Discrete-Time Gaussian Fourier Transform Pair, and Generating a Random Process with Gaussian PDF and Power Spectrum

Mark A. Richards

October 3, 2006

Updated April 5, 2010

1 Gaussian Transform Pair in Continuous and Discrete Time

The Fourier transform of a continuous-time Gaussian function of variance σ^2 is also a Gaussian shape, but with variance $1/\sigma^2$ [1]:

$$w_c(t) = \frac{1}{\sigma\sqrt{2\pi}} e^{-t^2/2\sigma^2} \quad \overset{\Im}{\leftrightarrow} \quad W_c(\Omega) = e^{-\sigma^2\Omega^2/2} = e^{-\Omega^2/2\left(1/\sigma^2\right)} \tag{1}$$

A discrete-time Gaussian sequence is created by sampling the continuous-time Gaussian $w_c(t)$ at a sampling interval of T:

$$w[n] \equiv w_c(nT) = \frac{1}{\sigma\sqrt{2\pi}} e^{-n^2T^2/2\sigma^2} \tag{2}$$

The effect of sampling is to scale and alias the spectrum, giving the discrete-time Fourier transform (DTFT) [2]

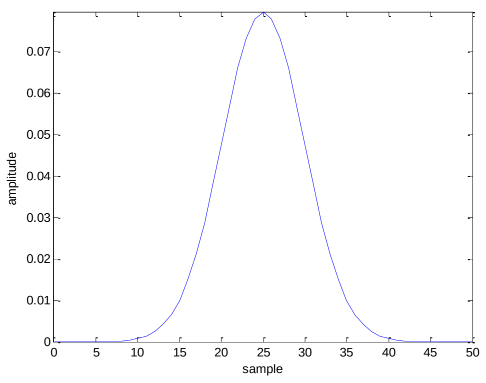
$$W(\omega) = \frac{1}{T}\sum_{k=-\infty}^{+\infty} W_c\left(\frac{\omega - 2\pi k}{T}\right) \tag{3}$$

If the standard deviation of $W_c(\Omega)$ is less than $\pi/3T$ ($\pi/4T$ is even better), then only the k=0 term of (3) will be significant in the interval $-\pi \le \omega \le \pi$. Consequently, there will be no significant overlap of terms and $W(\omega)$ will exhibit a Gaussian shape. If we also arbitrarily set T = 1, we obtain a discrete-time Gaussian Fourier transform pair:

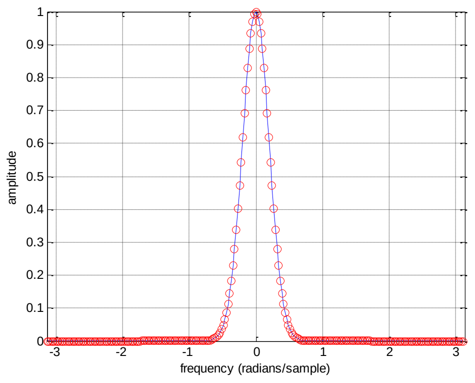
$$w[n] = \frac{1}{\sigma\sqrt{2\pi}} e^{-n^2/2\sigma^2} \quad \overset{\Im}{\leftrightarrow} \quad W(\omega) = e^{-\sigma^2\omega^2/2} = e^{-\omega^2/2\left(1/\sigma^2\right)} \tag{4}$$

This result is valid if the standard deviation $1/\sigma$ of $W(\omega)$ is less than $\pi/3$ radians per sample, meaning that the standard deviation σ of $w[n]$ must be at least 3 samples. As in the continuous time case, the variance in the frequency domain is the inverse of the time-domain variance.

Figure 1(a) illustrates a Gaussian impulse response with a standard deviation of 5 samples and a length of 51 samples. Part (b) of the figure overlays the 128-point DFT of the impulse response (blue curve) and the analytical expression from Eqn. (4) (green curve). The two curves are identical



(a)



(b)

Figure 1. (a) 51-point Gaussian impulse response with standard deviation σ = 5 samples. (b) DTFT of the impulse response in (a) (solid blue line), with the analytical expression for $W(\omega)$ from Eqn. (4) overlaid (red circles).

2 Creating a Gaussian Random Process with a Gaussian Power Spectrum

We will create the desired random process by passing a complex white Gaussian random process $x[n]$ through a linear filter with impulse response $h[n]$ and frequency response $H(\omega)$ as shown in Figure 2:

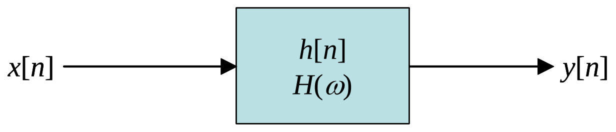


Figure 2. System for producing colored Gaussian noise.

We will assume the filter is FIR of length N. If the input random process has variance σ_x^2, then its autocorrelation function and power spectrum are [2]

$$\begin{aligned} s_{xx}[l] &= \sigma_x^2 \delta[l] \\ S_{xx}(\omega) &= \Im\{s_{xx}[l]\} = \sigma_x^2 \end{aligned} \tag{5}$$

The output random process will have a Gaussian probability density function (pdf) because each output sample is a linear combination of Gaussian-distributed input samples. The output variance, autocorrelation function, and power spectrum will be [2]

$$\sigma_y^2 = \sigma_x^2 \sum_{n=0}^{N-1} h^2[n]$$
$$s_{yy}[l] = s_{xx}[l] * s_{hh}[l] = \sigma_x^2 s_{hh}[l] \tag{6}$$
$$S_{yy}(\omega) = \Im\left\{s_{yy}[l]\right\} = \sigma_x^2 \left|H(\omega)\right|^2$$

In (6) we are being a little informal in using the notation s_{zz} for both the statistical autocorrelation function (in the case of s_{yy} and s_{xx}) and the deterministic autocorrelation function (s_{hh}); a similar statement applies to the notation for the power spectra, S_{zz}.

Suppose our goal is to have a Gaussian power spectrum with a standard deviation of σ_{HH}. From Eqn. (4), and avoiding any concerns on amplitude for the moment, the desired Gaussian power spectrum shape is

$$\left|H(\omega)\right|^2 = \exp\left(-\omega^2 / 2\sigma_{HH}^2\right) \tag{7}$$

It follows that the desired form for $|H(\omega)|$ and the corresponding time-domain Gaussian impulse response are

$$\begin{aligned} \left|H(\omega)\right| &= \exp\left(-\omega^2 / 4\sigma_{HH}^2\right) = \exp\left(-\omega^2 / 2\left(2\sigma_{HH}^2\right)\right) \\ h[n] &= \frac{\sigma_{HH}}{\sqrt{\pi}} \exp\left(-\sigma_{HH}^2 n^2\right) \end{aligned} \tag{8}$$

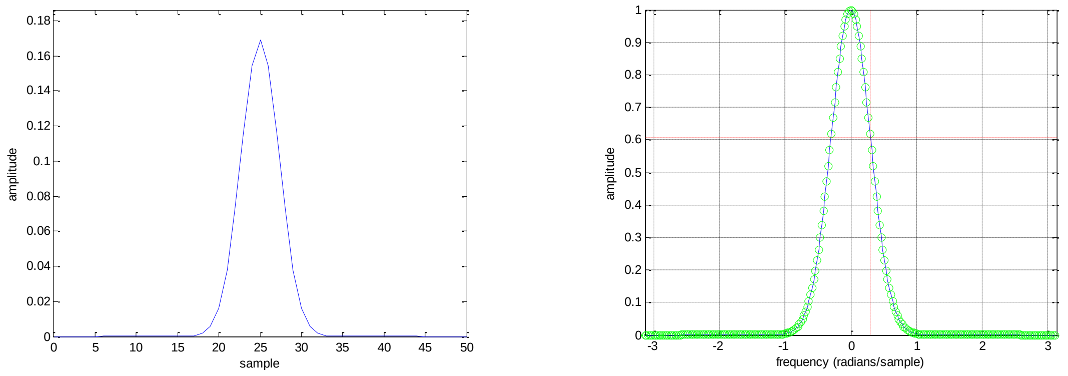
The required impulse response $h[n]$ is a Gaussian with standard deviation σ_{hh} = $1/\sqrt{2}\,\sigma_{HH}$.

Figure 3 shows an example created starting with a specification of σ_{HH} = 0.3 radians/sample and an impulse response length of N = 51 samples. The impulse response, shown in part (a) of the figure, was computed using Eqn. (8). Part (b) of the figure shows the magnitude-squared of the DTFT of $h[n]$, with the analytical expression of Eqn. (7) overlaid and marker lines indicating that the resulting power spectrum passes through the 1-sigma point (a value of exp(-1/2)) at the expected value of ω.

3 Summary and Example

To summarize, suppose we wish to create a discrete random process with the following characteristics:

- Complex-valued
- Zero mean
- Gaussian probability density function for each of the real and imaginary parts



(a) (b)

Figure 3. (a) Gaussian impulse response h[n] designed to give a power spectrum with standard deviation of 0.3 radians/sample. (b) Corresponding $|H(\omega)|^2$ (solid blue line) and overlaid analytical result (green circles).

- Total power σ_y^2, with half of the power in each of the real and imaginary parts of the process
- Gaussian power spectrum with variance σ_{HH}^2

The following procedure will achieve this result:

1. Form a Gaussian-shaped impulse response *h*[*n*] in the time domain with standard deviation $1/\sqrt{2}\,\sigma_{HH}$. The length *N* of the impulse response should be at least 6 standard deviations, and probably longer, to ensure that the impulse response tails are not significantly truncated. The amplitude scaling of *h*[*n*] is not important.
2. Create a complex, white, zero-mean Gaussian random process *x*[*n*] with variance $\sigma_x^2 = \sigma_y^2 / \sum_n h^2[n]$. Both the real and imaginary parts will have half of the total variance, *i.e.* they should be i.i.d.
3. Filter the process *x*[*n*] with the impulse response *h*[*n*] to create the output random process *y*[*n*]. Discard the transient samples of the convolution output; there are *N*-1 such samples at each of the beginning and end of the filter output.

The remainder of *y*[*n*] will have the desired properties.

Following is an example, created using the code in the Appendix, of an approximately 100,000 sample random process created with specifications of $\sigma_{HH} = 0.3$

radians and σ_y^2 = 0.005. A filter impulse response of length N = 51 samples was selected.

The Gaussian impulse response and the corresponding magnitude-squared frequency response $|H(\omega)|^2$ for this case were previously illustrated in Figure 3. Figure 4 shows the 100-bin histograms of the real and imaginary parts of the 100,000-sample filtered noise process $y[n]$, as well as theoretical pdfs. The fit of the theoretical pdfs to the histograms is excellent, and the match between the two channels indicates that the real and imaginary parts are i.i.d. Gaussians. Measurements of the mean and variance of $y[n]$ are ($4.7\text{x}10^{-4} + j5.6\text{x}10^{-4}$) and 0.005, respectively. Thus the variance is exactly as intended for this realization, and the mean is reasonably close to zero.

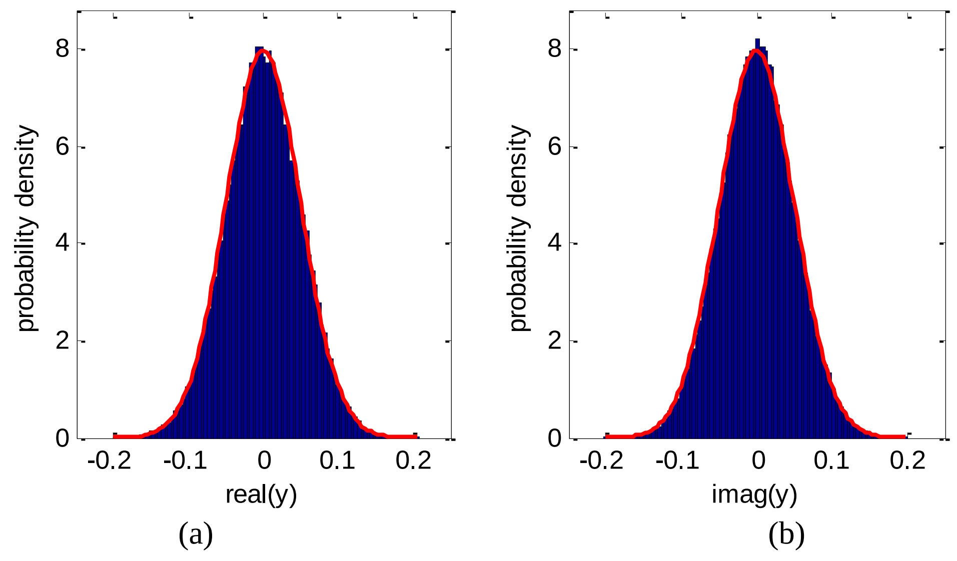


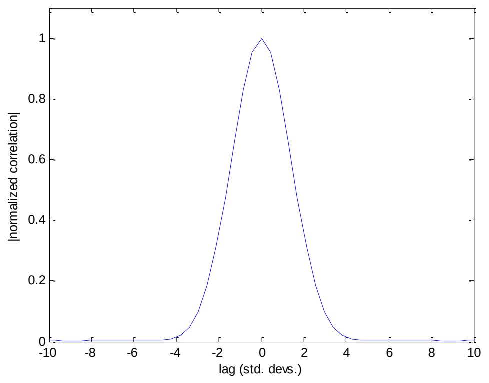
Figure 4. (a) Histogram and theoretical pdf for real part of y[n].. (b) Same for imaginary part of y[n].

Finally, Figure 5(a) shows the central 20-σ portion of the autocorrelation of one realization of $y[n]$, illustrating that it is also Gaussian, as would be expected. Figure 5(b) is the DFT of the autocorrelation function, and thus an estimate of power spectrum of $y[n]$, overlaid with the corresponding theoretical Gaussian of standard deviation σ_{HH}. The dotted red marker lines indicate that the power spectrum does indeed have the expected standard deviation of σ_{HH} = 0.3.

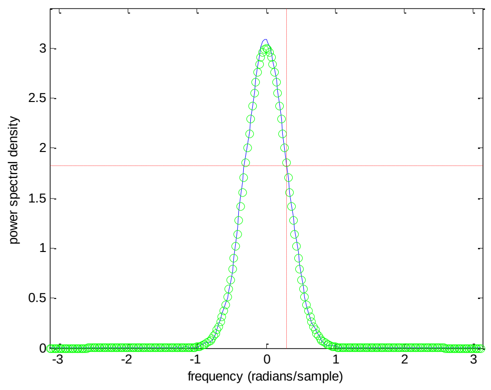
4 References

[1] http://mathworld.wolfram.com/FourierTransformGaussian.html

[2] A. V. Oppenheim, R. W. Schafer, and J. R. Buck, *Discrete-Time Signal Processing*, 2nd edition. Prentice-Hall, New York, 1999.



(a)



(b)

Figure 5. (a) Central portion of autocorrelation of y[n]. (b) Power spectrum S_{yy} obtained as DFT of the autocorrelation function. Solid blue line is the FFT of the autocorrelation in (a) green circles are the theoretical power spectrum. Red lines indicate the 1-σ point of the theoretical curve.

5 Appendix: MATLAB Demonstration Code[1]

The following MATLAB code implements and demonstrates various results discussed above, and was used to generate the plots above. Most of the code is devoted to a number of calculations and plots intended to check or demonstrate the results; some of these calculations are commented out in the code below.

```
% gauss_noise
%
% M-file to create a complex random process with Gaussian pdf, Gaussian
% power spectrum (and therefore Gaussian autocorrelation)

close all
clear all
format compact

% first make sure we have the Gaussian DTFT pair understood correctly.
% Start by generating gaussian window using gausswin.  Scale to be a true
% Gaussian pdf.  Test variance visually, test peak amplitude and area.

sig = input('Enter window standard deviation (samples): ')
N = input('Enter window length (odd; at least 6 std devs): ')
w = gausswin(N,(N-1)/2/sig)/sqrt(2*pi)/sig;
% sum(w)  % this should be 1.0
% max(w)/(1/sqrt(2*pi)/sig)  % this should be 1.0
figure
plot((0:N-1),w);
axis([0 N-1 0 1/sqrt(2*pi)/sig])
xlabel('sample'); ylabel('amplitude')
```

[1] The author thanks Weiguang Hou of Garmin, LTD. for discovering and correcting an error in the use of the MATLAB `gausswin` function in implementing $w[n]$ and $h[n]$ in an earlier version of this code.

```
% figure
% plot((0:N-1),w/max(w)/exp(-0.5));  % plot should go through 1.0 at n=sig
% grid; vline((N-1)/2+sig); vline((N-1)/2-sig);



% now let's test the DTFT of our time-domain Gaussian against the
% analytical formula
K = 2^(nextpow2(N)+2);
W = fft(w,K);
omega = 2*pi*(0:K-1)'/K - pi;
Wtest = exp(-omega.^2*sig^2/2);
figure
plot(omega,fftshift(abs(W))); hold on
plot(omega,Wtest,'or'); hold off
grid
xlabel('frequency (radians/sample)'); ylabel('amplitude')
axis([-pi pi 0 1])

%  Now let's test eqns. 7 and 8 in the memo.
sigHH = input('Enter std. dev. of power spectrum in radians/sample: ');
sighh = 1/sqrt(2)/sigHH;
Nh = input(['Enter size of impulse response (at least ',num2str(ceil(6*sighh)),'
samples): ']);
h = gausswin(Nh,(Nh-1)/2/sighh)/sqrt(2*pi)/sighh;
h = h/sum(h);

figure
plot((0:Nh-1),h);
axis([0 Nh-1 0 1.1/sqrt(2*pi)/sighh])
xlabel('sample'); ylabel('amplitude')

K = 2^(nextpow2(Nh)+2);
HH = abs(fft(h,K)).^2;
omega = 2*pi*(0:K-1)'/K - pi;
HHtest = exp(-omega.^2/2/sigHH^2);
figure
plot(omega,fftshift(HH)); hold on
plot(omega,HHtest,'og'); hold off
grid
xlabel('frequency (radians/sample)'); ylabel('amplitude')
axis([-pi pi 0 1])
vline(sigHH); hline(exp(-0.5));

% Now let's filter some noise.  Create complex white Gaussian nosie
% sequence.  Test mean and variance.

M = input('Enter input sequence length (at least a few 1000): ')
varx = input('Enter complex input noise variance: ');
x = sqrt(varx/2)*randn(M,1)+j*sqrt(varx/2)*randn(M,1);
mx_obs = mean(x)  % should be close to zero
varx_check = var(x)/varx  %should be close to 1
% figure
% subplot(1,2,1)
% hist(real(x),M/100)
% subplot(1,2,2)
% hist(imag(x),M/100)

% now do the filtering, using h as an impulse response
y = conv(x,h);
y = y(length(h):end);  % get rid of the filter start-up transient
% meany = mean(y)  % mean should be close to zero
vary = var(y);
% vary_check = vary/sum(h.^2)/var(x)  % check the output power; should be close
to 1
% vary_real = var(real(y))  % vary_real and vary_imag should be approximately
equal
% vary_imag = var(imag(y))
% check the histogram of the real and imaginary parts of y to see that it's
% still Gaussian pdf.  Normalize histogram to have unit area, plot agains
% expected theoretical Gaussian pdf
figure
subplot(1,2,1)
hist(real(y),100)
[count,centers]=hist(real(y),100);
bin_size = centers(2)-centers(1);
area = sum(count)*bin_size;
density = count/area;
bar(centers,density,1);
hold on
pdf = exp(-centers.^2/2/(vary/2))/sqrt(2*pi)/sqrt(vary/2);
plot(centers,pdf,'r','LineWidth',2)
axis([-5*sqrt(vary/2) 5*sqrt(vary/2) 0 1.1/sqrt(2*pi)/sqrt(vary/2)])
xlabel('real(y)'); ylabel('probability density')
hold off
subplot(1,2,2)
[count,centers]=hist(imag(y),100);
bin_size = centers(2)-centers(1);
area = sum(count)*bin_size;
density = count/area;
bar(centers,density,1);
hold on
pdf = exp(-centers.^2/2/(vary/2))/sqrt(2*pi)/sqrt(vary/2);
plot(centers,pdf,'r','LineWidth',2)
axis([-5*sqrt(vary/2) 5*sqrt(vary/2) 0 1.1/sqrt(2*pi)/sqrt(vary/2)])
xlabel('imag(y)'); ylabel('probability density')
hold off

% let's look at the power spectrum of the noise we have created.  I'll do
% it by computing the autocorrelation function and then Fourier
% transforming; I get a smoother spectral estimate that way.  Also I'm only
% going to use the central part of the autocorrelation where all the action
% is; I can estimate this from the expected variance, which will be
% 1/sigHH.
syy = xcorr(y)/length(y);  % with this normalization, value at lag 0 = vary
pause
syy = syy(length(y)-ceil(10*sighh):length(y)+ceil(10*sighh));
figure
lag = (-ceil(10*sighh):ceil(10*sighh))/sighh;
plot(lag,abs(syy)/max(abs(syy)))
xlabel('lag (std. devs.)'); ylabel('|normalized correlation|')
axis([-10 10 0 1.1])
K = 2^(nextpow2(length(syy))+2);
omega = 2*pi*(0:K-1)'/K - pi;
Syy = abs(fftshift(fft(syy,K)));
Syy = Nh/(Nh-1)*Syy;
figure
plot(omega,Syy); hold on
% we want to overlay the expected theroetical power spectrum and see what
% we get.  The std. dev. should be sigHH; that's what we started the whole
% exercise with.  The power is 'vary'; this will be the peak of the Gaussian
% autocorrelation function.  The discrete-time Gaussian transform pair then
% tells us that the power spectrum will have a peak amplitude of
pspec = exp(-omega.^2/2/sigHH^2)*vary*sqrt(2*pi)/sigHH;
% pspec = pspec*(max(Syy)/max(pspec));
plot(omega,pspec,'og')
axis([-pi pi 0 1.1*max(abs(Syy))])
xlabel('frequency (radians/sample)'); ylabel('power spectral density')
vline(sigHH); hline(exp(-0.5)*vary*sqrt(2*pi)/sigHH);
```