For mobile systems, a compromise is necessary, and the driver must be mounted in a box of some sort. This box may be sealed, which limits the low frequency reproduction of the loudspeaker, or vented in some way. Theile and Small developed several models to describe these systems in the 1970’s, where a closed box was modeled as a 2nd-order high pass filter and a vented box as a 4th-order high pass filter. This allowed loudspeaker designers to mathematically predict the effects that changes on an enclosure system would have: changes in enclosure would result in a new resonant frequency and system Q.

2.2.2 Consequences of Analog Design

These systems are all integrated in the sense that whatever enclosure is chosen, combined with the electrical and mechanical characteristics of the loudspeaker, will result in a specific electrical characteristic for the entire system. This means that the transfer function of a loudspeaker system is a function of both the loudspeaker and the enclosure. Therefore, an analog control system must be designed as a whole from the ground up. The substitution of a different loudspeaker, amplifier, or enclosure would upset the entire system, and it would no longer function correctly. However, systems employing these methods have been successfully marketed which demonstrate distortion levels more than an order of magnitude lower than other, passive systems of comparable size.
Based on the success of such tests, it appears that there is much promise for motion-sensing feedback systems in low frequency reproduction. However, the control systems employed currently for such purposes are all analog systems. This may explain why the Velodyne systems, which actually compete against some extremely expensive, higher-distortion, passive systems, have not dominated the consumer market. Two very important measurements of a sound transducer’s performance are its transient response and its phase distortion. While the above analog systems may result in low steady-state distortion, they are inherently slow to responding to an incoming signal. This delay may be accounted for by delaying the signals to the higher frequency drivers, but only if the delay is constant for all frequencies which it most likely is not.

Such phase distortion reduces the coherency of the reproduced sound. On the other hand, a high-speed digital system should be able lower the harmonic distortion without adding excessive delays or phase distortion. Additionally, the robustness of a digital system will allow the control system to be designed independently of the amplifier and the loudspeaker, thus allowing the consumer increased choice in purchasing a system or upgrading their current system.
3. **Digital Filters**

For the past few decades, digital systems have been slowly replacing the analog systems which were used to perform a variety of tasks. Analog filters, consisting of resistors, capacitors, inductors, and transistors, were and still are used to change some electrical signal into one which is more desirable. Many years of experience has led to analog filters which are very effective. However, analog filters are subject to the constraints which govern their physical components. To implement some filters using analog components, very elaborate configurations are required.

Digital filters deal not with physical entities such as magnetic and electric fields, but with information. Digital filters take the information present in a signal and affect it directly. Of course, the resolution of this information is finite, but dealing with signals in this way opens many new possibilities. To start with, the behavior of any analog filter can be approximated arbitrarily well by an appropriate digital filter. This alone is an improvement, as, for instance, changing a digital filter requires merely a software change and not the replacement of physical components. This is the key behind adaptive filters, which will be discussed in the next section, which change themselves in response to changing signals.

There are two types of digital filters: those with a finite impulse response duration, or FIR filters, and those with an infinite impulse response duration, or IIR filters. The impulse response of a filter is its output when the input is a pulse with a short time duration. The output of an FIR filters.
filter depends only on its inputs for some finite amount of time. On the other hand, the output of an IIR filter depends on what its previous *output* was. All analog filters, it turns out, are equivalent to IIR filters. This is because the input and output of a component such as an inductor are always connected together through electric and magnetic fields. Since an IIR filter’s output depends recursively on what its previous output was, an IIR filter “struck” by a short impulse will continue to generate an output signal forever. This output may die down eventually, but never disappear. The digital filters discussed in this paper are all of the FIR form.

The general equation for an FIR filter is as follows:

\[ y(n) = \sum_{k=0}^{N} w(k) \cdot u(n-k) \]  \hspace{1cm} (3.1)

where \( u(n-k) \) represents the kth previous input, \( w(k) \) is the kth filter coefficient, and \( y \) is the output of the filter. This operation can also be represented by the vector operation:

\[ y(n) = w^T u(n) \]  \hspace{1cm} (3.2)

For reference, an IIR filter would have the following form:

\[ y(n) = b(0) \cdot u(n) + \sum_{k=1}^{N} \left[ b(k) \cdot u(n-k) + a(k) \cdot y(n-k) \right] \]  \hspace{1cm} (3.3)

The \( w \)'s have been replaced with two sets of tap weights, \( b \) for the inputs and \( a \) for the outputs. Note how the output at time \( n \) depends on the output from as far back as time \( n-N \). Since the output at time \( n-N \) was dependent on the output at time \( n-N-N \), it is clear how the output of the system at an infinite time in the future can rely on the current conditions of the system.

Figure 3.1 shows a fourth order FIR digital filter. The \( z^{-1} \) represents a delay of one sampling period, such that a delay line of previous inputs (the past 4 inputs in this case) can be set up.
The w’s are called tap weights, numbers that those past inputs are multiplied by to get the current output. The tap weights determine the type of filter that is implemented. For example, if all of the tap weights are positive then it will perform low-pass filtering. Note that the filter in Figure 3.1 is FIR because the output depends only on the current and past inputs, not on previous outputs.
4. **ADAPTIVE FILTERS**

As mentioned earlier, digital filters have the advantage that they are continuously adjustable. The only factor limiting their adjustability is the resolution at which they operate, which is determined by the number of bits used to represent the signals they deal with. For modern systems, the resolution is usually high enough for them to be considered continuously adjustable. By exploiting this, it is possible to make a filter which changes its own parameters to adapt to different situations. These filters are known as adaptive filters. Adaptive digital filters were first developed more than 30 years ago, but technology has only advanced enough in the past five or ten years such that they are feasible for a wide range of applications. This is because the adaptive process takes more time than a usual filtering process: the system must both filter the signal, evaluate current conditions, and change if necessary. At minimum, this cuts in half the speed of the filter.

Now that computers are fast enough, adaptive filters are finding their way into many different areas such as telecommunications and speech processing. There are a number of different adaptive filter algorithms, such as the method of steepest descent and least-mean-square algorithm. The least-mean-square (or LMS) algorithm, which is used in this project to continually adjust the tap weights (the numbers by which the current and past inputs are summed together to create the output) of an FIR filter, are adjusted continuously. The tap weights are adjusted so as to minimize the square of the error between the output and a desired signal. The system takes some time to converge, or reach its optimal configuration, but this is usually a very short amount of time due to the high speeds involved. Since a desired signal is readily available (the signal before it reaches the loudspeaker), and only low frequencies are involved, distortion reduction for
4.1 ADAPTIVE FILTER LAYOUT

Figure 4.1 is an adaptive digital filter in which the multiplier coefficients, \( w_j \) where \( 0 \leq j \leq 4 \), are continually changing according to the adaptive control algorithm. The output of the filter is compared to a desired signal, and this comparison produces the error signal. The goal is to change the tap weights of the filter such that the output signal is the same as the desired signal, thus minimizing the error. The particular algorithm used for this project is the least mean-square (LMS) algorithm.

4.2 LMS ALGORITHM

The LMS algorithm is a based on feedback and belongs to a family of stochastic gradient based algorithms. This algorithm produces an error signal based on the difference between the desired signal and the signal actually output after filtering. The tap weights of the filter are then changed according to the error. The tap weights change as long as the signal is input to the filter. The algorithm is an iterative one, always trying to produce an optimal set of tap weights, or con-
Adaptive Distortion Correction

This optimal state exists for most situations, and is the state where the power in the error signal is minimized. Under the right circumstances, the algorithm converges sufficiently quickly and produces an acceptable output, at which point the filter coefficients have stabilized and the filter is close to optimal. The filter never converges completely due to minimal gradient noise which is always present. The LMS algorithm has the distinct advantage of being very easy to implement, while still providing good results. The LMS algorithm does not require correlation of the input or output signals to be calculated, nor does it require that matrix inversion be done. The algorithm simply requires a few multiply and accumulate operations (MACs); roughly as many as the length of the filter.

The following equations form the LMS algorithm:

Filter Output: \[ y(n) = w^H(n)u(n) \]  \hspace{1cm} (4.1)

Estimation Error: \[ e(n) = d(n) - y(n) \]  \hspace{1cm} (4.2)

Tap-weight adaptation: \[ w(n+1) = w(n) + \mu u(n)e^*(n) \]  \hspace{1cm} (4.3)

where \( d(n) \) is the desired signal, \( \mu \) is the step size parameter, \( u(n) \) is the filter input vector, consisting of the present input and past values stored in the delay line, and \( w^H(n) \) is the hermitian transpose of the tap-weight vector. The term \( \mu u(n)e^*(n) \) in equation (4.3) represents the tap weight correction. If this step size parameter is too small, the algorithm will converge too slowly; if it is too big, the system may become unstable. Therefore, its value must be selected carefully.
4.3 APPLICATIONS OF ADAPTIVE FILTERS

Adaptive filters using the LMS and other algorithms have many possible uses. The basic purpose of an adaptive filter is to approximate a transfer function. This may be the transfer function of another part of the system, the corresponding inverse transfer function, or an ideal transfer function which does not exist in the actual system. For adaptive noise cancellation, the noise and the signal plus noise are both input to the system. The filter operates on the noise itself, and the signal plus noise is used as the desired signal. With this setup, the ideal error should be the signal itself with no noise. This type of system falls into the third category, since the optimal filter coefficients do not mirror the actual transfer function of any other part of the system.

One important topic in adaptive filtering is that of orthogonality. The proper correlation of the three signals in an adaptive filter, u, d, and e, is essential to the operation of the filter. For an adaptive noise cancellation system, the input (noise) and desired signal (signal plus noise) should be as orthogonal (or uncorrelated) as possible. On the other hand, in an adaptive system which attempts to determine the transfer function of another part of the system, the error between the filter output and the desired signal will be orthogonal to both of them. The original idea for this project was using an adaptive filter to remove the “noise” added by the loudspeaker to the original system. Since this noise is highly correlated to the original signal, then the error of the system would be the orthogonal signal.
5. SYSTEM SETUP AND MEASUREMENTS

The LMS algorithm and digital signal processor are really the heart of a larger system. The entire system also consists of a loudspeaker, amplifier, accelerometer, and source. A diagram of the system is given below in Figure 5.1. The source, a CD player, is used both as the desired signal d(n) and as the input to the filter u(n). The signal is filtered, then amplified and sent to the loudspeaker. A piezoelectric accelerometer measures the loudspeaker’s motion, and creates a signal which is sent to the adaptive weighting control. The weighting control compares the loudspeaker’s output to the CD player’s signal and updates the filter coefficients appropriately. The analog-to-digital (A/D), digital-to-analog (D/A) converters, the adaptive filter, and the adaptive weighting control are all located inside the PC (see section 6 for details).

![Figure 5.1 – System block diagram](image)

5.1 LOUDSPEAKER AND AMPLIFIER

The loudspeaker used in this experiment is an MCM Audio select 15” ProSeries Woofer. This driver is robust enough to withstand the frequent testing required in this experiment. The driver also has a desirable response characteristic down to about 28Hz, as do most well made drivers. This provides a solid base for this project. While the ultimate goal of this project is to be able to correct distortion in any low frequency driver, it is important to first perform testing on a
system which introduces less distortion than most. In such a way the system can be fine tuned in an environment which introduces as few extraneous variables as possible. Similarly, the amplifier which is used in this project, a Carvin F-300, is of fairly high quality. A low quality amplifier can introduce distortion by itself. The amplifier must be able to amplify the signal input to it without clipping the signal. The amplifier used in this project was tested upon receipt and is capable of amplifying a signal without causing appreciable distortion.

5.2 ACCELEROMETER

The device which provides the information on the true action of the loudspeaker is the accelerometer. The accelerometer used in this project is a Kistler K-SHEAR 8614A500M1. This is a very lightweight device (0.7g) which is mounted on the cone of the speaker. As the name implies, the accelerometer measures acceleration. Like most accelerometers, this one exploits the piezoelectric properties of quartz crystals to measure acceleration. Figure 5.2 shows the different types of cuts which are made in a block of quartz to extract a piece useful for a specific type of measurement.

If the crystal is cut to exploit the excitation by one of these effects it is said to be in that mode, and this excitation causes the crystal to produce a charge - this is the piezoelectric effect. The two first modes are transverse and longitudinal compression modes. Until recently most accelerometers operated in one of these modes and were made to produce
measurable charges from this type of effect. The accelerometer used in this project is based on the third effect, the shear effect. Figures 5.3 and 5.4 show accelerometers of both designs.

The shear mode element is sensitive to shearing forces and unaffected by other orthogonal forces. The shear mode element has three advantages over the previous elements. First, the shear mode element produces twice as much charge as the previous two elements, which allows the use of smaller crystals, which in turn allows the entire device to be smaller and more lightweight. The last advantage of the shear mode based accelerometer is that it is more resilient to case strain due to the nature of the elements and the fashion in which they are placed in the case.

The accelerometer has a built in charge-to-voltage converter for low impedance voltage output. The converter must be powered by an external power supply coupler which powers the electronics in the device. A Kistler 5118A1 Power Supply / Coupler performs this duty in this system. The 5118A1 and the accelerometer itself are high quality pieces of equipment which should not introduce any extraneous noise or distortion.

5.3 Measurements

With the accelerometer in place, distortion measurements were taken using a Hewlett-
Packard (HP) model 35660A Dynamic Signal Analyzer. For comparison, a power spectral density (PSD) measurement was taken of the output the HP model 3325B signal generator. As seen in Figure 5.5, the signal generator’s output was very clean, showing only traces of second and fourth harmonics.

Harmonic distortion is the ratio of the sum of the power in the second through fifth harmonics to the power in the fundamental, in percent. The overall total harmonic distortion (THD) of the generator’s output was less than $1 \times 10^{-4}$ percent at 15 Hertz. The loudspeaker’s output at only 0.15 watts and 25Hz is quite distorted, with a THD of 1.3%. At lower frequencies and higher power levels, for instance 2.5 watts at 15 Hz, the THD increased to over 15%. The distortion peaks can clearly be seen in Figure 5.6. The distortion measured here is unacceptable, as the other components of a hi-fi system are expected to have THD below one percent.

![Figure 5.5 -- Power Spectrum of Signal Generator Output, 15Hz](image.png)
5.4 ADDITIONAL EQUIPMENT

Several other items required for this system were built in the lab using common parts. Specifically, two low-pass filters, a variable-gain amplifier, and a cable-conversion box were built.

The filters are necessary to condition the signals input and output from the D/A and A/D converters. Since a sampling rate of 2kHz is employed, incoming signals must be filtered to remove any frequencies above 1kHz. Also, the output signal from the D/A converter must be filtered to remove any imaging introduced by the sampling process.

Because this system is designed to operate below 100Hz, this frequency was used for the
low pass filters’ design. Each is a 2nd order Butterworth active analog low-pass filter with a cutoff frequency of 100Hz. A single BiFET opamp is used for each, along with several resistors and capacitors. Their frequency response was confirmed using the HP Spectrum analyzer, as shown in Figure 5.4.

Despite the Power Supply / Conditioner which the accelerometer uses, additional conditioning is required of the accelerometer’s output signal. First, the DC bias which is present and which varies with temperature must be removed. Additionally, the accelerometer signal is fairly weak at low power levels (several millivolts). Another important concern is that the overall forward gain of the system varies with changes in the gain of the main amplifier used. In the final system this gain would be constant, but for testing purposes it is kept very low and adjusted manually. This means that the accelerometer amplifier’s gain must be adjustable, so a non-inverting opamp amplifier was designed an constructed. A potentiometer is used to adjust the gain from one to 100. A first-order high pass filter, consisting of a capacitor and resistor, was set at 3Hz to remove
the DC bias as well.

The last piece of extra equipment required is a converter box to provide ease of input and output to and from the DSP board. The board has a 26-pin high density D-Sub connector, which provides analog inputs and outputs as well as clock and other signals. The converter constructed sends these signals to several BNC connectors so that they may be connected to lab equipment. These are all fairly robust, noise-free signals, so the converter’s connections are passive.

Figure 5.10 -- Accelerometer Amplifier F.R.,
Gain = 3

Figure 5.11 -- Accelerometer Amplifier F.R.,
Gain = 70
6 DSP SYSTEM

To implement the LMS algorithm, a digital processing system is necessary. Many digital platforms are available on which this processing could be done, such as a PC or a dedicated embedded system. For this project a DSP chip on a PC-compatible ISA card is used. First of all, DSP chips are ideally suited to signal processing applications such as this one. The hardware of a DSP chip is optimized for tasks such as filtering which involve repetitive multiply-and-accumulates (MACs). Its register design allows for processing of data arrays (here, signal vectors) with a minimum of overhead. Additionally, the use of pipelining allows simultaneous operations to occur, such as simultaneous shifting of registers and multiplication.

6.1 TEXAS INSTRUMENTS TMS320C50

Texas instruments was one of the first companies to market a dedicated DSP chip, the TMS320C10, in the early eighties. The TMS320C50 is one of the two more recent successors of that chip, the other being the ‘C40, and is the chip utilized in this system. The ‘C50 is a fixed point processor, whereas the ‘C40 is a floating point processor. The ‘C50 operates at a clock speed of 40MHz, which allows it to perform 20 million instructions per second (MIPS). The ‘C50 is a 16-bit processor, which means that data and memory values and addresses are 16 bits in length. However, several registers within the ‘C50, such as the product register (PREG1) and the accumulator (ACC), are 32 bits. This allows the multiplication of two 16-bit numbers, or the accumulation of several smaller products. These 32-bit numbers must scaled down to 16 bits before being further processed. However, only a small error is introduced by this scaling, whereas the pre-scaling of the values to be added/multiplied would result in a potentially much larger...
error.

The ‘C50’s 20MIPS rating allows it to process audio signals sampled at up to 48kHz or very-low-bit rate video, both in real-time. For this project, which only involves frequencies below 200Hz, and where any delay less than a millisecond is trivial, sampling rates as low as 1kHz are acceptable. This doesn’t mean that the use of such a powerful chip is overkill, however, as fairly high-order filters (and therefore a large number of MACs) are necessary to reduce distortion acceptably (see section 7.2). The estimated requirement on the chip for this system is around 5 MIPS, which leaves just enough room for overhead and the unexpected.

6.2 SPECTRUM PC/C5x

An ISA-card mounted DSP chip is used for this project for ease of programming and debugging. The computer interface allows programs to be written on the PC and uploaded quickly into the DSP board’s memory. Also, debugging software allows every internal register of the DSP to be displayed on screen for each step of the program. This makes programming the DSP much easier, with a minimum amount of time spent debugging using a trial-and-error method. Spectrum manufactures several ISA boards which allow DSP chips to be used in conjunction with PCs. The board used for this project, the PC/C5x, has one TMS320C50 DSP chip, 64k of SRAM, and various I/O chips on board. Installation was simple, with the appropriation of the proper DMA and IRQ being the only necessary configuration. Spectrum’s debugging program, View5x, was also installed on the PC to facilitate debugging. View5x displays the contents of data and program memory and of each register in the ‘C50. It also de-assembles the program memory, which allowed step-by-step observation of the program.
6.3 **SPECTRUM AM/D16SA DAUGHTER MODULE**

To connect the DSP board to the rest of this system, several analog-to-digital (A/D) and digital-to-analog (D/A) converters are required. Spectrum offers several daughter modules (DMs) which provide conversion for analog inputs and outputs. For this project, a Spectrum AM/D16SA Burr-Brown ADC/DAC daughter module is used. The module implements 16-bit conversion of two separate channels at up to 200kHz using the successive-approximation method. The signal-to noise ratio (SNR) of this module is an excellent 110dB, and any distortion added by the module is low enough to be considered negligible. Figure 6.1 shows a 30Hz sine wave which has been sampled at 1kHz, passed through the DSP system, and played back at 1kHz. The need for the 100Hz LPF can be seen in this figure. Since the DM is meant to be operated at output sampling rates up to 48kHz, its 18.3kHz anti-imaging filters were not useful for such a low sampling rate. These filters are disabled when the 100Hz LPF is in use.

6.4 **‘C50 ASSEMBLY LANGUAGE**

Programming the TMS320C50 requires that a chip-specific compiler be used. Included with the Spectrum PC/C5x board is a C50 assembler/linker program and a set of C libraries.
These files allow programs to be optimized and configured correctly for the C50. They include support for C50-specific commands and routines, as well as memory maps which compliment the PC/C5x board’s memory layout. For speed and increased control, the program for this system is written in assembly language. The LMS algorithm is straightforward enough that this presents no significant challenges.

As stated earlier, the C50 consists of several 16-bit registers and a couple of 32-bit registers. A second multiplication section compliments the main ALU, to allow for simultaneous multiply-and-accumulate operations. Included in the lineup of 16-bit registers are the auxiliary registers, AR0 through AR7. These are general-purpose registers which the user can temporarily store data in. A separate register, the auxiliary register pointer register, or ARP, is used when indirect addressing is utilized. In this usage, the ARP points to one of the auxiliary registers, which in turn points to a data or program memory address. The ARP can be conveniently invoked in assembly programs by the asterisk (‘*’), as shown in the following example:

```
lar ar3,#3000h
larp ar3
lacc *
```

The value 3000h is loaded into AR3, and then AR3 is made the auxiliary register pointer. The third line then represents the usage of the ARP: the value which resides in the memory address stored in AR3 (3000h), is loaded into the accumulator. The asterisk can be used in any command which requires a memory address. Until the ARP is changed, the asterisk will continue to point to the memory address which AR3 points to.
6.4.1 **CIRCULAR BUFFERING**

One important feature of the ‘C50 is its ability to use circular buffers. Circular buffers are actually linear portions of memory to which data is written or read from in a circular manner. When a pointer aimed at a location in a circular buffer reaches the end of the mapped memory space, and is incremented, it then points to the beginning of the memory space. This is useful for filtering operations which require a vector of previous input or output samples to be stored. The operation steps through the data, returning to the beginning when the end is reached. When used as an input buffer, the most recent input sample should overwrite the oldest sample. This is easily achieved by incrementing the pointer by one before storing the newest value: it then overwrites the oldest sample, the previous sample is directly behind it in the buffer, and the oldest sample in front of it. The C50 can be set up to control these buffers automatically, removing the need for pointer checks and subroutines within the filter code.

Control of the circular buffers, two of which can be active at one time, is set through five registers: the Circular Buffer Start Registers (CBSR1 and CBSR2); the Circular Buffer End Registers (CBER1 and CBER2); and the Circular Buffer Control Register (CBCR). The CBSRs and CBERs contain the memory addresses which locate the start and end of each circular buffer.

<table>
<thead>
<tr>
<th>Bit</th>
<th>Name</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>0-2</td>
<td>CAR1</td>
<td>Maps AR0-AR7 to circular buffer 1</td>
</tr>
<tr>
<td>3</td>
<td>CENB1</td>
<td>Circular buffer 1 enable=1, disable=0</td>
</tr>
<tr>
<td>4-6</td>
<td>CAR2</td>
<td>Maps AR0-AR7 to circular buffer 2</td>
</tr>
<tr>
<td>7</td>
<td>CENB2</td>
<td>Circular buffer 2 enable=1, disable=0</td>
</tr>
</tbody>
</table>

**Table 6.1 – Circular Buffer Control Register**

The CBCR dictates which buffers are active, and which auxiliary register controls which buffer.
The CBCR is an 8-bit register, which is as shown in Table 6.1. For instance, to set up two circular buffers from #1200h-#13FFh and #1400h-#15FFh, where AR6 controls buffer one and AR7 controls buffer two, the CBCR would be set to #0FEh and the start and end registers set appropriately.

### 6.4.2 AMELIA

AMELIA, which stands for Application ModulE Link Interface Adapter, is the hardware and software interface between the PC/C5x board and the AM/D16SA daughter module. AMELIA’s control is mapped to #00h-#0Fh in the C50’s I/O memory. This allows the selection of sampling rates, the setup of interrupts, and access to inputs and outputs through memory-mapped spaces. Table 6.2 shows AMELIA’s memory map:

Data is sent to and read from these locations using the “in” and “out” commands. For instance, to send the data input in channel one to the ARP, the following command would be used:

```
in *,06h
```

The AMELIA Control and User Control locations are used to set up clock and sampling rates, and the Interrupt Mask and Interrupt Status locations are used to setup and check interrupt functions.

<table>
<thead>
<tr>
<th>Address</th>
<th>Read Register</th>
<th>Write Register</th>
</tr>
</thead>
<tbody>
<tr>
<td>2h</td>
<td>Channel 0 Input Data</td>
<td>Channel 0 Output Data</td>
</tr>
<tr>
<td></td>
<td>Timer 1</td>
<td>Timer1</td>
</tr>
<tr>
<td>6h</td>
<td>Channel 1 Input Data</td>
<td>Channel 1 Output Data</td>
</tr>
<tr>
<td></td>
<td>Interrupt Mask</td>
<td>Interrupt Mask</td>
</tr>
<tr>
<td>7h</td>
<td>User Control</td>
<td>User Control</td>
</tr>
<tr>
<td>8h</td>
<td>--Not Used--</td>
<td>AMELIA Control</td>
</tr>
<tr>
<td>Ah</td>
<td>Configuration</td>
<td>Interrupt Mask</td>
</tr>
<tr>
<td>Bh</td>
<td>Interrupt Status</td>
<td>Configuration</td>
</tr>
<tr>
<td>Fh</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Table 6.2 -- AMELIA Register Mapping**
Adaptive Distortion Correction

Timer1 sets the final sampling rate, according to the following formula:

\[ \text{Timer1} = 65537 - \left( \frac{\text{FCLK}}{\text{Fs}} \right) \]

FCLK is the selected pre-scaled clock signal, and Fs is the desired sampling rate. For instance, for a 2kHz sampling rate using a 12.288Mhz clock pre-scaled by 8, the value #FD01h would be written to Timer1.
7. **SIMULATION**

Simulations are very useful for determining parameters and discovering problems with a system before actually assembling anything. This saves the hassle of rewiring or reprogramming an entire system each time a change is made. It can also prevent damage which would be caused by unstable systems: instability can be monitored in the simulation and avoided in the real system. For this system, software simulation was used to determine the proper filter parameters which would yield acceptable results: a significant distortion reduction where the computational power required was within the ‘C50’s range.

### 7.1 MATLAB SETUP

The software simulation of this system is performed using MATLAB, which stands for MATrix LABoratory. MATLAB is a very powerful numerical analysis program which deals exclusively with matrices (a scalar is stored as a 1x1 matrix). Thus, all matrix operations are built in functions. This is especially useful for signal analysis, where signals and filters are represented by matrices.

The MATLAB simulation for this system can be found in Appendix A. The first goal of the simulation is the implementation of the LMS algorithm. In MATLAB, operations are most efficient when they are reduced to matrix operations, thus reducing extraneous indexing steps inside the processor. For instance, an inner product of two column vectors x and y can be represented either as a matrix multiplication, $x'y$, or as a `for` loop multiplying each element in one vector by the respective element in the other:
While both methods represent the same process, the former is much faster in MATLAB since the two vectors are simply put in arrays and sent straight through the ALU, while the latter both keeps track of the index ‘n’ and writes ‘sum’ to memory for each run through the loop. Unfortunately, some algorithms, such as the LMS, cannot be vectorized in such a way. The changing parameters due to the adaptive process in an LMS filter must be changed after each filtering operation, and therefore a for loop is required. The computation time for ‘for’ loops in MATLAB appears to rise as the square of the length of the loop, which severely limits the length of simulations using this method. Fortunately, a signal length of 4096 samples produces useable information in about five minutes on a 486DX4/120 computer.

The LMS algorithm is implemented inside the ‘for’ loop with the input and filter coefficients represented as vectors and the error and output as scalars. Required next for the simulation is a desired signal and a distortion model. A source consisting of multiple equal-amplitude sine waves is used as the desired signal, representing the audio signal reaching the loudspeaker. The frequencies present in the source are variable and stored as a vector \( f \), and the input signal is sampled at a variable rate \( f_s \). To model the distortion of a loudspeaker, frequencies of integer multiples of the input frequencies are added to the signal. Additionally, a small amount of additive white Gaussian noise (AWGN) is added to the signal using MATLAB’s \texttt{randn} command to represent noise picked up by the accelerometer circuit.

Note that any delay present from the loudspeaker itself and the accelerometer is taken into
account by the variable k. For this simulation, k is set to 20 samples, or 10ms. This is a slight overestimate of a typical delay present at low frequencies for a large loudspeaker in a sealed enclosure.

Figures 7.1 and 7.2 show time domain plots of the system output versus the system input. The input is plotted in red, and the output in blue. Note that, initially, the filter has not converged and the output does not match the input. However, Figure 7.2 shows the system after convergence, and the output matches the input almost exactly.

### 7.2 ANALYSIS

To analyze the effectiveness of the system, the total harmonic distortion (THD) must be measured. First, the spectrum of each signal (desired, unfiltered output, and filtered output) is found using a fast Fourier transform (FFT). The spectra are then normalized such that the fundamentals represented in each are of the same magnitude. Then the THD is calculated as the ratio of the extraneous power present in distortion frequencies to the input signal’s power. For $\mu=0.0001$ and $N=512$ at a sampling rate of 2kHz, the THD of the filtered output signal is less than
5% for unfiltered THD of up to 50%. This can be seen in Figures 7.3 and 7.4, where the distortion peaks in blue are shown to be absent from the system output in green. The noise which was added in the simulation was attenuated by the filter as well.

This filter length and sampling rate represent a total number of computations per second, or MIPS, of around 3 MIPS, which is within the TMS320C50’s capabilities. For a constant frequency input, the filter converges after about 100ms as shown in Figure 7.2. This does not represent any delay in the signal, but rather the amount of time taken for distortion to be eliminated. Also note that, once the filter coefficients converge, they do not remain constant – a small amount of oscillation occurs around the desired state. This adds a small amount of noise to the system, but, as shown in Figure 7.2, not enough to be problematic.

Another important aspect of the system, quantization and its effects, is also simulated. The inputs and outputs of the DSP board are quantized to 16 bits, signed. Therefore all signals present in the simulation are limited to a relative value of $2^{15}$, and rounded to the nearest integer. The ‘C50 has a 32-bit main accumulator in its ALU, which means that for a filter length of 512, and 16-bit data, this restricts the filter coefficients to 8 bits of resolution, or a relative $\pm 2^7$ (the


8. RESULTS

Various factors involved led to results which did not meet the goals set out at the beginning of the project. The theoretical background for the system was slightly flawed, so the final architecture of the system did not match that of the simulation. This, combined with overestimates of the C50’s processing power, led to systems which were either ineffective or unstable.

8.1 CHANGES IN THE SYSTEM

Several changes were made to the overall system configuration during the course of this project. Initially, a standard adaptive filter setup was utilized, with one minor change. The traditional setup of an adaptive noise cancellation system involves the filter operating on the signal after the noise has already been added:

\begin{align}
    x(n) &= \text{original signal} \\
    u(n) &= \text{signal plus noise} \\
    d(n) &= \text{music from CD} \\
    y(n) &= \text{filter output} \\
    y(n) &= w^H \cdot u \quad \text{(8.1)} \\
    w(n+1) &= w(n) + \mu \cdot u(n) \cdot (d(n) - y(n))^* \quad \text{(8.2)}
\end{align}

However, for this system, the source of “noise” is the loudspeaker itself, and the filter must precede the loudspeaker for the effects to be audible. So, initially, the filter and the loudspeaker were swapped such that the signal was pre-filtered before reaching the loudspeaker.

Unfortunately, this design was flawed, and the filter was not able to adequately converge for any signal. Based upon suggestions in [2], the system was changed such that two adaptive fil-
Adaptive Distortion Correction

ters were implemented. The first filter acts upon the output of the loudspeaker, and attempts to find the transfer function which approximates the inverse of the loudspeaker. This is accomplished by using the input to the loudspeaker as the desired signal, so in effect what passes through the loudspeaker and then the filter is as close to the original signal as possible. Then, the first filter’s coefficients are used in the second filter, which pre-filters the signal before it reaches the loudspeaker. This can be seen in Figure 8-1.

For the first filter:

\[
\begin{align*}
\mathbf{u}_1(n) &= \text{output of loudspeaker} \\
\mathbf{d}_1(n) &= \text{music from CD} \\
\mathbf{y}_1(n) &= \mathbf{w}_1^H \cdot \mathbf{u}_1 \\
\mathbf{w}_1(n+1) &= \mathbf{w}_1(n) + \mu \cdot \mathbf{u}_1(n) \cdot (\mathbf{d}_1(n) - \mathbf{y}_1(n))^* 
\end{align*}
\]

(8.3)  (8.4)

For the second:

\[
\begin{align*}
\mathbf{u}_2(n) &= \text{music from CD} \\
\mathbf{y}_2(n) &= \text{output to loudspeaker} \\
\mathbf{y}_2(n) &= \mathbf{w}_2^H \cdot \mathbf{u}_2 \\
\mathbf{w}_2(n+1) &= \mathbf{w}_1(n+1)
\end{align*}
\]

(8.5)  (8.6)

When implemented in this fashion, the first filter is attempting to find the inverse transfer function, \( H^{-1} \), of the loudspeaker. This transfer function is then sent to the second filter so that the music can be pre-filtered with \( H^{-1} \) before the loudspeaker operates on it with \( H \) — the net product should be only a delay (since \( H^{-1} \) must still be causal).
8.2 RESULTS

Several unfortunate side effects appeared due to the changes in configuration. First, our estimate of the MIPS required by the algorithm was off by a factor of two, so the sampling rate was reduced to 1kHz when 512th order filters were used. Second, the second adaptive filter implemented which operates directly on the output of the loudspeaker doubled the computational intensity of the system, necessitating either another reduction in sampling rate or a halving of the filter length. Tests concluded that the latter option resulted in a more stable system, so the final configuration was 256th order filters at a sampling rate of 1kHz.

These changes unfortunately reduced the effectiveness of the system. The system has difficulty following signals below 20Hz, and has only a slight beneficial effect on those above 20Hz.
Figures 8.2 through 8.7 show time and frequency domain plots of filtered versus unfiltered outputs. The first set are at 15Hz, and the second are at 30Hz. In the time domain plots, the top trace is the input to the system, and the bottom is the output of the loudspeaker. The system reacted poorly to the 15Hz signal, as shown by the very high noise level in the frequency plot. A slight increase of noise floor is also present in the 30Hz frequency plot; however, this level is consistent with the MATLAB simulation and arises from the filter’s continuous adjustment.

The distortion at 30Hz was reduced only slightly, from 0.24% to 0.15%. At higher power levels, the system has difficulty with the increased distortion and is less effective. This behavior is supported by the MATLAB simulation: when the filter order was too low, the system would be less effective in general, and have difficulties with high distortion levels. It would appear that a more powerful system with a higher sampling rate and/or filter order might be more successful.

The convergence time of the system is adequate such that no audible delay exists in the low frequencies of music played through it. As noted above, convergence time doesn’t correspond to an actual delay; however, if the system converges slowly enough, transients could be attenuated and the result would be an apparent
Adaptive Distortion Correction

Figure 8.2 -- Input and Output of System, 15Hz

Figure 8.3 -- PSD of System Output, Filters Inactive, 15Hz

Figure 8.4 -- PSD of System Output, Filters Active, 15Hz
**Adaptive Distortion Correction**

Figure 8.5 -- Input and Output of System, 30Hz

Figure 8.6 -- PSD of System Output, Filters Inactive, 30Hz

Figure 8.7 -- PSD of System Output, Filters Active, 30Hz
9. **CONCLUSION**

Although the ultimate goal of this project, to implement an actual system by which distortion was reduced significantly, was not successful, the project itself was still a success. Knowledge into the design and implementation of adaptive filters was gained, as well as specific knowledge of ‘C50 programming and systems integration.

9.1 **WHAT WORKED AND WHAT DIDN’T**

While the power of the TMS320C50 was not sufficient to meet the goals presented in this project, each other part of the system performed adequately. With the addition of the amplifier/HPF, the accelerometer system provided excellent data on the loudspeaker’s movements. A stronger HPF could be used to remove more of the DC offset present, or a different Power Supply/Conditioner model could be selected which has this feature built-in. Truthfully, the accelerometer used for this project was too good: Kistler only manufactures high-precision, calibrated accelerometers. The model used cost over $400, which is too high of a cost for this accelerometer to be used in a commercial system. While the small size of the accelerometer is crucial to avoid altering the driver’s response, some of the precision and calibration could be sacrificed. This system does not depend on a specific acceleration producing a specific voltage; only a linear relationship between the two is required.

The loudspeaker itself performed as expected, displaying high amounts of distortion despite being an expensive, high-technology piece of equipment. It did, however, withstand the abuse that many unstable systems threw at it without failing: the loudspeaker was driven many times by subsonic noise at power levels over ten watts, and did not sustain any damage. A less
rugged driver would most likely have been destroyed, and the necessary replacement would most likely cancel any monetary benefits from using a cheaper driver.

For the most part, programming the C50 was straightforward. The TI series of User’s Guides was essential, and most of the code was written in the span of one week. The AMELIA system performed very well, as sampling rates, interrupts, and I/O signals were all set up with only one day of programming. At times the system would not act consistently, however, and most runs (but not all: once the program ran all night) would stop for an unknown reason after a minute or two. This was most likely due to stray signals from the PC sent over the bus, and would not be a problem in an embedded system.

The MATLAB simulation was extremely useful in some ways, namely that by the time actual programming started, large chunks of it had already been written. Even though MATLAB and assembly are vastly different, the order in which operations occur in the filter is consistent between the two programs. Unfortunately, the changes which were necessary negated much of the simulation’s usefulness, as the filter order and sampling rate changed, and a new μ was used. If time had allowed, the simulation could have been altered to reflect these changes. It was unfortunate that these changes were not foreseen until so late in the project.

9.2 CONCLUSIONS

It’s not known whether or not a faster processor and therefore longer filter length and higher sampling rate would make the system perform as successfully as the initial simulation. However, it is apparent that loudspeaker distortion reduction is a perfect application for adaptive digital filters. The demand for improved sound reproduction is immense, as shown by the tremen-
dous abundance of and technological level of audiophile equipment. However, loudspeakers remain the weak link in terms of their ability to reproduce sound accurately. While several analog systems have successfully eliminated distortion from specific loudspeakers, a large market exists for a device which would eliminate low-frequency distortion from any loudspeaker system.
REFERENCES


2. Fontaine, Fred. Personal Communication, March 1997


MATLAB Simulation Program

fs=2000; %Sampling Rate
del =2^15;
t=(1:(1024*8))/fs; %time vector for 8192 samples
f=[20 23 47 73 31]'; %input frequencies
x=sum(sin(2*pi*(f*t))); %input signal
Z=x+1*(0.5*sum(sin(2*pi*2*(f*t)))+0.3*sum(sin(2*pi*3*(f*t)))); %distortion model
Z=Z+randn(1,length(Z))*0.05; %addition of noise
maxx=max(abs(x));
x=x/max(abs(Z))*del; %Put x into +or- 2^15
maxz=max(abs(Z))*del;
x=round(x); %quantize x
Z=round(Z); %quantize Z
N=512; %Length of FIR filter
w=[zeros(1,N-1) x]; %Setup initial conditions
Z=[zeros(1,N-1) z];
w=w(:); %columnize vectors
Z=Z(:);
x=x(:);
lenx=length(x);
mu=0.0001; %Stepsize parameter µ
k=20; %delay of ‘k’ samples
for n=N:lenx-k
    u=Z((n+k-N):(n+k-1)); %Note delay of k samples
    y=w'*u/2^16; %filtering — note shift down since
    %w’s shifted up
    y=(round(y)); %quantize y
    if abs(y) > 2^15
        disp('Output Overflow') %check for overflow
    end
    y=y-(2^15<abs(y)).*(y-2^15); %truncate y at 15 bits
    e=x(n)-y; %calculate error
    e=round(e); %quantize e
    w=w/64+mu*u*e/2^15; %calculate new coefficients
%w’s must be 7 bit s.t. 512*(2^15 * 2^7) =2^31
%since C50 has 32-bit accumulator
%also, limit w’s to +or- 1
w=round(w*64); %quantize w
    if abs(w) > 2^6
        disp('Coefficient Overflow') %check for overflow
    end
    w=w-(2^6<abs(w)).*(w-2^6); %truncate w to +or- 2^6
output(n-N+1)=y; %write to output vector
error(n-N+1)=e; %keep track of error
end
x=x/2^15; %bring down from 16 bit

Appendix A: MATLAB Simulation Program
Z = Z/2^15;
error = error/2^15;
output = output/2^15;
t2 = 1:length(output);
figure(1)
plot(t,x(N:length(x)),t2/fs,output); %plot input and output
hold;
plot(t,Z(N:length(x)),'r'); %overlay distorted signal
hold off
figure(2)
hold off
plot(t2/fs,error) %plot error
figure(3)
hold off
lenout = length(output);

fout = 20*log10(abs(fft(output.*hamming(lenout)',4096*8)));
% spectrum of output
fz = 20*log10(abs(fft(Z(N:lenout+N-1).*hamming(lenout),4096*8)));
% spectrum of distorted signal
fx = 20*log10(abs(fft(x(N:lenout+N-1).*hamming(lenout),4096*8)));
% spectrum of input

len = length(fx);
Q = min(fz(f*len/fs) - fx(f*len/fs));
fz = fz - Q; % normalize spectra
Q2 = min(fout(f*len/fs) - fx(f*len/fs));
fout = fout - Q2; % normalize spectra
omega = (1:len)/len*fs;
omega2 = (1:len)/len*fs;
semilogx(omega2, fout,'m'); %plot output
hold on
semilogx(omega, fz,'r'); % overlay distortion
semilogx(omega, fx,'y'); % overlay input
axis([10 300 -20 max([max(fz) max(fout)])]);
hold off
Qout = fout > (fx' + 10); % find distortion in output
Qz = fz > (fx + 10); % find distortion in distorted signal
D1 = 10.^(Qout.*fout/20);
D2 = 10.^(Qz.*fx/20);
D1P = (1/length(D1))*sum(D1.^2); % Calculate distortion power
D2P = (1/length(D2))*sum(D2.^2);
sig_pow = (1/length(fx))*sum(10.^(2*fx/20)); % calculate signal power
THD1 = D2P/sig_pow % calculate old THD
THD2 = D1P/sig_pow % calculate new THD

Adaptive Distortion Correction

Appendix A: MATLAB Simulation Program
Appendix B: Assembly Program

```
.sect "init"
b START ; Reset vector
rete
rete
b AISR ; Interrupt service routine
rete
rete
rete
rete
.text
.mmregs

START

lacc #00100h ; load 100h into the accumulator
samm pdwsr ; Set program and data memory wait
 ; state register to 100h
lacc #00010h ; load 10h into the accumulator
samm cwsr ; Set wait state control register to 10h
lacc #0AAAAh ; load AAAAAh into the accumulator
samm iowsr ; Set i/o space wait states register to
 ; AAAAAh
ldp #0h ; Set data page pointer to 0
clrc intm ; Set interrupt mode bit to be one
lacc #0 ; Enable int2
or #2 ; logical OR accumulator with #2
samm imr ; Processor initialised....
lar ar2,#3000h ; Define auxiliary register usage
larp ar2 ;

; Amelia resides at address 0 in i/o space
lacc #0A4C0h ;
sacl * ;
out *,08h ; Set up User Control reg — 1kHz

lacc #00F2h ;
sacl * ;
out *,0Ah ; Set up Amelia Control register

lacc #0FD01h ;
sacl * ;
out *,05h ; SET TIMER 1 — Now set for 1kHz

lacc #8DFFh ;
sacl * ;
out *,0Fh ; Set up Configuration register
```
lacc  #01h  ;
sacl  *      ;
out  *,0Bh   ; Set up interrupt mask register

; SET UP CIRCULAR BUFFERS AND POINTERS
***buffer1 = 1600h -> 16FFh***
lacc  #01600h ; load accumulator with start address of
               ; first circular buffer.
sacl  CBSR1  ; send low part of accumulator to the
               ; first circular buffer start register.
lacc  #016FFh ; load accumulator with end address of
               ; first circular buffer
sac   CBER1  ; send accumulator to the first circular
               ; buffer end register.

***buffer2 = 1400h -> 14ffh***
lacc  #01400h ; load accumulator with start address of
               ; second circular buffer.
sacl  CBSR2  ; send low part of accumulator to the
               ; second circular buffer start register.
lacc  #014FFh ; load accumulator with end address of
               ; second circular buffer
sac   CBER2  ; send accumulator to the second circular
               ; buffer end register.

lar   AR6,#01600h ; load AR6 register with the begining
                   ; address of the first circular buffer:
                   ; u(n) will be stored in this buffer.
lar   AR7,#01400h  ; load AR6 register with the begining
                   ; address of the first circular buffer:
                   ; d(n) will be stored in this buffer.
lacc  #0FEh  ; 0FEh = 0000000011111110b
               ; buf1=AR6: enable AR6
               ; buf2=AR7: enable AR7
sacl   CBCR  ; send accumulator to cicular buffer
            ; cotrol register.
.eval 0,x  ; initiate counter for “zero out” loop
lar  ar5,#1200h ; load AR5 with first address of filter
                 ; coefficient.
LOOP
   lacc  #00000000h ; zero out accumulator
   larpa ar5
   sacl  *+  ; send the low part of the accumulator
            ; to the address in AR5, then auto
            ; increment.
   .eval x+1,x  ; increment counter.
   .break x = 256 ; if counter = 256; break out of loop.
.endloop ; END “ZERO OUT” LOOP
lar   ar5,#01200h ; set address in AR5 to 1200h
larpa ar5  ; set aux. reg. pointer to value in AR5
Adaptive Distortion Correction

Appendix B: Assembly Program

```assembly
lacc #0000001h ; load accumulator with one
sacl * ; send the low half of the accumulator
          ; to the address in AR5.

LOOP
  nop
  nop
  b LOOP ; Idle Loop - wait for interrupts

*************AMELIA INTERRUPT SERVICE ROUTINE*************

AISR
  lar ar5,#01200h ; load AR5 with first address of the
                  ; second circular buffer
  lar ar3,#3001h ; load AR5 with an available memory
                  ; location
  larp ar3 ; set aux reg pointer to value in AR3
  in *,0Bh ; Read interrupt status to clear int.
  in *,06h ; read d(n) from pin 10 on DSP board
            ; d(n) = filtered signal, Put d(n) in
            ; memory location 3001h
  lacl * ; load accum with value in 3001h and
           ; clear the high part of the accum.
  larp ar7 ; set aux reg pointer to first address
            ; in buffer2
  sacl * ; Put d(n) in proper place in buffer2
  lar ar3,#3002h ; load AR3 with next available memory
                 ; location.
  larp ar3 ; set aux reg ptr to 3002h
  in *,02h ; Put u(n) in 3002h
  lacl *
  larp ar6
  sacl *

.eval 0,x
  spm #0 ; set PM shift to no shift
  zpr ; reset PREG
  lar ar5,#1200h
  lacc #00000000h ; Clear the accumulator
  .LOOP
    larp ar5 ; Get w’s
    lt *+
    larp ar7 ; get u’s (CD)
    mpya *+ ; multiply and accumulate
    .eval x+1,x
    .break x = 256
  .ENDLOOP
  apac ; ACC final product
  larp ar3
  lar ar3,#3006h
  sach * ; out *,06h
```
lacc #00000000h ; Clear the accumulator
.eval 0,x
spm #0 ; set PM shift to no shift
zpr ; reset PREG
lar ar5,#1200h

.DOOP
  larp ar5 ; Get w’s
  lt *+
  larp ar6 ; get u’s (accel)
  mpya *+

  .eval x+1,x
  .break x = 256

.ENDLOOP
apac ; ACC final product
lar ar3,#3003h
larp ar3
sach *

lar ar3
lar ar3,#3006h
lacc #00000000h
lacc *
lar ar3,#3003h
sub * ; compare y2(n-1) and y1(n)
lar ar3,#3004h
bsar 1 ; error could be 17 bits
sacl *
out *,02h
ssxm ; set for arithmetic shift
lar ar5,#01200h
.eval 0,x

.DOOP
  larp ar6
  lt *+
  out *,02h
lar ar3,#3004h
lar ar3
mpy *
apac
call NEGCHK
; rptk #0dh
; sfr
lar ar3,#3005h
lar ar3
; sacl *
; out *,02H
lacc #00000000h
lacc *
Adaptive Distortion Correction

Appendix B: Assembly Program

```assembly
larp ar5
add *
sacl *+
.eval x+1,x
.break x = 256
.endloop
lacc #00000000h
lar ar3,#3006h
larp ar3
out *,06h
larp ar7
lacc *+
larp ar6
lacc *+
rete

NEGCHK
BLZ     NEGA
lar     ar3,#3005h
larp    ar3
rptk    #0eh
sfr
sacl    *
ret

NEGA
neg
rptk    #0eh
sfr
lar     ar3,#3005h
larp    ar3
sacl    *
lacl    *
neg
sacl    *
ret
```
Adaptive Distortion Correction

Appendix C: Equipment List

Sony D-242CK Compact Disc Player

Carvin F300 300 Watt Amplifier

100Hz Active Lowpass Filter

MCM Audio Select ProSeries 15” Subwoofer

Kistler 8614A500M1 Accelerometer and Kistler 5118A1 Power Supply / Coupler
Appendix C: Equipment List

Spectrum PC/C5x ISA Board with Texas Instruments TMS320C50 DSP

Spectrum AM/D16SA ADC/DAC Daughter Module

Hewlett Packard 3325B Signal Generator

Hewlett Packard 35660A Dynamic Signal Analyzer

Lambda Dual Regulated Power Supply
actual coefficients lie between -1 and 1).