

Estimating and Graphing the Amplitude Probability Distribution Function of Complex-Baseband Signals

Robert J. Achatz, Michael G. Cotton, and Roger A. Dalke
US Department of Commerce, National Telecommunications and Information
Administration
Institute for Telecommunication Sciences
Boulder, CO USA

The amplitude probability distribution (APD) function has been found to be useful in characterizing signals and evaluating effects of interference on victim receivers. The following discussion defines the APD mathematically and demonstrates how it can be estimated and plotted with the MATLAB[®] language.

1. Estimating the APD

The mathematics and terminology of the APD are based on the concept of random variables. A random variable is a function, X , which assigns a real number, $X(v)$, to every element, v , of a sample space. In our case, the random variable assigns a real number representing the amplitude of a complex-baseband signal drawn from a sample space via signal simulation or measurement. The APD is expressed as

$$F(a) = P(X(v) > a) ,$$

or more commonly

$$F(a) = P(X > a) ,$$

where a is an amplitude value and $P()$ means probability.

A discrete estimate of the APD can be obtained from a finite set of samples. This is accomplished by sampling the complex-baseband signal N times, converting these samples to the amplitudes

$$a(n) , \quad n = 1, \dots, N ,$$

ordering the amplitudes from smallest to largest

$$a[n] , \quad n = 1, \dots, N ,$$

where the brackets distinguish the ordered amplitudes from the unordered amplitudes, and computing

$$P(X > a[n]) = 1 - n/N .$$

The routine “[a, p] = apd(s)” (shown in Appendix A) estimates the APD from N amplitude samples. Elements in the vector “s” are checked to make sure they are amplitudes, i.e., that they are real and positive. “s” is ordered from smallest to largest via

$$“a = \text{sort}(s)” ,$$

where “sort” is a MATLAB function, and corresponding probabilities are computed via

$$“p = 1-(1 : N)/N” .$$

2. Plotting the APD

APDs are typically displayed on a Rayleigh graph whose axes are transformed by functions that linearize the APD of the Rayleigh-distributed amplitudes of complex Gaussian noise. Derivation of these functions begins with the expression of the APD of a Rayleigh-distributed random variable

$$P(X>a) = \exp(-a^2/\sigma^2) ,$$

where σ represents the standard deviation of zero-mean complex noise. This function is linearized by the natural logarithm function and converted to decibels by

$$10\log_{10}(a^2) = 10\log_{10}(\sigma^2) + 10\log_{10}(-\ln(P(X>a))) .$$

Graphical coordinates, x and y, are introduced by

$$y = 10\log_{10}(a^2)$$

and

$$x = 10\log_{10}(-\ln(P_0)) - 10\log_{10}(-\ln(P(X>0))) ,$$

where P_0 is the probability at the graph origin.

The x-axis is in percentage units and labeled “percentage of time ordinate is exceeded” or “percent exceeding ordinate”. The x-axis probabilities generally range from 0.0001 to 99 percent for problems associated with modern receiver design.

Signals are often normalized. Common normalization factors are the root-mean-square (rms) voltage of the signal or the rms voltage of the white Gaussian noise introduced by the measurement system. If normalized by the rms voltage of the signal, the y-axis is labeled “dB relative to rms voltage” or “dB relative to average power”. If normalized by the average power of the white Gaussian noise introduced by the measurement system, the y-axis is labeled “dB relative to measurement system average noise power” or “dB relative to kTB ”. Otherwise, units appropriate to the samples such as dBV or dBW are used.

The routine “plotapd(a,p)” plots the APD on a Rayleigh graph. The routine begins by checking for zero amplitudes and probabilities that will generate log function error messages. Zero amplitudes are possible for signals simulated without added noise. The largest or peak amplitude of the APD estimate is assigned zero probability. “Plotapd” circumvents these problems by replacing the zero value with the MATLAB constant “not a number” (NaN). The MATLAB “log” or “log10” functions will not generate error messages when operating on NaNs and the MATLAB “plot” routine will not plot points in which abscissa or ordinate is NaN. The remainder of the routine transforms data and axes to the graphical coordinate system using the equations for x and y above. Y-axis labeling is dependent on how or if the samples were normalized and therefore is left outside the plotting routine.

3. Testing the Routines

The routine “testapd” demonstrates the use of the “apd” and “plotapd” routines by checking the median and mean values of two test signals (corresponding plots are shown in Appendix B). The first test signal is a ten-thousand element vector of linearly increasing voltages ranging from 0 to 9999. The median (i.e., the amplitude exceeded by 50 percent of the samples) of this signal is 5000 V or 74.0 dBV. The second test signal is a one-million element vector of complex Gaussian noise voltages with zero mean and a variance of 2 V^2 . The rms voltage is equal to the square root of the variance or 3.0 dBV. The corresponding probability is equal to $1/e$ or 0.367 and expressed on the graph as 36.7%.

4. Additional Comments

Several important amplitude statistics, including rms voltage or average power, can be computed with the information found in the APD. However, the APD cannot provide information concerning the signal time-domain behavior without other a priori information such as knowing that it is a periodically pulsed signal.

Additionally, the APD is strongly affected by the type of band-limiting filter; the APD estimate accuracy and precision are dependent on the sample interval and number of samples used. Consequently, these parameters should be documented with the APD graph.

Appendix A. MATLAB Code

A.1. testAPD.m

```
% testAPD
% demonstrates how to use apd(s) and plotapd(a,p).

% ramp
N = 10^4;
v = [0:N-1];
[a,p] = apd(abs(v));
plotapd(a,p);
title(sprintf('Simulated ramp, 0-9999 V, N = 10^4, Peak %f dBV',20*log10(a(N))))
ylabel('dBV')
pause

%complex Gaussian noise
N = 10^6;
v = randn(1,N) + i*randn(1,N);
[a,p] = apd(abs(v));
plotapd(a,p);
title(sprintf('Simulated complex Gaussian noise, variance 2 V^2, N = 10^6, Peak %f V',20*log10(a(N))))
ylabel('dBV')
```

A.2. apd.m

```
function [a, p] = apd(s);
% [a, p] = apd(s) estimates the amplitude probability distribution function.
%
% input parameters:
% s = amplitude samples
%
% return variables:
% a = ordered amplitudes
% p = probability that the ordered ampitude is exceeded

if isreal(s) & min(s)>=0
    a = sort(s);
    N = length(a);
    p = 1 - [1:N]/N;
else
    disp('Input values must be amplitudes i.e. real and positive values.');
```

A.3. plotapd.m

```
function plotapd(a,p)
% plots (a,p) pairs representing the APD on Rayleigh graph.
%
% input variables:
% a = ordered amplitudes
% p = probability of ordered amplitude

% x-axis labels (labels must be same length)
xticklabel = ['0.0001;' 0.01;' 0.1;' 1;' 5;' 10;' 20;' 30;' 40;' 50;' 60;'
70;' 80;' 90;' 95;' 98;' 99'];
ptick = [0.0001 .01 0.1 1 5 10 20 30 40 50 60 70 80 90 95 98 99]/100;
porigin = ptick(1);

% replace 0 valued amplitudes with NaN to avoid logarithm error messages
if min(a) == 0
    idx = find(a==0);
    a(idx) = NaN;
end

% replace peak amplitude 0 probability with NaN to avoid logarithmic error messages
p(length(a)) = NaN;

% map a and p and plot
x = 10.0*log10(-log(porigin)) - 10.0*log10(-log(p));
y = 20*log10(a);
plot(x,y,'LineWidth',2);
grid

% customize x axis
xlabel('percent exceeding ordinate');
xtick = 10.0*log10(-log(porigin))-10.0*log10(-log(ptick));
xLim([min(xtick) max(xtick)]);
set(gca,'XTick',xtick);
set(gca,'XTickLabel',xticklabel);

return
```

Appendix B. Example APD Plots

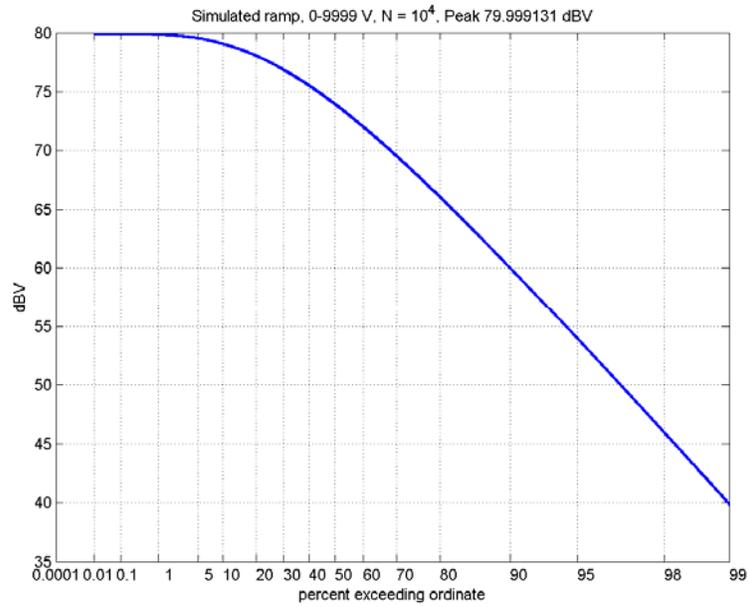


Figure B.1. APD of a ramp function increasing from 0 to 9999 V.

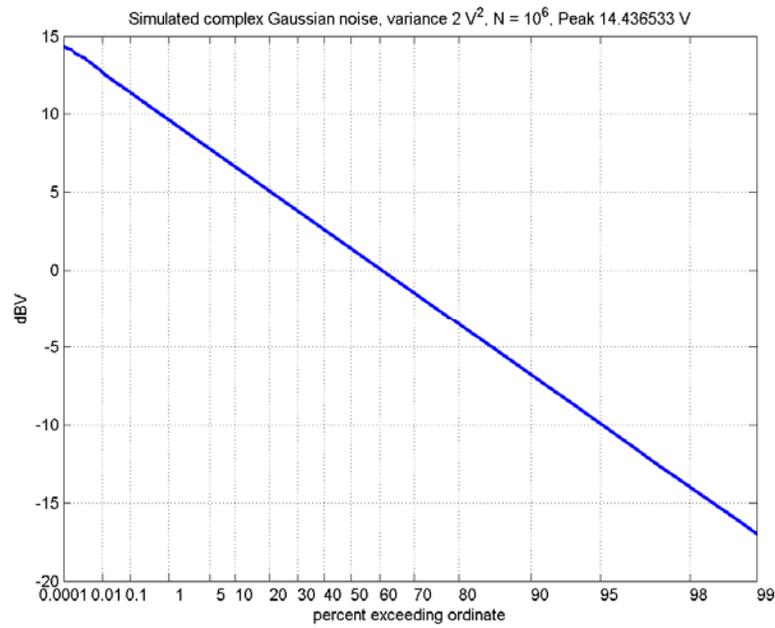


Figure B.2. APD of complex Gaussian noise with mean = 0 V and variance = $2 V^2$.