Signals and Systems Project

Andrew Buck, Nick Chao, and Micah Thornton

***Abstract*—In this report we discuss the methodology behind implementing a particular system using matlab code. The aim of this project is to utilize built in libraries and functions from matlab for the purpose of creating a software system capable of emulating a message being transmitted and received electronically. There are many built in capabilities and software packages that are capable of creating such an emulator, we are using matlab because of its helpful built in features for designing filters and other critical components.**

CONTENTS

I. INTRODUCTION

The nature of this paper is laid out by the four internal sections, each describes the process undergone for implementing the particular component of the system. Due to the tiered nature of this project, each section of the report is broken down further into four subsections, indicating the difficulty level of each contribution made to the overall component.

The system that we are implementing begins at a very simple level. The stages of the project are re-iterated here:

The project itself is to design and implement a system capable of emulating the parameters specified on the bottom row of 1.

Fig. 1. Project Component Specifications

message	**transmitter**	**channel**	**receiver**
sinusoid	simple AM Mixer	Noiseless & lossless	Simple Mixer and Low Pass Filter
sqaure & triangle wave & pulse trains		Attenuation	Simple mixer out of phase or frequency
Voice (recorded)	AM-QAM	Attenuation, Dispersion, Nonlinearity	AM-QAM
Pseudo-random pulse trains		Channel fading with random fades	Noise injected before received

II. THE MESSAGE

The nature of representing a message on a discrete device (computer) already poses some challenges. For instance, a continuous sine wave as was suggested as the message in the first level. A sine wave is a continuous function of it's parameter (in the case of a message the parameter is the time or t), and as such can not be perfectly represented on a machine, as there are an infinite(uncountable) number of values in the domain, mapping to an infinite number of values in the range. This problem is initially solved using a sampling frequency that determines the rate of sampling for a particular continuous function, and storing off all of the function values for each time t_i for which we are sampling. Hence we are able to store an approximation of a continuous function to enough resolution by fine tuning the sampling frequency such that it is virtually indistinguishable from it's continuous counterpart. It is important to note that the sampling frequency of the original message constricts the range of frequencies that can be adapted for the carrier frequency of the modulated carrier waveform to be less than that of the sampling frequency of the original message.

A. *level 1*

The message for level one of the project is a simple sine wave. To address some of the problems discussed above, we set up a sampling frequency. The below listing is a display of the methodology used to set up the message for level one of the project.

```
F0 = 2000; % Frequency of the sine wave or
    the message.
fSampling = 30000; % Sampling rate for
    message in hertz.
tSampling = 1/fSampling; % The sampling
    period for the message seconds.
t = -0.005:tSampling:0.005; % The range of
    specific times for which the function is
    evaluated.

yt = sin(2*pi*F0*t); % The message is a sine
    wave at frequency F0, for the values of t
    above.
```

B. level 2

C. level 3

D. level 4

III. Transmission

A. level 1

B. level 2

C. level 3

D. level 4

IV. Channel

A. level 1

B. level 2

C. level 3

D. level 4

V. Receiver

A. level 1

B. level 2

C. level 3

D. level 4

VI. Conclusion