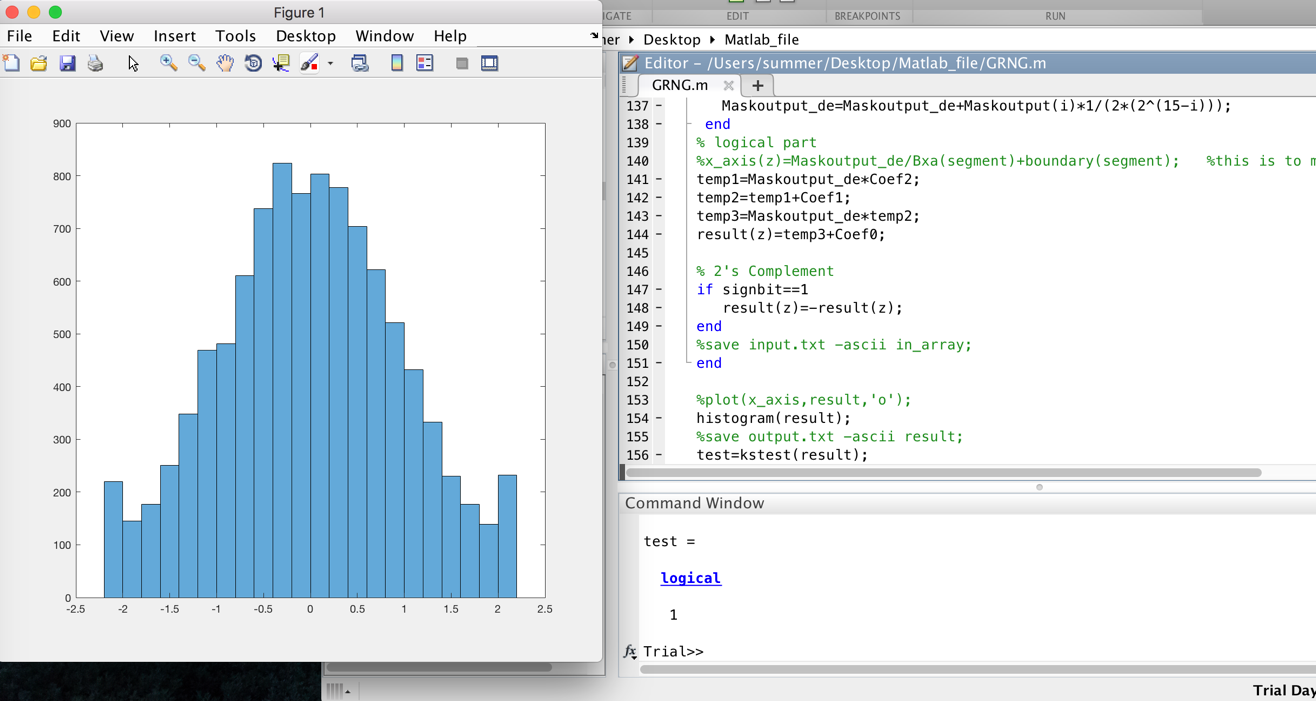
Reference: R.Gutierrez,V.Torres, and J.Valls “Hardware Architecture of a Gaussian Noise Generator Based on the Inversion Method”. Vol.59, No.8, AUGUST,2012

**Steps to recreate Matlab result:**

1. Open Matlab, navigate the folder to Matlab\_file
2. Run segment.m file, get the segment.txt file (this includes the limits of segmentation)
3. Run Coefficient\_gen.m to generate Coefficients.
4. Run GRNG to get the result in result.txt file. Also the histogram will pump up.
5. Type test in Command Window, push enter. The kstest result will pump up.



**RTL part:**

1. Open ModelSim, create a new project, add existing file, browse to RTL\_file.
2. Add all Verilog files and top\_level\_tb.v
3. Start simulation. (sorry I am still working on debugging the top-level)
4. The Modelsim\_prj folder includes the Modelsim projects I did and files.

**Explanations for the Logic of my project:**

1. Matlab model:
2. Function.m file is to draw the ICDF function. And found out that the function is symmetric at 0.5, so I chose the approximation interval to be (0.5,1).
3. Use the minimax method to calculate an Error max and use the Error max to calculate the segmentation limits.(Emax.m segment.m )
4. Referred to paper[2] and decided to use hierarchical segmentation scheme with P2SR. This determined my LZD output to be two parts. The [6:1] bits represent the position of leading zero. While the [0] bit is the first bit following the leading zero. In this way, a two hierarchical segmentation is achieved.
5. With the segmentation, I use the polyfit function to get the coefficients (Coefficient\_gen.m), stored in a\_coef.txt file.
6. Referred to paper[6], I transformed Coefficients and stored in trans\_coef.txt.
7. Mask to zero part and MUX part, and the addition, multiplication are the same as paper mentioned. While in matlab, I didn’t considered the position of decimal point as in binary in the calculation, but using decimal. I did the arrangement in RTL part.
8. As for the uniform random number generation. Since the rand function in matlab can generate random number in (0,1), I used it and transform the decimal into binary. One problem is that the last 12 bits of the random number are always 0s. It is fixed by concatenating a random 12 bits with it.
9. RTL model
10. For top level, I used FSM to guarantee the correct result calculated.
11. The logic of each part is similar to matlab ones. Except that I combined the adders, multipliers, and rom\_coef into datapath. The calculation here is fixed point.
12. For the datapath, input 15 bits values, assume it has 1 bit of integer and 14 bit of fractional part. So for the three coefficients, I arrange the integer and fractional bits width as the table shows.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Coef2 (18 bit) | | Coef1 (18 bit) | | Coef0 (21 bit) | |
| integer | fractional | integer | fractional | integer | fractional |
| 1 | 17 | 2 | 16 | 4 | 17 |