Department of Electrical & Computer Engineering
McGill University

ECSE-490 DSP Laboratory
Experiment 2

Decoding a Signal in Noise

2.1 Purpose

Imagine that you have obtained through some, possibly suspect, means a data signal containing some information of interest. Unfortunately, the signal has been corrupted and is unusable in its current form. Your job is to extract the signal and decode its message. As a secondary objective, you are also trying to minimize the computational complexity of your solution – perhaps you would like to decode long messages in real-time. In the process you will learn to interactively design and analyze Finite Impulse Response (FIR) and Infinite Impulse Response (IIR) digital filters using the MATLAB Filter design and analysis tools (SPTool and FDATool). You should also gain a better understanding of *when* and *why* to use different filters and how some of the tradeoffs affect your choices.

2.2 Background

2.2.1 Clean Data

The transmitter output signal is $s[n]$ (see Fig. 2.1 and 2.2). This signal consists of 3 parts:

(i) Leading silence containing only a single impulse. The impulse is 10 times larger than the data.

(ii) The data, coded using biphase encoding. This is described in more detail below.

(iii) A trailing section of zeroes.

The data part is an ASCII string encoded using its binary representation with the most significant bit first. The table of the hexadecimal values for the ASCII characters is shown in Table 2.1. For example, the letter "`M`" would be encoded as `01001101`, which is the binary value for hexadecimal 4D with the order of the bits most-significant bit first.

Biphase encoding is an example of a 1b2b code, i.e. it maps one bit of input to two binary symbols.[1] Biphase encoding always causes a transition of the signal level at the beginning of each

[1] Biphase coding is the signalling format for S/PDIF which is used to interconnect digital audio devices.

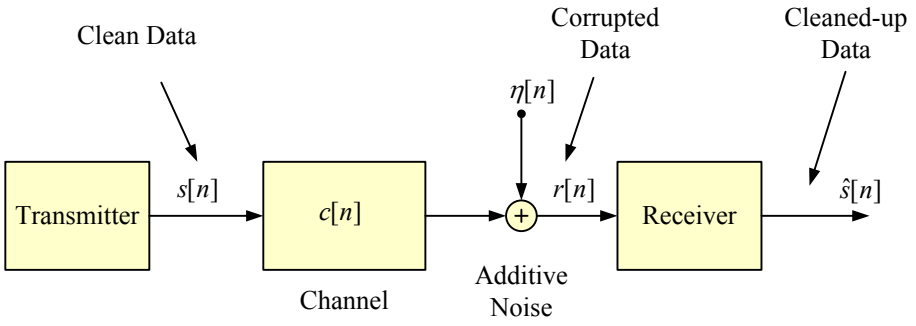

Fig. 2.1 System model

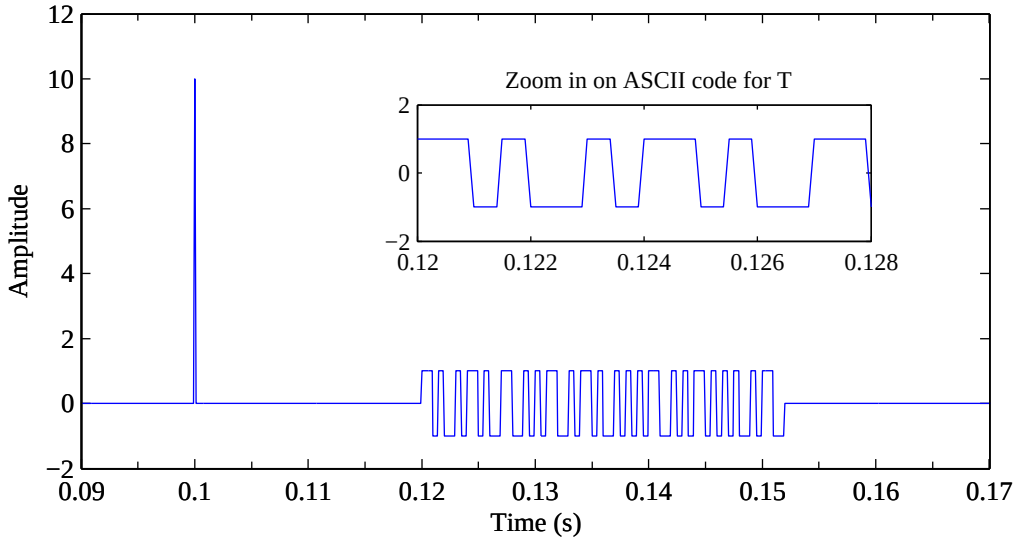

Fig. 2.2 Clean data signal

Table 2.1 ASCII Hexadecimal Codes

00 *nul*	01 *soh*	02 *stx*	03 *etx*	04 *eot*	05 *enq*	06 *ack*	07 *bel*
08 *bs*	09 *ht*	0a *nl*	0b *vt*	0c *np*	0d *cr*	0e *so*	0f *si*
10 *dle*	11 *dc1*	12 *dc2*	13 *dc3*	14 *dc4*	15 *nak*	16 *syn*	17 *etb*
18 *can*	19 *em*	1a *sub*	1b *esc*	1c *fs*	1d *gs*	1e *rs*	1f *us*
20 *sp*	21 !	22 "	23 #	24 $	25 %	26 &	27 ’
28 |	29)	2a *	2b +	2c ,	2d -	2e .	2f /
30 0	31 1	32 2	33 3	34 4	35 5	36 6	37 7
38 8	39 9	3a :	3b ;	3c <	3d =	3e >	3f ?
40 @	41 A	42 B	43 C	44 D	45 E	46 F	47 G
48 H	49 I	4a J	4b K	4c L	4d M	4e N	4f O
50 P	51 Q	52 R	53 S	54 T	55 U	56 V	57 W
58 X	59 Y	5a Z	5b [	5c \	5d]	5e ^	5f _
60 ‘	61 a	62 b	63 c	64 d	65 e	66 f	67 g
68 h	69 i	6a j	6b k	6c l	6d m	6e n	6f o
70 p	71 q	72 r	73 s	74 t	75 u	76 v	77 w
78 x	79 y	7a z	7b {	7c &	7d }	7e ~	7f *del*

bit. In addition, if the bit is a “1”, there is also a signal level reversal at the middle of the bit. Thus, in Fig. 2.2, the first 8 bits are 01010100 which is the code for letter “T” (Hex 54). The entire message contained in Fig. 2.2 is the word “Test”.

A few more things that you need to know:

- The sampling rate of the signal: 10^4 samples/second.
- The bit rate of the data: 10^3 bits/second, hence there are 10 samples per bit.
- The first 1,200 samples of the signal are all zero except for sample 1,000 (sample zero is the first sample) which has a value of 10, i.e., there are exactly 1,000 zeroes at the beginning of the file, followed by the impulse, and then by 199 more zeroes.

2.2.2 The channel

The transmitted signal, $s[n]$, passes through a channel that introduces two effects:

(i) It filters $s[n]$ by the channel filter $c[n]$.

(ii) The noise $\eta[n]$ is added to the output of the channel filter.

The channel output (and received siganl) is $r[n] = c[n] * s[n] + \eta[n]$. This output signal is shown in Fig. 2.3.

2.2.3 The receiver

Your job is to build a receiver that generates $\hat{s}[n]$, an approximation to $s[n]$, from the received signal $r[n]$.

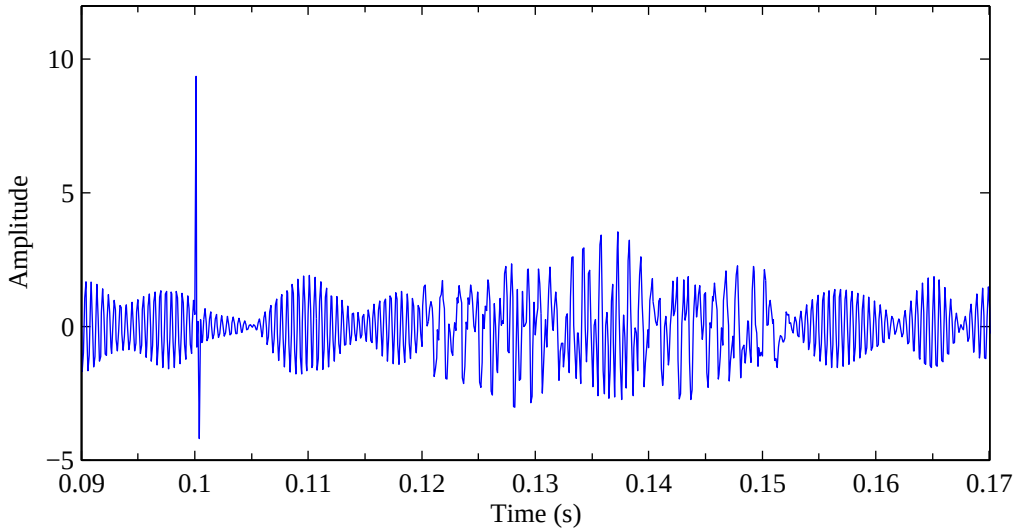

Fig. 2.3 Noisy data signal

2.3 Preparation:

- Biphase coding of equiprobable data can be shown to have the following power spectrum,

$$G_s(f) = \frac{2}{T}|S_h(f)|^2\big(1 - \cos(3\pi f T)\big),$$

  where $S_h(f)$ is the Fourier transform of a rectangular pulse of width $T/2$ and T is the time interval for a symbol. Plot this function for the value of T that is used in this experiment. Identify the nulls of the spectrum.

- Plan a method of attack to determine the structure and parameters of a system to extract the signal (Draw a block diagram and specify the function of each block). Be specific about how you will treat the effects of noise and how you will treat the channel effects. It is important to note that the channel filter may not be minimum-phase and as such may not be directly invertible (see Section 5.5 of Proakis and Manolakis [2]. Explain how you would handle this case.

- You will need to design a noise suppression filter. In your preparation, detail how you will determine the filter specifications (passband ripple, stopband attenuation, etc.).

- The signal is contaminated by noise. Discuss how to determine if that noise is Gaussian or not. What measurements are needed to determine this property? Discuss how to determine if the noise is white. What measurements are needed to determine this property?

- Read Section 10.3 of Proakis and Manolakis [2], or Sections 5.7, 7.2.1 and 7.2.2 of Oppenheim and Schafer [1] and familiarize yourself with linear phase filter properties.

- Read Section 8.3 of Proakis and Manolakis [2] or Section 7.1 of Oppenheim and Schafer [1] to familiarize yourself with the tradeoffs between Butterworth, Chebyshev and elliptic designs for IIR filter.
- The Simulink system uses IEEE 754 standard double float representation for signals and values (unless otherwise directed). As such it uses finite precision arithmetic. For each arithmetic operation, there may be an error of half of a least significant bit. Find the equivalent signal-to-computation noise ratio in dB. There is a discussion of single precision IEEE 754 arithmetic in Section 9.4 of Proakis and Manolakis [2]; double float has 64 bits of which 53 bits (1 hidden) are for the mantissa, 11 bits for the exponent and 1 bit for the sign. A good review of floating point arithmetic is in the article by Goldberg [3].

2.4 Experiment

2.4.1 Input signal

Data files, one for each lab group, are available on the DSP Lab web page as `DSPLab`*xx*`.wav`, where *xx* is your group number. The file contains floating point data. There is also a data signal `CleanSig.wav` to use as a reference. This signal contains an example of clean data. Note that your signal does *not* contain the same data as `CleanSig.wav`; it is only there for you to use as an example of a clean data signal.

2.4.2 Part 1 of the receiver: Noise filter

- You will first need to analyze the corrupted signal in MATLAB to determine the noise content of the signal. Specifically, plot the power spectrum of the clean data, the noise, and the corrupted data signals. You can look at `CleanSig.wav` to characterize a clean data signal. Then, decide whether the noise could be removed without affecting the data.
- In the preparation, you plotted the power spectrum of a biphase signal. Compare the measured power spectrum with the theoretical one.
- If the noise could be removed without affecting the data, decide what kind of filter you will need to pass the noisy signal through to reduce the noise without affecting the data. Specifically, determine the frequency bands and attenuation of the filter. To determine the stopband attenuation, measure the input signal-to-noise ratio and estimate the SNR needed for reliable decoding of the data. The difference of the SNR values in dB is the required noise attenuation. Rationalize a choice for the passband ripple (Is 3 dB too much or too little – what is a reasonable value?).
- Once you have all the above information, use SPTool to design the filter. At this stage you will also need to decide which design method is best for your purpose.
- Once you have created the appropriate filter, for both FIR and IIR filter types, you can use SPTool to apply it, or create a Simulink model, using a signal source to read your data file, and your filter block to filter it.

- Decide which filter type is the best, and why.

2.4.3 Part 2 of the receiver: Channel compensation (equalization)

With the noise reduced, you need to compensate for the channel. This is made easier by the impulse included in the data signal. The channel will smear this impulse in the same way it distorts the data. You can use the impulse response of the channel to identify the channel transfer function. Then you need to design a filter that is the inverse of the channel response.

Note: You will not be able to get rid of all the noise or fully compensate for the channel. Remember that the objective is to decode the data, not to make the signal look perfect.

2.5 Report

Include in your report the following:

- What you did and why.
- Biphase coding has been described above. It can generated by a two-state model with transitions. Prepare a diagram showing the two-states, the transitions between states which depend on the input bits (0 or 1), and the output generated with each transition. These outputs are [+1,+1], [+1,-1], [-1,-1], and [-1,+1].
- A detail history of your signals, as well as plots of the frequency response of the filters you used.
- An estimate of the number of operations/sec that is required by your filtering operations (see Sections 9.2–9.3 in Proakis and Manolakis [2] or section 6.3 in Oppenheim and Schafer [1]). Think about whether your solution could be implemented in real-time to descramble a continuous stream of data.
- Estimate the impulse response of the system including the noise suppression filter, but excluding the channel compensation filter? Comment on the signal distortion caused by the noise suppression filter.
- The decoded message.
- Ideally, the transfer function of the compensator should be the inverse of the channel transfer function. Think about how you estimated the response of your compensator and write down its transfer function as a function of the channel filter and noise filter transfer functions. Comment on the effectiveness of your compensator.
- The channel compensation filter is the inverse of the estimated channel response. Can you still find an inverse filter if the channel response has a z-transform with zeros outside the unit circle (i.e, it is non-minimum phase). Suggest work-arounds for such a situation.

References

[1] A. V. Oppenheim, R. W. Schafer, with J. R. Buck, *Discrete-Time Signal Processing*, 2nd edition, Prentice-Hall, 1998 (ISBN 978-0-13-754920-7).

[2] J. G. Proakis and D. G. Manolakis, *Digital Signal Processing*, 4th edition, Prentice-Hall, 2007, (ISBN 978-0-13-187374-2).

[3] D. Goldberg, "What Every Computer Scientist Should Know About Floating-Point Arithmetic", *ACM Computing Surveys*, vol. 23, no. 1, pp. 5-48, March 2001 (Available electronically on the DSP Lab web site).