HW#6 Solutions

(a) Using the MATLAB script provided in lecture, we generate the N=10 bit "M"-sequence whose length is 2^N -1 = 1023 "chips". The sequence is ordered columwise starting in the upper left and ending in the lower right, the last element being an "X", meaning it doesn't exist.

1	1	-1	1	-1	1	1	-1	-1	-1	1	1	-1	-1	1	1	1	1	1	1	-1	-1	1	-1	1	-1	1	-1	1	-1	-1	1
1	-1	-1	1	1	-1	-1	1	-1	1	-1	-1	1	1	1	1	1	-1	1	-1	-1	1	1	1	-1	-1	-1	-1	1	-1	-1	-1
1	1	-1	1	1	-1	-1	1	-1	-1	-1	1	-1	1	-1	-1	1	1	-1	1	1	1	1	-1	1	1	1	-1	1	-1	1	-1
1	1	1	-1	-1	1	1	-1	-1	1	1	1	-1	1	1	1	-1	1	1	1	-1	-1	1	1	1	-1	1	-1	1	-1	-1	1
1	1	-1	1	-1	-1	-1	-1	-1	1	1	1	1	-1	1	1	-1	1	1	1	-1	-1	-1	-1	1	1	-1	-1	-1	1	-1	-1
1	-1	1	-1	-1	1	-1	1	1	-1	-1	1	1	-1	1	-1	-1	-1	1	-1	-1	-1	1	-1	1	1	-1	1	-1	-1	1	-1
1	1	1	-1	1	-1	-1	1	1	-1	-1	-1	1	-1	1	1	-1	-1	-1	-1	-1	-1	1	-1	1	-1	-1	1	-1	-1	1	-1
1	1	1	1	1	-1	1	1	1	1	-1	-1	-1	1	1	-1	-1	-1	1	1	-1	1	1	1	-1	1	1	-1	-1	-1	-1	1
1	1	1	-1	-1	1	-1	-1	1	1	1	-1	-1	1	-1	1	1	-1	-1	-1	1	-1	-1	-1	-1	1	1	-1	1	1	1	1
1	1	1	-1	-1	1	1	1	-1	-1	-1	1	1	-1	1	-1	1	-1	-1	1	-1	1	-1	-1	-1	-1	1	-1	-1	-1	-1	1
-1	-1	1	-1	1	1	-1	1	1	1	1	1	-1	1	-1	1	1	1	-1	-1	-1	1	-1	1	1	1	-1	-1	1	-1	-1	-1
-1	1	1	1	1	1	-1	1	1	-1	1	-1	1	-1	1	-1	-1	-1	1	-1	1	1	1	1	-1	1	1	-1	-1	1	1	1
-1	-1	1	1	1	1	1	-1	-1	-1	-1	-1	1	1	1	-1	-1	1	-1	-1	1	-1	1	-1	1	1	-1	-1	1	-1	1	1
1	1	-1	-1	1	-1	1	1	1	-1	-1	-1	-1	-1	1	-1	1	-1	1	1	-1	1	1	-1	-1	1	1	-1	-1	-1	-1	1
1	-1	1	-1	1	1	-1	1	1	1	-1	1	-1	-1	-1	1	-1	1	-1	1	1	1	1	1	1	-1	1	-1	-1	-1	1	1
1	-1	-1	1	1	-1	1	1	1	-1	-1	1	-1	1	-1	-1	-1	1	1	-1	1	-1	-1	-1	-1	-1	-1	1	1	-1	-1	1
-1	-1	1	-1	-1	-1	1	-1	-1	-1	-1	-1	1	-1	-1	1	1	-1	1	1	-1	1	1	-1	-1	1	1	1	1	-1	-1	1
-1	1	-1	1	-1	1	1	-1	-1	-1	-1	1	-1	1	1	1	-1	1	1	-1	1	-1	-1	-1	1	1	-1	-1	-1	-1	1	-1
-1	1	1	1	1	1	1	1	-1	1	1	1	-1	-1	-1	1	1	1	1	-1	-1	-1	-1	-1	-1	1	1	1	-1	1	1	-1
1	1	-1	-1	-1	1	-1	1	-1	-1	-1	1	-1	-1	1	1	-1	-1	1	-1	-1	-1	-1	-1	1	1	-1	1	-1	-1	1	-1
-1	1	1	1	1	-1	1	1	1	1	1	-1	-1	-1	1	-1	1	-1	1	-1	1	1	-1	1	-1	-1	-1	-1	1	-1	1	-1
-1	-1	-1	-1	-1	-1	1	-1	1	1	-1	1	1	-1	1	1	1	1	-1	-1	1	1	1	1	-1	-1	-1	1	1	1	-1	1
1	1	1	1	1	-1	1	1	-1	-1	-1	1	1	1	-1	1	-1	1	1	-1	1	-1	1	-1	-1	-1	-1	1	1	-1	-1	1
1	-1	1	1	-1	-1	-1	-1	-1	1	1	-1	-1	-1	-1	-1	-1	-1	-1	1	1	-1	1	1	-1	1	1	-1	1	-1	1	1
1	-1	1	-1	1	1	1	1	-1	-1	-1	-1	1	-1	1	-1	1	-1	-1	1	-1	-1	-1	-1	1	-1	1	1	-1	-1	1	-1
-1	1	1	-1	-1	-1	-1	-1	1	-1	-1	-1	1	-1	-1	1	-1	-1	-1	-1	-1	-1	1	-1	-1	-1	-1	-1	-1	-1	-1	-1
1	-1	-1	1	-1	-1	1	-1	-1	1	1	-1	1	-1	-1	1	1	-1	1	-1	1	1	1	1	1	-1	-1	1	1	-1	-1	-1
1	1	1	1	1	-1	-1	1	-1	-1	-1	1	1	1	-1	1	1	1	1	1	1	-1	-1	-1	-1	1	1	1	-1	-1	-1	-1
-1	-1	-1	1	1	1	1	1	1	1	1	1	1	-1	-1	-1	1	1	1	-1	-1	-1	1	-1	-1	1	1	1	-1	1	1	-1
-1	1	-1	1	-1	1	1	1	-1	1	1	1	1	-1	1	-1	1	-1	-1	-1	1	1	1	1	-1	1	-1	-1	1	-1	1	-1
1	-1	-1	-1	-1	-1	1	-1	1	1	1	1	1	1	1	1	-1	1	-1	1	-1	1	-1	1	-1	1	1	1	1	-1	1	-1
-1	-1	-1	1	1	1	-1	1	-1	-1	1	-1	-1	-1	1	1	-1	-1	1	-1	1	1	-1	1	-1	1	1	-1	-1	1	1	X

The chips are in signed-binary format, being either -1 or +1. Like the NRZ (non-return to zero) format in digital communications, this has benefits in the digital signal processing, such as allowing multiplications involving this sequence to be replaced by summations [in general, summations requires far less hardware resources (i.e., logic gates) than multiplications]. As mentioned in the lecture, the M sequences have many special properties that can be tested readily in the MATLAB workspace. Assuming the sequence is stored in the 1023-element vector prn, we can execute the command "sum(prn)" and get a return of "+1". This means, of course, that there is one more +1 than -1 in the 1023 sequence.

(b) The standard deviation of any vector can be carried out very quickly in the MATLAB workspace using the command "std". By typing in "std(prn)", we get a return "1.0005". Since the amplitude of the sequence is 1 (i.e., all chips have the same magnitude of 1), this makes sense. Note the standard deviation would be exactly 1.0 if there were an equal number of +1s and -1s in the sequence.

(c) The cross correlation is computed by the following routine that is best copied into a .m file and then executed as a MATLAB script.

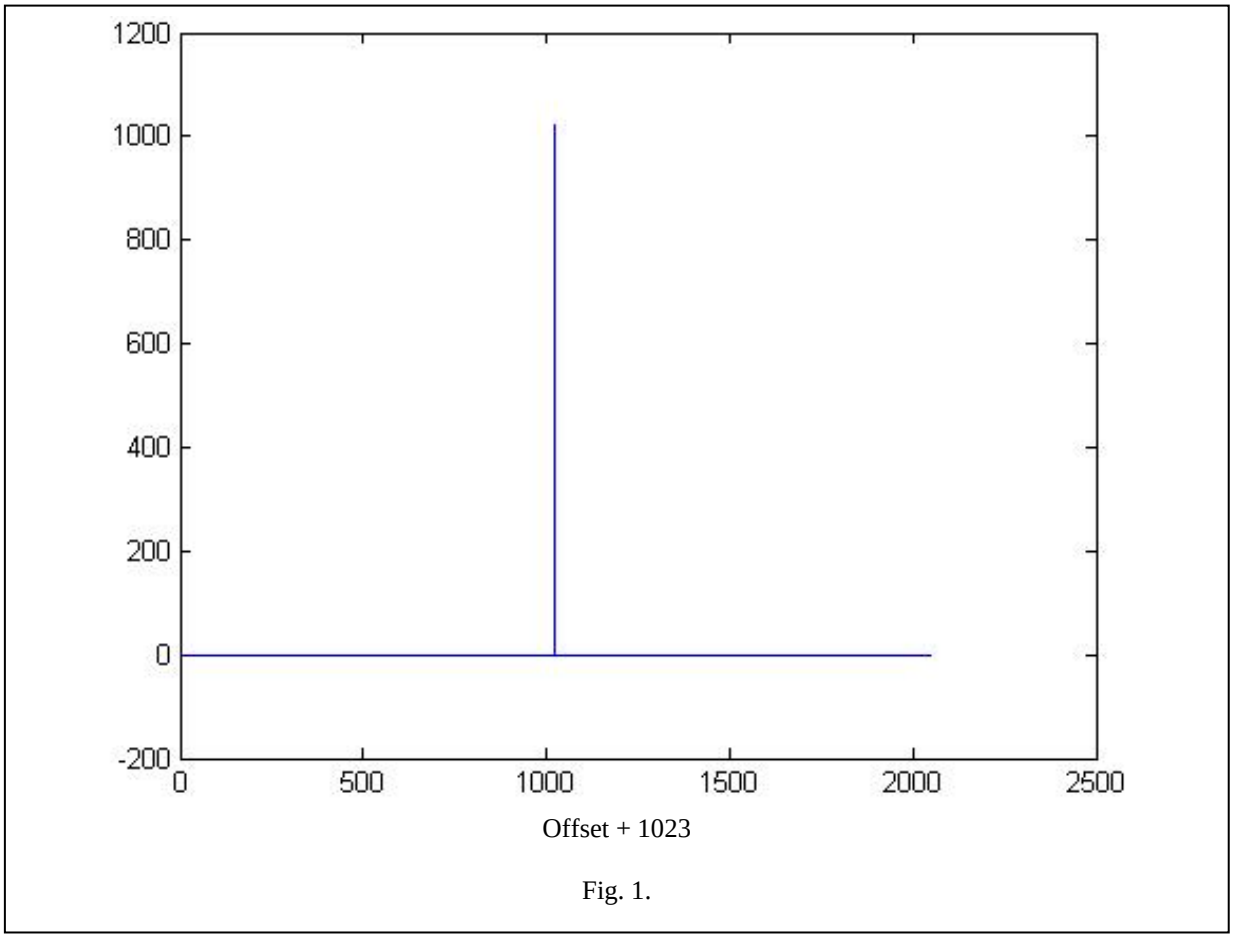
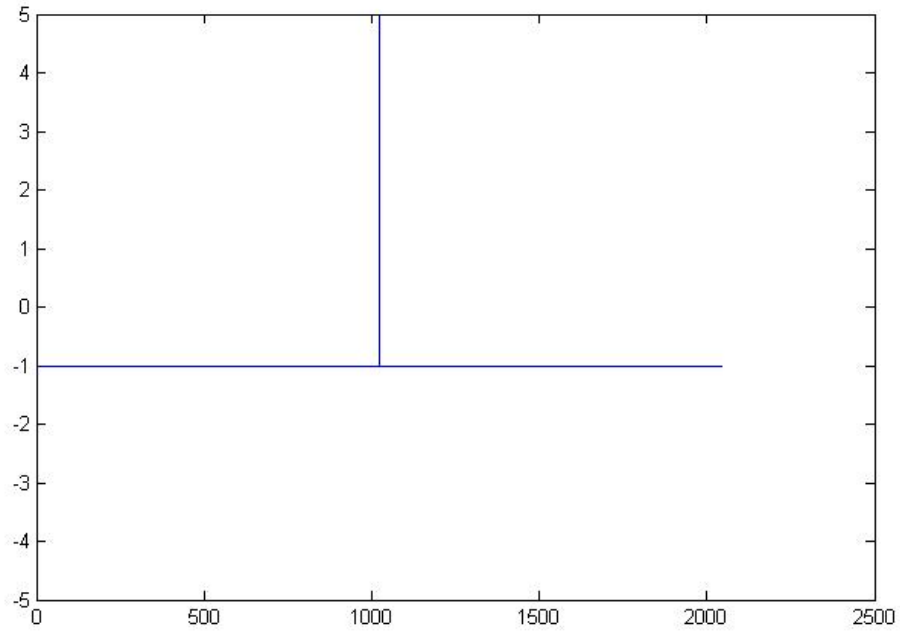


Fig. 1.

```
prnprime=prn';
crosscorr1=[]
prn3 = [prn; prn; prn];
for icorr=-nprn+1:1:nprn-1
%crosscorr is the inner product of the row vector xprime and column vector prn3
%the length of both vectors is nprn
      crosscorr1(icorr+nprn) = prnprime(1:nprn)*prn3(nprn+icorr+1:2*nprn+icorr);
end
```

This results in the cross correlation plot shown in the Fig. 1 above. Note that the horizontal axis is shifted so that the autocorrelation for an offset of zero occurs at an index of 1023, "crosscorr1(1023)", which equals "1023". This makes perfect sense. Any sequence of +1s and -1s, no matter how long, will produce a zero-offset value of the autocorrelation function equal to the length of the sequence.

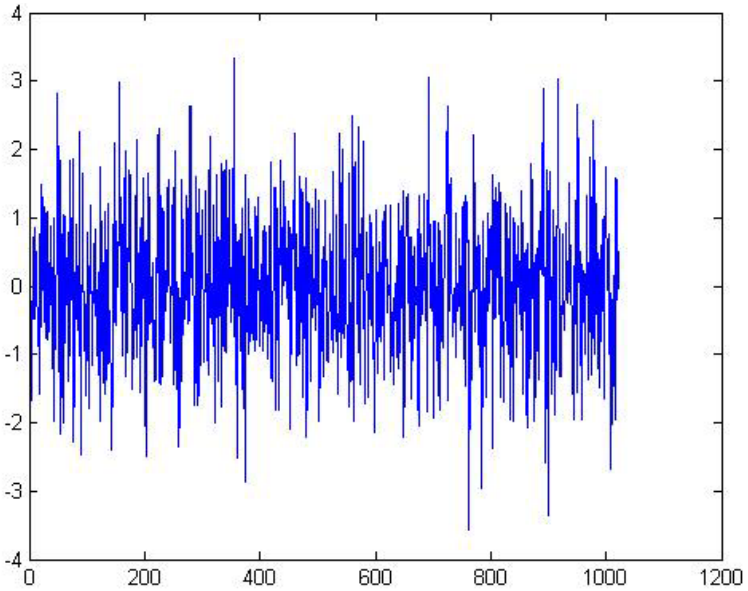
More interesting is the autocorrelation function at offsets other than zero. This is seen clearly below by expanding the vertical axis of Fig. 1, as is done in Fig. 2 above. It is clear that the autocorrelation function away from zero offset is *always -1*. In other words, the autoccorelation function $R_{xx}(i)$ for offset i satisfies the relation $R_{xx}(i) = \sum_{m}^{N} x_m x_{m-i}$ = N if i = 0, = -1 otherwise. If you happen to have a background in statistics or probability theory, you know how unique this is.



Offset + 1023

Fig. 2.

As a demonstration of the uniqueness, let's repeat the computation of the autocorrelation function for a vector representing discrete Gaussian noise. This is done easily in MATLAB starting with the "randn(1023,1)", which means random number generation of a 1023-length vector having a standard deviation of 1. The resulting vector is shown as a discrete time-domain waveform in Fig. 3 (note: this vector will change every time you execute "randn" in MATLAB, representative of a true "random" number generator). Putting this vector through the same autocorrelation routine as the PRN sequence, we get the autocorrelation plot in Fig. 4.



Offset + 1023

Fig. 3.

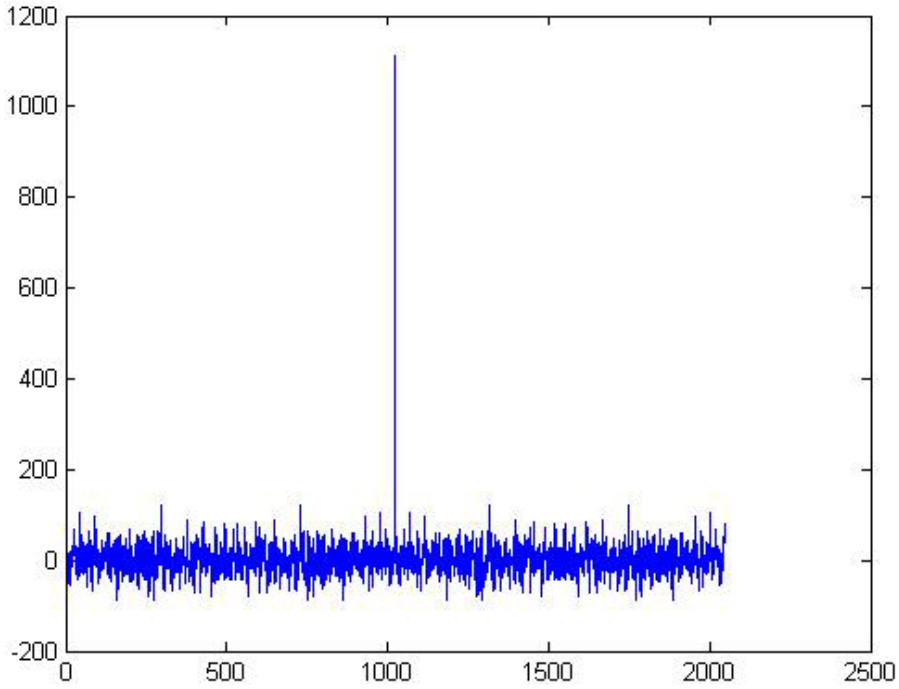
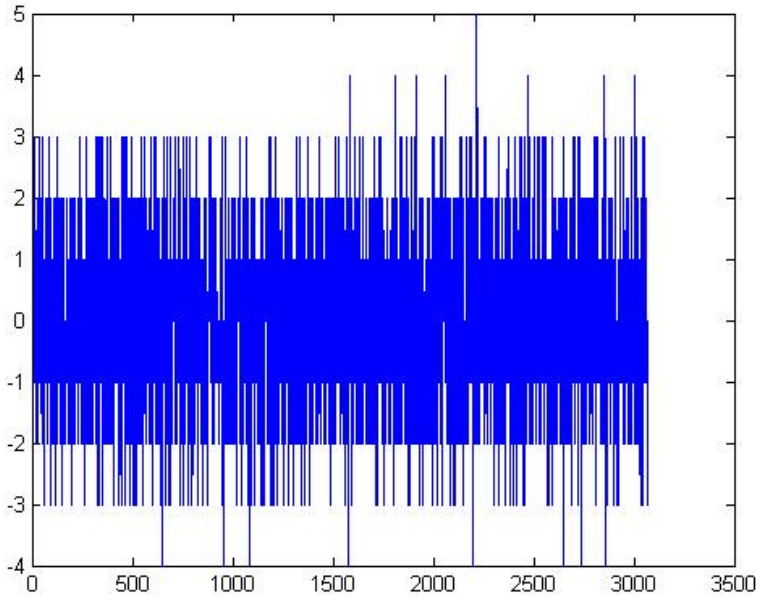


Fig. 4.

Two key results are obvious from Fig. 4. At zero offset, the autocorrelation indeed shows a peak as one would expect intuitively. And it is even somewhat larger than 1023, representing the fact that the Gaussian statistics allows for fluctuations well beyond the standard deviation (1.0 in the present case). These are clearly evident as the upward or downward going "spikes" in Fig. 3.

(d) The unique autocorrelation function of the PRN sequence in Figs. 1 and 2 arises largely because the sequence has only two possible amplitudes, +1 and -1, and because the sequence repeats over-and-over again in time. The randomness arises in the mostly random occurrence of these in the discrete-time waveform during a given $2^{\wedge N}$-1 sequence. So the PRN sequence displays both deterministic and random properties, and therefore is commonly used in digital communications, GPS, and radar as either a code or as the signal itself.

Therefore, an interesting exercise is to assume the signal is a 10-bit PRN sequence waveform and add to it Gaussian noise of the same standard deviation, 1.0. The resulting waveform is shown in Fig. 5. The presence of the PRN sequence is clearly seen as the dark zone between -1 an +1. The presence of the Gaussian component is seen as the spikes going as lows as -4 or as high as +5. The autocorrelation function is plotted in Fig. 5. Not surprisingly, it appears very much like the sum of the autocorrelation for the pure-PRN in Fig. 1 and the autocorrelation for the pure-Gaussian in Fig. 4.

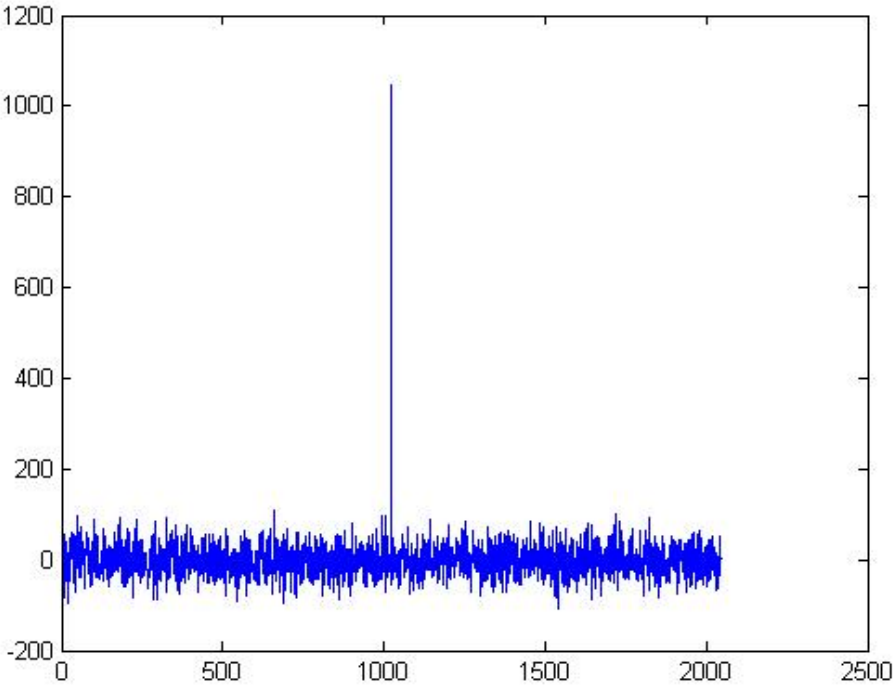


Offset + 1023

Fig. 5.

(e) The input signal-to-noise ratio (SNR) is computed as the ratio of the input-signal variance to the input-noise variance, where the variance is the square of the standard deviation. We take the variance because the SNR is always in terms of power (or energy), and the PRN and Gaussian sequences are signals (voltages or currents). The variance of the input PRN sequence portion of Fig. 5 is very close to 1.0. The variance of the Gaussian component in Fig. 5 is also 1.0 (by design). So the input SNR is 1.0.

(f) The output SNR (after the correlation) requires some explanation. In the discussion of the "matched-filter" concept, the intuitive goal was to "compress" all of the energy available from the signal into a narrow "spike" in the time-domain output, and then sample the output at this time only. From Fig. 6, the output signal power is proportional to the square of the peak of the autocorrelation function at zero offset, which is 1046^2 = 1.09×10^6. The noise is the fluctuation of this value over an ensemble containing a large number of samples. Because the autocorrelation function of a PRN M-sequence has no fluctuations, all the fluctuations seen in Fig. 6 can be associated with the random Gaussian noise added. For these, an ensemble average is the same as a long-time average (ergodic hypothesis). So the fluctuation in peak in Fig. 6 can be estimated from the variance of a large number of values of the autocorrelation function away from zero. Let's consider the first 1022 samples of Fig. 6 below the peak. In the MATLAB workspace we take "std(crosscorr1(1:1023)", which returns 33.4. Then we square this to get a variance of 1.11×10^3. The SNR after the correlator is thus SNR=$1.09 \times 10^6 / 1.11 \times 10^3$ = 982 – almost 1000 times higher than the input SNR of 1.0. According to rigorous digital-signal processing theory for an M-sequence embedded in Gaussian noise, it should simply be $SNR_{out}/SNR_{in} = 2^N - 1 = 1023$. If we repeated the MATLAB experiment many times, this would be the average value.



Offset + 1023

Fig. 6.

In the language of modern RF systems, this enhancement in SNR between the input and the output of a correlator is called the “processing gain.” It presupposes that the correlator is behaving as a matched filter for the signal component of the input waveform, which is generally true of all signals consistent with North’s theorem.