Chapter 5 MATLAB EXERCISE

Mu-Law Encoding in MATLAB

Mu-law encoding (μ-law) is a non-linear companding method that can be used to reduce the bit depth of a digital audio signal in a way that preserves the dynamic range of samples at low amplitudes. The word "companding" is used because this method works by *compressing* the bit depth of an audio signal and then *expanding* it again after the signal has been transmitted. Mu-law encoding is non-linear because it uses more bits to quantize low amplitude samples as opposed to high amplitude ones. This method works well for telephone communication because it reduces the signal-to-noise ratio in the area where it matters most – in low amplitude values, which are common in human speech and are particularly subject to noise distortion. The equation for non-linear compression by mu-law encoding is

$$m(x) = sign(x)\left(\frac{\ln(1+\mu|x|)}{\ln(1+\mu)}\right)$$

where $-1 \le x \le 1$, and *sign(x)* is –1 if *x* is negative and 1 otherwise. μ is 255 when samples are being quantized to 8 bits. (In some sources, you will see this equation using log_2 rather than the natural log. The definitions are equivalent.)

Decompression operates by the inverse equation:

$$d(x) = sign(x)\left(\frac{(\mu+1)^{|x|}-1}{\mu}\right)$$

Here is a scenario in which mu-law encoding could be used. A 16-bit digital audio signal – for example, a telephone signal – is to be transmitted across a network. It is reduced to 8-bits by mu-law encoding and then transmitted. At the receiving end, it is decoded back to a 16-bit signal. Because the amount of error for low amplitude samples is less than it would have been in a linear encoding, the effect is that a dynamic range of 72 dB is achieved with mu-law encoding (as opposed to 48 dB that you would expect from an 8-bit audio signal).

Your assignment is to write a MATLAB function or sequence of commands that implements mu-law encoding and decoding with the equations above. Compare the error you get with mu-law encoding versus the error you get with a linear companding method.

This material is based on work supported by the National Science Foundation under CCLI Grant DUE 0717743, Jennifer Burg PI, Jason Romney, Co-PI.

Solution:

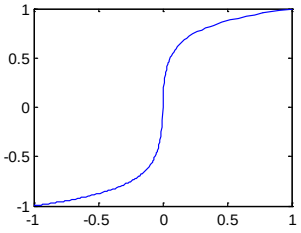
You can create the μ-law function for μ = 255 with the following MATLAB inline function:

```
f = inline('sign(x)*((log(1+255*abs(x)))/log(256))', 'x');
```

If you plot the μ-law function over the domain interval *x*=[-1 1], you see it has the expected shape.

```
fplot(f, [-1 1]);
```

The graph should look something like this:



This function assumes that the input values are in the range of –1 to 1. If we actually want to begin with 16-bit quantized samples in the range of –32768 to 32767, we need to divide the quantized value by 32768 before sending it as input to the function and requantize at 8 bits before returning it. We can do this by adapting the function above in the following manner:

```
mu = inline('sign(x) * floor(128*((log(1+255*abs(x/32768))/log(256))))',
'x');
```

The mu-inverse function is given by the following MATLAB inline function:

```
mu_i =inline('ceil(32768*sign(y)*(1/255)*(256^abs(y/128)-1))', 'y');
```

Linear companding would be accomplished by the following two functions:

```
lin = inline('floor(x/256)', 'x');

lin_i = inline('y * 256', 'y');
```

Here's a comparison of mu-law encoding and linear companding for some example values:

```
mu(100) = 13
mu_i(13) = 98
lin(100) = 0
lin_i(0) = 0

mu(1000) = 50
```


This material is based on work supported by the National Science Foundation under CCLI Grant DUE 0717743, Jennifer Burg PI, Jason Romney, Co-PI.

```
mu_i(50) = 993
lin(1000) = 3
lin_i(3) = 768

mu(10000) = 100
mu_i(100) = 9652
lin(10000) = 39
lin_i(39) = 9984

mu(31000) = 126
mu_i(126) = 30038
lin(31000) = 121
lin_i(121) = 30976

mu(-50) = -7
mu_i(-7) = -45
lin(-50) = -1
lin_i(-1) = -256

mu(-2000) = -64
mu_i(-64) = -1927
lin(-2000) = -8
lin_i(-8) = -2048

mu(-12000) = -104
mu_i(-104) = -11502
lin(-12000) = -47
lin_i(-47) = -12032

mu(-28000) = -124
mu_i(-124) = -27534
lin(-28000) = -110
lin_i(-110) = -28160
```


This material is based on work supported by the National Science Foundation under CCLI Grant DUE 0717743, Jennifer Burg PI, Jason Romney, Co-PI.