EE 567 Project

Communication Systems

by

Wanwiset Peerapatanapokin
USC ID: 4178-8245-40

Table of Contents

Part 1 – Analytical Comparison of I&D and IIR Low Pass Filter

In the first part we are going to investigate the operation of applying an LPF to a discrete time signal. We consider the received signal after being down converted by a mixing operation. The next step is to apply an LPF.

Assume that at this point the signal to be input of the LPF is

$$s(k) = \sqrt{E} + double\, frequency\, terms + n(k), k = 0, 1, ..$$

Where n(k) is standard gaussian random variable where each k is independent of each other for all k. In this model we will ignore the double frequency terms since they will be sufficiently suppressed by the LPF.

Now the idea of I&D – integrate and dump LPF is to compute the average of N terms, the output of the LPF is:

$$y(n) = \frac{1}{N} \sum_{k=n-N+1}^{n} s(k) \ , \qquad \mathrm{n} = 0,1, \dots$$

Where y(n) = 0 for n < 0. It is called integrate and dump filter due to its continuous time counterpart, here we are looking at the discrete version.

Next, the IIR – infinite impulse response LPF is defined as the following:

$$\tilde{y}(n) = (1 - \alpha)s(n) + \alpha\tilde{y}(n - 1), \qquad n = 0, 1 ..$$

Where ỹ(-1) = 0 and 0 < α < *1.*

First, we want to find the value of α*.* To make a fair comparison we want both filters' output to have the same mean and variance as $n \to \infty$. For simplicity we will assume that only noise is present in the input (let s(k) = n(k)).

I&D Filter mean and variance:

$$E[y(n)] = E\left[\frac{1}{N} \sum_{k=n-N+1}^{n} s(k)\right] = \frac{1}{N} \sum_{k=n-N+1}^{n} E[s(k)] = \frac{1}{N} \sum_{k=n-N+1}^{n} E[n(k)]$$

$$= \frac{1}{N} \sum_{k=n-N+1}^{n} 0 = 0$$

$$V[y(n)] = V\left[\frac{1}{N}\sum_{k=n-N+1}^{n} s(k)\right] = \frac{1}{N^2}\sum_{k=n-N+1}^{n} V[s(k)] = \frac{1}{N^2}\sum_{k=n-N+1}^{n} V[s(k)]$$

$$= \frac{1}{N^2}\sum_{k=n-N+1}^{n} V[n(k)] = \frac{1}{N^2}\sum_{k=n-N+1}^{n} 1 = \frac{1}{N^2} \times N = \frac{1}{N}$$

Where mean and variance of a standard gaussian random variable is 0 and 1 respectively.

The IIR LPF in its current iterative form may prove difficult to work with. We can simplify it to a more closed form before finding the mean and variance. First, try to expand out the terms:

$$\tilde{y}(n) = (1-\alpha)s(n) + \alpha\tilde{y}(n-1), \qquad n = 0, 1 ..$$

$\tilde{y}(n) = (1-\alpha)s(n) + \alpha\big((1-\alpha)s(n-1) + \alpha\tilde{y}(n-2)\big)$

$\tilde{y}(n) = (1-\alpha)s(n) + \alpha(1-\alpha)s(n-1) + \alpha^2\tilde{y}(n-2)$

$\tilde{y}(n) = (1-\alpha)s(n) + \alpha(1-\alpha)s(n-1) + \alpha^2\big((1-\alpha)s(n-2) + \alpha\tilde{y}(n-3)\big)$

$\tilde{y}(n) = (1-\alpha)s(n) + \alpha(1-\alpha)s(n-1) + \alpha^2(1-\alpha)s(n-2) + \alpha^3\tilde{y}(n-3)$

$\tilde{y}(n) = (1-\alpha)s(n) + \alpha(1-\alpha)s(n-1) + \alpha^2(1-\alpha)s(n-2) + \cdots + \alpha^n(1-\alpha)s(0) + \tilde{y}(-1)$

$\tilde{y}(n) = (1-\alpha)s(n) + \alpha(1-\alpha)s(n-1) + \alpha^2(1-\alpha)s(n-2) + \cdots + \alpha^n(1-\alpha)s(0)$

$\tilde{y}(n) = (1-\alpha)(s(n) + \alpha s(n) + \alpha^2 s(n) + \cdots + \alpha^n s(0))$

$\tilde{y}(n) = (1-\alpha)\sum_{k=0}^{n}\alpha^k s(n-k)$

Then, finding the mean and variance:

$$E[\tilde{y}(n)] = E\left[(1-\alpha)\sum_{k=0}^{n}\alpha^k s(n-k)\right] = (1-\alpha)\sum_{k=0}^{n}\alpha^k E[s(n-k)]$$

$$= (1-\alpha)\sum_{k=0}^{n}\alpha^k E[n(n-k)] = (1-\alpha)\sum_{k=0}^{n}\alpha^k \cdot 0 = 0$$

$$V[\tilde{y}(n)] = V\left[(1-\alpha)\sum_{k=0}^{n}\alpha^k s(n-k)\right] = (1-\alpha)^2\sum_{k=0}^{n}\alpha^{2k}\,V[s(n-k)]$$

$$= (1-\alpha)^2\sum_{k=0}^{n}\alpha^{2k}\,V[n(n-k)] = (1-\alpha)^2\sum_{k=0}^{n}(\alpha^2)^k\cdot 1$$

$$= (1-\alpha)^2\frac{1-(\alpha^2)^{n+1}}{1-\alpha^2} \quad , by\ geometric\ sum\ formula$$

$$= (1-\alpha)^2\frac{1-(\alpha^2)^{n+1}}{(1-\alpha)(1+\alpha)} = (1-\alpha)\frac{1-(\alpha^2)^{n+1}}{(1+\alpha)}$$

Now, the mean of both LPF is 0 but we still need to find the α that matches the variance of both filters as $n \to \infty$, so:

$$\lim_{n\to\infty}(1-\alpha)\frac{1-(\alpha^2)^{n+1}}{(1+\alpha)} = \lim_{n\to\infty}\frac{1}{N}$$

$$(1-\alpha)\frac{1-\lim_{n\to\infty}((\alpha^2)^{n+1})}{(1+\alpha)} = \frac{1}{N}$$

$$\lim_{n\to\infty}\alpha^{2n+2} = 0 \quad , Since\ 0 < \alpha < 1$$

$$\frac{1-\alpha}{1+\alpha} = \frac{1}{N}$$

$$(1-\alpha)N = 1+\alpha$$

$$N-1 = N\alpha+\alpha$$

$$\boldsymbol{\alpha} = \frac{\boldsymbol{N-1}}{\boldsymbol{N+1}}$$

Next, we will try to compute the impulse response and step response of each LPF. Where the impulse response is the output when the input is a delta function $\delta[n]$

$$where\ \delta[n] = \begin{cases} 1 \ , & n = 0 \\ 0 \ , & else \end{cases}$$

And the step response is the output when the input is a unit step function $u[n]$

$$where\ u[n] = \begin{cases} 1 \ , & n \geq 0 \\ 0 \ , & n < 0 \end{cases}$$

I&D Low Pass Filter impulse response $h[n]$:

$$y[n] = \frac{1}{N} \sum_{k=n-N+1}^{n} s[k] \ , \qquad \mathrm{n} \ = \ 0,1, \dots$$

$$let\ s[k] = \delta[k]$$

$$h[n] = \frac{1}{N} \sum_{k=n-N+1}^{n} \delta[k] \ , \qquad \mathrm{n} \ = \ 0,1, \dots$$

Here, the $\delta[k]$ term only has value when k = 0

Hence, the summation bounds must include 0, we have

$$n - N + 1 \leq 0 \ \leq n$$

$$-N + 1 \leq -n \leq 0$$

$$0 \geq -n \geq -N + 1$$

$$0 \leq n \leq N - 1$$

$$Then, \sum_{k=n-N+1}^{n} \delta[k] = \delta[n - N + 1] + \dots + \delta[0] + \dots + d[n] = \ \delta[0] = 1$$

$$Substituting, \qquad h[n] = \frac{1}{N} \cdot 1 \ , \qquad 0 \leq n \leq N - 1$$

$$\boldsymbol{h[n] = \frac{1}{N}} \ , \qquad \boldsymbol{0 \leq n < N}$$

I&D Low Pass Filter step response $g[n]$:

$$y[n] = \frac{1}{N} \sum_{k=n-N+1}^{n} s[k] \ , \qquad \mathrm{n} \ = \ 0,1, \dots$$

$$let\ s[k] = u[k]$$

$$g[n] = \frac{1}{N} \sum_{k=n-N+1}^{n} u[k] \ , \qquad \mathrm{n} \ = \ 0,1, \dots$$

Expanding the sum,

$$g[n] = \frac{1}{N} \left(u[n-N+1] + u[n-N+2] + \cdots + u[0] + u[1] + \cdots + u[n]\right)$$

If $k < 0 \ or\ n - N + 1 < 0\ we\ only\ consider\ terms\ u[0]\ to\ u[n]$,

$$g[n] = \frac{n+1}{N} \ , \ \ n - N + 1 < 0$$
$$g[n] = \frac{n+1}{N} \ , \ \ n < N - 1$$

If $k \geq 0\ or\ n - N + 1 \geq \ 0\ the\ summation\ considers\ every\ terms\ which\ is\ N\ terms$,

$$\sum_{k=n-N+1}^{n} u[k] = N$$

$$g[n] = \frac{1}{N}\ N = 1, \qquad n - N + 1 \geq \ 0$$

$$g[n] = 1, \qquad n \geq \ N - 1$$

Also, from definition y(n) = 0 for n < 0. In summary,

$$\boldsymbol{g[n]} = \begin{cases} \boldsymbol{\frac{n+1}{N}} \ , & \boldsymbol{0 \leq n < N - 1} \\ \boldsymbol{1}, & \boldsymbol{n \geq \ N - 1} \end{cases}$$

IIR Low Pass Filter impulse response $\hat{\mathrm{h}}[n]$:

$$\tilde{\mathrm{y}}(n) = (1-\alpha)\sum_{k=0}^{n}\alpha^k s(n-k)\ , \quad n = 0,1,\ldots$$

$$let\ s[n-k] = \delta[n-k]$$

$$\hat{\mathrm{h}}[n] = (1-\alpha)\sum_{k=0}^{n}\alpha^k\delta(n-k)$$

$\delta(n-k)\ only\ has\ value\ when\ n-k = 0, therefore\ k = n.\ \ Substituting,$

$$\hat{\mathrm{h}}[n] = (1-\alpha)\alpha^n\delta(n-n) = (1-\alpha)\alpha^n\cdot 1$$

$$\hat{\mathrm{h}}[\boldsymbol{n}] = (\mathbf{1}-\boldsymbol{\alpha})\boldsymbol{\alpha}^{\boldsymbol{n}}\ , \quad \boldsymbol{n}\geq\mathbf{0}$$

IIR Low Pass Filter step response $\hat{\mathrm{g}}[n]$:

$$\tilde{\mathrm{y}}(n) = (1-\alpha)\sum_{k=0}^{n}\alpha^k s(n-k)\ , \quad n = 0,1,\ldots$$

$$let\ s[n-k] = u[n-k]$$

$$\hat{\mathrm{g}}[n] = (1-\alpha)\sum_{k=0}^{n}\alpha^k u(n-k)$$

$$u(n-k)\ is\ always\ considered\ since\ k =\ 0\ to\ n, \qquad n-k\ is\ always\ \geq 0$$

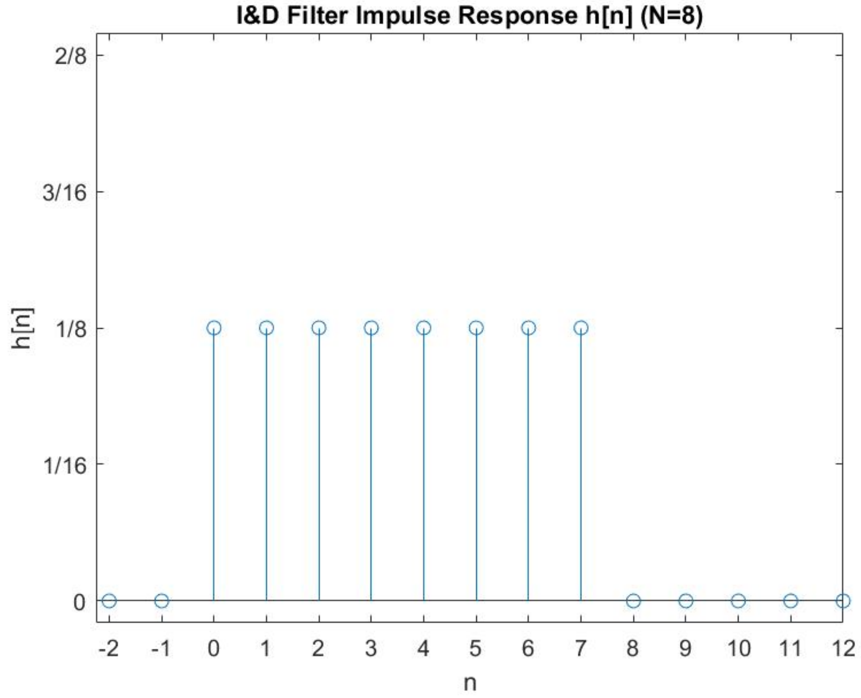
$$\hat{\mathrm{g}}[n] = (1-\alpha)\sum_{k=0}^{n}\alpha^k$$

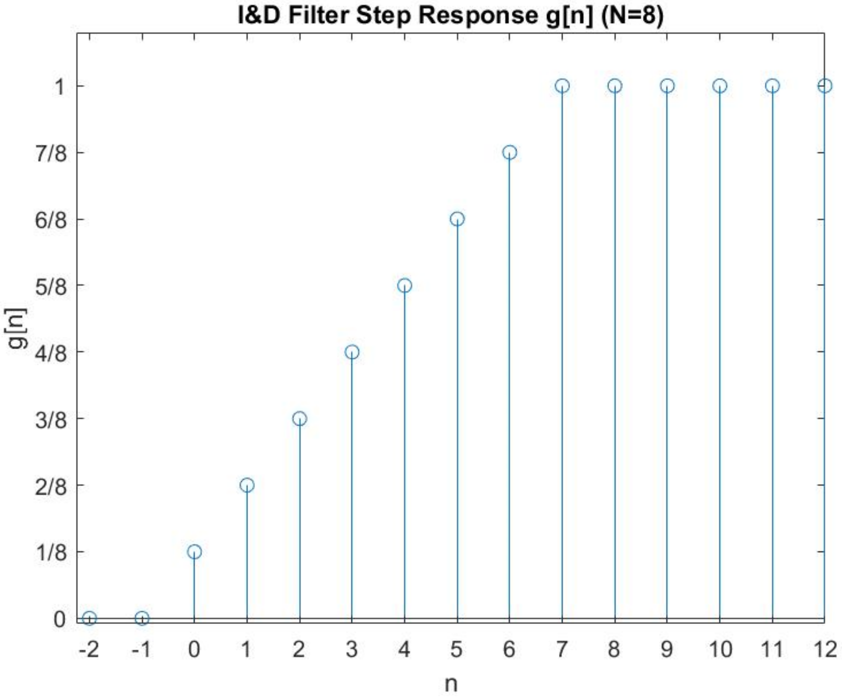
Using geometric series formula,

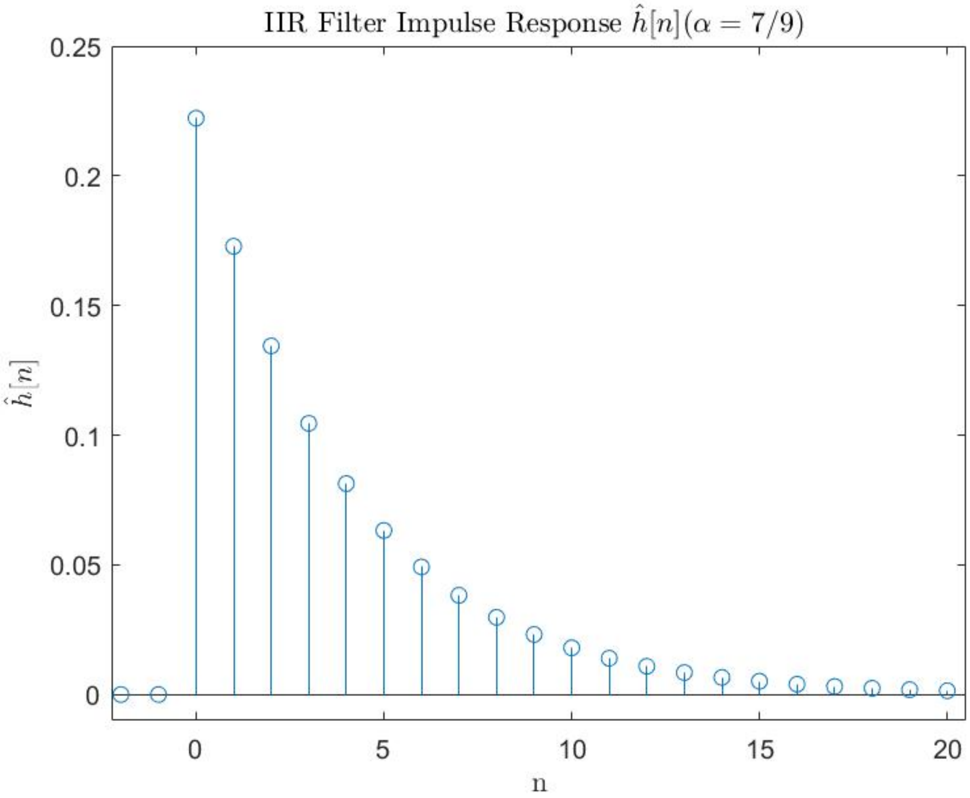
$$\hat{\mathrm{g}}[n] = (1-\alpha)\frac{1-\alpha^{n+1}}{(1-\alpha)}$$

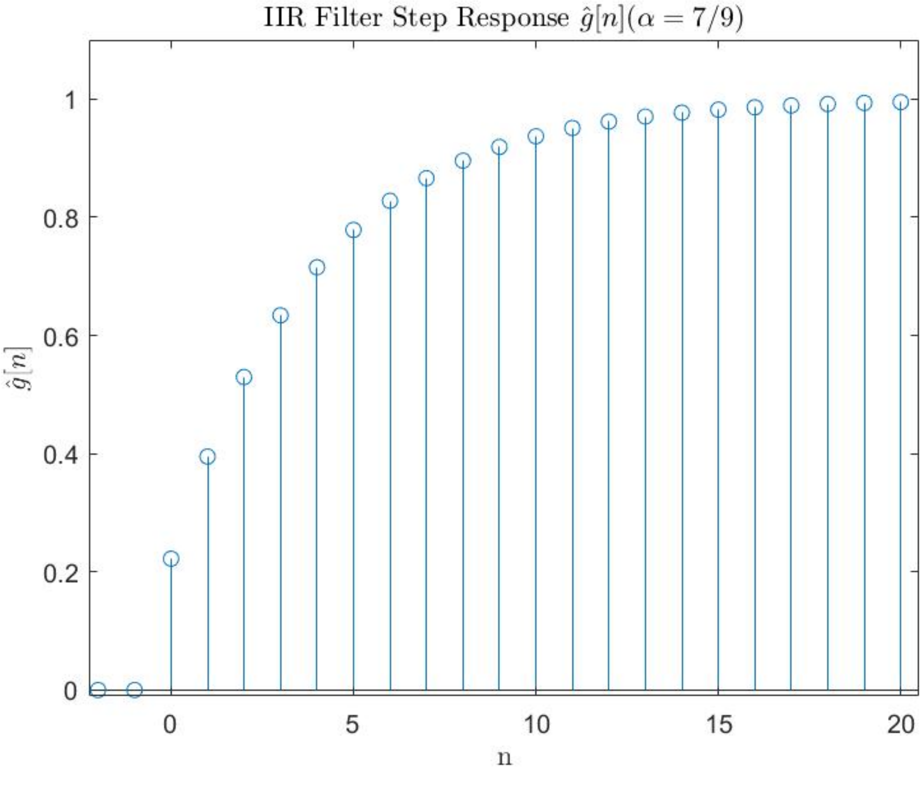
$$\hat{\mathrm{g}}[\boldsymbol{n}] = \mathbf{1}-\boldsymbol{\alpha}^{\boldsymbol{n}+\mathbf{1}}, \quad \boldsymbol{n}\geq\mathbf{0}$$

Finally, to better visualize, let's plot the impulse responses and step responses of the I&D and IIR LPF. Let's use $N = 8$ for the I&D filter, then, $\alpha = \frac{N-1}{N+1} = \frac{7}{9}$ for IIR.






IIR Filter Impulse Response $\hat{h}[n] (\alpha = 7/9)$
$\hat{h}[n]$
0.25
0.2
0.15
0.1
0.05
0
0
5
10
15
20
n


IIR Filter Step Response $\hat{g}[n] (\alpha = 7/9)$
$\hat{g}[n]$
1
0.8
0.6
0.4
0.2
0
0
5
10
15
20
n

Part 2 – I&D and IIR LPF Probability of Bit Error Analysis through Simulation

In this part the I&D and IIR filter will be implemented using MATLAB and we will simulate the bit error rate (BER) also known as probability of bit error (P_b) while varying E_b/N_0. Again, we will start with a down converted BPSK signal, ready to be the input for the LPF.

$$s(k) = \ A \ + \ double\,frequency\,terms \ + \ n(k), k \ = \ 0, 1, ..$$

Where A is a constant and n(k)'s are standard gaussian random variable with mean zero and variance one ($\sigma^2 = 1$) and are independent of each other for all k's. Let's first look at I&D filter.

$$y(n) = \frac{1}{N}\sum_{k=0}^{N-1} s(k)$$

Again, we can assume that the LPF operation will sufficiently suppress the double frequency terms. Then, the output y of this LPF will produce a simulation result with E_b/N_0 as its bit-energy-to-noise ratio. It is also well known that the bit error rate of BPSK follows the function $P_b = \ Q\left(\sqrt{\frac{2E_b}{N_0}}\right)$ where Q is the tail distribution of the standard normal. In the simulation we will fix the E_b/N_0 on the x-axis so we must find the signal component A that corresponds to the E_b/N_0.

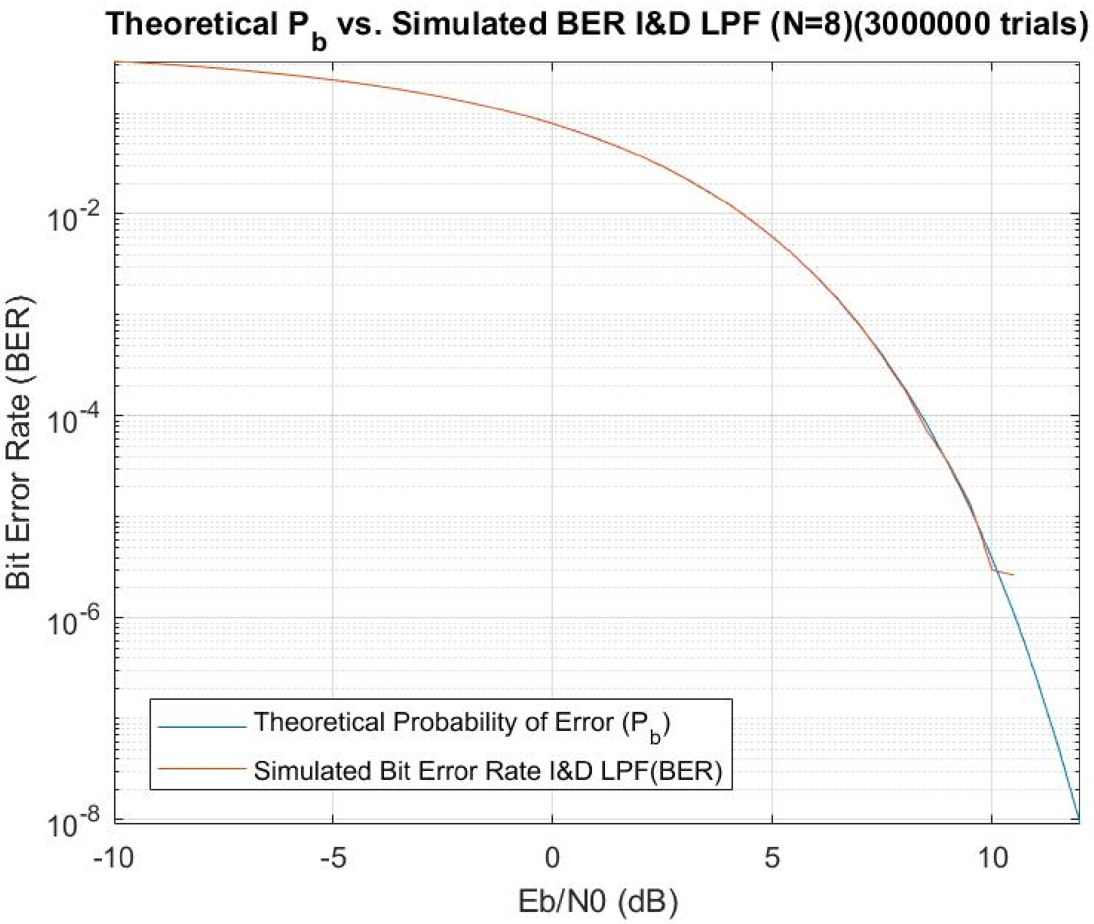
From $s(k)$, A is the signal component and in $y(n)$, $\frac{1}{N}\sum_{k=0}^{N-1} A = A$. So $E_b = \ A^2$

$n(k)$ is the noise component with $\sigma^2 = 1$, but here the noise component of $y(n)$ is composed of the average of N noises. So, by averaging out many independent gaussian noises the variance is reduced, and by Central Limit Theorem we know that $\sigma_y^2 = \frac{\sigma^2}{N}$.

Then, from power spectrum of white gaussian noise $\sigma_y^2 = \frac{N_0}{2}$. So, $N_0 = \sigma_y^2 = 2 \cdot \frac{\sigma^2}{N}$.

Then $\frac{E_b}{N_0} = \frac{A^2}{\sigma_y^2}$, solve for A, $A = \ \sqrt{\frac{E_b}{N_0} \cdot 2 \cdot \sigma_y^2} = \sqrt{\frac{E_b}{N_0} \cdot 2 \cdot \frac{\sigma^2}{N}}$

In the simulation we will use N = 8 and plot its result versus theoretical P_b, so the simulation should provide equal results. We want results to include up to 10^{-6} BER so there must be at least 10^6 trials, let's try $3 * 10^6$ trials.

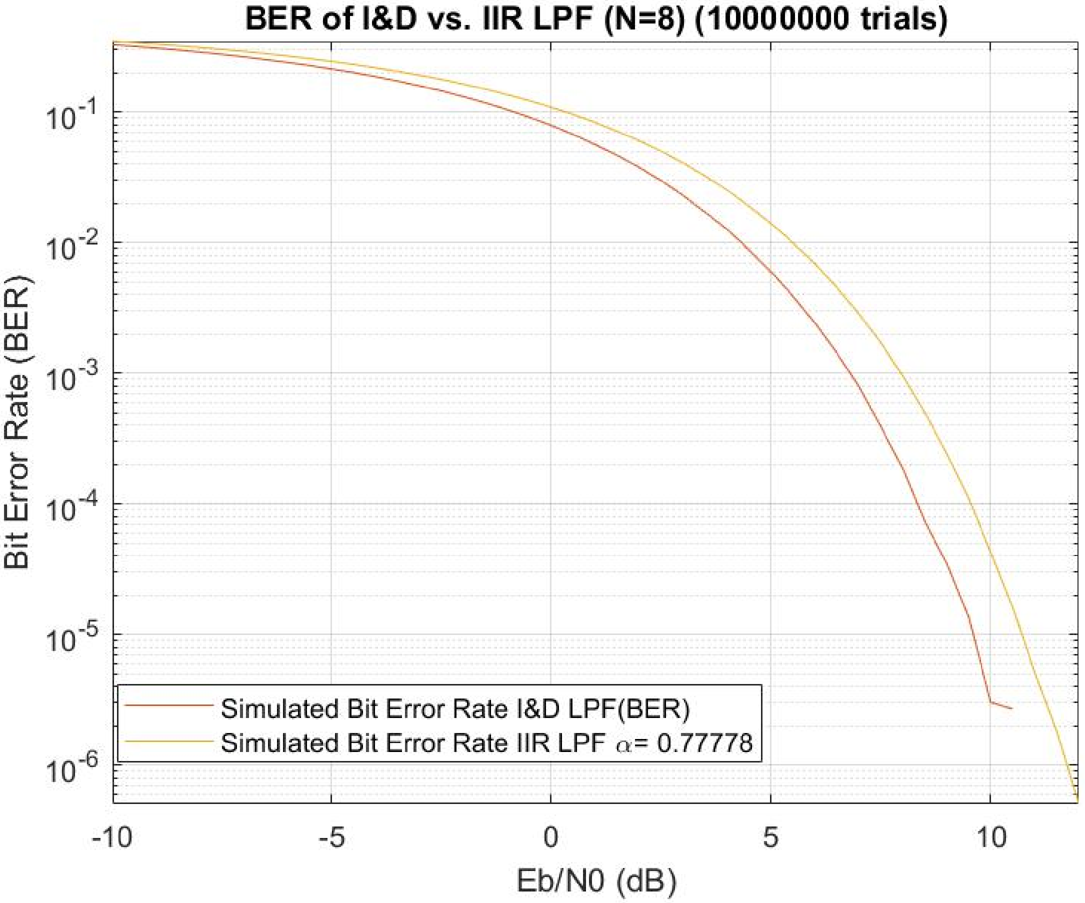

Theoretical P_b vs. Simulated BER I&D LPF (N=8)(3000000 trials)
Bit Error Rate (BER)
10^{-2}
10^{-4}
10^{-6}
10^{-8}
-10
-5
0
5
10
Eb/N0 (dB)
Theoretical Probability of Error (P_b)
Simulated Bit Error Rate I&D LPF(BER)

Next, we will compare the simulation BER result for I&D filter with IIR filter. The IIR filter is defined as

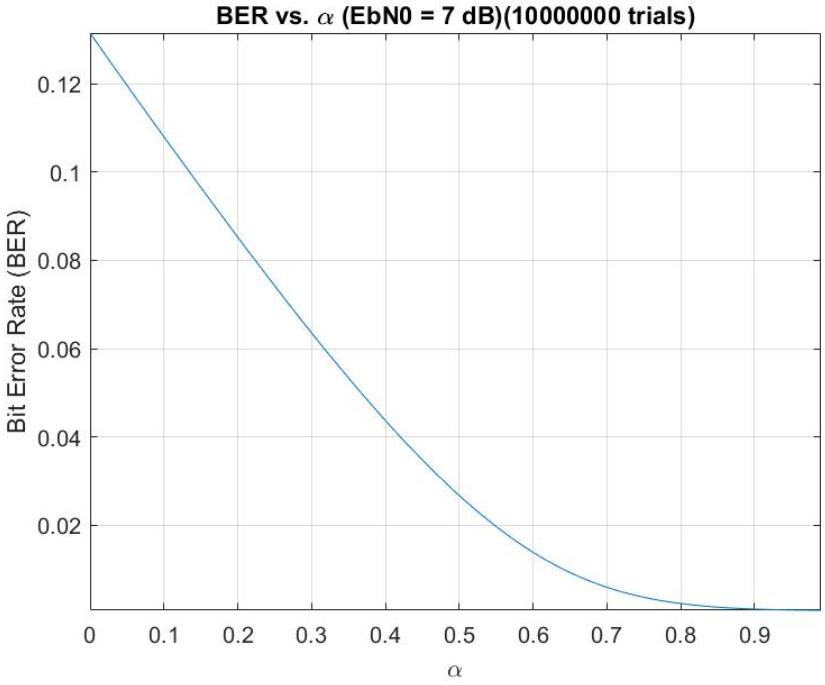
$$\tilde{y}(n) = (1 - \alpha)s(n) + \alpha\tilde{y}(n - 1), \qquad n = 0, 1, .., N - 1$$

Where $\tilde{y}(-1)$ =0

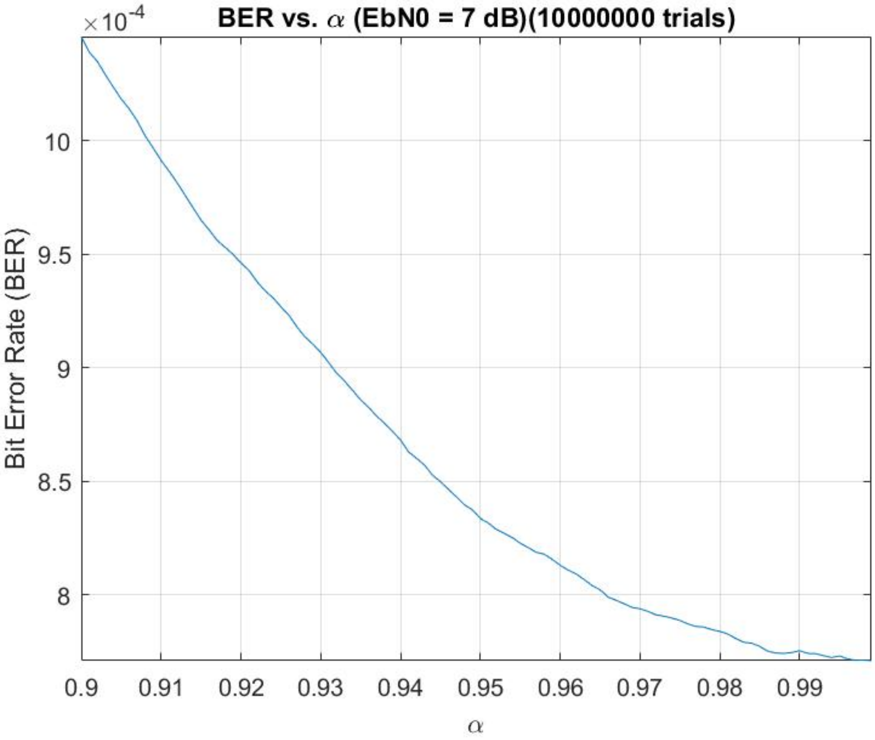
Again, we will take N=8 for the I&D LPF which also applies to the output of the IIR filter, we will take the highest iteration, $\tilde{y}(N-1) = \tilde{y}(7)$ and using the α value found in part 1: $\alpha = \frac{N-1}{N+1} = \frac{7}{9}$



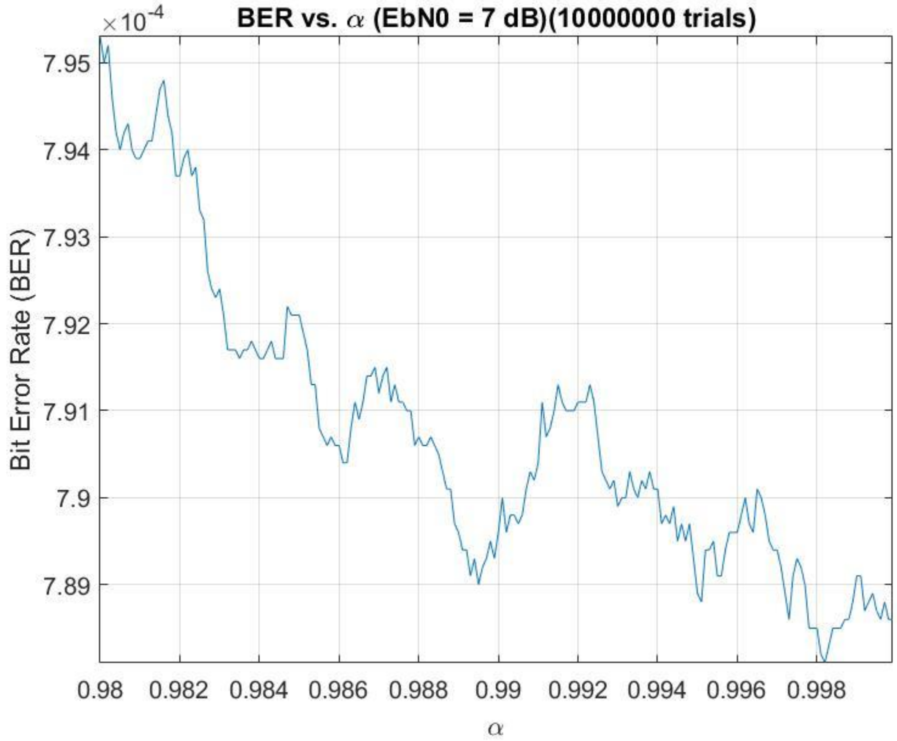
It seems that the IIR LPF is performing worse than the I&D. Maybe the α value used was not exactly optimal. Next, we will investigate various α values for the IIR filter. Here, we fix $E_b/N_0 = 7$ dB and vary the α.



Now try to magnify on $\alpha = 0.9\ to\ 1$

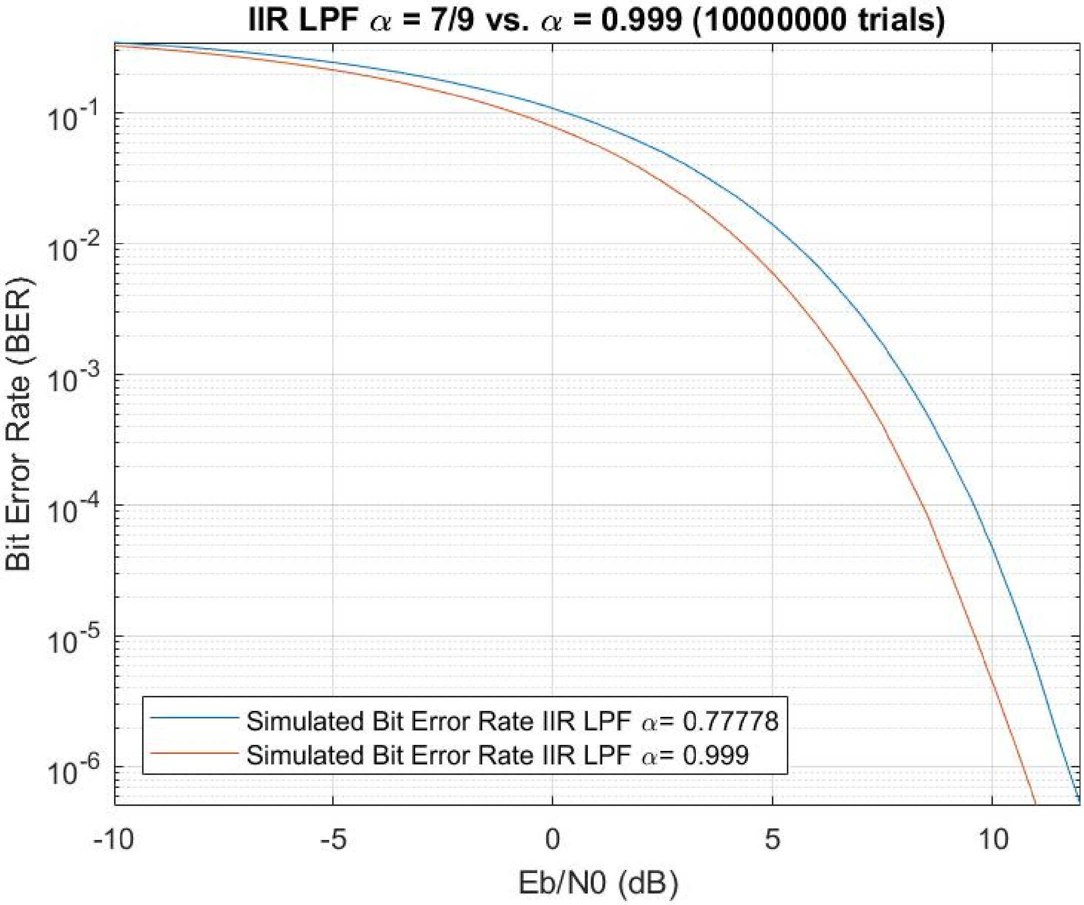


Further magnify on 0.98 to 1



It seems that $\alpha = 0.999..$ or approaching 1 is optimal for the simulation. In the real world, however, this might not be the case since it would involve other factors such as the double frequency terms we are choosing to ignore in this simulation.

Finally, we will compare the perfomance of IIR filter with optimal $\alpha = 0.999..$ to the original $\alpha = 7/9$



As expected, the optimal $\alpha = 0.999$ performs better for this simulation.

Part 3 – Signal Detection in Noise

In this part we will compare integration detection and M of N logic detection. The criteria of performance is how well can each method detect signal in presense of noise. The metrics we will consider are P_{fa} – Probability of false alarm and P_d – Probability of detection along with varying SNR.

We begin with a received signal

$$r(t) = Acos(2\pi f_c t + \varphi) + n(t),\ 0\leq t\leq T$$

Where A is constant, taking on the value $A = 1$ or $A = 0$ depending on the presense of message, $f_c = 1{,}000{,}000$ Hz , $T = 0.001$ second, and $n(t)$ is a gaussian random variable with mean 0 and variance σ^2 for all t.

First let's simulate the direct integration approach. After mixing the signal and integrated over T seconds we obtain

$$r_1(T) = Acos(\varphi) + n_1$$

$$r_2(T) = Asin(\varphi) + n_2$$

Here, n_1 and n_2 are independent gaussian noise with mean 0 and variance σ^2. Then combining we have the detector output

$$z(T) = r_1^2 + r_2^2$$

Then $z(T)$ is compared to a threshold T_0. If $z(T)$ is higher, signal is present. If $z(T)$ is lower, no signal is present. To compare simulations, for all simulationslet's fix P_{fa} to a constant 10^{-4} and vary the SNR, then, P_d will be the output. To fix the P_{fa} we must find the corresponding T_0. P_{fa} considers the case when only noise is present, here, $z(T) = n_1^2 + n_2^2$. The summation of squared independent gaussian noises is well-known to be Chi-square distributed, in this case with 2 degrees of freedom (also assume $\sigma^2 = 1$).

The PDF of Chi-square k=2 is $\frac{1}{2}e^{-\frac{x}{2}}$ we have,

$$P_{fa} = \int_{T_0}^{\infty} \frac{1}{2}e^{-\frac{x}{2}}dx$$

$$P_{fa} = -e^{-\frac{x}{2}} \Big|_{x = T_0}^{x = \infty}$$

$$P_{fa} = e^{-\frac{T_0}{2}}$$

$$\ln(P_{fa}) = \ln(e^{-\frac{T_0}{2}})$$

$$\ln(P_{fa}) = -\frac{T_0}{2}\ln(e)$$

$$T_0 = -2\ln(P_{fa})$$

Substitute $P_{fa} = 10^{-4}$

$$T_0 = 18.4207$$

Now let's simulate the P_d vs. SNR where we vary the SNR from 0 to 15 dB. Similar to part 2, in the simulation we will vary the SNR by varying A and let the noise term constant $\sigma^2 = 1$.

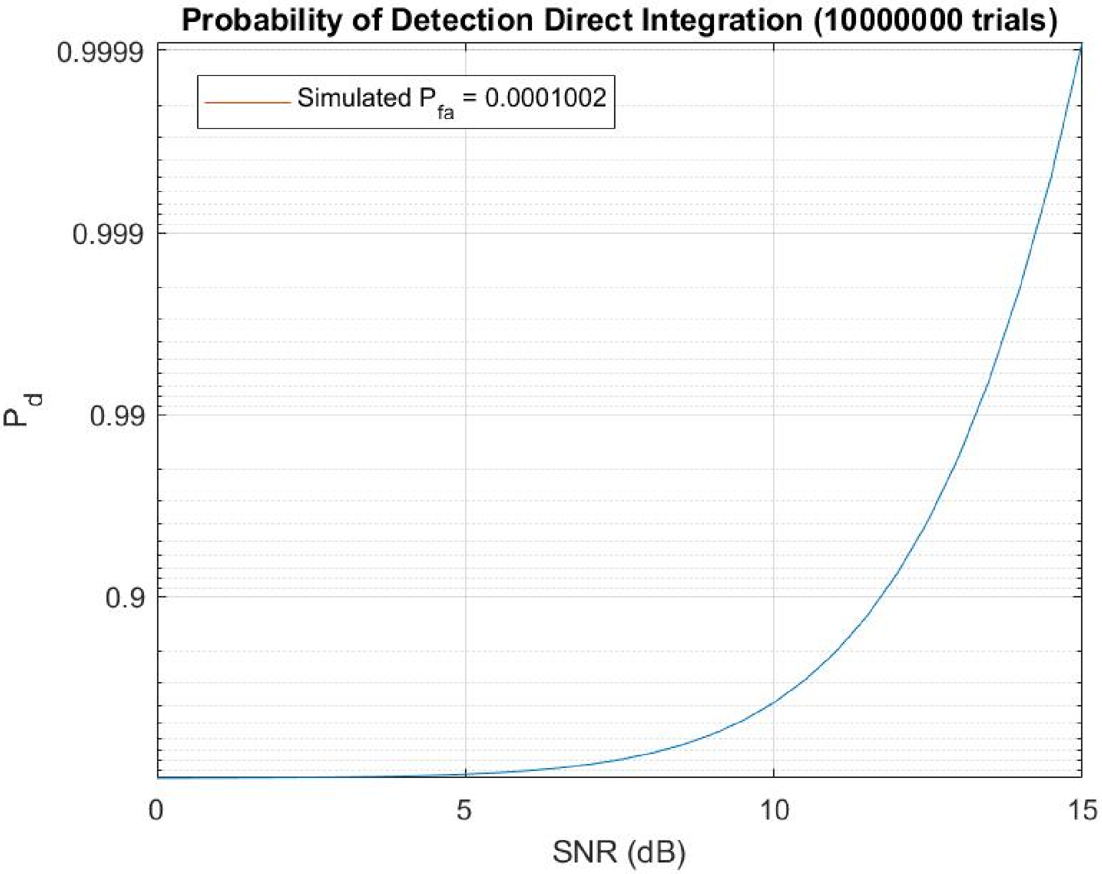
From the received signal $r(t) = Acos(2\pi f_c t + \varphi) + n(t),\ 0 \leq t \leq T$

The signal component is $Acos(2\pi f_c t + \varphi)$ which has power $\frac{A^2}{2}$

And the noise component $n(t)$ has power $\sigma^2 = 1$

So the SNR $= \frac{Signal\ power}{Noise\ power} = \frac{A^2}{2\sigma^2} = \frac{A^2}{2}$

Solve for A, we get $A = \sqrt{SNR * 2}$



Next we will simulate the post detector integrator method, where N = 16. Here $z(T)$ is sampled 16 times over T and scaled by $\frac{1}{16}$, then the results are summed up which averages out the noise then compared to a new threshold. By trial and error the new $T_0 = 4.41$ which yields $P_{fa} \approx 10^{-4}$. And we know that this T_0 is accurate because of confidence interval. In this case that we don't know the true variance the t-distribution should be used but as sample size grows large t-distribution converges to standard normal. So we can use standard normal version since it is more convenient to calculate. We will try to calculate the interval for P_{fa} using the formula $I = (\bar{X} - \frac{s}{\sqrt{n}} z_{\alpha/2}, \bar{X} + \frac{s}{\sqrt{n}} z_{\alpha/2})$ where

n is number of samples which is $trials * SNR\ points = 31 * 10{,}000{,}000 = 310{,}000{,}000$

$\bar{X}$ is the sample mean which is the $P_{fa} = \frac{signal\ detected}{n} = 0.00010071$,

$z_{\alpha/2}$ is the corresponding statistic for the confidence interval, in this case let's choose 99% which give $z_{\alpha/2} = 2.576$

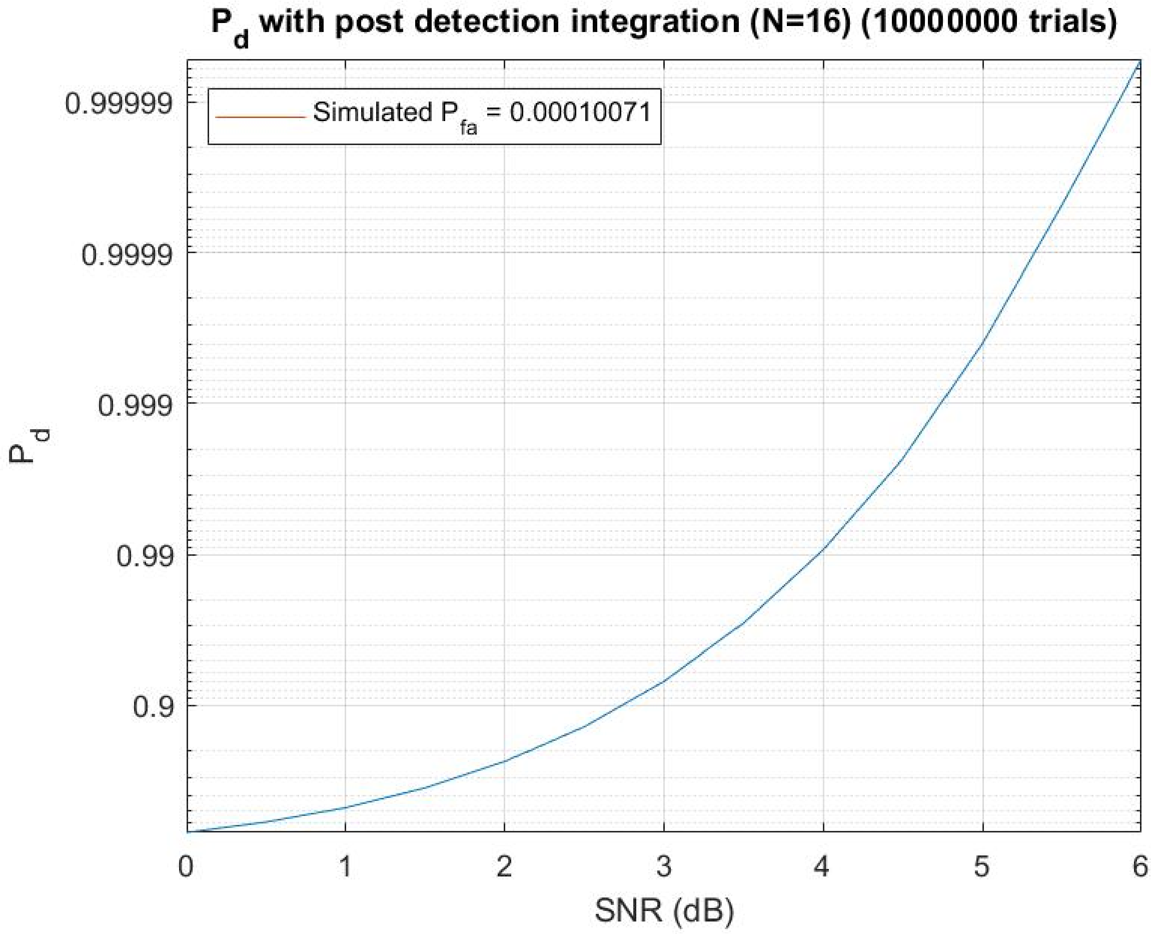
$s = \sqrt{\frac{\sum_{i=1}^{n}(X_i - \bar{X})^2}{n-1}} = 0.010035$ is the sample standard deviation, where X_i is the individual sample result, taking on 0 if no signal is detected and 1 if signal is detected, where only noise is present.

Finally we have $I_{99\%} = (\bar{X} - \frac{s}{\sqrt{n}} z_{\alpha/2}, \bar{X} + \frac{s}{\sqrt{n}} z_{\alpha/2})$

$$= \left(0.00010071 - \frac{0.010035}{\sqrt{310{,}000{,}000}} \cdot 2.576,\ 0.00010071 + \frac{0.010035}{\sqrt{310{,}000{,}000}} \cdot 2.576\right)$$

$$= (9.9241 \times 10^{-5}, \qquad 1.0218 \times 10^{-4})$$

So we can be 99% sure that $P_{fa}\ is\ between\ (9.9241 \times 10^{-5},\ 1.0218 \times 10^{-4})$ which includes the desired value 10^{-4}.



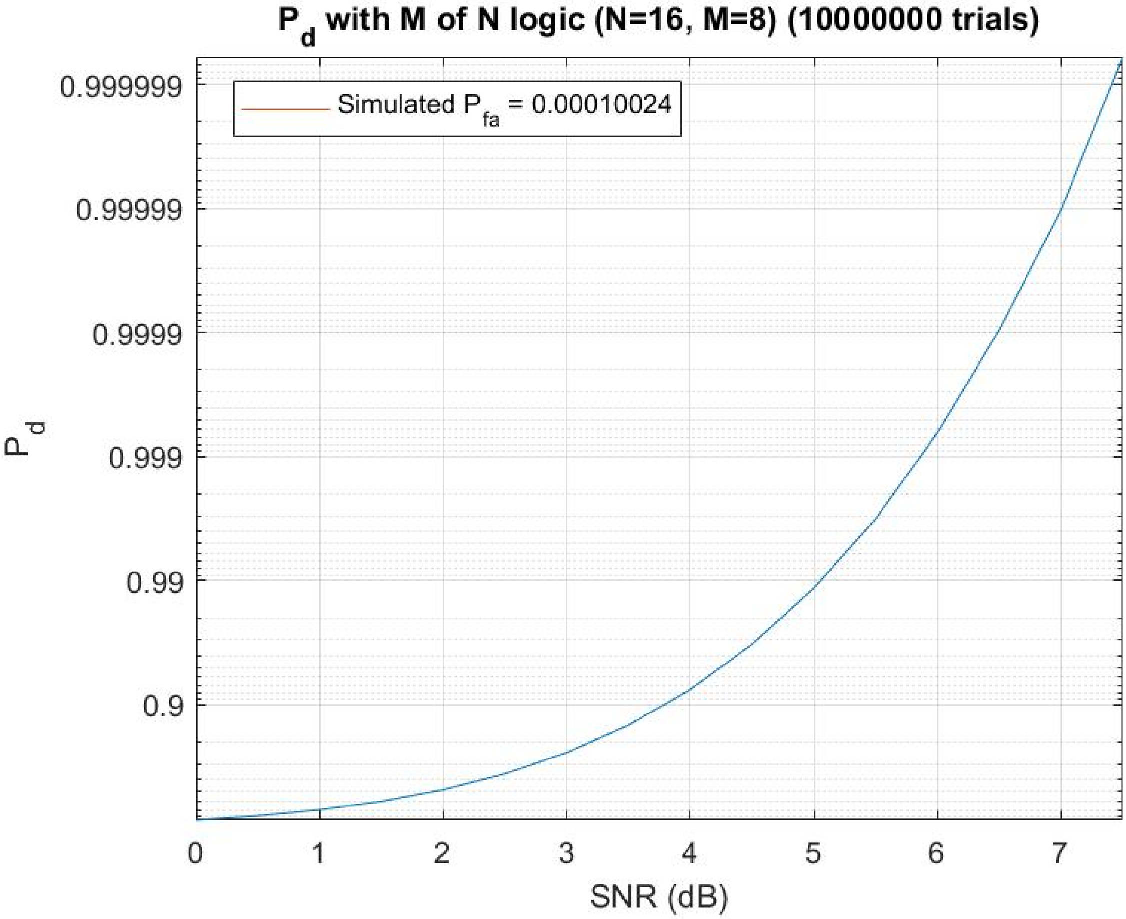
Here the SNR range does not go above 6 dB because the simulation doesn't produce any more detection error beyond that point ($P_d = 1$).

Finally, we will look into the performance of M of N logic detection with N = 16 and M = 8. In M of N logic the signal is sampled N = 16 times over T and after each sample the signal is immediately compared to a threshold. If by the end of T there are at least M = 8 signals above threshold then the detector thinks the signal is present. Again, we require $P_{fa} = 10^{-4}$ so by trial and error we obtain $T_0 = 0.27925$. Again, we will calculate the 99% confidence interval

$$I_{99\%} = (\bar{X} - \frac{s}{\sqrt{n}} z_{\alpha/2}, \bar{X} + \frac{s}{\sqrt{n}} z_{\alpha/2})$$

$$I_{99\%} = (0.00010024 - \frac{0.010011}{\sqrt{310{,}000{,}000}} \cdot 2.576, 0.00010024 + \frac{0.010011}{\sqrt{310{,}000{,}000}} \cdot 2.576)$$

$$I_{99\%} = (9.877 \times 10^{-5}, 1.017 \times 10^{-4})$$



If we compare the M of N logic result to Post Detection Integration it can be seen that M of N logic is worse by about 1 dB. This agrees with the theory that M of N performance is a little worse but it is easier to implement actual systems.

Appendix – All MATLAB Codes are shown here

EE567_project_1.m

```
n = -2:1:20;
N = 8;
alpha = (N-1)/(N+1);

h_n = zeros(1,length(n));
g_n = zeros(1,length(n));
h2_n = zeros(1,length(n));
g2_n = zeros(1,length(n));


for i = 1:length(n)
    if n(i)>=0 && n(i)<N
        h_n(i) = 1/N;
        g_n(i) = (n(i)+1)/N;
    else
        if n(i)>= N
            g_n(i) = 1;
        end
    end
    if n(i)>=0
        h2_n(i) = (1-alpha)*(alpha^n(i));
        g2_n(i) = 1-(alpha^(n(i)+1));
    end
end



figure;
stem(n,h_n)
axis([-2.25 12 -0.01 0.26])
set(gca, 'YTick', [0 1/16 1/8 3/16 2/8]);
set(gca, 'YTickLabel', {'0' '1/16' '1/8' '3/16' '2/8'});
xticks(n);
xlabel('n')
ylabel('h[n]')
title('I&D Filter Impulse Response h[n] (N=8)')

figure;
stem(n,g_n)
axis([-2.25 12 -0.01 1.1])
yl = 0:1/8:1;
set(gca, 'YTick', yl);
set(gca, 'YTickLabel', {'0' '1/8' '2/8' '3/8' '4/8' '5/8' '6/8' '7/8' '1' });
xticks(n);
xlabel('n')
ylabel('g[n]')
title('I&D Filter Step Response g[n] (N=8)')

figure;
stem(n,h2_n)
axis([-2.25 20.5 -0.01 0.25])
xlabel('n','Interpreter','Latex')
ylabel('$$\hat{h}[n]$$','Interpreter','Latex')
```

```
title('IIR Filter Impulse Response $$\hat{h}[n] (\alpha = 7/9)$$
','Interpreter','Latex')


figure;
stem(n,g2_n)
axis([-2.25 20.5 -0.01 1.1])
xlabel('n','Interpreter','Latex')
ylabel('$$\hat{g}[n]$$','Interpreter','Latex')
title('IIR Filter Step Response $$\hat{g}[n] (\alpha =
7/9)$$','Interpreter','Latex')
```

EE567_project_2a.m

```
EbN0_dB = -10:0.5:12;
 EbN0 = 10.^(EbN0_dB/10);

P_b = qfunc(sqrt(2*EbN0));
N = 8;
sigmasq = 1;
A = sqrt(EbN0*sigmasq*2*(1/N));
y = zeros(1,length(EbN0));
count = zeros(1,length(EbN0));
trials = 3000000;

for j = 1:trials
    for i = 1:length(EbN0)
        n_k = randn([1,N]).*sqrt(sigmasq);
        s_k = A(i)+n_k;
        y(i) = (1/N)*(sum(s_k));
        if y(i) <= 0
            count(i) = count(i)+1;
        end
    end
end

BER = count/trials;

figure;
p1 = semilogy(EbN0_dB,P_b);
hold on;
p2 = semilogy(EbN0_dB,BER);


grid on;
axis tight;
xlabel('Eb/N0 (dB)')
ylabel('Bit Error Rate (BER)')
title("Theoretical P_b vs. Simulated BER I&D LPF (N=8)("+trials+" trials)")

m1 = "Theoretical Probability of Error (P_b)";
m2 = "Simulated Bit Error Rate I&D LPF(BER)";
legend([p1;p2],[m1;m2]);
```

EE567_project_2b_1.m

```
%Run after EE567_project_2a.m w/o clearing workspace
alpha = 7/9;
EbN0_dB = -10:0.5:12;
EbN0 = 10.^(EbN0_dB/10);
N = 8;
sigmasq = 1;
A = sqrt(EbN0*sigmasq*2*(1/N));
y = zeros(N,length(EbN0));


count = zeros(1,length(EbN0));
trials = 10000000;

for k = 1:trials
        n_k = randn([1,N]).*sqrt(sigmasq);
    for i = 1:length(EbN0)
        s_k = A(i)+n_k;
        y(1,i) = (1-alpha)*s_k(1);

        for j = 2:N
            y(j,i) = (1-alpha)*s_k(j) + alpha*y(j-1,i);


        end
        if y(N,i) <= 0
            count(i) = count(i)+1;
        end
    end
end
```

EE567_project_2c.m

```
alpha =  0.98:0.0001:1-0.0001;
EbN0_dB = 7;
EbN0 = 10.^(EbN0_dB/10);
N = 8;
sigmasq = 1;
A = sqrt(EbN0*sigmasq*2*(1/N));
y = zeros(1,N);
count = zeros(1,length(alpha));
trials = 10000000;

for k = 1:trials
        n_k = randn([1,N]).*sqrt(sigmasq);
        s_k = A+n_k;
    for i = 1:length(alpha)
        y(1) = (1-alpha(i))*s_k(1);
        for j = 2:N
            y(j) = (1-alpha(i))*s_k(j) + alpha(i)*y(j-1);
        end
        if y(N) <= 0
            count(i) = count(i)+1;
        end
    end
end


BER = count/trials;

plot(alpha,BER);
grid on;
axis tight;
xlabel('\alpha')
ylabel('Bit Error Rate (BER)')
title(['BER vs. \alpha (EbN0 = ',num2str(EbN0_dB),' dB)(', num2str(trials) ,'
trials)'])
```

EE567_project_2c_2.m

```
alpha = 7/9;
alpha2 = 0.999;
EbN0_dB = -10:0.5:12;
EbN0 = 10.^(EbN0_dB/10);
N = 8;
sigmasq = 1;
A = sqrt(EbN0*sigmasq*2*(1/N));
y = zeros(N,length(EbN0));
y2 = zeros(N,length(EbN0));

count = zeros(1,length(EbN0));
count2 = zeros(1,length(EbN0));
trials = 10000000;

for k = 1:trials
        n_k = randn([1,N]).*sqrt(sigmasq);
    for i = 1:length(EbN0)
        s_k = A(i)+n_k;
        y(1,i) = (1-alpha)*s_k(1);
        y2(1,i) = (1-alpha2)*s_k(1);
        for j = 2:N
            y(j,i) = (1-alpha)*s_k(j) + alpha*y(j-1,i);
            y2(j,i) = (1-alpha2)*s_k(j) + alpha2*y2(j-1,i);

        end
        if y(N,i) <= 0
            count(i) = count(i)+1;
        end
        if y2(N,i) <= 0
            count2(i) = count2(i)+1;
        end
    end
end


BER = count/trials;
BER2 = count2/trials;


p3 = semilogy(EbN0_dB,BER);
hold on;
p4 = semilogy(EbN0_dB,BER2);
grid on;
axis tight;
xlabel('Eb/N0 (dB)')
ylabel('Bit Error Rate (BER)')
title(['IIR LPF \alpha = 7/9 vs. \alpha = 0.999 (', num2str(trials),' trials)']);
m3 = "Simulated Bit Error Rate IIR LPF \alpha= "+alpha;
m4 = "Simulated Bit Error Rate IIR LPF \alpha= "+alpha2;
legend([p3;p4],[m3;m4]);
```

EE567_project_3a.m

```
SNR_dB = 0:0.5:15;
SNR = 10.^(SNR_dB/10);
sigmasq = 1;
A = sqrt(SNR.*2*sigmasq);
phi = 0;
Pfa = 10^-4;
T0 = -2*sigmasq*log(Pfa);
trials = 10000000;
signal = zeros(1,length(SNR));
nosignal = zeros(1,length(SNR));

for k = 1:trials
    n1 = randn(1,length(SNR)).*sqrt(sigmasq);
    n2 = randn(1,length(SNR)).*sqrt(sigmasq);
    for i = 1:length(SNR)
        r1T = A(i)*cos(phi)+n1(i);
        r2T = A(i)*sin(phi)+n2(i);
        zT = r1T.^2+r2T.^2;
        if  zT > T0
            signal(i) = signal(i) + 1;
        end
        zT = n1(i).^2+n2(i).^2;
        if  zT > T0
            nosignal(i) = nosignal(i) + 1;
        end
    end
end

P_d = signal./trials;
P_fa = sum(nosignal)/(trials*length(SNR));  %simulated value

semilogy(SNR_dB,(1-P_d));
hold on;

grid on; axis tight;
xlabel('SNR (dB)')
ylabel('P_d')
title("Probability of Detection Direct Integration ("+trials+" trials)");

set(gca,'ydir','reverse')
label = [0.000001 0.00001 0.0001 0.001 0.01 0.1];
revlabel = 1-label;
yticks(label)
yticklabels(revlabel);

p1 = plot(0,0);
m1 = "Simulated P_f_a = "+P_fa;
legend([p1],[m1]);
```

EE567_project_3b.m

```
SNR_dB = 0:0.5:15;
SNR = 10.^(SNR_dB/10);
sigmasq = 1;
A = sqrt(SNR.*2*sigmasq);
phi = 0;
Pfa = 10^-4;
T0 = 4.41;
N=16;
trials = 10000000;
signal = zeros(1,length(SNR));
nosignal = zeros(1,length(SNR));
zT = zeros(1,N);
zT2 = zeros(1,N);

for k = 1:trials
    n1 = randn(N,length(SNR)).*sqrt(sigmasq);
    n2 = randn(N,length(SNR)).*sqrt(sigmasq);
    for i = 1:length(SNR)
        for j = 1:N
            r1T = A(i)*cos(phi)+n1(j,i);
            r2T = A(i)*sin(phi)+n2(j,i);
            zT(j) = (1/N)*(r1T.^2+r2T.^2);
            zT2(j) = (1/N)*((n1(j,i).^2+n2(j,i).^2));
        end
        if  sum(zT) > T0
             signal(i) = signal(i) + 1;
        end
        if  sum(zT2) > T0
             nosignal(i) = nosignal(i) + 1;
        end
    end
end

P_d = signal./trials;
P_fa = sum(nosignal)/(trials*length(SNR));  %simulated value

semilogy(SNR_dB,1-P_d);
hold on;
grid on; axis tight;
xlabel('SNR (dB)')
ylabel('P_d')
title("P_d with post detection integration (N=16) ("+trials+" trials)")

set(gca,'ydir','reverse')
label = [0.0000001 0.000001 0.00001 0.0001 0.001 0.01 0.1];
revlabel = 1-label;
yticks(label)
yticklabels(revlabel);

p1 = plot(0,0);
m1 = "Simulated P_f_a = "+P_fa;
legend([p1],[m1]);
```

EE567_project_3c.m

```
SNR_dB = 0:0.5:15;
SNR = 10.^(SNR_dB/10);
sigmasq = 1;
A = sqrt(SNR.*2*sigmasq);
phi = 0;
Pfa = 10^-4;
T0 = 0.27925;
N=16;
M = 8;
trials = 10000000;
signal = zeros(1,length(SNR));
nosignal = zeros(1,length(SNR));
zT = 0;
zT2 = 0;

for k = 1:trials
    n1 = randn(N,length(SNR)).*sqrt(sigmasq);
    n2 = randn(N,length(SNR)).*sqrt(sigmasq);
    for i = 1:length(SNR)
        for j = 1:N
            r1T = A(i)*cos(phi)+n1(j,i);
            r2T = A(i)*sin(phi)+n2(j,i);
        if  1/N*(r1T.^2+r2T.^2) > T0
             zT = zT+1;
        end
        if  1/N*(n1(j,i).^2+n2(j,i).^2) > T0
             zT2 = zT2+1;
        end
        end
        if zT >= M
            signal(i) = signal(i)+1;
        end
        if zT2 >= M
            nosignal(i) = nosignal(i)+1;
        end
        zT = 0;
        zT2 = 0;


    end
end

P_d = signal./trials;
P_fa = sum(nosignal)/(trials*length(SNR));  %simulated value

semilogy(SNR_dB,1-P_d);
hold on;
grid on; axis tight;
xlabel('SNR (dB)')
ylabel('P_d')
title("P_d with M of N logic (N=16, M=8) ("+trials+" trials)")

set(gca,'ydir','reverse')
label = [0.000001 0.00001 0.0001 0.001 0.01 0.1];
revlabel = 1-label;
```

```
yticks(label)
yticklabels(revlabel);
p1 = plot(0,0);
m1 = "Simulated P_f_a = "+P_fa;
legend([p1],[m1]);
```