# PathSim User

# and

# Technical Guide

## Ver. 1.0

## Dec 1, 2000

**by**

**Moe Wheatley, AE4JY**
**ae4jy@mindspring.com**

# Table of Contents

# 1. Getting Started

## *1.1. Introduction*

PathSim is a program that allows simulation of various radio propagation conditions using a PC and Windows soundcard. It's main purpose is to be able to take audio input from either a soundcard or wave file, and distort this audio signal in a manner similar to how a signal is distorted by the ionosphere and other affects. This distorted signal can then be sent to a sound card or wave file. This allows comparing various modulation modes against various signal conditions.

This program is made available free of charge for amateur radio use only and is licensed under GPL.

## *1.2. System Requirements*

PathSim requires Windows 95,98, or NT 4.0 running on at least a 133MHz Pentium in order to simulate in real time using the soundcard. The program and its help documents eat up several Megs of disk space. The program needs several Meg of RAM. If one does not use the soundcard and uses wave files for I/O, a slower system can be used.

## *1.3. Program Installation*

The program consists of a single executable file "PathSim.exe". This file can be placed anywhere but is probably best to create a new folder(directory) for it say at C:\PathSim\. If online help is desired then you also need the file "PathSim.hlp" and "PathSim.cnt" to also be in the same directory. Windows will create a few other files if Help is used. Several “canned” HF simulation condition setup files are available as well that have a suffix of *.sim.

## *1.4. Program Removal*

To remove this program, just delete all the files in the folder where PathSim.exe file resides.

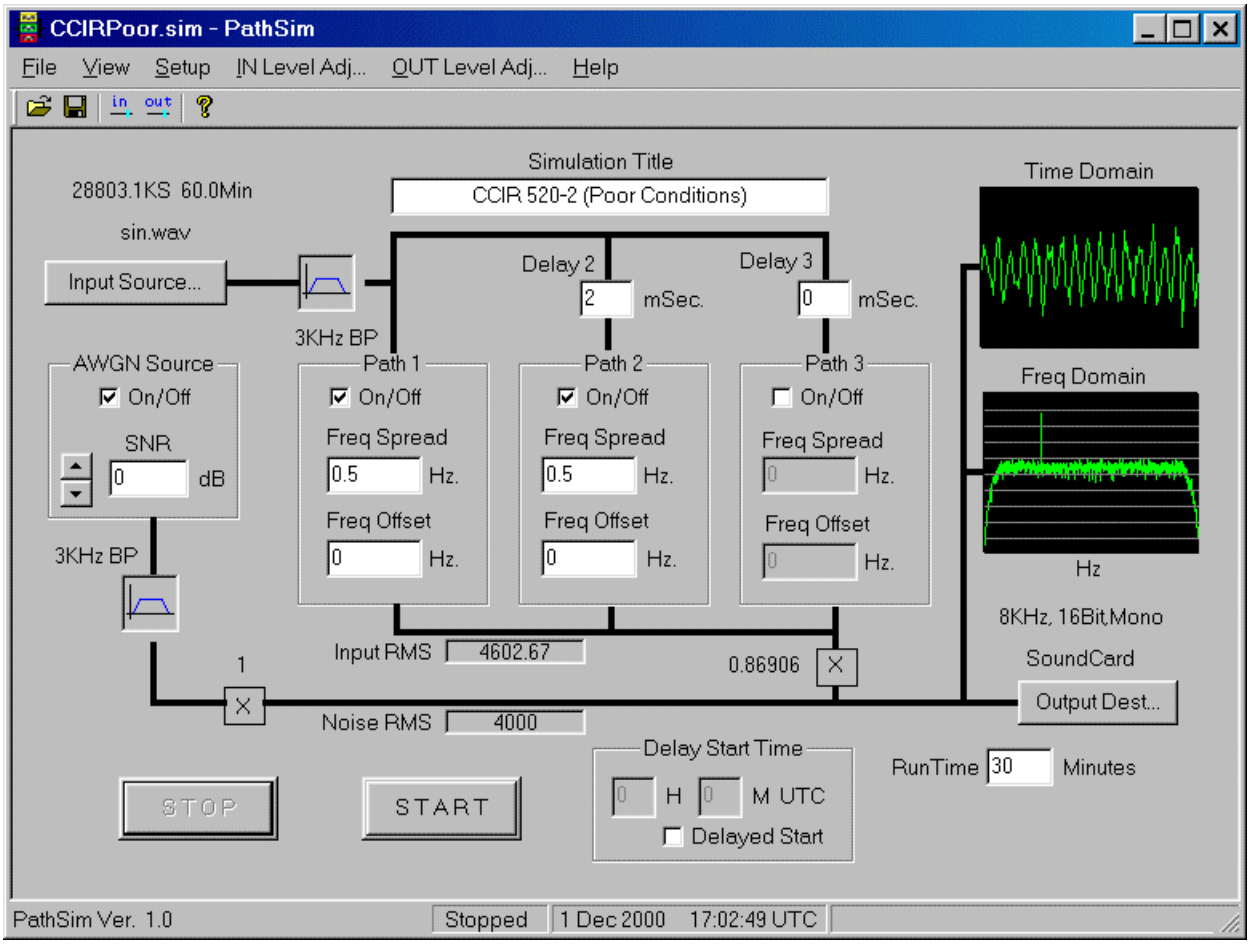
Purists may also want to go into "Regedit" and look in the "HKEY_CURRENT_USER\Software" folder for a registry key called "AE4JY Software/PathSim". Sselect it and hit delete and your system should no longer have any knowledge of PathSim. Leaving this key may eat up a few bytes of disk space but will not affect any other programs.

## 1.5. Running PathSim

Once the program is placed in the directory you wish, just double click on it in Explorer and it should begin operation.

Hint: while in Explorer, select PathSim.exe with the mouse and RIGHT click on it. A menu will pop up and select "Create Shortcut". A Windows shortcut to PathSim will now appear in Explorer and you can drag it off onto your desktop or anywhere else. PathSim can now be executed by clicking on the new shortcut, even though it is not located in the folder as all the rest of the files.

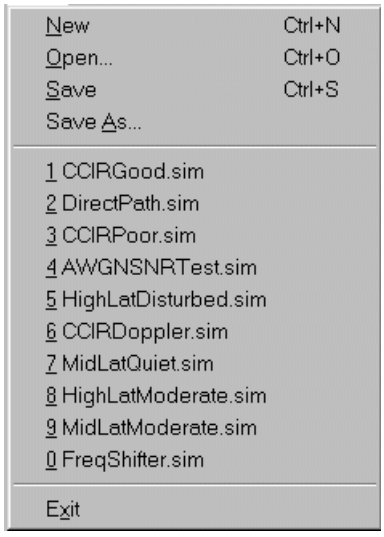
The program's main screen gives an overall block diagram of the signal flow as well as all the user settings that can be adjusted.



# 2. PathSim Reference

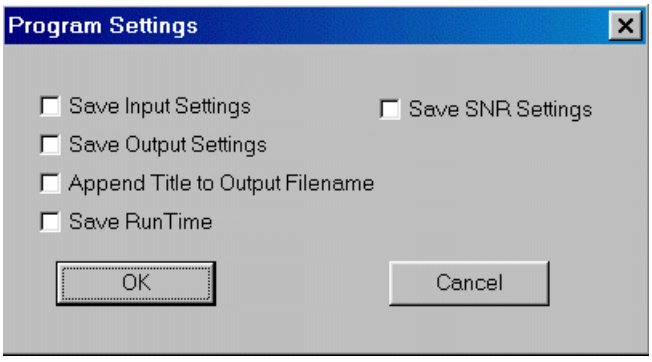
## *2.1. Menu Options*


The file menu allows the saving and reading back simulation setups. A MRU list show the last 10 files that have been used.




The View menu allows setting the screen to display the toolbar and status bar or not. The Always on Top can be selected to keep the program on top of all others.


The Setup menu allows one to specify what is saved into the simulation setup files. The items that can be selected are the input and output source/destination as well as the run time limit. Another setting allows appending the simulation Title to the output wave filename. This can be useful when running many different simulation setups and having to rename the output file each time.

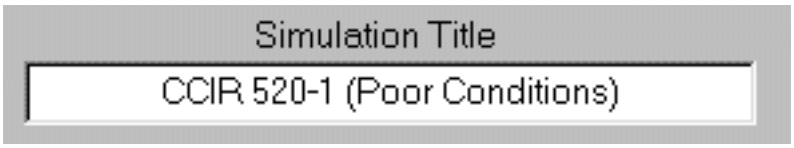



The INLevel and OUTLevel menus bring up the Windows mixer controls for adjusting the soundcard levels.

The Help item brings up any online help or other program information.

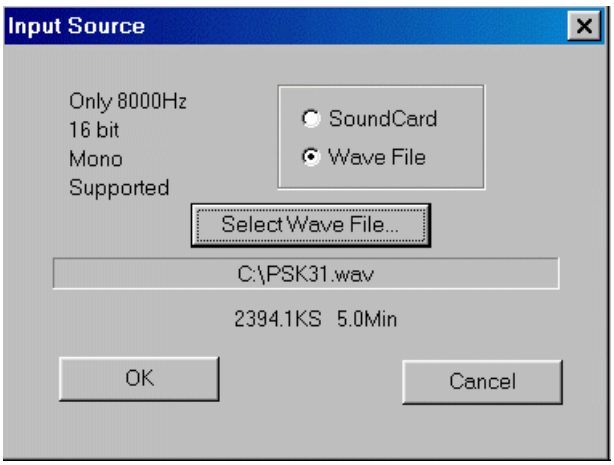
### 2.2. Title Edit box

A Title can be entered for a simulation session and is saved off along with the simulation settings. This text can also be automatically appended to the output filename if desired.

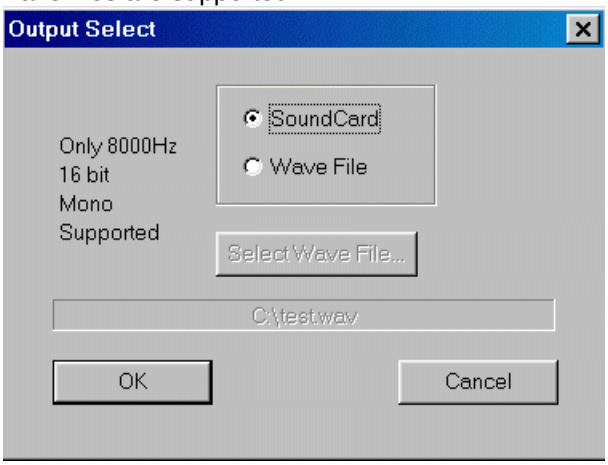



### 2.3. Input/Output Controls

The Input source allows the selection of either the soundcard or a wave file as a source for the input audio. Only 8000Hz, 16 bit, mono sound and wave files are supported.

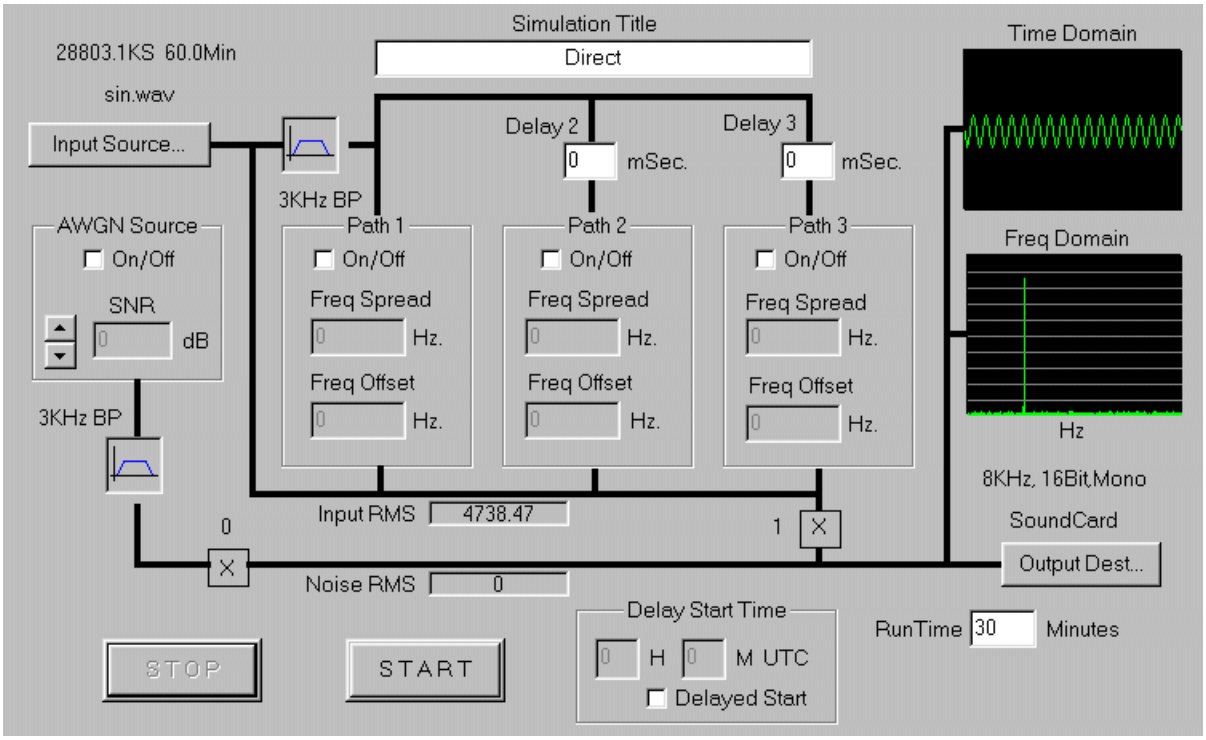



The Output Source allows the selection of either the soundcard or a wave file as a source for the input audio. Only 8000Hz, 16 bit, mono sound and wave files are supported.



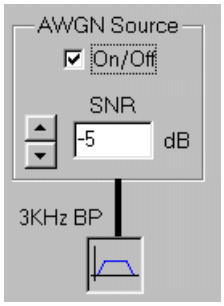
## 2.4. Direct Path Mode

The simplest mode of operation is when the HF path sections are all disabled. The audio then goes directly to the output with only optional AWGN added. If the AWGN source is disabled, then the program operates as a simple digital audio recorder and playback device. The output will be the same as the input.



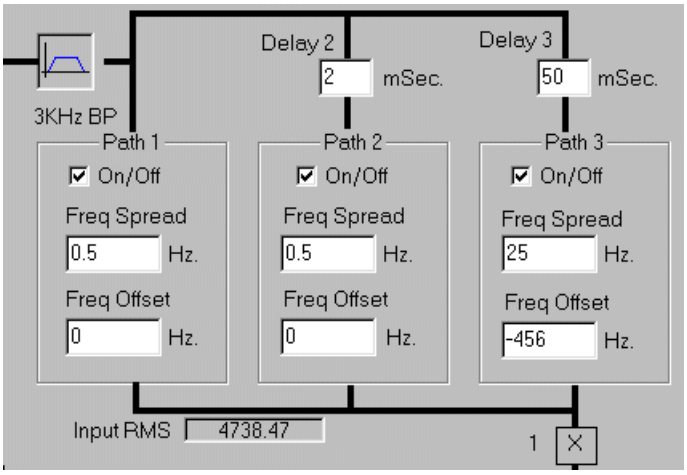

## 2.5. AWGN Source

If the AWGN source is enabled then Gaussian white noise can be added to the input signal in order to simulate various SNR ratios. An SNR of 0 means that the input signal rms level is equal to the noise rms level as measured through the 3KHz bandpass filter.



## 2.6. HF Path Controls

If any of the HF Paths are enabled, then the direct connection goes away and the input signal is first Band-pass filtered then divided among the active paths with a variable time delay. The first path has no delay but paths 2 and 3 can have time delays from 0 to 50 mSec.




Each path can be enabled or disabled.

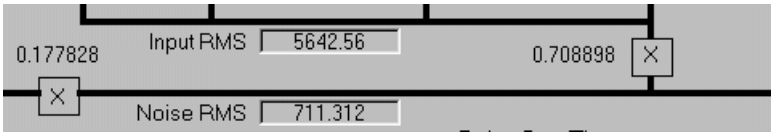
The frequency spread for each path can be specified ranging from .1 Hz to 30Hz.

A frequency offset can also be applied to each path. This can be used to simulate Doppler shift. The range is +/- 1000Hz.

## 2.7. Signal Displays

Several signal measurements can be viewed in real time. One is the average "Input RMS" signal value of the input signal after passing through any paths. This value, multiplied by the displayed multiplier value, will be added with any AWGN to create the final output signal.
The AWGN value that is added is displayed in the "Noise RMS" box.

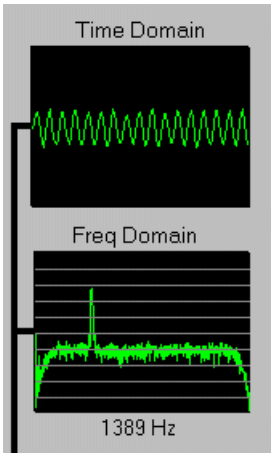
In the example below, the SNR was set to +15dB.




The output RMS is 5642.56 x 0.708898 = 4000.
The Noise RMS is 711.312.

$$SNR = 20\log(\frac{4000}{711.312})$$ The SNR is 15dB

Two graphics windows show the output signal in the time domain and frequency domain for observation purposes. If the output signal is too high, the time domain display will turn red indicating that the input needs to be turned down.

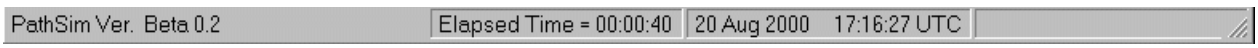



The frequency display is a simple 2048 point FFT plot with each vertical division equal to 10 dB. If the mouse cursor is in the plot area, then the frequency of the pointer is displayed bellow the plot. Due to the size of the plot, the resolution is not very good.

## *2.8. Other Functions/Displays*


The simulation start can be delayed until a specified UTC time by selecting delayed start, and entering the hour and minute that you wish to start the simulation. This gives up to 24 hours of start delay. The Run Time can be selected from 1 minute to 500 minutes.
This option is useful if you select a direct simulation without any AWGN and just use the program as a digital audio recorder. Audio can be recorded to a wave file at a predetermined time say in the middle of the night to check signals on a band while you are asleep or away from the radio. The data rate is about a megabyte per minute so make sure you have sufficient disk space before starting. The program will show about how much disk space is free and how much is needed.

## *2.9. Status bar*




The Status bar shows the UTC time and data, as well as the current running status of the simulation.

# 3. Technical Description

## *3.1. Watterson Ionospheric Channel Model*

Radio propagation via the earth's ionosphere contain reflections from 3 general effects:

- Density variations versus altitude in the ionosphere
- Magnetic-ionic effects that cause polarization-dependent paths
- Non-uniformity of the ionospheric layers

The Watterson HF channel model has been used for some time for simulating the propagation of radio waves reflecting off of the ionosphere by simulating both amplitude and phase distortion in multiple propagation paths each with a specified fixed delay.

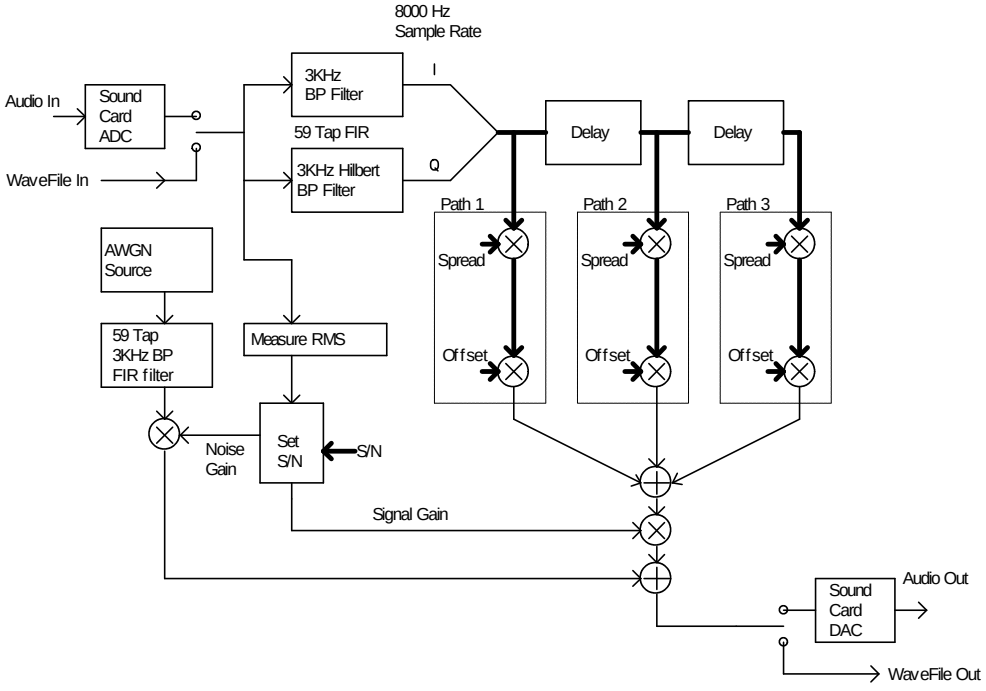
For a much more detailed description, one is referred to an article in the May/June 2000 issue of QEX by Johan Forrer describing the details of such a model. Another reference can be found in "Wireless Digital Communications: Design and Theory" by Tom McDermott.

## *3.2. Program Description*

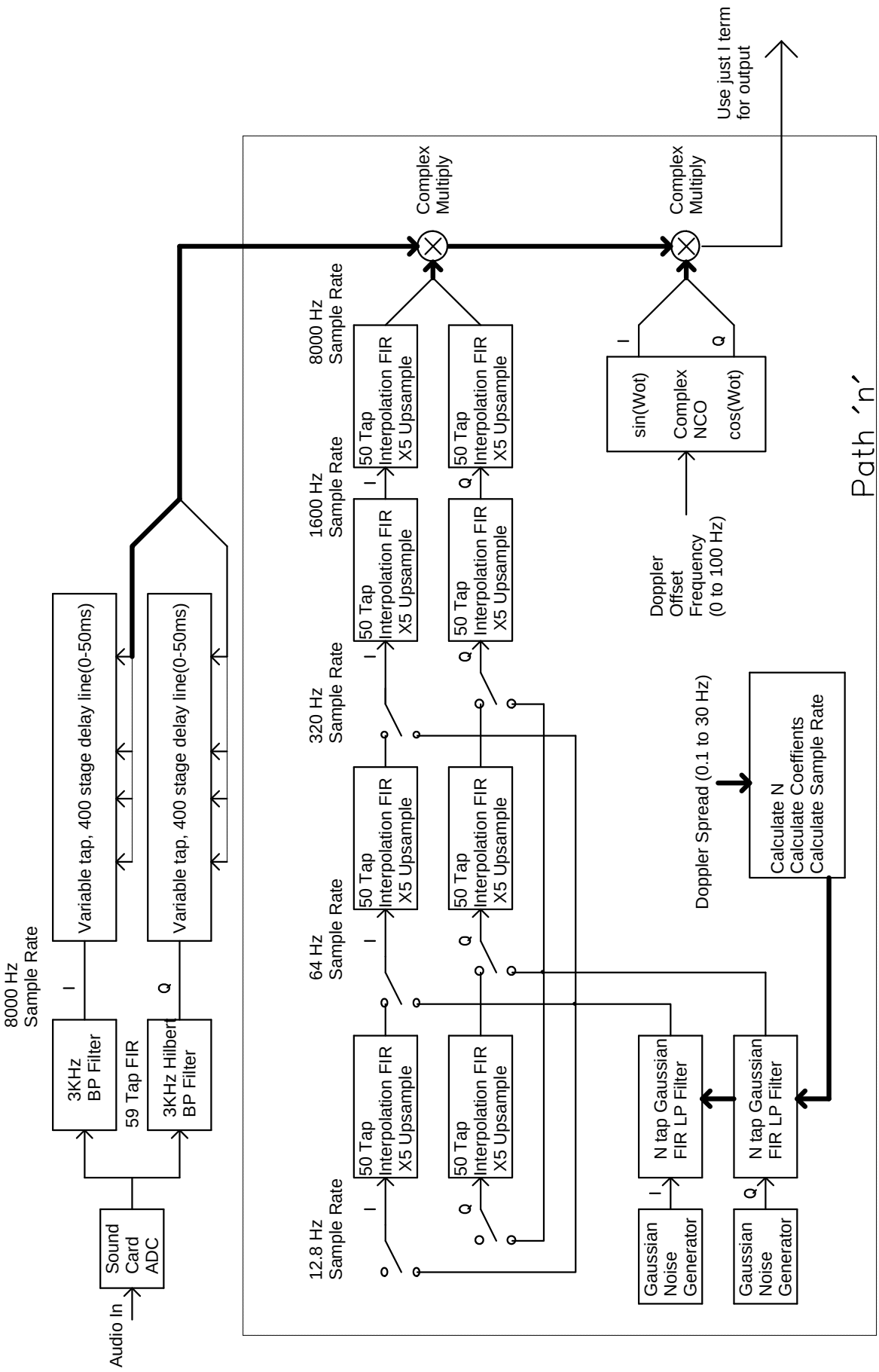
### 4.1.1 Simulation Path Block Diagram

The following block diagram shows the overall simulator data paths.

The basic goal is to have unity gain between the input signal and the final summed signal from all the paths. So a 1 volt" RMS input sine wave will produce a 1 "volt" RMS output signal after being delayed, frequency and phase modulated, and frequency shifted by the various paths. This final output signal can then have Gaussian noise added to it to produce a specified S/N ratio over the 3KHz bandwidth of the simulator.




The following block diagram shows the components used to create one of the three possible simulation paths.


Audio In
Sound Card ADC
8000 Hz Sample Rate
3KHz BP Filter
59 Tap FIR
3KHz Hilbert BP Filter
I
Q
Variable tap, 400 stage delay line(0-50ms)
Variable tap, 400 stage delay line(0-50ms)
Path 'n'
12.8 Hz Sample Rate
64 Hz Sample Rate
320 Hz Sample Rate
1600 Hz Sample Rate
8000 Hz Sample Rate
50 Tap Interpolation FIR X5 Upsample
Gaussian Noise Generator
N tap Gaussian FIR LP Filter
Doppler Spread (0.1 to 30 Hz)
Calculate N
Calculate Coeffients
Calculate Sample Rate
Doppler Offset Frequency (0 to 100 Hz)
sin(Wot)
Complex NCO
cos(Wot)
Complex Multiply
Complex Multiply
Use just I term for output

### 4.1.2 Hilbert Transform BP FIR Filter

The input audio signal is first BP filtered and converted into a complex signal I and Q. This is done using two matched 3KHz wide bandpass filters where one of them also is a Hilbert transform filter that provides the 90 degree phase shift over the bandpass of the filter.
The basic design method is as follows:

a. Design a lowpass FIR filter with a passband of ½ the desired BP passband and also to whatever stopband goals.
b. Convert the LP coefficients to I and Q BP FIR coefficients using the following equations.

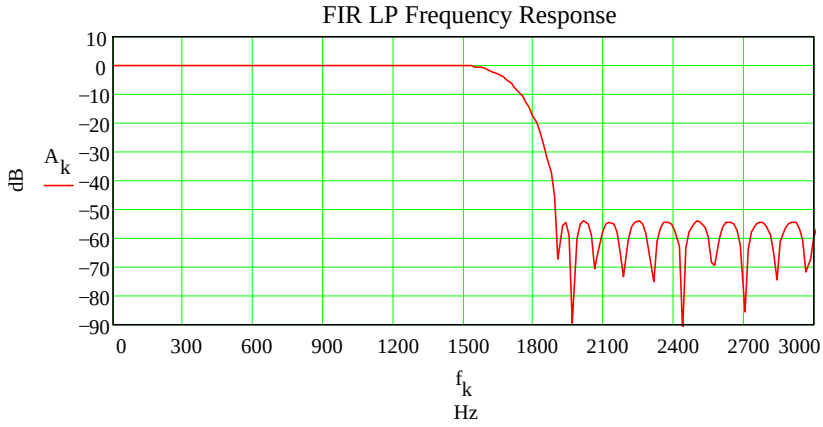
$$h_{IBP}(n) = 2h_{LP}(n)\cos(2pf_0[n - \frac{(N-1)}{2}]T)$$

$$h_{QBP}(n) = 2h_{LP}(n)\sin(2pf_0[n - \frac{(N-1)}{2}]T)$$

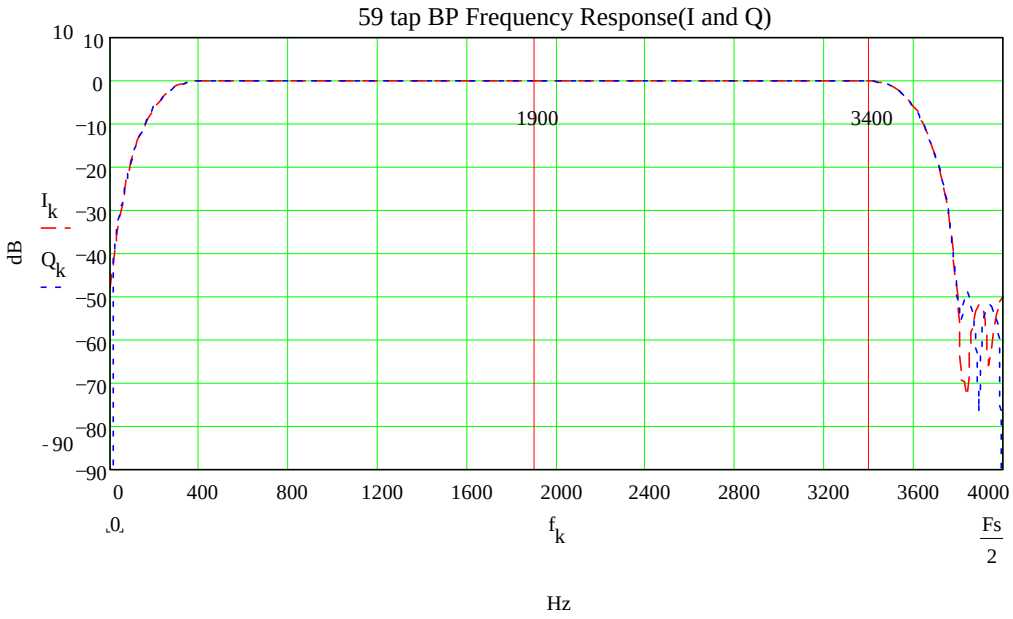
where:

$h_{LP}(n) = \text{n}^{\text{th}}$ Lowpass FIR Coefficient
$f_0 = $ Bandpass Center Frequency
$N = $ Number of Coefficients
$T = $ Sample Period

For PathSim, the low pass filter was designed for a 1500 Hz pass band and a 1900Hz stop band.



Converting to a band pass filter centered at1900Hz resulted in the following pair of filters:




### 4.1.3 Delay Block

The delay line delays both the I and Q signals by "n" samples up to 400 samples (50 mSec max delay). This is performed using a circular buffer with pointers that lag behind by the specified amount of delay.

### 4.1.4 Gaussian Noise Generator

Two random Gaussian numbers are first generated, one for the I, and one for the Q channel. When finally multiplied by the incoming signal, they create a Rayleigh distribution and create the desired random phase and amplitude modulation required by the Watterson model. The Gaussian numbers are generated by generating two uniformly distributed numbers that are constrained to lie inside a unit circle. The following code segment creates 2 Gaussian distributed variables with a standard deviation of STD.

```
// Generate two uniform random numbers between -1 and +1
// that are inside the unit circle
do {
        u1 = 1.0 - 2.0 * (double)rand()/(double)RAND_MAX ;
        u2 = 1.0 - 2.0 * (double)rand()/(double)RAND_MAX ;
        r = u1*u1 + u2*u2;
} while(r >= 1.0 || r == 0.0);
rad = sqrt(-2.0*log(r)/r);
G1 = u1*rad*STD;
G2 = u2*rad*STD;
```

### 4.1.5 Gaussian Shaped LP Filter

The Gaussian variables are then LP filtered to create the desired spreading frequency. A FIR LP filter is used that is pre-calculated to have a Gaussian shaped amplitude response. It turns out that a Gaussian shaped FIR impulse response creates a Gaussian shaped frequency response. The trick is figuring out the frequency and amplitude scaling factors. Mathcad was used to design the algorithm and plot the FIR response versus the ideal Gaussian shape.

Several requirements need to be met to accurately implement a Watterson model simulator.

a. The spreading frequency is the 2-Sigma frequency of the Gaussian shaped LP filter.
b. The LP filter shaped should be Gaussian shaped down to at least –35 dB.
c. The filter needs to be calculated 30 times faster than the specified fading rate.
d. The spreading frequency range is from .1Hz to 30 Hz.

The basic equation for a Gaussian(Normal) distribution is shown below.

$$y(x) = \frac{1}{s\sqrt{2p}} e^{\left(-\frac{1}{2s^2}(x-m)^2\right)} \quad \text{where:} \quad s = \text{Standard Deviation}, \quad m = \text{Mean}$$

If one creates a set of FIR coefficients(Impulse response) that have a Gaussian shape, then the resulting filter is a low pass filter also with a Gaussian shape. No proof is given here, but could probably be derived using the properties of the Fourier transform of exponentials.

The following Mathcad program was used to develop and examine various Gaussian shaped filters. The one shown is for a 1Hz 2 sigma version with a length of 91 and a sample frequency of 64 Hz.

## Gaussian Shaped FIR Lowpass Filter Design

## Filter Specification

Sampling Frequency in Hz: $Fs := 64$

Shape Quality Constant $KQUAL := 1.4$ (larger the constant, the better the match)

Two Sigma Frequency in Hz: $F2\sigma := 1$

FIR Length: $N := ceil\left(KQUAL \cdot \frac{Fs}{F2\sigma}\right) + 1 \qquad N = 91$

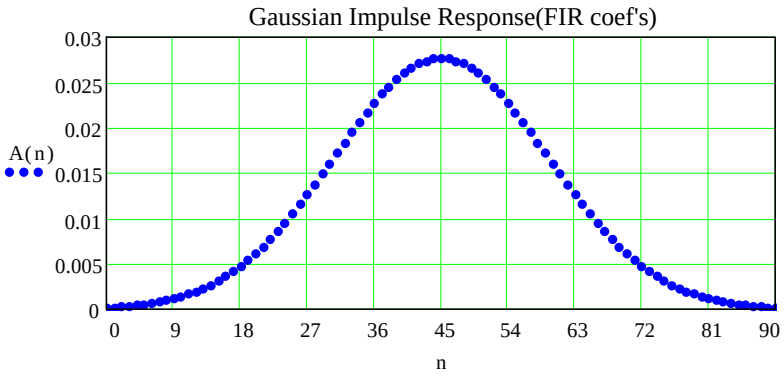
$n := 0 .. N - 1 \qquad NHALF := \frac{N-1}{2}$

$\sigma := \frac{Fs \cdot \sqrt{2}}{2 \cdot \pi \cdot F2\sigma} \qquad \sigma = 14.405$

### *Gaussian Filter Impulse Response*

$\mu := NHALF \qquad \mu = 45$

$A(f) := \frac{dnorm(f, \mu, \sigma)}{\sqrt{2 \cdot \pi} \cdot \sigma \cdot dnorm(0, 0, \sigma)}$




## *Create Ideal Gaussian reference function for plot comparisons*

$$\mu := 0$$

$$Ar(f) := 10 \cdot \log\left[\frac{dnorm\left(f, \mu, \frac{F2\sigma}{2}\right)}{dnorm\left(0, \mu, \frac{F2\sigma}{2}\right)} + 1 \cdot 10^{-100}\right]$$

## *Calculate Gaussian FIR Frequency Response from Impulse Response*

Extend the length to a power of 2 to use fft for frequency response evaluation

$$M := 16 \cdot 2^{floor\left(\frac{\log(N)}{\log(2)}\right)} \qquad m := 0 .. M - 1$$

$$h2_m := 0 \qquad h2_n := A(n)$$

$$k := 0 .. 0.5 \cdot M \qquad \delta f := \frac{Fs}{M}$$

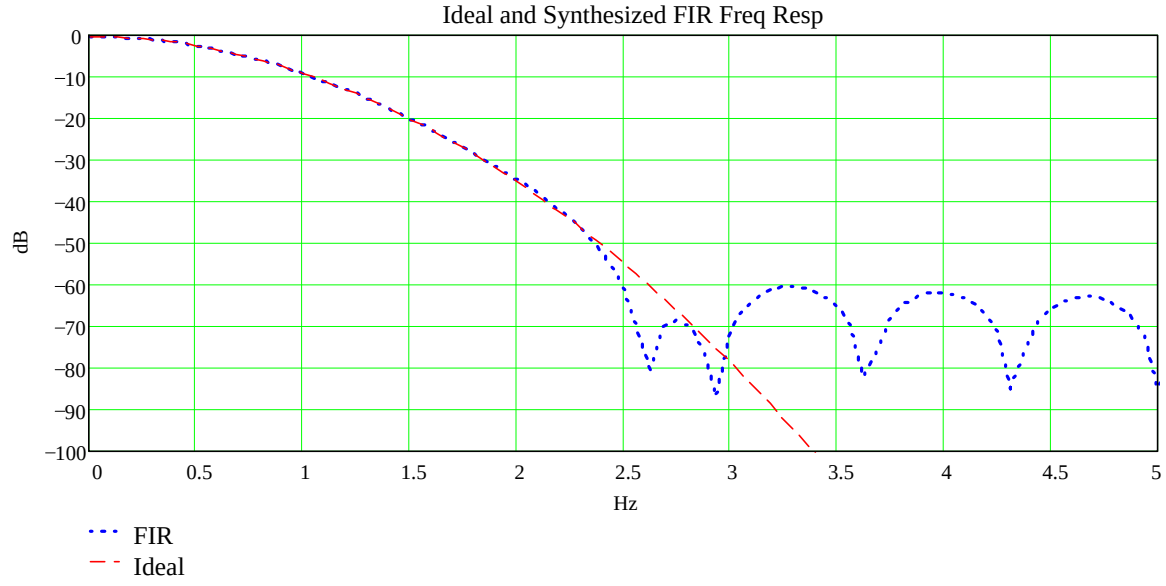
$$M = 1.024 \cdot 10^3 \qquad f_k := k \cdot \delta f$$

## *Find the frequency response from 0.0 to fmax in steps of..* $\delta f = 0.063$ Hz

$$fmax := 5 \cdot F2\sigma \qquad fmax = 5$$

$$H := fft(h2)$$

$$A_k := 20 \cdot \log\left(\left|H_k\right| \cdot \sqrt{M} + 1 \cdot 10^{-100}\right) \qquad ar_k := Ar\left(f_k\right)$$



In order to maintain a constant power bandwidth throughout the system, the equivalent noise bandwidth is calculated for each Gaussian filter setting. The input to the Gaussian filters is multiplied by the following factor to maintain a constant gain of 1 throughout the simulator.

$$Gain = \sqrt{\frac{F_s}{2F_{2s}K_{ENB}}} \quad \text{where:} \quad F_S = \text{Sample Frequency} \quad F_{2s} = \text{Two Sigma Frequency}$$

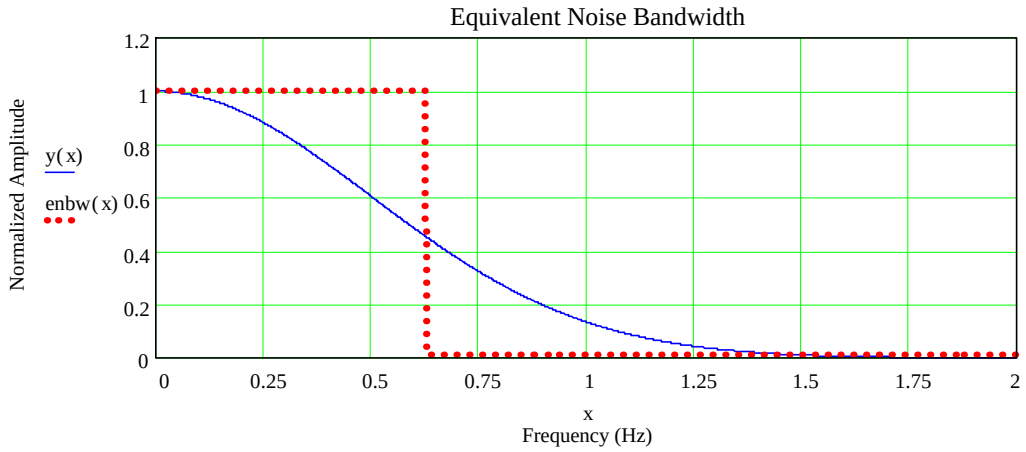
$$K_{ENB} = \text{Equivalent Noise Bandwidth of 1Hz 2 Sigma filter}$$

## Calculate Equivalent(rectangular) Noise Bandwidth of filter:

$$Nbw := \frac{F2\sigma}{4 \cdot dnorm(0,0,1)} \qquad Nbw = 0.626657 \ \text{Hz}$$

$$ymax := dnorm\left(0,0,\frac{F2\sigma}{2}\right) \qquad x := 0, .001 .. 2 \cdot F2\sigma$$

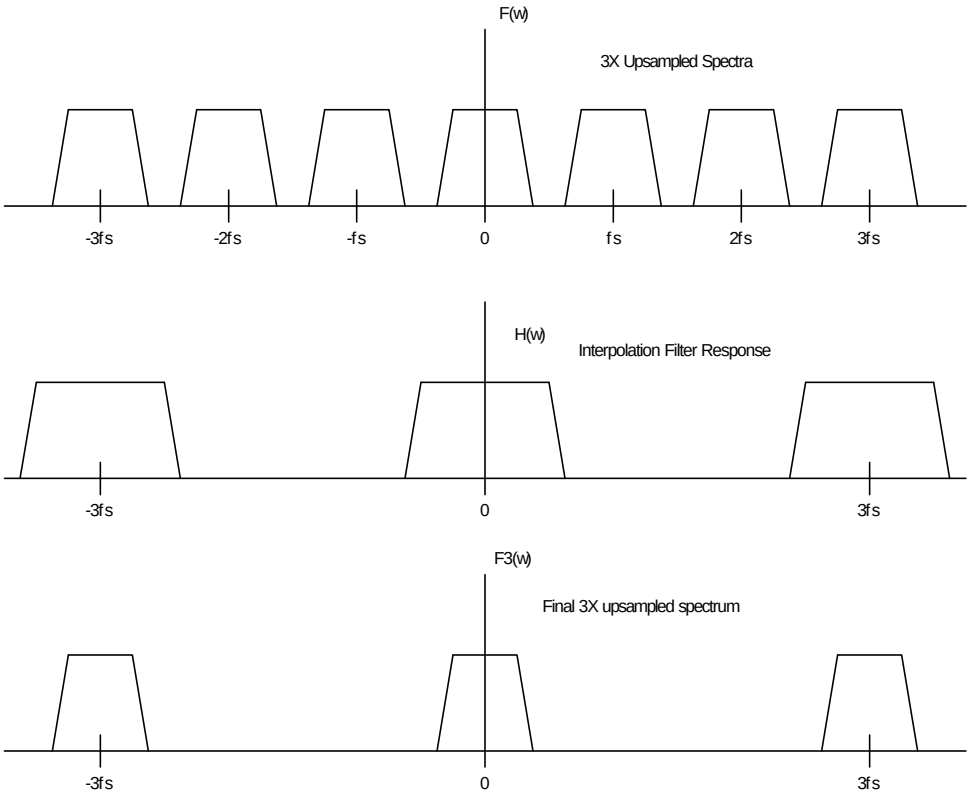
$$enbw(x) := if((x > Nbw), 0.01, 1) \qquad y(x) := \frac{dnorm\left(x,0,\frac{F2\sigma}{2}\right)}{ymax}$$



### 4.1.6 Interpolation Filter and Up-Sampling

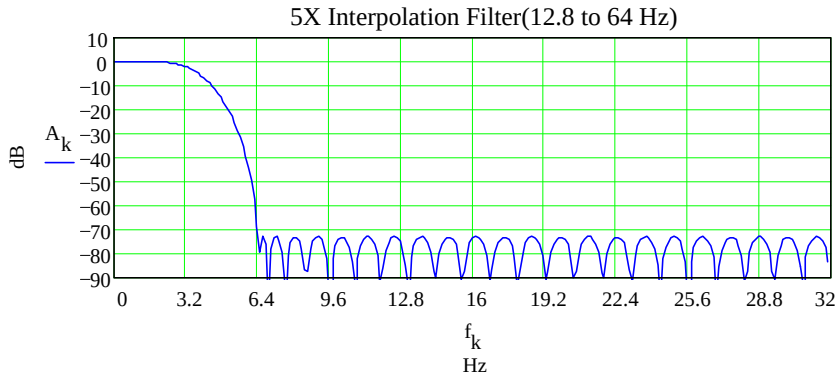
Since the main signal sample rate is 8000Hz, the samples generated by the Gaussian generator and FIR LP filters need to be up-sampled to 8000 Hz from 12.8, 64 or 320 Hz. Multiple stages of 5x up-sampling were used to keep the interpolation filters a reasonable size.
Interpolation is done by adding N-1 zero data points in between each sample to create the desired N times up samples. These new samples must be LP filtered to remove all the lower sample rate aliases.

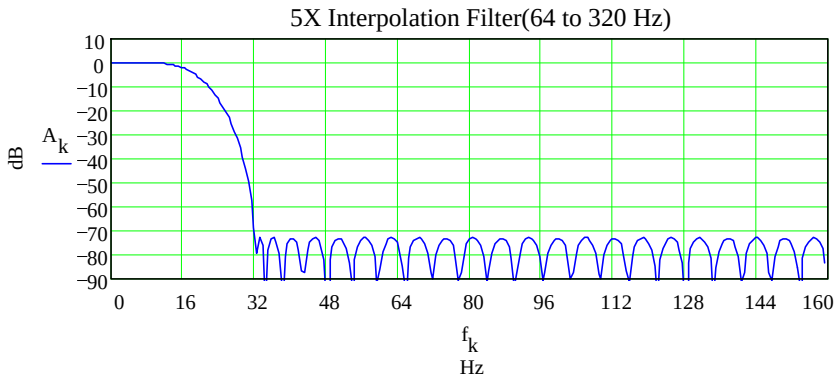
The figure below is an example of an interpolation by a factor of three.

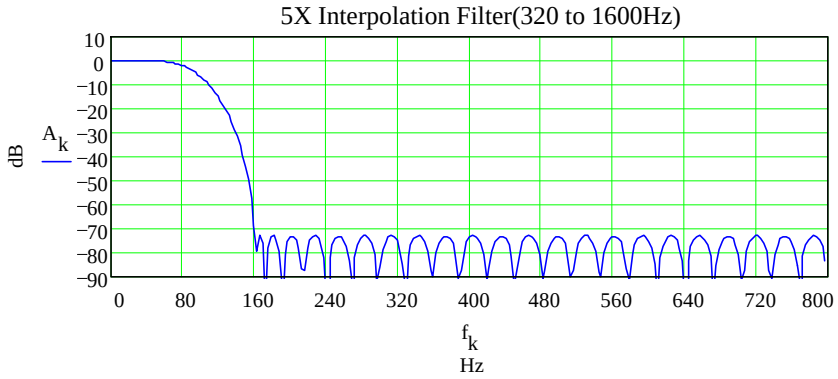


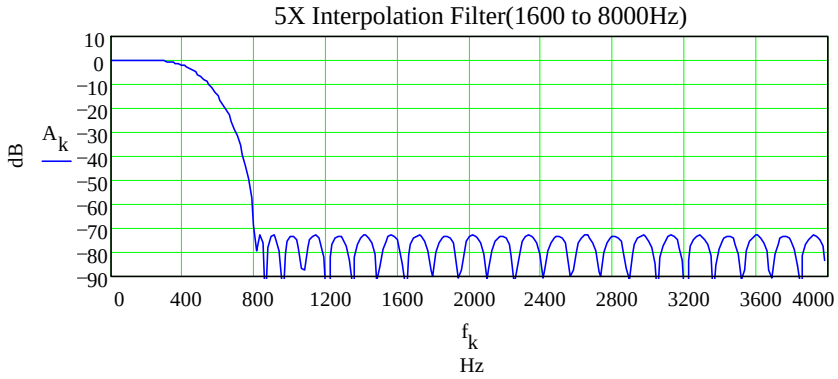

The total interpolation is broken up into stages of 5x up-sampling. This reduces the size of the filters and CPU usage. Depending on the spreading frequency, not all of the stages will be used. The lower the spreading frequency, the lower the sample frequency that is used.

The exact same FIR filter coefficients are used in all the stages. Only the sample rate is different. The following are the frequency responses of all the stages, each with a filter length of 50.



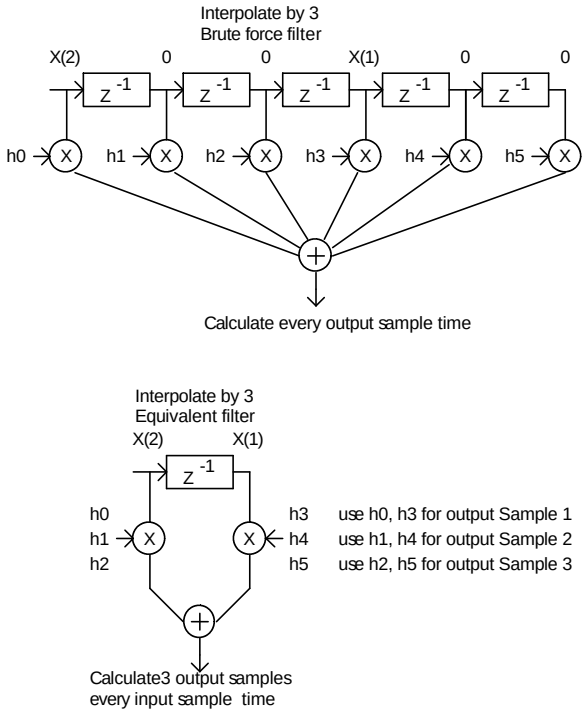



When designing an interpolation FIR filter, the pass-band gain must be multiplied by the up-sampling ratio so that the final gain is one. This is due to the adding of zeros to the input data stream.

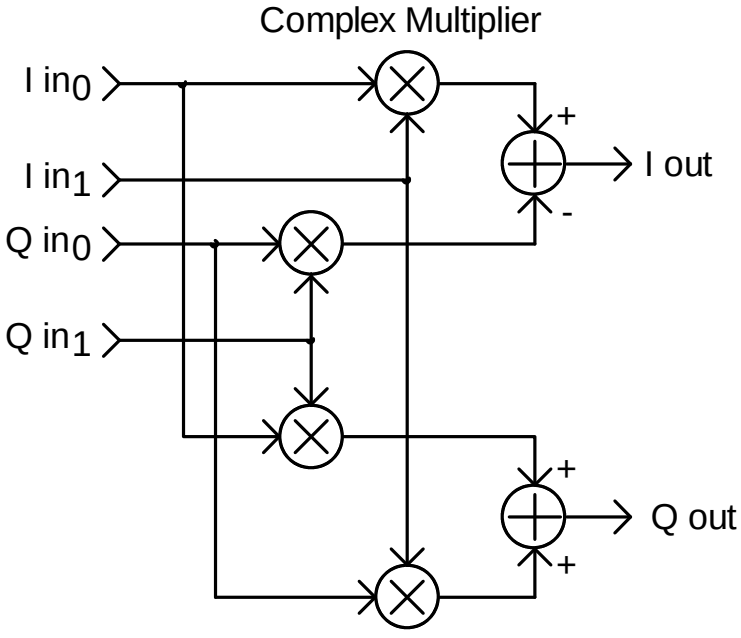
By playing with the interpolation FIR filter structure, one can implement it without actually adding zeros to the input data stream just by skipping all the multiplies by zero.




In order for this to work, the FIR length must be an integer multiple of the interpolation ratio.

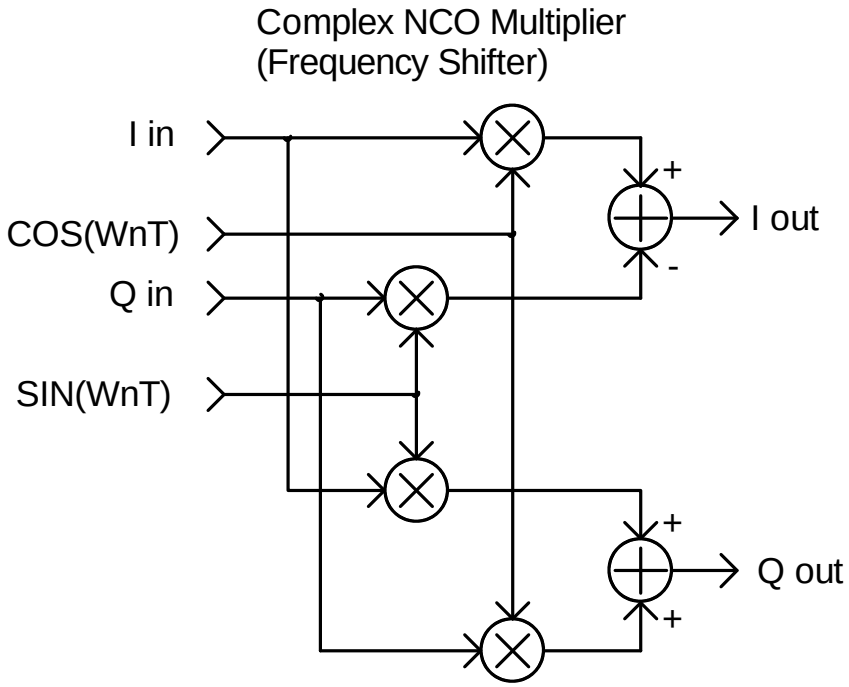
### 4.1.7 Complex Multiplier / NCO Frequency shifter

The following structure is used to perform the complex multiply of the input signal by the interpolated spreading signal. This is just the product: $(I_0 + jQ_0)(I_1 + jQ_1)$



The constant frequency offset is added to the signal by a similar complex multiply by a complex NCO signal. This shifts the incoming signal spectrum by the frequency of the NCO. Because it is complex, a negative or positive frequency can be generated to shift the signal up or down in frequency.

$$Signal_{Shifted} = e^{-jw_{shift}} Signal_{Input}$$

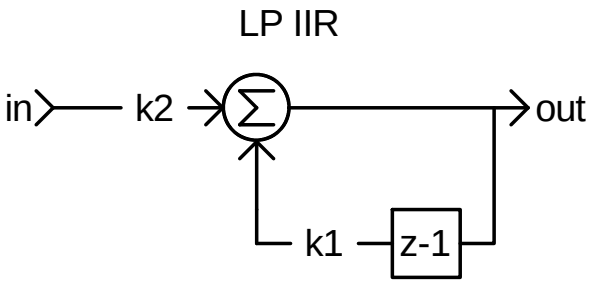


### 4.1.8 AWGN SNR Calculations

The input signal is sampled to measure its RMS power so that a specified S/N ratio can be added after all the path's have been generated.

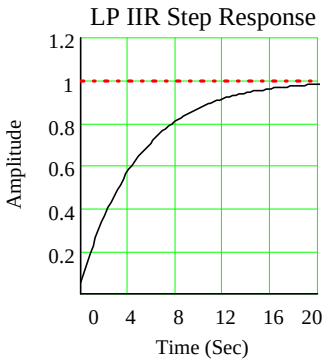
$$RMS = \sqrt{\frac{\sum_{i=0}^{N-1} x_i^{\,2}}{N}}$$

The RMS value is calculated over 2048 samples(.256 Seconds) and then low pass filtered using a simple IIR to create a slower average of the RMS signal.




The Step response of this filter is as follows:

$y_n := k1 \cdot y_{n-1} + k2 \cdot x_n$




The ratio of Signal to Noise is calculated from the set value that is in dB.

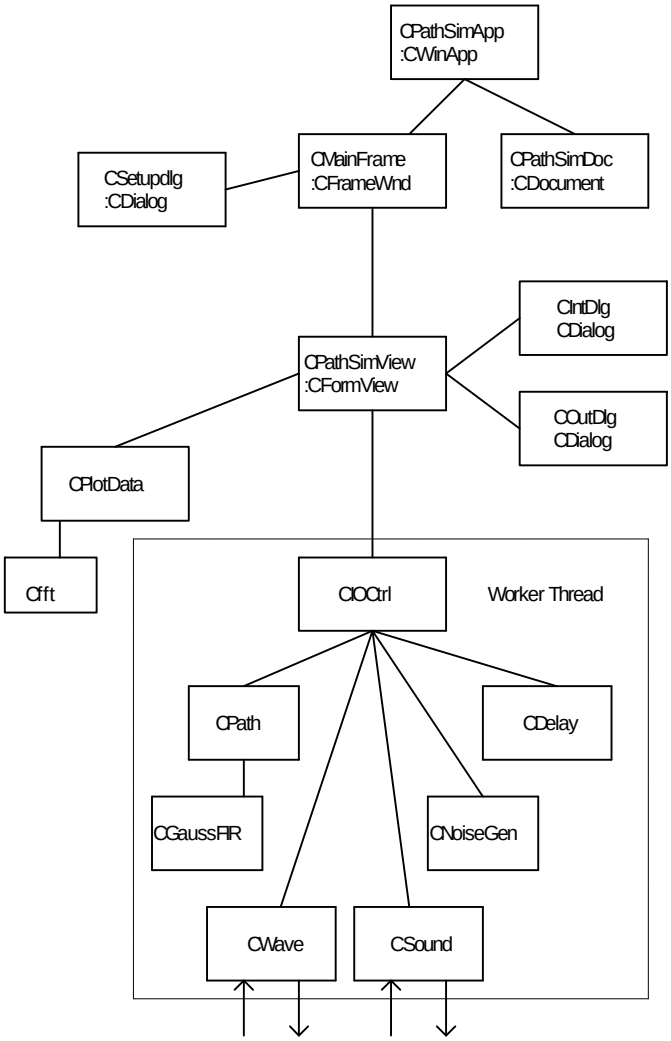
$$SNR = 10^{\frac{SNR_{dB}}{20}}$$

If the ratio is greater than 1, then the input signal gain is clamped to a maximum value and the noise level is reduced accordingly.
If the ratio is less than 1, then the noise level gain is clamped to a maximum value and the signal gain is reduced accordingly.

The Gaussian noise generator for the S/N function is also band-pass filtered using the identical FIR filter shape used to band pass filter the input signal. The equivalent rectangular noise bandwidth of this filter was measured experimentally and used to calibrate the noise power used for The S/R function.

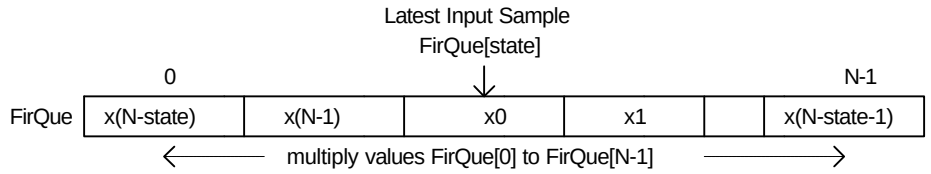
### 4.1.9 Program Organization

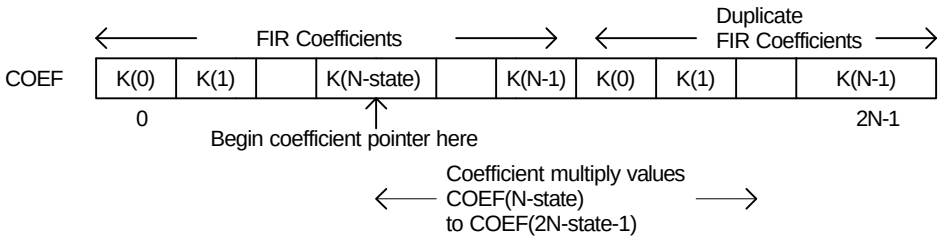
The program is written as an MFC Windows project. The user interface uses the usual range of dialog boxes and check boxes for displaying and obtaining user inputs.



### 4.1.10 FIR Filter Implementation

The FIR filters used by PathSim are optimized by utilizing a “flat” coefficient table that is twice as long as the actual coefficient table. This method eliminates a compare instruction inside the inner multiply loop of the filter making it a bit faster at the expense of bit more memory usage.
The following shows the basic buffer and pointer structure used.




**An example FIR filter calculation:**

```
double CalcFilter(double in)
{
double acc;
double* Firptr;
double* Kptr;
      pQue[FirState] = in;
      Firptr = pQue;
      acc = 0.0;
      Kptr = COEFTABLE + FIRLENGTH - FirState;
      for(INT i=0; i<FIRLENGTH; i++)
            acc += ( (*Firptr++) * (*Kptr++) );
      if( --FirState < 0)
            FirState += FIRlen;
      return acc;
}
```

The interpolation FIR filters use a similar structure but handle the pointers differently due to the sample rate change.

### 4.1.11 Standard Simulation Program Settings

Several HF simulation parameter sets have been recommended by CCIR 520-2.
The parameter sets are (two paths with equal attenuation and Doppler spread, zero Doppler shift)

| | Differential Time delay | 2sigma Doppler spread |
|---|---|---|
| Good Conditions: | 0.5 ms | 0.1 Hz |
| Moderate Conditions: | 1.0 ms | 0.5 Hz |
| Poor Conditions: | 2.0 ms | 1.0 Hz |
| Flutter fading | 0.5 ms | 10.0 Hz |

Roald Otnes/ LA5NJA forwarded some preliminary draft ITU-R recommendations he found.

| | Differential Time delay | 2sigma Doppler spread |
|---|---|---|
| Low-latitude quiet | 0.5 ms | 0.5 Hz |
| Low-latitude moderate | 2 ms | 1.5 Hz |
| Low-latitude disturbed | 6 ms | 10 Hz |
| Mid-latitude quiet | 0.5 ms | 0.1 Hz |
| Mid-latitude moderate | 1 ms | 0.5 Hz |
| Mid-latitude disturbed | 2 ms | 1 Hz |
| Mid-latitude disturbed NVIS | 7 ms | 1 Hz |
| Highlatitude quiet | 1 ms | 0.5 Hz |
| Highlatitude moderate | 3 ms | 10 Hz |
| Highlatitude disturbed | 7 ms | 30 Hz |

# 4. Design Verification

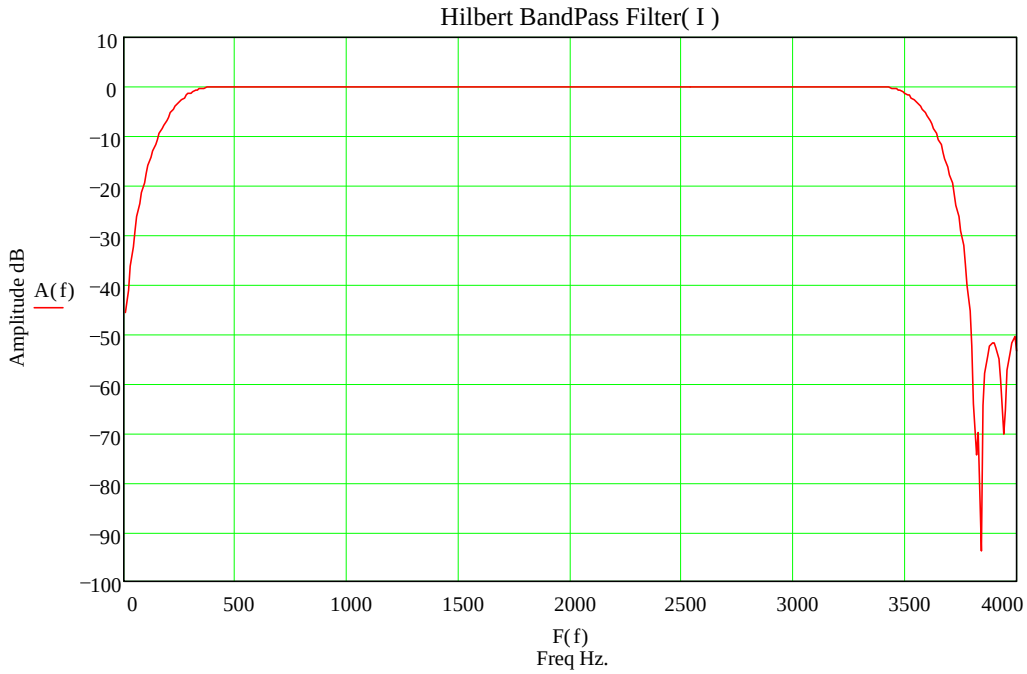
An attempt was made to verify the basic design and implementation of the major components of PathSim. An assortment of sweep generators, RMS measuring blocks, time measuring, and statistical histogram tools were implemented in a separate module( perform.c, perform.h). They can be enabled by un-commenting the line #define TESTMODE 1 in the PathSim.h project file. Some of them write to a fixed file "plot.prn" which I used as an input to MathCad to format and plot the results. Other routines just write to global variables, which are displayed on the screen.

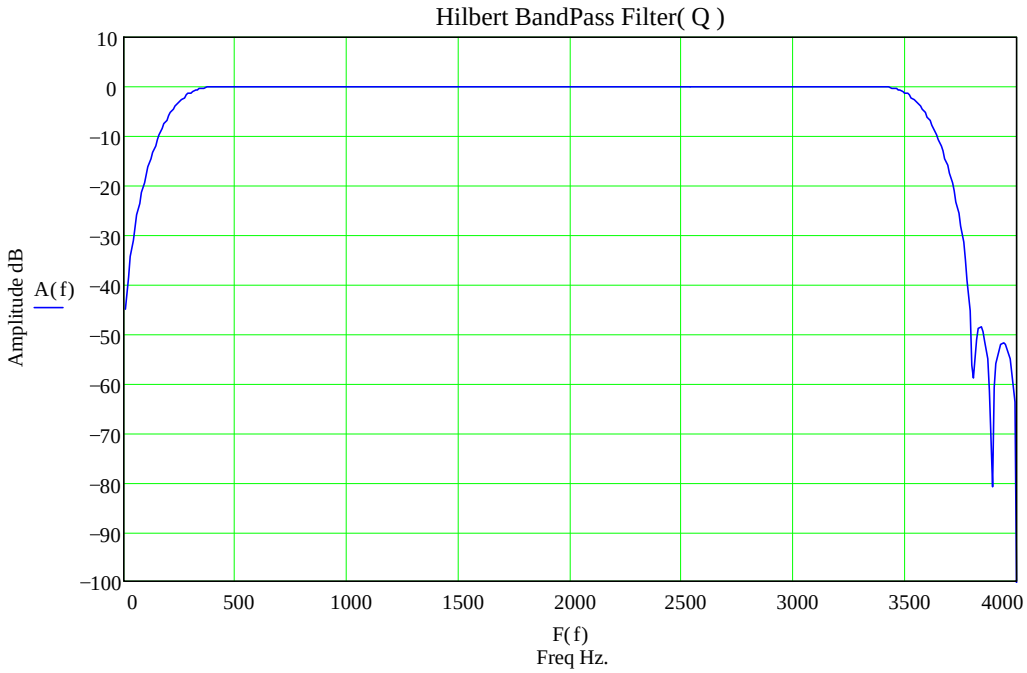
## *4.1. Input Hilbert Bandpass Filter*

### 4.1.1 Amplitude Response

First the frequency response and phase response of the input Hilbert band pass filter was measured using a sweep generator input and measuring the I and Q output responses. Both the I and Q channels were measured by measuring the RMS over 10000 samples at each sweep step.
Note that the pass-band amplitude response is identical but the stop-band is somewhat different.

Sweep 10 to 4000Hz. Step size 10Hz., Sample rate 8000Hz. RMS sample size 10000.






### 4.1.2 Phase Response

The phase difference between the I and Q channels was measured by using a trig identity

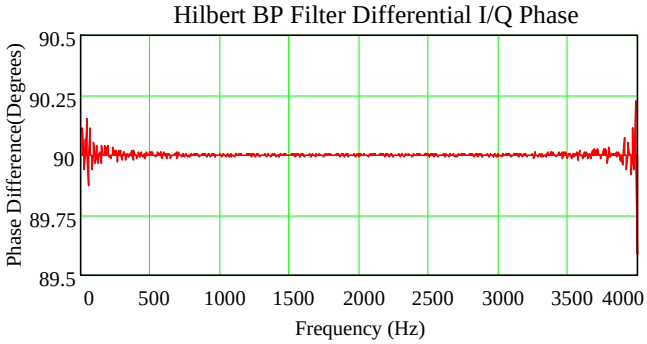
$$\sin(a)\sin(b) = \frac{\cos(a-b)}{2} - \frac{\cos(a+b)}{2}$$

With a sine input to the Hilbert BP filters the output should be a sin wave on each channel separated by a constant phase offset. Multiplying the I and Q channels together generates the following signal:

$$\sin(wt)\sin(wt+\Phi) = \frac{\cos(\Phi)}{2} - \frac{\cos(2wt+\Phi)}{2}$$

Now the second term can be averaged to zero(low pass filtered) leaving the constant term of the phase differential.

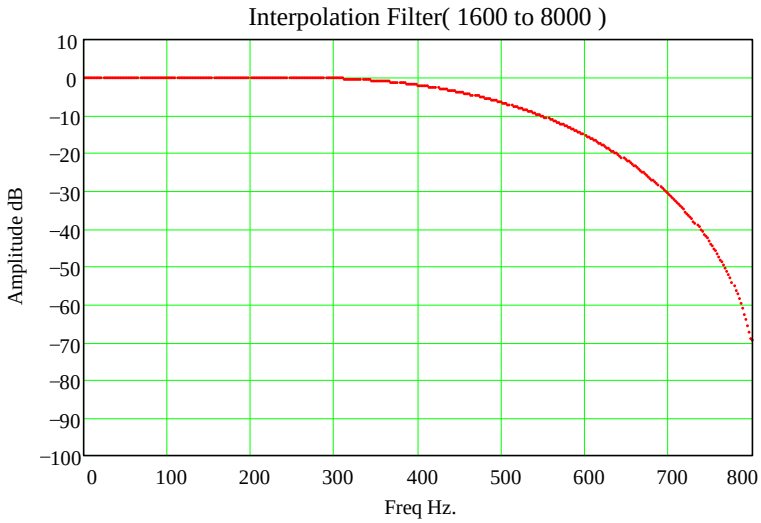
Taking the inverse cosine and scaling it to degrees provides a measure of the phase difference between the I and Q channels with a pure sin wave input.
The following is a sweep from 10 to 4000 Hz with a step size of 5 Hz and averaging the phase over 10000 samples to remove the 2x frequency component leaving the DC phase difference.



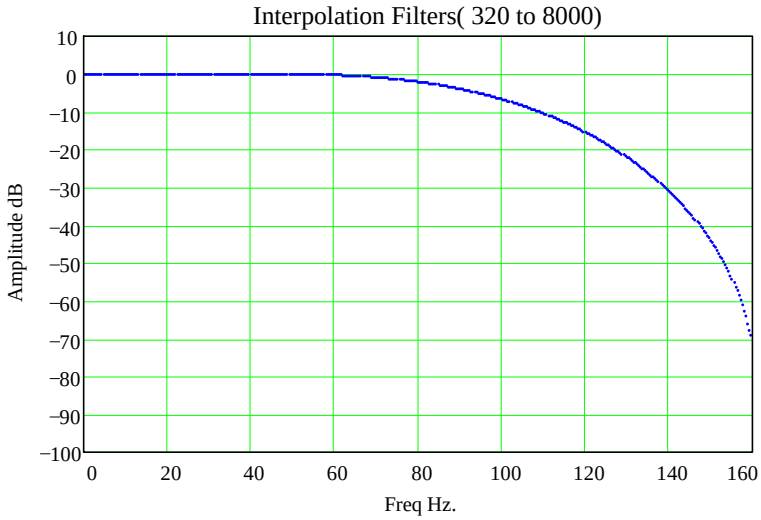
## 4.2. Interpolation Filters

The amplitude versus frequency response of the interpolation stages was measured at each of the 5x stages using a complex sweep generator and measuring the complex RMS value at each frequency step.

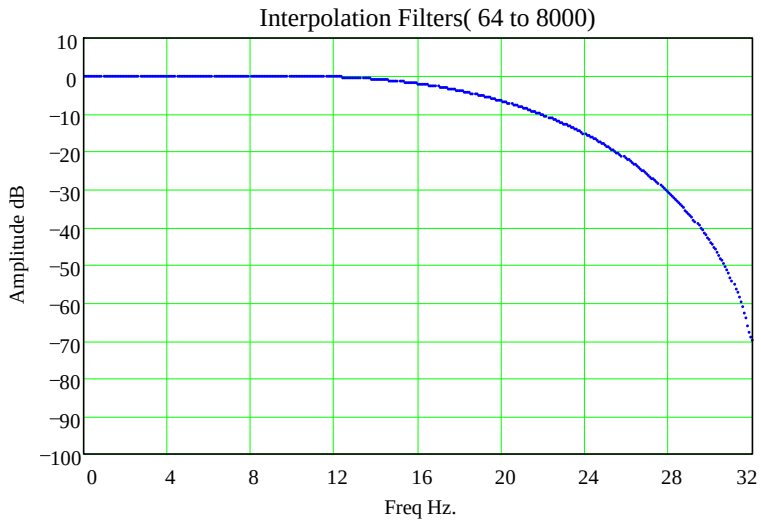
Sweep 0 to 800Hz. Step size 2Hz., Sample rate 8000Hz. RMS sample size 8000.



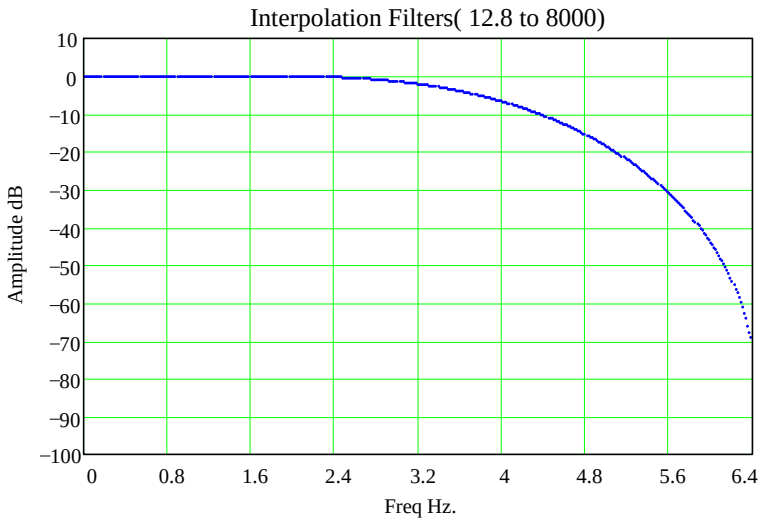

Sweep 0 to 160. Step size .4Hz., Sample rate 320Hz. RMS sample size 8000.



Sweep 0 to 32Hz. Step size .08Hz, Sample rate 64Hz. RMS sample size 8000.



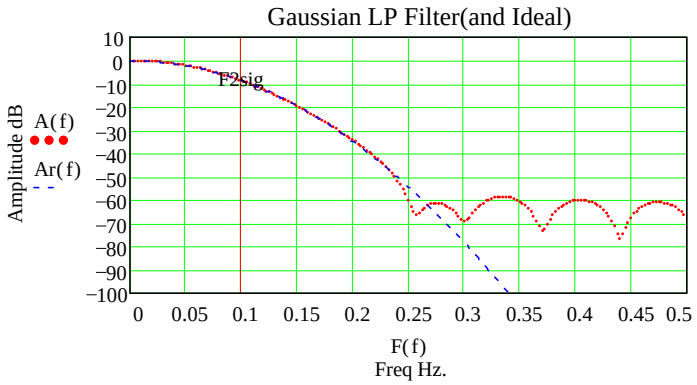

Sweep 0 to 6.4Hz. Step size .016Hz., Sample rate 12.8Hz. RMS sample size 8000.



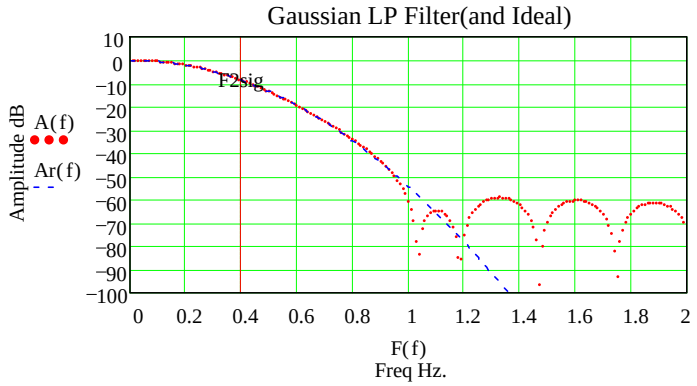

## *4.3. Gaussian Low Pass Filters*

The amplitude versus frequency response of the Gaussian Low Pass filter stage was measured at each extreme of the 2 Sigma frequency ranges for each sample rate and measuring the complex RMS value at each frequency step.

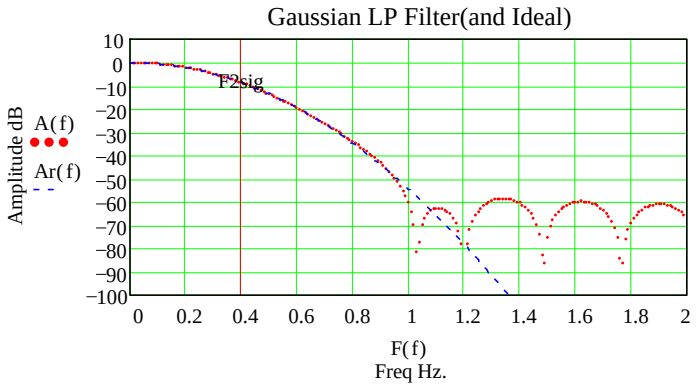
Sweep 0 to .5Hz. Step size .0025Hz., Sample rate 12.8Hz. RMS sample size 100.



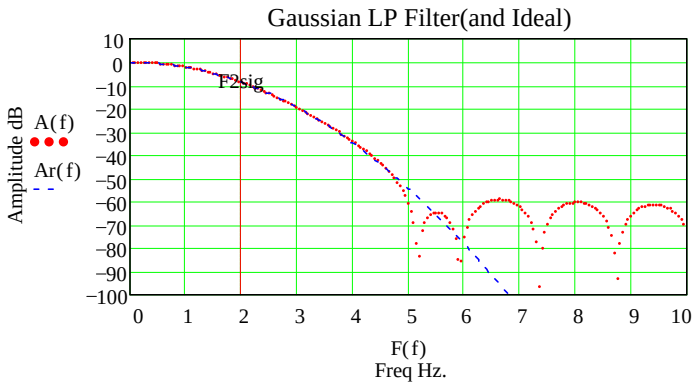

Sweep 0 to 2.0Hz. Step size .0025Hz., Sample rate 12.8Hz. RMS sample size 100.



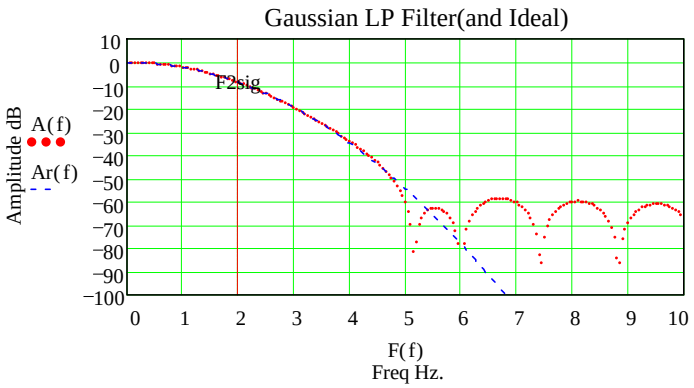

Sweep 0 to 2.0Hz. Step size .0025 Hz., Sample rate 64Hz. RMS sample size 500.



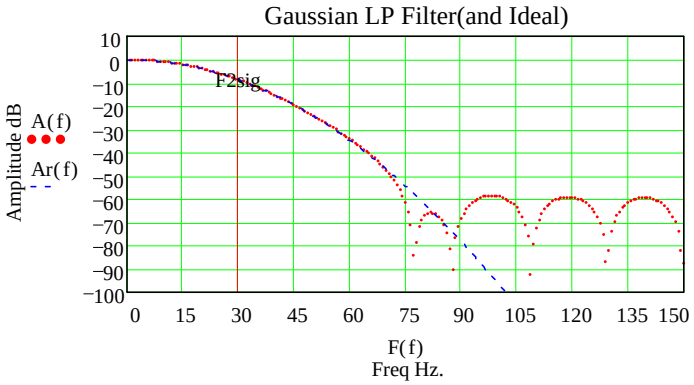

Sweep 0 to 10.0Hz. Step size .05 Hz., Sample rate 64Hz. RMS sample size 500.



Sweep 0 to 10.0Hz. Step size .05 Hz., Sample rate 320. RMS sample size 500.




Sweep 0 to 150.0Hz. Step size .75 Hz., Sample rate 320. RMS sample size 500.

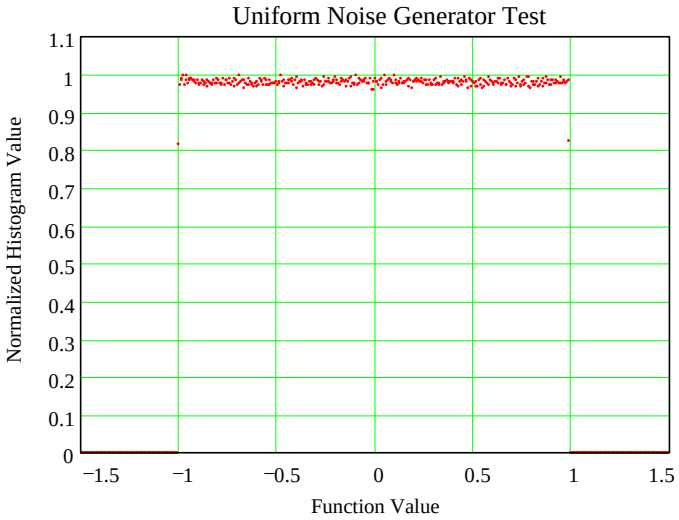


## *4.4. Noise Generators*

A histogram was created using a 500 bin array and used to show the shape of the noise generator outputs over a lot of samples.

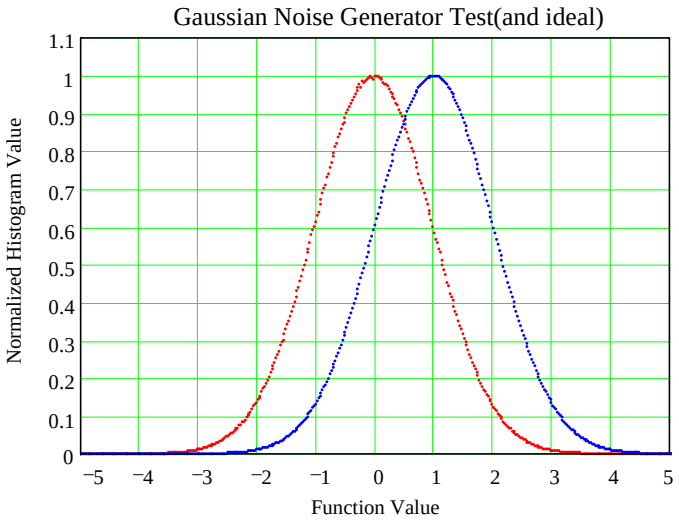
### 4.4.1 Uniform Generator

The uniform number generator used by PathSim is the Visual C++ library one. It generated the following histogram over 10,000,000 samples.




### 4.4.2 Gaussian Generator

The Gaussian noise generator algorithm produced the following histogram over 10,000,000 samples. The ideal Gaussian shape is shown on the plot but is shifted to the right by one unit for comparison.




The noise generator that is used for AGWN to the final signal is band pass filtered using the identical coefficients as the input BP filter. The equivalent noise bandwidth was measured experimentally and used to compensate the noise generator so that it produced a known RMS power through the filter.

## 4.5. Noise Power Measurements

To verify that the equivalent rectangular noise bandwidth of the Gaussian filters is correct, a fixed power sine wave is applied to the input of PathSim. The RMS power is measured after being processed by a single path with various 2 sigma spreading frequencies. The goal is to have the output RMS value equal the input RMS value for any combination of spreading frequencies as well as multiple paths.

Conditions:
Input RMS value = 4738 sin wave
S/N disabled
Delays = 0, Offsets = 0
RMS calculated over about 14,000,000 samples.

| Spread Frequency( Hz) | 1 Path Active | 3 Paths Active |
|---|---|---|
| 0 | 4720 Output RMS | - |
| 0.1 | 4811 Output RMS | 4697 Output RMS |
| 0.2 | 4790 Output RMS | 4732 Output RMS |
| 0.5 | 4685 Output RMS | 4747 Output RMS |
| 1.0 | 4690 Output RMS | 4738 Output RMS |
| 2.0 | 4694 Output RMS | 4730 Output RMS |
| 5.0 | 4724 Output RMS | 4695 Output RMS |
| 10.0 | 4723 Output RMS | 4699 Output RMS |
| 30.0 | 4714 Output RMS | 4707 Output RMS |

Just as a sanity check, the following setup was used to test all paths, offsets and delays:
Spread1 = 0.1Hz, Spread2 = 1.0Hz, Spread3 = 10.0Hz
Offset1 = -100Hz, Offset2 = 0HZ, Offset3 = 1000Hz
Delay2 = 3mSec, Delay3 = 50mSec
Input was sine wave with an RMS of 4738. S/N disabled.

The output RMS value after 14,000,000 samples was 4766.

## 4.6. Program Timing Measurements

The processing time for several of the major blocks was measured by using the built in Pentium timer and some API calls that read it.

The following test data is from a 133 MHz Pentium PC running Windows 95.

| Function Block | Time per sample (uSec) | %CPU useage |
|---|---|---|
| Input RMS Measuring | 0.15 uSec | 0.12% |
| Input Hilbert BP Filter | 5.59 uSec | 4.5% |
| Create Delays | 1.08 uSec | 0.86% |
| Path Calculation( 0 Hz Spread) | 5.08 uSec | 4.1% |
| Path Calculation( 0.1Hz Spread) | 6.06 uSec | 4.8% |
| Path Calculation( 1.0 Hz Spread) | 5.72 uSec | 4.6% |
| Path Calculation( 30.0 Hz Spread) | 5.48 uSec | 4.4% |
| Gaussian Noise Generator | 1.98 uSec | 1.6% |
| Gaussian Noise BP Filter | 2.88 uSec | 2.3% |
| 2048 pt. FFT | 3.07 uSec | 2.4% |
| Screen Plotting | 11.0 uSec | 8.8% |

Maximum processing time for all three paths active was about 32.5uSec/sample or 26%. CPU usage.
For a 400MHz Pentium II the same test condition yielded about 9.5uSec/sample or 7.6% CPU usage.

# 5. Further References and Acknowledgements

The following are some references used in this document and can be used for further reading on the subject.

MathCad ver.6.0. MathSoft 101 Main St.,Cambridge, MA 02142

Marvin E. Frerking. "Digital Signal Processing in Communication Systems"p.146, 377. ISBN0-442-01616-6

CRC Press, Jr. "Standard Mathematical Tables" 21st Edition p.231 ISBN 0-87819-621-8

Bernard Sklar. "Digital Communications Fundamentals and Applications" ISBN 0-13-211939-0

Tom McDermott, N5EG. "Wireless Digital Communications: Design and Theory" ISBN 0-9644707-2-1

Johan Forrer, KC7WW. "A Low-Cost HF Channel Simulator for Digital Systems" May/June 2000 QEX.

Paul M. Embree, Damon Danieli. "Algorithms for Digital Signal Processing" 2nd Edition, ISBN 0-13-179144-3

Thanks to the following individuals for their comments and inputs.

Johan Forrer, KC7WW
Roald Otnes, LA5NJA
Alexander Kurpiers
Michael Keller, DL6IAK
Adam Zajac