VANIER COLLEGE – Computer Engineering Technology – Winter 2021

Telecommunications (247-410-VA)

Leonardo Fusser (1946995)

LABORATORY EXPERIMENT #7

Introduction to MATLAB

NOTE:

To be submitted using the typical lab format, one week later – at the start of your lab session.

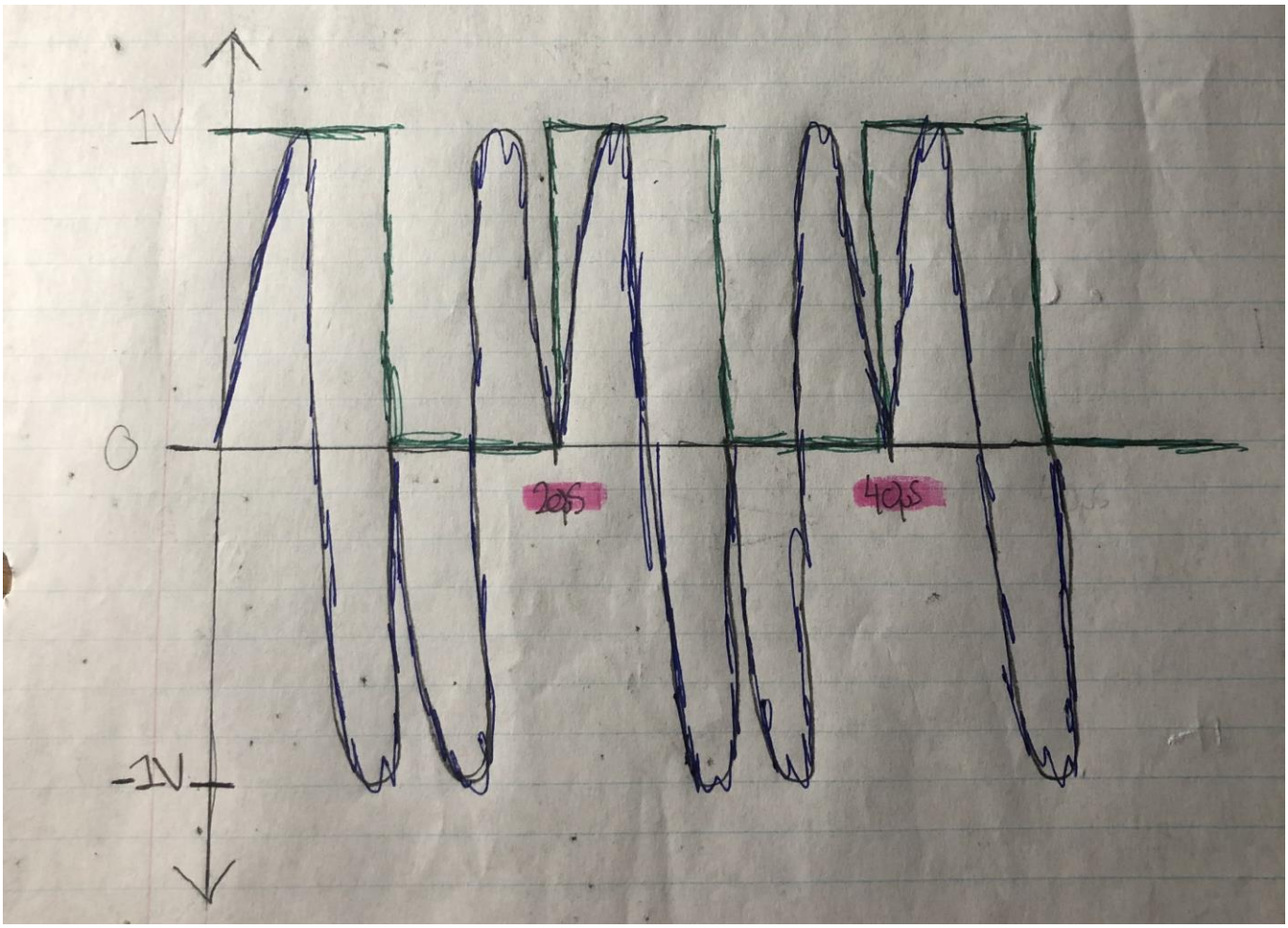
OBJECTIVES:

To become familiar with the use of MATLAB Simulink as a modeling tool for Communication systems.

- Use MATLAB Simulink to Model Simple Circuits
- Simulate & Interpret signal behavior with added Noise
- Simulate a simple digital modulation technique.

PRELAB:

Read the theory part of lab 5 and draw the output a PSK modulator in time domain with a sin wave carrier with frequency of 100kHz and modulation signal a square wave with frequency of 50kHz. Both amplitude 1V.



Output of PSK modulator shown above. Green signal shows original pulse signal and blue signal shows output of PSK modulator. There are 180° phase shifts to show the two different logic levels (1 or 0).

PROCEDURE

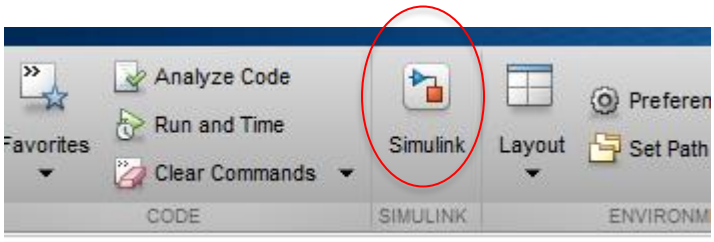
PART 1: Download and install MATLAB and Simulink from the following link
https://www.mathworks.com/campaigns/products/trials.confirmation.html?elqsid=1616786003030&potentialuse=Student&prodcode=ML#

If you can, install wireless communication toolbox as well. It takes 15Gbyte of hard disk. If you don't have space just download MATLAB and Simulink.

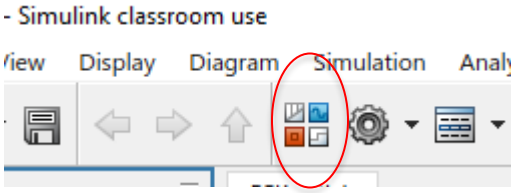
PART 2

Introduction

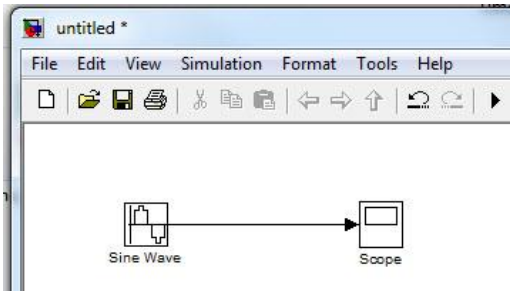
(1) Start MATLAB and launch Simulink (this is found on an icon in the menu bar).



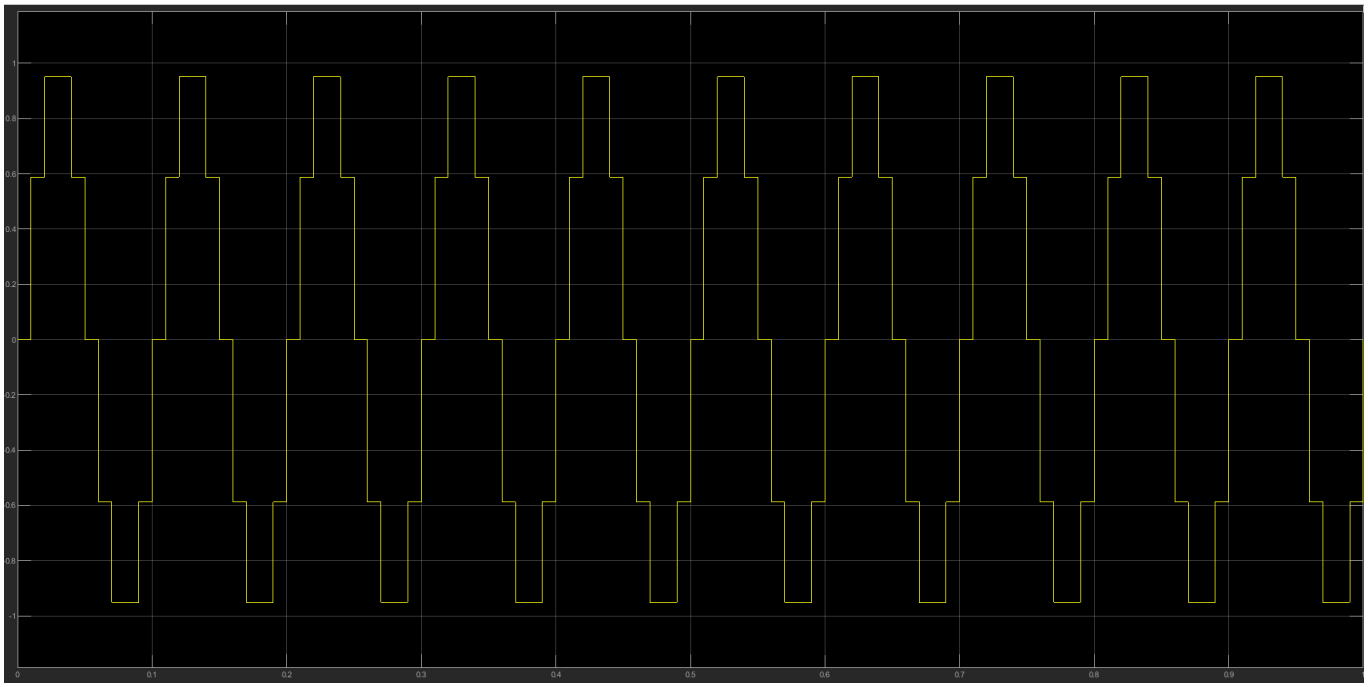
(2) Start a new model. In Simulink, a model is a collection of blocks which, in general, represents a system.
(3) Brows the library and from source library select sine wave and from sink scope



(4) build a simple simulated system that looks like the diagram below:

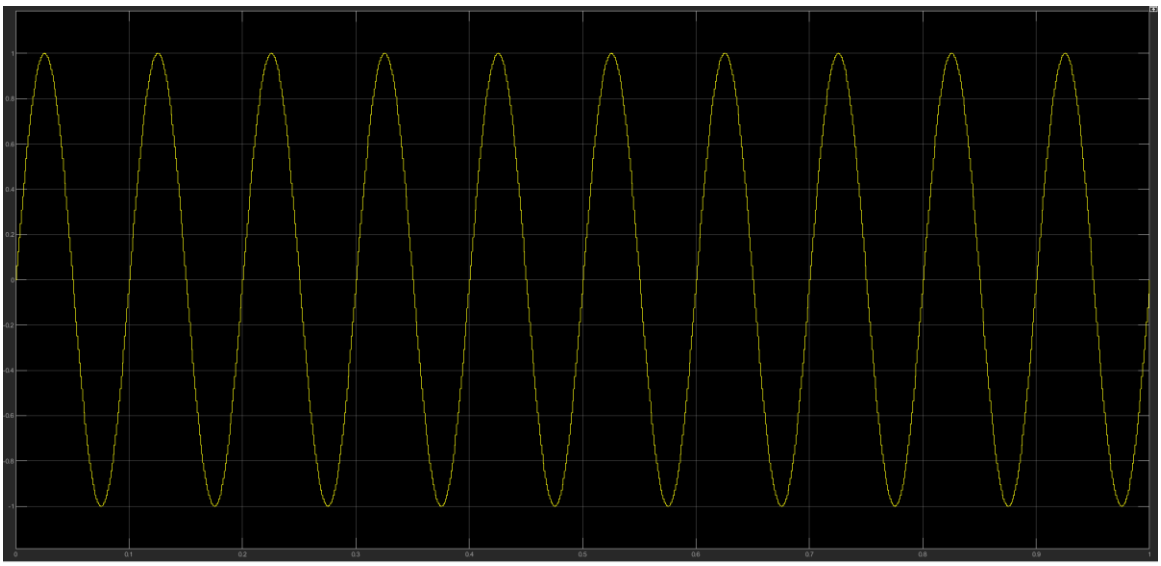


(5) set up the Sine Wave parameters (double click on the icon to access the settings) to give a sample time of 10ms and a source frequency of 10Hz – you will need to convert to required units. Use a Simulation Stop time of 1 (second). Play the simulation and observe the output on the Scope (again, double click to see its output). Summarize the results (including a sketch of the scope display).

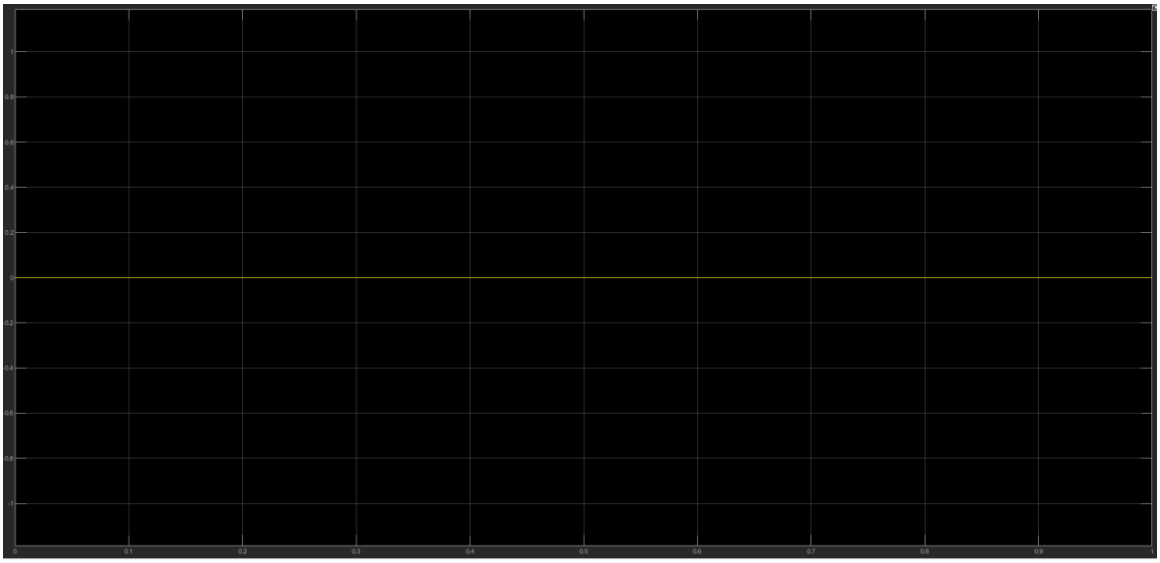


Result of under-sampling a sine wave signal shown above. Frequency of signal is 10Hz. As seen above, the signal is sampled every 10mS but this is too slow of a sampling rate because the reconstruction of the signal shown on the scope is far from the original signal (original signal is a sine wave). The signal that is being shown on the scope above is more like a square wave. In order for the reconstruction of the signal to look more like a sine wave, a faster sampling time is needed. This is shown on the next page.

(6) Leaving all other parameters the same, adjust the Sample Time to 1ms and run the simulation again. Summarize your results. Comment on the changes. What happens if you change the Sample Time to 100ms? Why?



a: signal with 1mS sampling rate shown above.



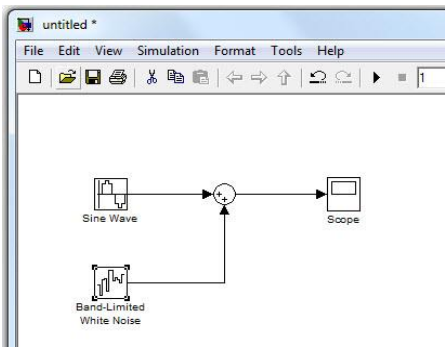
b: signal with 100mS sampling rate shown above.

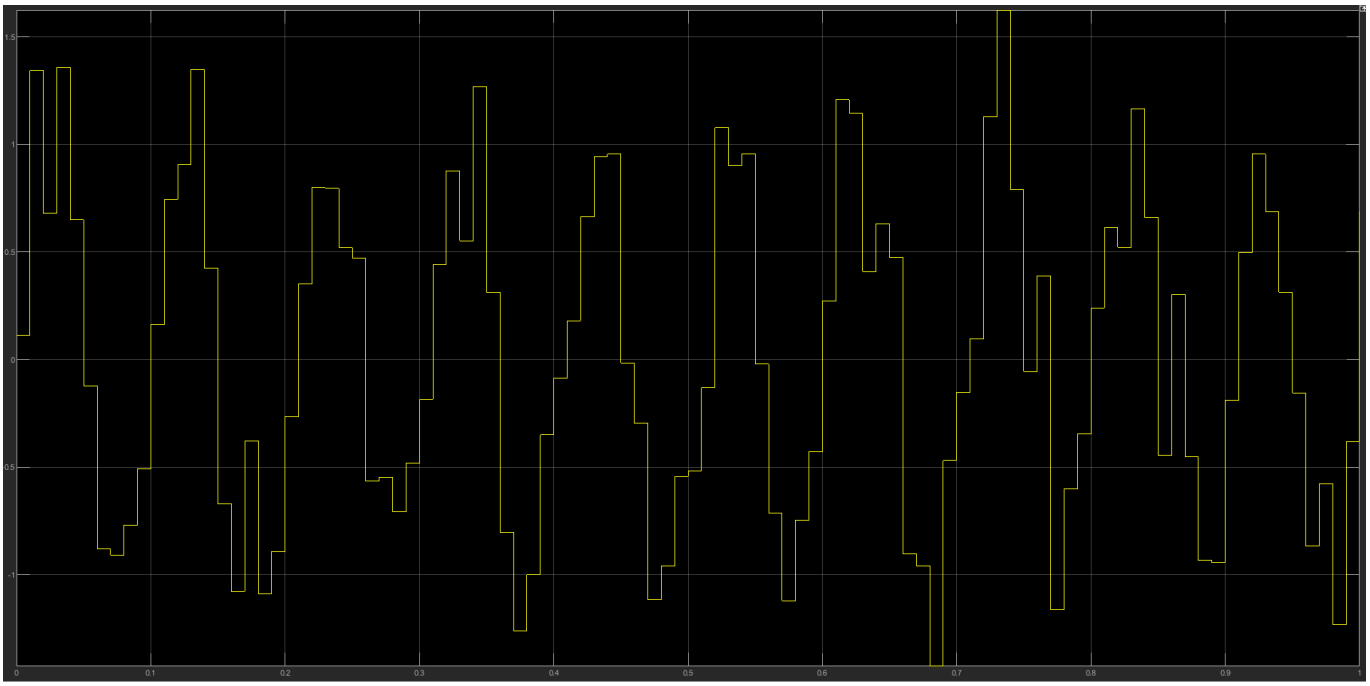
Result of adequately sampling a sine wave signal (a) and severely under-sampling a sine wave signal (b) shown above. As mentioned on the previous page, in order to correctly reconstruct the signal on the scope, a faster sampling rate is needed, which the result is seen in a above. On the other hand, in b above, the signal is very under-sampled and the reconstruction of the signal on the scope display just shows a constant flat line. This is because the sampling rate is set to be very slow (100mS) and the reconstruction of the signal cannot be done properly.

(7) Change the Sample Time back to 10ms, then adjust the Stop time to 'inf' and run the simulation. Comment on your observations.

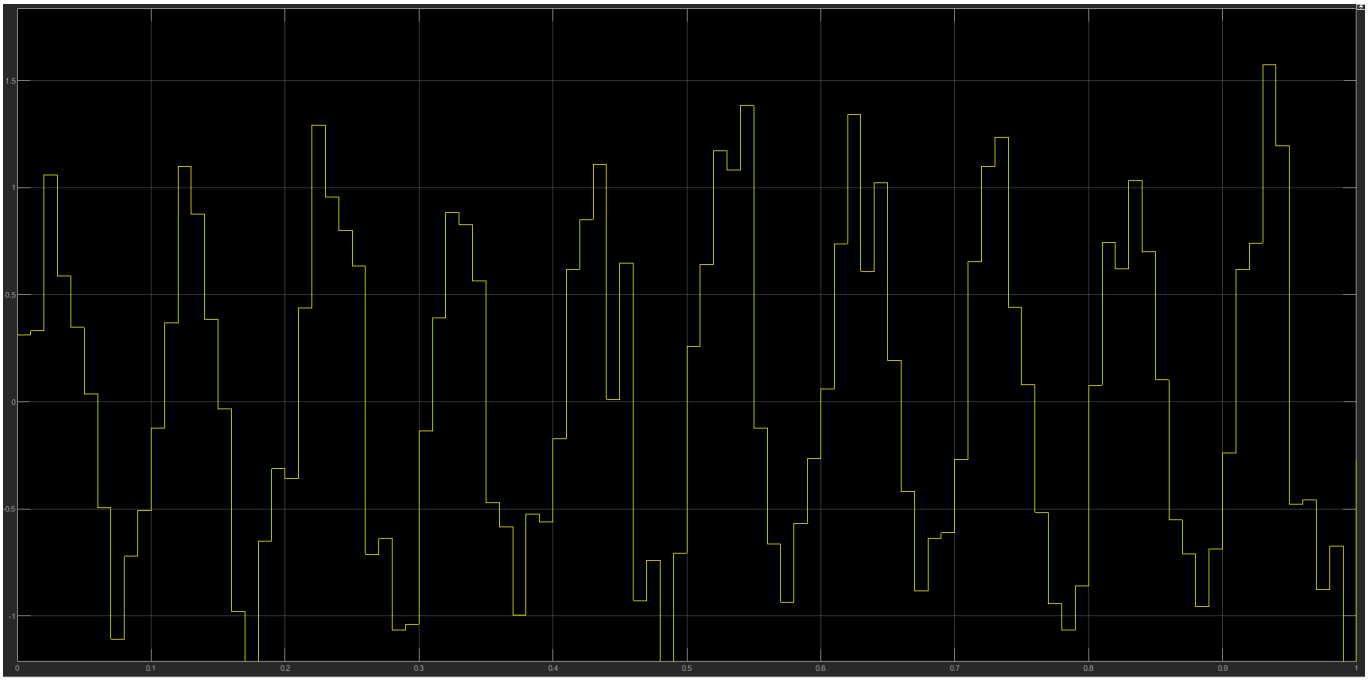
- When the stop time is set to 'inf' (infinite), the simulation is running forever until the simulation is stopped by the user. The resulting signal for the 10mS sampled signal is the same signal shown in (5) above.

(8) Return the Stop time to 1 sec, and change the simulation setup to include Noise as shown in the diagram below. The Circular component is a Sum block which adds the two input signals together. Use a Noise Power of 1mW and a sample time of 10ms. Run the simulation and sketch the resulting scope display. Does the display change if you re-run the simulation? What is the effect of changing the Seed? Change the Sample Time to 1ms and comment on the effect. Also comment on the effects of changing the Noise Power level.

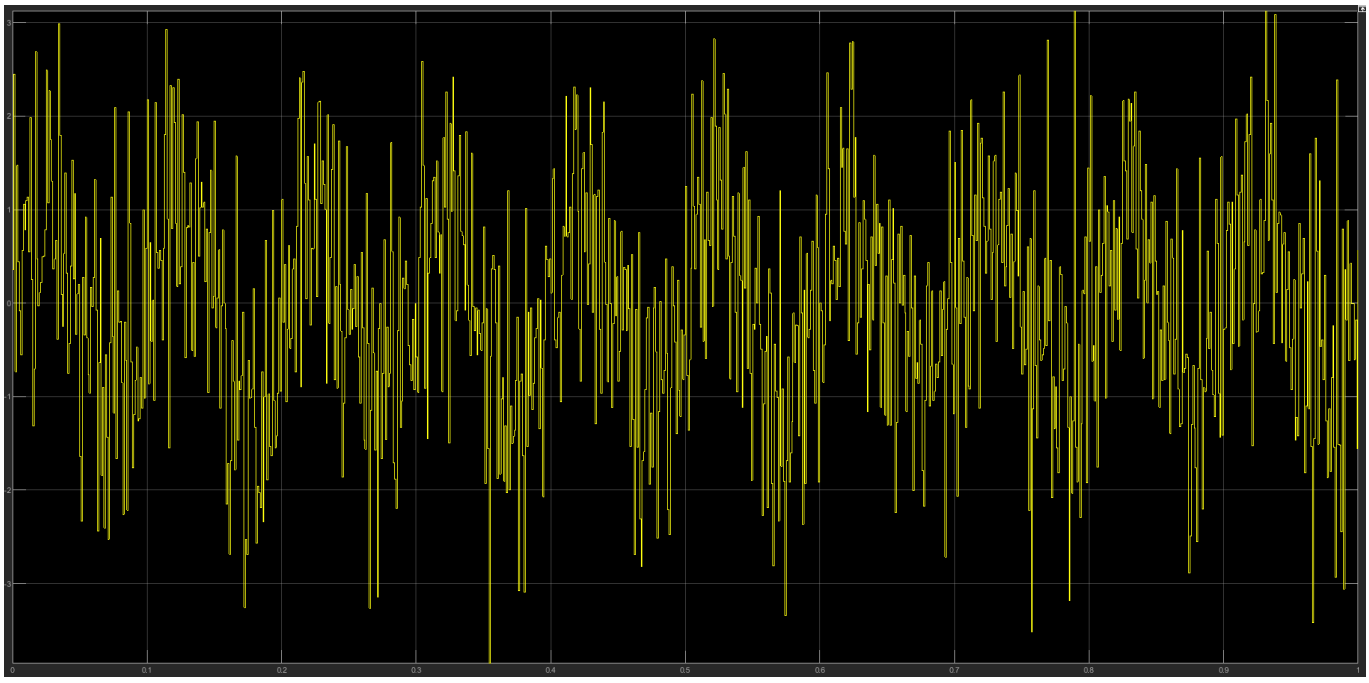




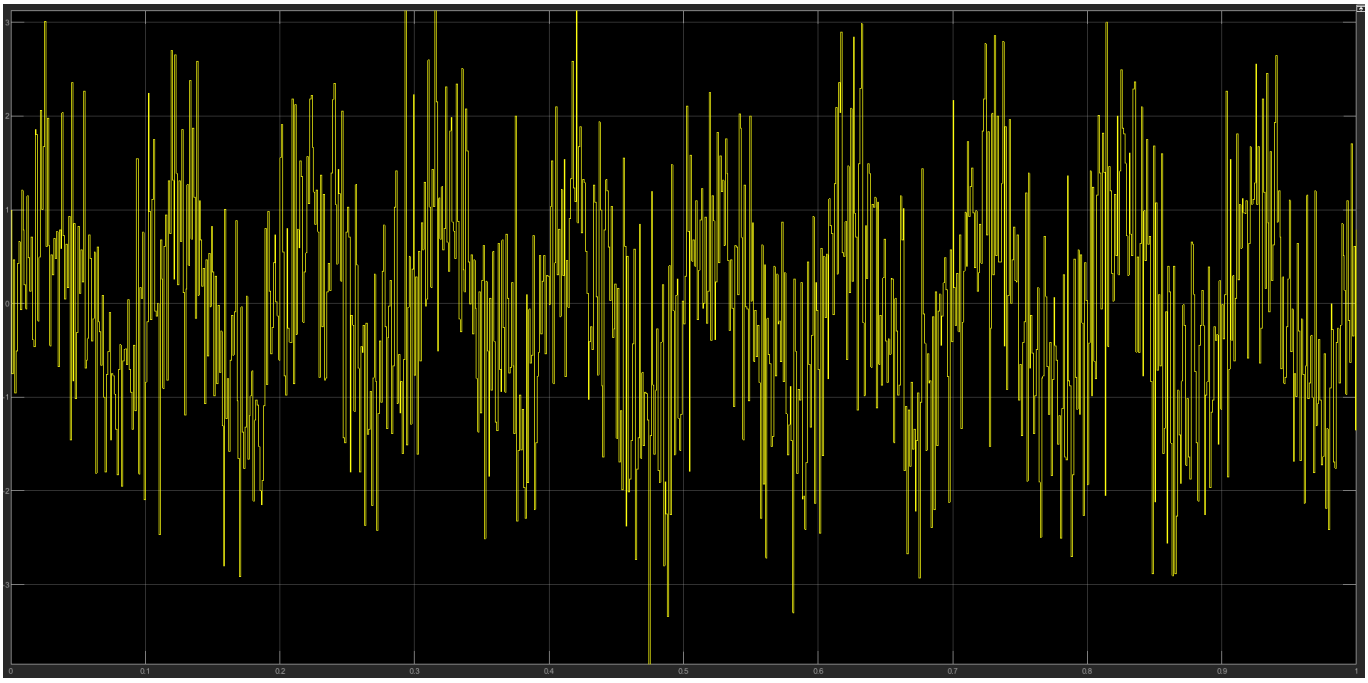
Signal a: seed is 23341, sample time is 10mS and noise power is 1mW.



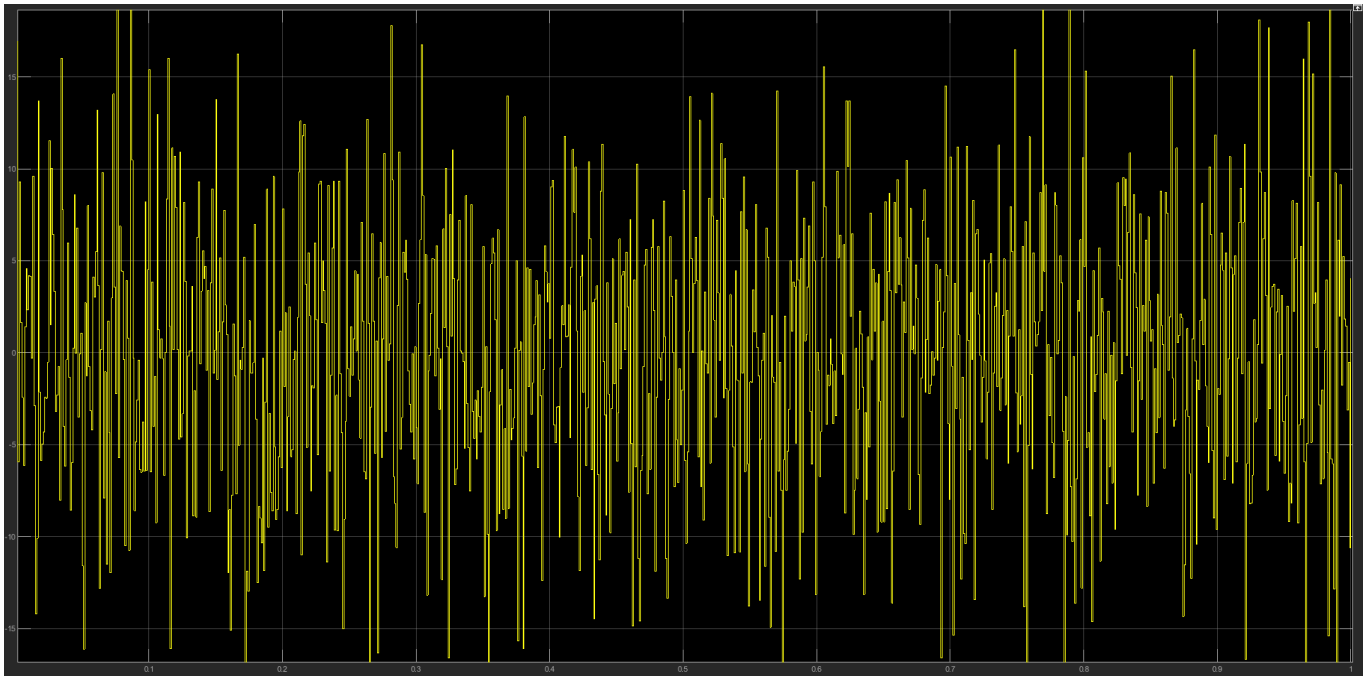
Signal b: seed is 46682 (doubled), sample time is 10mS and noise power is 1mW.



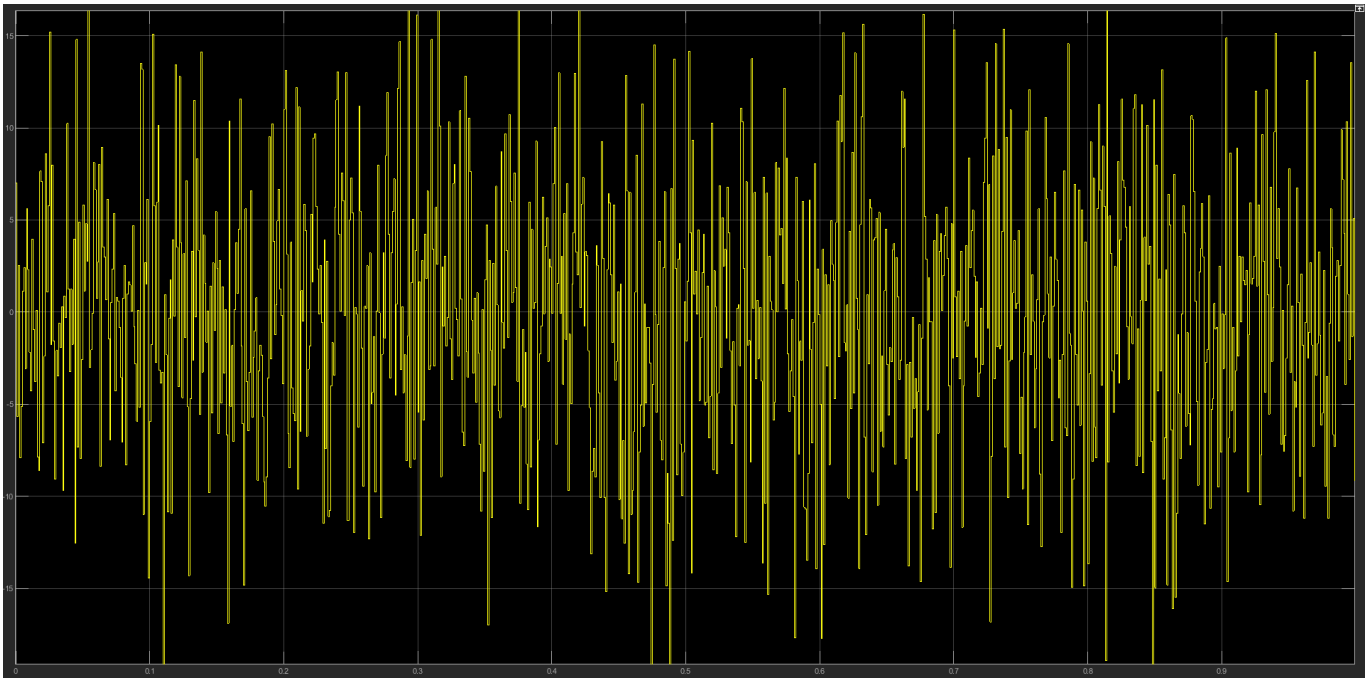
Signal c: seed is 23341, sample time is 1mS and noise power is 1mW.



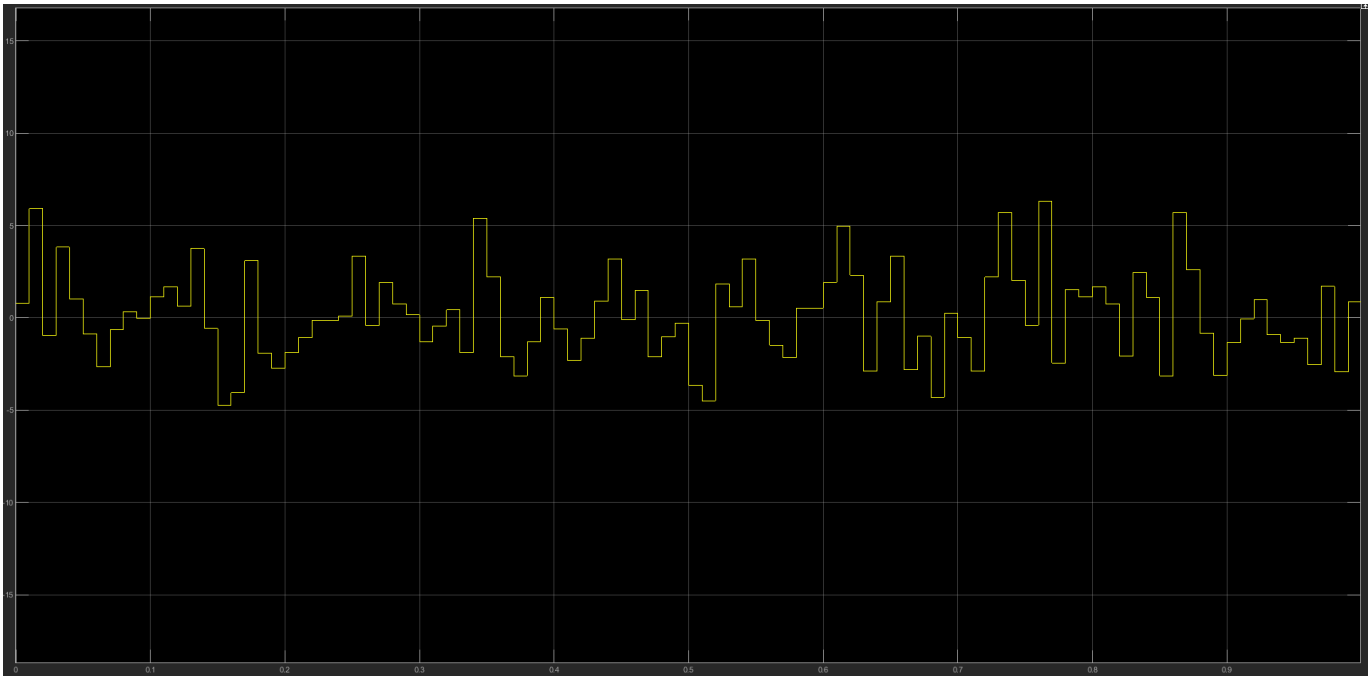
Signal d: seed is 46682, sample time is 1mS and noise power is 1mW.



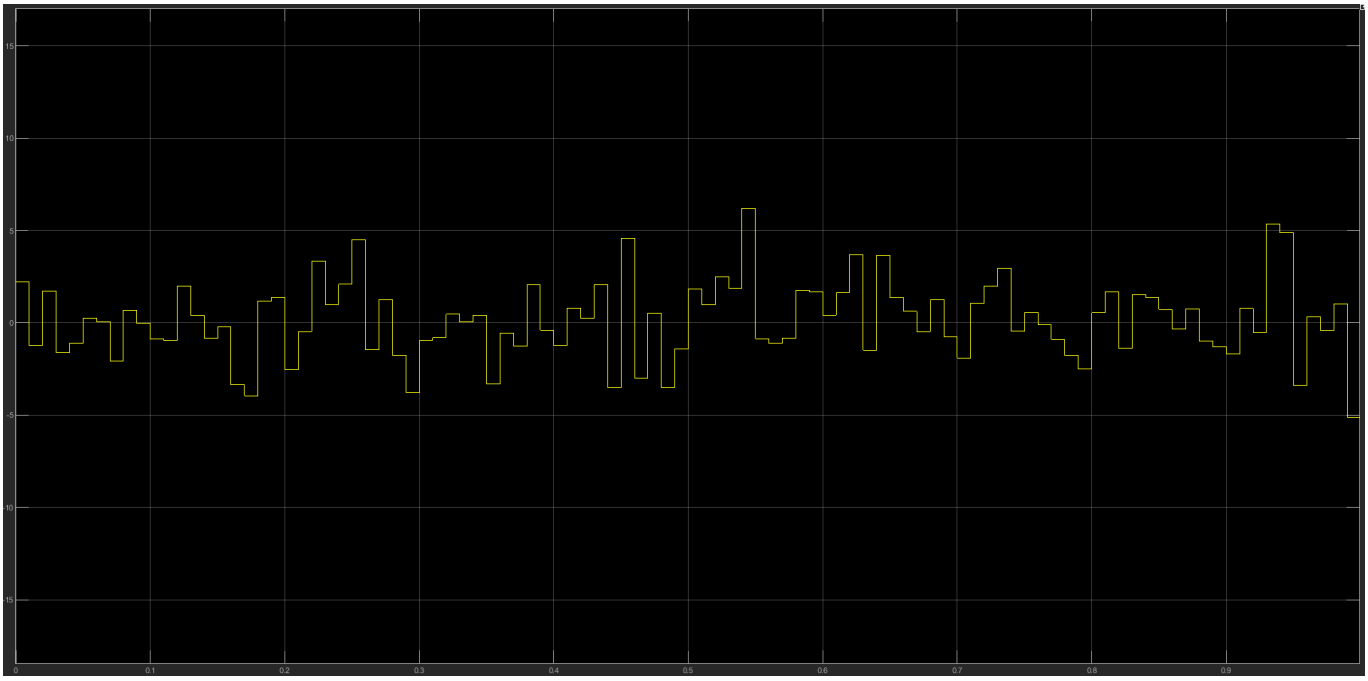
Signal e: seed is 23341, sample time is 1mS and noise power is 50mW.



Signal f: seed is 46682, sample time is 1mS and noise power is 50mW.



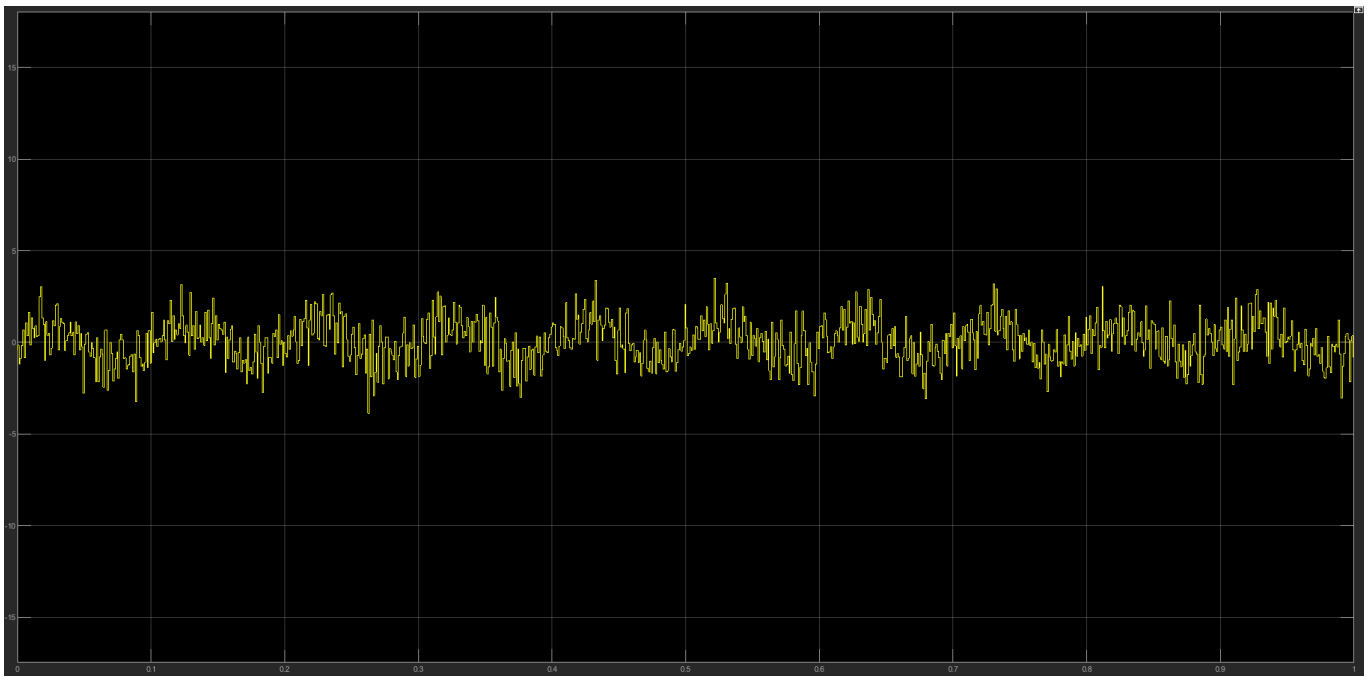
Signal g: seed is 23341, sample time is 10mS and noise power is 50mW.



Signal h: seed is 46682, sample time is 10mS and noise power is 50mW.

Overall, for any of the signals shown above with their unique parameters set, nothing drastically changes if the simulation is re-run a second time. On the other hand, as you can see above, depending on the characteristics set, the resulting signal on the scope display will vary. For all the signals, when the seed is changed, the pattern of noise is changed as well. For example, when the seed is doubled, there are more patterns of noise simulated. This is shown on the scope display in signals b, d, f and h. When the sample time is changed from 10mS to 1mS, the effect is drastic. As seen above in signals c to f no matter what noise power or seed, the reduced sample time would normally work and the reconstruction of the signal would be close to the original signal if there was no noise at all (as seen in (6) above). Since there is noise present, it completely changes the way the signal is reconstructed and with the faster sample time, the reconstruction of the signal shown on the scope display is virtually unrecognizable and is not the same as the original signal. When the noise power is changed, it creates even more problems with the reconstruction of the signal and no matter the seed, it will still be worse unlike if it were lower. This is shown in signals e to f. Unlike in signals g to h, even though the noise power is 50mW and despite the seed, the reconstruction of the signal on the scope display doesn't look as bad as the previous signals because the sample time is 10mS.

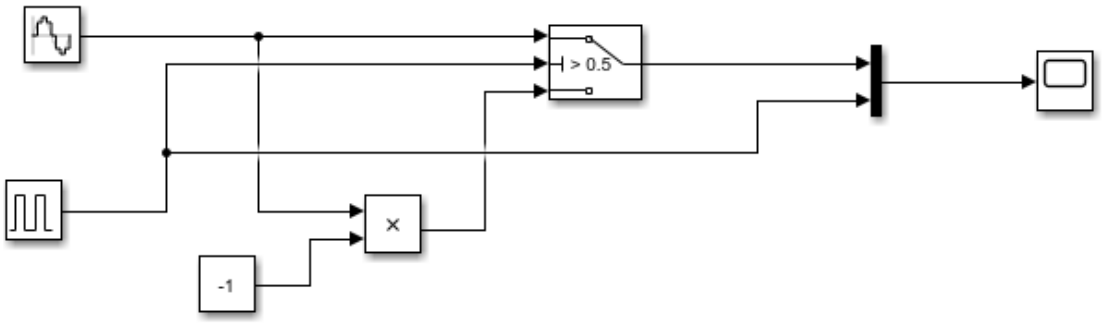
(9) Leave the Sample Time at 1ms and Noise Power at 1mW. Once again adjust the Stop time to 'inf'. Run the simulation. Comment on your observations.



Signal shown above is same signal that is in (8), "signal c" with the same parameters set. The only difference with this signal is that the stop time is set to 'inf' (infinite) so that the simulation can run forever until the user stops it. When the simulation is running and the stop time is set to 'inf', we can see the signal as if it were in real-time on the scope in Simulink and we see how the noise interferes with the reconstruction of the sine wave on the scope display. As I would expect, even though the sample time would normally be adequate to reconstruct the original sine wave signal (as shown above in (6)), the addition of noise proves that the signal cannot be reconstructed correctly. The result is what you see in the above screenshot. What is shown above in (6) is somewhat unrealistic because in the real world, there is always noise, even though we try to mitigate it, it will be always present.

PART 3 PBSK

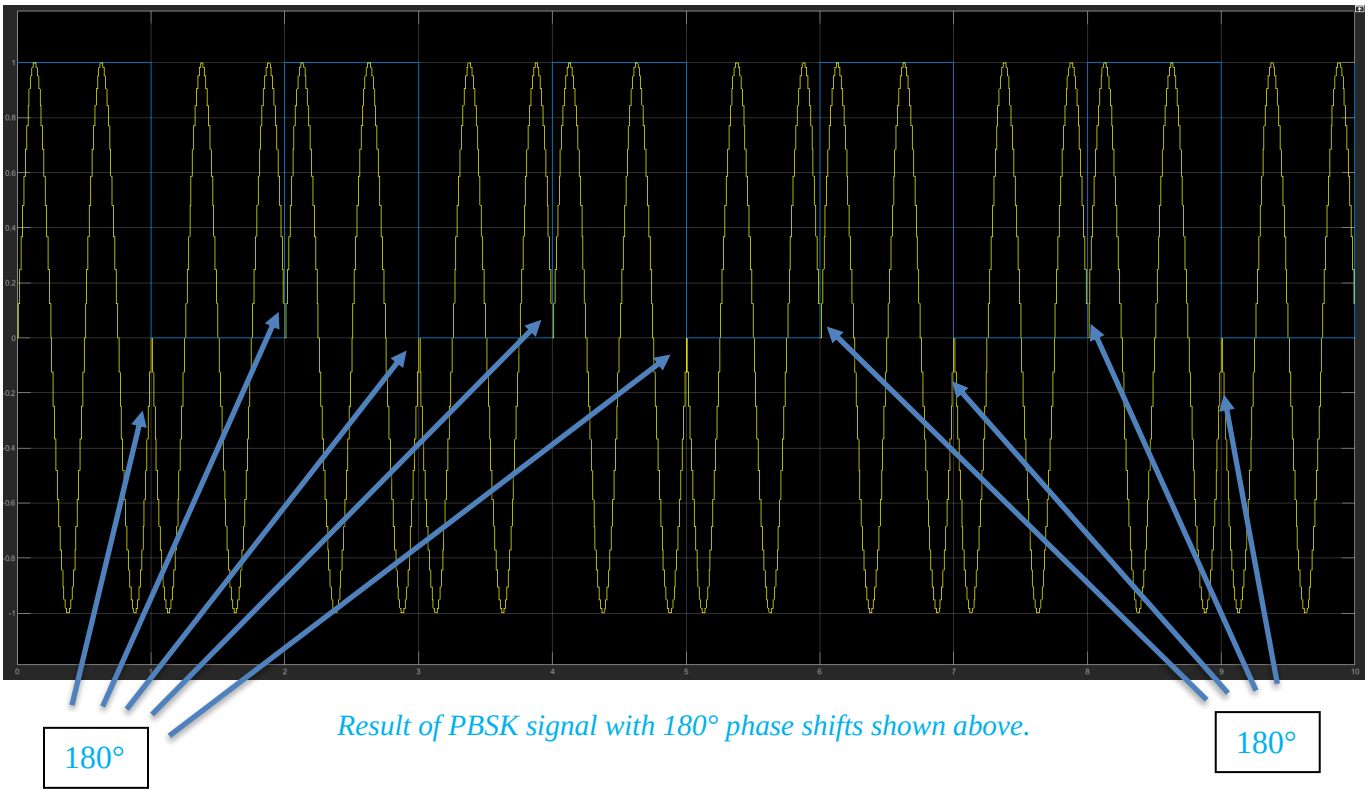
Build the following block diagram



a. Set Clock generator period 2Sec, and duty cycle 50%
b. Set sin wave Amplitude 1V, 2Hz, and sampling rate 0.01sample/sec
c. Add constant (-1) and, multiplier (product) from math operations and switch from signal routing. Set the threshold for switch 0.5.
d. Change the setting the type to fixed-step and fundamental sample time to 0.01.
e. From the signal router add a mux to be able to add two signals to the scope.
f. Question: explain how the circuit works and what you are expecting to see on the Scope.

- As shown in the title above, the circuit above is a simple PBSK circuit. The way this PBSK is implemented in MATLAB's Simulink is by using the components shown above in Simulink (a sine wave generator, a pulse generator, a constant value, a multiplier (product), a switch, a multiplexer and a scope). The pulse generator is used as one of the input signals to the switch which is used to compare the threshold level with another signal (a sine wave) and another signal (another sine wave) with opposite phase and same frequency (this second signal is generated from the multiplier shown above; it takes the original sine wave used as the first input to the switch and multiplies it with a constant value of -1. The result is a new sine wave with same frequency but opposite phase). The output of the comparison done by the switch is MUXed with the original pulse signal from the pulse generator and the output of the MUX is fed to the scope. When the pulse signal falls to 0, the result would be a sine wave with a 180° phase change and when the pulse signal goes to 1, the result would be the same sine wave but with another 180° phase change. What is expected to be shown on the scope is a typical PBSK signal where two logic states (1 and 0) are represented as two different phases but with the same frequency (this signal will be a sine wave). A square wave will be shown as well to compare it with the generated sine wave signal. The expected output on the scope display can be found on the next page.

g. Run the Simulink and compare the result on the scope with your expectation in f.



Result of PBSK signal with 180° phase shifts shown above.

The result shown above is what I had expected as I had explained in f on the previous page. Here, it is shown that there are 180° phase shifts to indicate two logic states (1 and 0) from the yellow sine wave. The frequency stays the same throughout but the phase changes in order to indicate a different logic level (0 or 1). The addition of the original pulse signal, shown above in blue, shows that each time the logic state changes (from 1 to 0 or from 0 to 1), proves that there is a 180° phase shift. This is what a typical PBSK would look like and how it would behave in real life.