LAB Assignment #6 for ECE 525

Assigned: Tue., Mar. 7, 2017
Due: Thur., March. 9, 2017

Description: Process PND into PND_c, PND_{co} and $modPND_{co}$

1) This lab adds to the code you created for lab5, and assumes you did not do the extra credit portion of creating the PND_c.

2) From lab5, you have an array of floating point values, PND, of size 2048. Write a routine that computes the PND_c as follows. Functional definition should look similar to this:

```
void ComputePNDc(int max_PNDiffs, float PND[max_PNDiffs], float
PNDc[max_PNDiffs], float reference_mean, float reference_range)
```

3) Use the following equation to convert from PND to PND_c:

$$zval_i = \frac{(PND_i - \mu_{TVx})}{Rng_{TVx}} \qquad \textbf{Eq. 1.}$$

$$PND_c = zval_i Rng_{ref} + \mu_{ref} \qquad \textbf{Eq. 2.}$$

The constants μ_{TVx} and Rng_{TVx} are computed as the mean and **3*standard deviation** using the values in the PND array. Multiply the computed standard deviation by 3 to obtain the Rng (range). Look up the mean and standard deviation formulas on wikipedia under Gaussian distributions. Make the *reference_mean* and *reference_range* equal to 0 and 100, respectively.

4) PND_c are then transformed by adding a random offset to another new array, PND_{co}. Functional definition should look similar to this:

```
void ComputePNDco(int max_PNDiffs, float PNDc[max_PNDiffs], float
PNDco[max_PNDiffs], int LFSR_seed, int Modulus)
```

The random offset is restricted to a value between 0 and *Modulus*/2. The random offset is computed individually for each PNDc using an LFSR as follows:

Use the two calls to the LFSR that I provided for lab5, one that initializes and returns the first LFSR value, one that returns only the next value in the LFSR sequence.
Divide the parameter *Modulus* by 32 and store this constant in a variable called *offset_delta*.
For each LFSR value that is returned, mask off (make zero) all bits from 4 to 31 (use ONLY the low order 4 bits, i.e., bits 0 through 3, of the LFSR value).
Multiply the 4-bit LFSR by the *offset_delta* and then add it to the PNDc. Store the new value in the same position but in the new PNDco array.

Use a seed of 0 for the LFSR and a *Modulus* of 20.

5) The modulus operator is then applied to the PND_{co} to create $modPND_{co}$. Functional definition should look similar to this:

```
void ComputeModulus(int max_PNDiffs, float PNDco[max_PNDiffs], float modP-
NDco[max_PNDiffs], int Modulus)
```

The PND_{co} are floating point values so you'll need to compute the Modulus of the PND_{co} using the following pseudo-code:

```
while(1)
   if PND_co < 0, add Modulus to modPND_co
   else if PND_co > Modulus, subtract Modulus from modPND_co
   else break
```

The $modPND_{co}$ are the floating point values from which you will generate the bitstring in the next lab.