How to generate AWGN noise in Matlab/Octave(without using in-built awgn function)

Mathuranathan Viswanathan
http://www.gaussianwaves.com

June 16, 2015

1 Introduction

 More such tricks can be found in the ebook : Simulation of Digital communication systems using Matlab by Mathuranathan Viswanathan

2 AWGN - the in-built function

Matlab/Octave communication toolbox has an inbuilt function named - $awgn()$ with which one can add an Additive Gaussian White Noise to obtain the desired $Signal-to-NoiseRatio$ (SNR). The main usage of this function is to add AWGN to a clean signal (infinite SNR) in order to get a resultant signal with a given SNR (usually specified in dB). This usage is often found in signal processing/digital communication applications. For example, in Monte Carlo simulations involving modeling of modulation/demodulation systems, the modulated symbols at the transmitter are added with a random noise of specific strength, in order to simulate a specific E_b/N_0 or E_s/N_0 point.

The function $y = awgn(x, SNR,' measured')$, first measures the power of the signal vector x and then adds white Gaussian Noise to x for the given SNR level in dB. The resulting signal y is guaranteed to have the specified SNR.

3 Custom function to add AWGN noise

If you do not have the communication toolbox, or if you would like to mimic the in-built AWGN function in any programming language, the following procedure can be used.

1) Assume, you have a vector x to which an AWGN noise needs to be added for a given SNR (specified in dB).

2) Measure the power in the vector x [1]

$$E_s = \frac{1}{L}\sum_{i=0}^{L-1} |x[i]|^2; \quad where \quad L = length(x) \tag{1}$$

3) Convert given SNR in dB to linear scale (SNR_{lin}) and find the noise vector (from Gaussian distribution of specific noise variance) using the equations below

$$noise = \begin{cases} \sqrt{\frac{E_s}{SNR_{lin}}} * randn(1, L) & \text{if x is real} \\ \\ \sqrt{\frac{E_s}{2*SNR_{lin}}} * [randn(1, L) + j * randn(1, L)] & \text{if x is complex} \end{cases}$$

4) Finally add the generated noise vector to the signal x

$$y = x + noise \tag{2}$$

3.1 The custom function

The custom function written in Matlab, that mimics the *awgn* function is given below. It can be easily ported to Octave.

```
%Function to add AWGN to the given signal
%Authored by Mathuranathan Viswanathan
%How to generate AWGN noise in Matlab/Octave by Mathuranathan Viswanathan
%is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 4.0
%International License.
%You must credit the author in your work if you remix, tweak, and build upon
%the work below
function [y,n] = add_awgn_noise(x,SNR_dB)
    %[y,n]=awgn_noise(x,SNR) adds AWGN noise vector to signal 'x' to generate a
    %resulting signal vector y of specified SNR in dB. It also returns the
    %noise vector 'n' that is added to the signal x
    L=length(x);
    SNR = 10^(SNR_dB/10); %SNR to linear scale
    Esym=sum(abs(x).^2)/(L); %Calculate actual symbol energy
    N0=Esym/SNR; %Find the noise spectral density
    if(isreal(x)),
        noiseSigma = sqrt(N0);%Standard deviation for AWGN Noise when x is real
        n = noiseSigma*randn(1,L);%computed noise
    else
        noiseSigma=sqrt(N0/2);%Standard deviation for AWGN Noise when x is complex
        n = noiseSigma*(randn(1,L)+1i*randn(1,L));%computed noise
    end
    y = x + n; %received signal
end
```

Listing 1: *add_awgn_noise.m*: Custom function to add AWGN noise to a signal vector

3.2 Comparison and Testing

Let's cross check the results obtained from the above function with that of the standard in-built *awgn* function in Matlab. Testing and comparison is done using two test waveforms - 1) sawtooth waveform (represented by a vector containing only real numbers) , 2) A complex sinusoidal waveform (vector having both real and imaginary part). The testing below is done for SNR=5 dB. Users are advised to test the function for various ranges of SNRs.

Test code is shown next and the results for sawtooth waveform (the input signal x is a vector of real numbers) are shown in Figures 1 and 2. Test of linearity in Figure 2 indicates that the results from the custom function matches with that of the in-built function.

```
SNR_dB = 5; %Signal to noise ratio in dB
t = 0:.1:10;%time base
x = sawtooth(t); % Create sawtooth signal.

%Method 1: using custom function 'add_awgn_noise
rng('default');%set the random generator seed to default (for comparison only)
%If the seed above is not set, the results may vary from run-run
y_custom = add_awgn_noise(x,SNR_dB); %our custom function

%Method 2:Using in-Built Function (needs comm toolbox)
rng('default');%set the random generator seed to default (for comparison only)
y_inbuilt = awgn(x,SNR_dB,'measured'); % Add white Gaussian noise.

%Plotting results
figure; subplot(1,2,1);
plot(t,x,'b',t,y_custom,'r') % Plot both signals.
legend('signal','signal with noise');
xlabel('timebase');ylabel('y_{custom}');
title('custom add\_awgn\_noise function')

subplot(1,2,2); plot(t,x,'b',t,y_inbuilt,'r') % Plot both signals.
legend('signal','signal with noise');
xlabel('timebase');ylabel('y_{inbuilt}');
```

```
title('Inbuilt awgn function')

%check for visual linearity between custom function and AWGN inbuilt function
figure; plot(y_inbuilt,y_custom); %check for linearity between custom function and AWGN
    inbuilt function
title('output of custom function Vs in-built awgn fn');
xlabel('y_{inbuilt}');ylabel('y_{custom}');
```

Listing 2: Test code for comparison with inbuilt awgn command

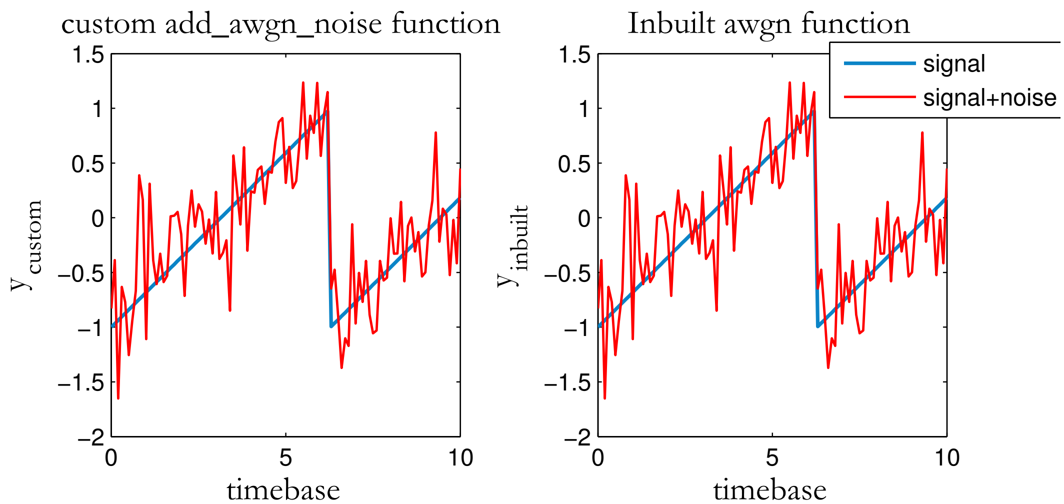

Figure 1: Sawtooth waveform with AWGN noise generated using custom method and Matlab's in-built method

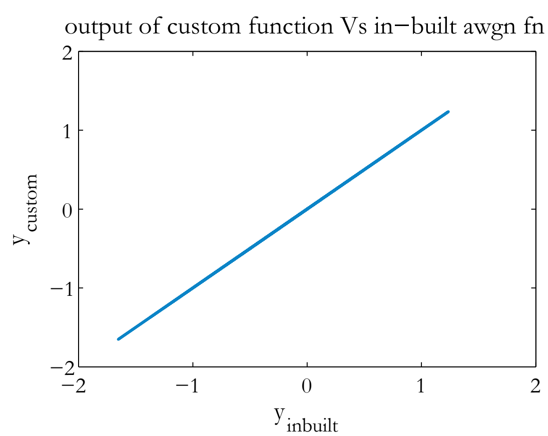

Figure 2: Output from add_awgn_function Vs inbuilt awgn function

Test code is shown next and the results for a complex sinusoida waveform (the input signal x is a vector of complex numbers) are shown in Figures 3 and 4 . Test of linearity in Figure 4 indicates that the results from the custom function matches with that of the in-built function.

```
SNR_dB = 5; %Signal to noise ratio in dB
t = 0:pi/8:6*pi;%time base
x = sqrt(2)*(sin(t)+1i*sin(t)); % Create complex sinusoidal signal.
```

```
%Method 1: using custom function 'add_awgn_noise'
rng('default');%set the random generator seed to default (for comparison only)
%If the seed above is not set, the results may vary from run-run
y_custom = add_awgn_noise(x,SNR_dB); %our custom function

%Method 2:Using in-Built Function (needs comm toolbox)
rng('default');%set the random generator seed to default (for comparison only)
y_inbuilt = awgn(x,SNR_dB,'measured'); % Add white Gaussian noise.

%Plotting results
figure;  subplot(1,2,1);
plot(t,abs(x),'b',t,abs(y_custom),'r') % Plot both signals.
legend('signal','signal with noise');
xlabel('timebase');ylabel('y_{custom}');
title('custom add\_awgn\_noise function')

subplot(1,2,2); plot(t,abs(x),'b',t,abs(y_inbuilt),'r') % Plot both signals.
legend('signal','signal with noise');
xlabel('timebase');ylabel('y_{inbuilt}');
title('Inbuilt awgn function')

%check for visual linearity between custom function and AWGN inbuilt function
figure;plot(abs(y_inbuilt),abs(y_custom));
title('output of custom function Vs in-built awgn fn');
xlabel('|y_{inbuilt}|');ylabel('|y_{custom}|');
```

Listing 3: Test code for comparison with inbuilt awgn command

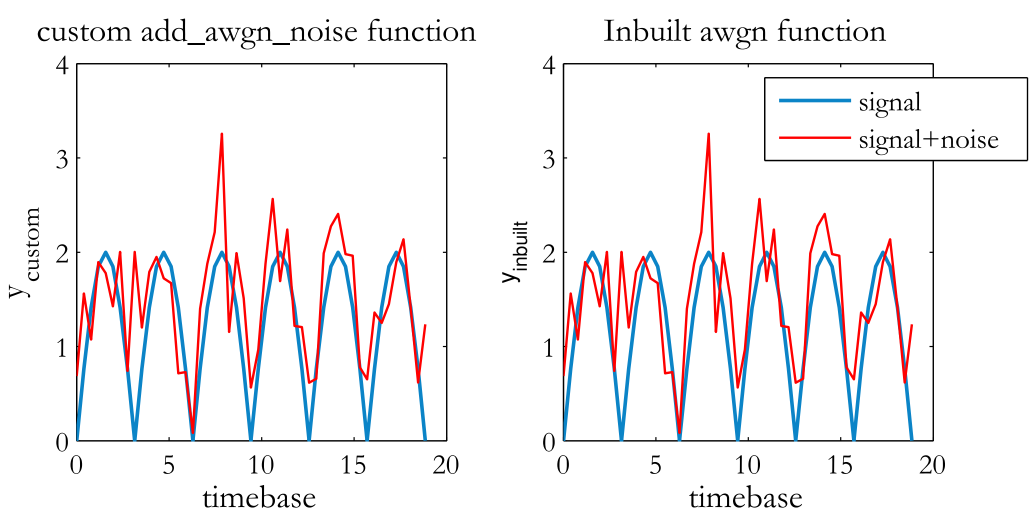

Figure 3: Complex Sinusoidal waveform with AWGN noise generated using custom method and Matlab's in-built method

References

[1] Mathuranathan Viswanathan, *Power and Energy of a signal*, gaussianwaves.com, December 2013

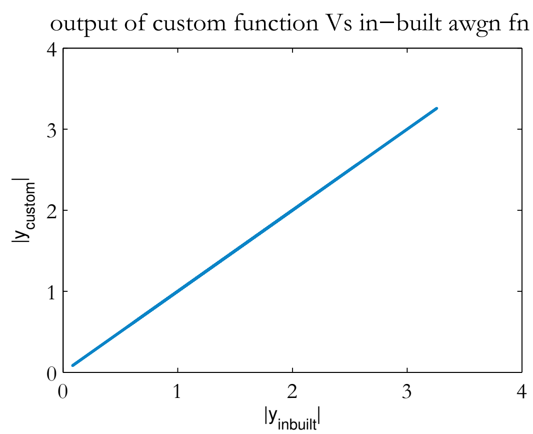

Figure 4: Output from add_awgn_function Vs inbuilt awgn function