GEORGIA INSTITUTE OF TECHNOLOGY
SCHOOL of ELECTRICAL and COMPUTER ENGINEERING

ECE 2025 Fall 2001
Lab #9999: Updates for the FSK Challenge

Date: 4 Dec. 2001

Rules of the contest remain the same. Please read the initial posted PDF file for the rules and a description of the scenario. This update will only describe new issues that you will now encounter in the data.
Have fun!

---

1 The Bulldog-301

Recent events have shown that the original *Bulldog-300* is not as secure as its designer had anticipated. Therefore, Dr. Rad Bleck has ordered his minions back to the pound with instructions to design a new model with a second layer of encryption.[1] It works as follows:

(a) The message bits are first encrypted with the XOR method, as described in the earlier memo. Call the result of this encryption, $b_x[n]$.

(b) Then the bit stream from the XOR, $b_x[n]$, is filtered with a digital filter whose z-transform is:

$$H(z) = \frac{1}{1 - z^{-1}}$$

(c) Call the filtered output $b_y[n]$. Make the length of $b_y[n]$ the same as the length of $b_x[n]$ by keeping the beginning and chopping off the tail. This is necessary because $H(z)$ is IIR, meaning that its response is infinitely long.

(d) The signal $b_y[n]$ is not guaranteed to have only zeros and ones, so we must fix that. Modulo-2 to the rescue.[2] The encrypted output is obtained by applying a modulo-2 operator to the filter's output:

$$b_o[n] = b_y[n] \bmod 2$$

where $b_o[n]$ is the output bit stream.

(e) Example: $b_x[n] = \{0, 1, 0, 1, 1, 1, 1, 0\}$ gives $b_y[n] = \{0, 1, 1, 0, 1, 0, 1, 1\}$.

You will need to discover the decryption algorithm on your own. So, here is one hint: *deconvolution.* You must undo the effect of $H(z)$ to recover the bits. We've used deconvolution in a earlier lab, and it is discussed in the textbook.

NOTE: remember that this encryption is done on top of the XOR encryption. Also, only the data bits are encoded; not the preamble nor the trailer.

[1]Some might call this new product the *Bandaid-300* because its encryption is meant to cover up the flaws of the XOR method.
[2]It is interesting to point out that the exclusive-OR of a and b is equal to $(a + b) \bmod 2$.

2 Combatting Noise

Real communication channels have noise added to the signal. This is often simulated in MATLAB by using the `randn` command to generate Gaussian random numbers, and then adding the random data to the actual signal.[3] For the data of this project, there is a *high noise level*. Therefore, we need a strategy to cope with the noise. If not, bad things can happen: random bits will be changed from 0 to 1, or 1 to 0. This will show up as random bad characters, or worse it will make it impossible to find the beginning of the message because locking on the synch pattern and/or the start flag will be tricky.

There is one simple way to fight the noise: use an averaging filter *before detecting* the sign bit of $d[n]$, the signal that comes out of the slicer. The underlying idea is that $d[n]$ should have several samples in a row that all have the same sign bit. Without noise, $d[n]$ would be constant for several samples, but with noise the values of $d[n]$ go up and down. However, if we average several consecutive values then we'll get something close to the true value. Try the following code snippet:

```
xx = 1+randn(1,8), xx_avg = sum(xx)/8
```

Notice that even though a few of the values of `xx` are negative, the average is still positive. Without noise, `xx` would be all ones, so its average should be one. As long as the averaging filter reduces the noise enough to give the correct sign bit, the resulting $b[n]$ sequence formed from the filtered $d[n]$ signal will be much easier to decode.

When you include an averaging filter in your processing chain, you will have to pick its length, but you should be able to make an educated guess of the length, by considering the bit rate of the data.

3 Recorded Signals Available

There are several recordings that are available for your analysis. Lots of random noise has been added to each signal, because the "real world" always has some noise. On a spectrogram, noise shows up at all frequencies.

In each case, the signal was sampled with an A/D converter whose sampling frequency is

$$\boxed{f_s = 9000 \text{ Hz}} \tag{1}$$

You can get the first signal from the ZIP file called

$$\boxed{\texttt{FSK\_3\_start.zip}}$$

which can be downloaded from WebCT. The ZIP file contains a MAT file, called `FSK_3_start.mat`. The MAT file contains one signal that was sampled at 9000 Hz.

[3]This is one major topic in ECE-3075 *Random Signals*.