RMIT UNIVERSITY

SCHOOL OF ELECTRICAL AND COMPUTER ENGINEERING

EEET1150 DATA&INTERNET TRANSMISSION SYSTEM

Lab Report and Major Assignment:

All-Pass Filter and their Application in Digital Transmission System

Student: *Wei Han* *S3204911*

Lecture: *Mr. C.Burden*

Table of Contents

1. Abstract:

In the first part and the second part of this paper, we studied the all-pass filters' principle, realization and their application in digital transmission system as the smearing filter. The detail design and factorization procedure was given with the assist of Matlab software. The final filter realization is with the all-pass second order digital filter section. The filter characteristic such as group delay has been studied for CCITT ALLPS1/2/3 and Gregorian filters.

In the third part, we made some transmit tests with above filters to evaluate the effects of smearing technique. It shows that the smearing technique is the most effective at the low noise level. With the increasing of noise level, the benefit from the smearing technique is gradually fading. When the noise level comes above a threshold noise level, the error rate in smearing system will go above the level in non-smearing system. All the test results were analyzed and explained based on the principle of smearing technique.

2. Introduction:

In transmission system, the impulse noise is one of the most vicious forms of interference affecting data communications. This kind of noise has a high peak value but last short time duration, it will punch "holes" in the signal and mask the detection process. A possible solution to this problem is to perform some linear transformation to the signal before the impulse noise is added, and then to perform the inverse transformation before the receiver signal detection. In such a way, the noise energy could be spread into wider time duration at the receiver and hence has a lower peak value.

This is the basic idea of Smearing technique. It was originally come from radar theory. Then it was used in digital transmission system design to smear noise when the impulse type noise is a limiting factor.

The signal transformation in this technique could be accomplished by letting the signal pass through a filter which has linear group-delay characteristic and smearing signal. The smeared signal was then send to the communication channel. At the receiver location, the signal passes through a de-smearing filter which has an opposite group-delay frequency characteristic. The combination of smear-desmearing filters for the signal thus equivalent to purely time-delay. But for the noise, because it lacks the smearing stage in sender, after passing the de-smearing filter in receiver, it was smeared.

In this paper, we will study the characteristic and realization of some smear filters and their application in transmission system. The filters include CCITT ALLPS1/2/3 all-pass filters and Gregorian all-pass filter. They all have a nearly const group delay characteristic at the working frequency bandwidth. Then we will implement these filters into the Modem transmission system to study the effects of the smearing technique.

The experimental configuration is set as both high count and low count mode. In high count test, the impulse noise is introduced between transmit Modem and the smearing filter. In the low count mode, the impulse noise is put into the transmission path between the smearing/de-smearing filters.

It could get that the impulse noise will be smeared in low count mode. But in high count mode, the impulse noise was processed as same as the sending signal, so the filters don't have effect to error rate and therefore the error count is a product of the modems only. So high count tests data will always be the same whenever the same modem is used.

The matched delay filters used in experiment is ALLPS2/3 filters. Originally the filter ALLPS1 was used. The ALLPS1 filter was clocked at 6kHz instead of the quoted frequency of 2.4kHz in order to cover the channel bandwidth of the 9600C modem (the ALLPS1 poles were taken from a design of a baseband filter and used in the tests as a pass-band modulated channel filter). To compensate for the loss of absolute delay caused by the higher clock frequency the filter was also used double and triple density poles. This gave the filters named ALLPS2 and ALLPS3. As the comparison, Gregorian all-pass filter was also used in testing.

Part I: Smearing Technique Review

3. Digital filter technique review:

Filtering is the most common form for signal processing, the purpose of it is to remove the frequencies in certain parts and to improve the magnitude, phase, or group delay in some other part(s) of the spectrum of a signal.

According to their usage, the filter could be divided into low-pass filter, high-pass filter, band-pass filter, band-stop filter and all-pass filter. Most of these filters have the obvious usage from their name. The only one which need further explain is the all-pass filter. According the definition in Wikipedia, "An all-pass filter is a signal processing filter that passes all frequencies equally, but changes the phase relationship between various frequencies. It does this by varying its propagation delay with frequency." In this paper we will study some digital all-pass filters and their application in transmission system.

Digital filters could be classified into two major categories according to their impulse responses length:

1) Finite impulse response (FIR) digital filters: their impulse response h(n) has a finite number of non-zero samples;
2) Infinite impulse response (IIR) digital filters: their impulse response h(n) has an infinite number of non-zero samples.

FIR filter transfer function is given by: $H(z) = {}^{Y(z)}\!/_{X(z)} = h_0 + h_1 z^{-1} + \cdots + h_{N-1} z^{-(N-1)}$.

From above transfer function, we could find FIR don't have pole point, or in other word, all the pole points of the FIR are located at original point where z=0, so it don't stability problem, which means it always stable.

Infinite impulse response (IIR) digital filter introduced the feedback from the output. The transfer function is given by:

So have: $$H(z) = \frac{Y(z)}{X(z)} = \frac{b_0 + b_1 z^{-1} + \cdots + b_M z^{-M}}{1 + a_1 z^{-1} + \cdots + a_N z^{-N}} = \frac{\sum_{i=0}^{M} b_i z^{-i}}{\sum_{k=0}^{N} a_k z^{-k}}$$, where a_0=1.

Hence, unlike FIR filters, IIR filters can have poles at locations other than z=0. So it has the stability problem. From above equation, if we put $a_1 = a_2 = \cdots = a_N = 0$, it will become a FIR filter. Normally, the number of poles N is larger than the number of zeros M, and the order of the filter is decided by the number of it poles. For the stability conditions of the IIR filter, is that all poles should be inside the unit circle.

In above IIR filter transfer function, when $z = e^{j\omega}$, we have IIR filter frequency response as following,

$$H(e^{j\omega}) = \frac{\sum_{k=0}^{M} b_k \cos(k\omega) - j\sum_{k=0}^{M} b_{(k)} \sin(k\omega)}{\sum_{k=0}^{N} a_k \cos(k\omega) - j\sum_{k=0}^{N} a_{(k)} \sin(k\omega)} \quad = \quad \left|H(e^{j\omega})\right| e^{j\theta(\omega)} \qquad (3.1)$$

Where

$\left|H(e^{j\omega})\right|$ is the magnitude response, $\left|\theta(e^{j\omega})\right|$ is the phase response.

Above equation could further change to:

$$\left|H(e^{j\omega})\right| = \left\{ \frac{\left[\sum_{k=0}^{M} b_k \cos(k\omega)\right]^2 + \left[\sum_{k=0}^{M} b_{(k)} \sin(k\omega)\right]^2}{\left[\sum_{k=0}^{N} a_k \cos(k\omega)\right]^2 + \left[\sum_{k=0}^{N} a_{(k)} \sin(k\omega)\right]^2} \right\}^{\frac{1}{2}} \qquad (3.2)$$

And

$$\theta(j\omega) = -\tan^{-1} \frac{\sum_{k=0}^{M} b(k)\sin(k\omega)}{\sum_{k=0}^{M} b(k)\cos(k\omega)} + \tan^{-1} \frac{\sum_{k=0}^{N} a(k)\sin(k\omega)}{\sum_{k=0}^{N} a(k)\cos(k\omega)} \qquad (3.4)$$

The group delay $\tau(j\omega)$ is defined as: $\tau(j\omega) = -\left[d\theta(j\omega)\right]/d\omega$ (3.5)

In IIR transfer function, if we let $a_N = b_0, a_{N-1} = b_1, \ldots, \ a_0 = b_N = 1$, we have following All-pass filter transfer function:

$$H_{ap}(z) = \pm \frac{a_{(N)} + a_{(N-1)}z^{-1} + \cdots + a_{(2)}z^{-(N-2)} + a_{(1)}z^{-(N-1)} + z^{-N}}{1 + a_{(1)}z^{-1} + a_{(2)}z^{-2} + \cdots + a_{(N-1)}z^{-(N-1)} + a_{(N)}z^{-N}} = z^{-N} \frac{D(z)}{D(z^{-1})} \qquad (3.6)$$

It is easy to see that the magnitude response of $\left|H(e^{j\omega})\right|$ is equal to one at all frequencies and is independent of all the coefficients:

$$\left|H_{ap}(e^{j\omega})\right| = \left| \frac{1 + a_{(1)}e^{j\omega} + a_{(2)}e^{j2\omega} + \cdots + a_{(N-1)}e^{j(N-1)\omega} + a_{(N)}e^{jN\omega}}{1 + a_{(1)}e^{-j\omega} + a_{(2)}e^{-j2\omega} + \cdots + a_{(N-1)}e^{-j(N-1)\omega} + a_{(N)}e^{-jN\omega}} \right| = 1$$

But the phase response (and the group delay) is dependent on the coefficients of the all-pass filter and far from a constant.

4. Gregorian's method for All-Pass Digital Filter Design:

From above section, we know that the all-pass filter phase response (and the group delay) — as defined by (3.4) and (3.5) — is dependent on the coefficients of the all-pass filter. They are the nonlinear function of ω and therefore its group delay defined by (3.5) is far from a constant value. The initial design for the all-pass filter is based on the iterative method, which is from an initial approximation and iterative to the requirement filter performance. So the design needs a good initial approximation.

B. Gregorian proposed a direct match design method for all-pass filter [1], which could be used as the initial approximation, and usually is good enough to be used as the final design directly. In this section, we will give a brief description for this direct match method.

We rewrite the all-pass transfer function as: $H(z) = Az^{-M} \dfrac{\sum_{i=0}^{N} c_i z^{-i}}{\sum_{i=0}^{N} c_{N-i} z^{-i}}$, (4.1)

The linear-phase factor (i.e. Az^{-M}) could be omitted or add to obtain a simple functions. Assuming that all c_i are real, that N is even, and that all zeros and poles occur in conjugate pairs, above equation could be written in the following second order form:

$$H(z) = \prod_{i=1}^{N/2} \frac{(z - r_i^{-1} e^{j\phi_i})(z - r_i^{-1} e^{-j\phi_i})}{(z - r_i\, e^{j\phi_i})(z - r_i\, e^{-j\phi_i})} \tag{4.2}$$

Where r_i is the magnitude and ϕ_i is the phase angle of the ith pole z_i. For $z = e^{j\omega}$, the transfer function is:

$$H(e^{j\omega}) = \frac{\sum_{k=0}^{N} b_k \cos(k\omega) - j\sum_{k=0}^{N} b_{(k)} \sin(k\omega)}{\sum_{k=0}^{N} a_k \cos(k\omega) - j\sum_{k=0}^{N} a_{(k)} \sin(k\omega)}, \tag{4.3}$$

$$\theta(j\omega) = --2\tan^{-1}\left[\frac{\sum_{i=0}^{N} c(i)\sin(i\omega)}{\sum_{i=0}^{N} c(i)\cos(i\omega)}\right] \tag{4.4}$$

The group delay is:

$$\tau(\omega) = -T\frac{\partial\theta(\omega)}{\partial\omega} = 2T\left(\sum_{i=0}^{N} c_i \cos(\omega i)\sum_{i=0}^{N} ic_i \cos(\omega i) + \sum_{i=0}^{N} c_i \sin(\omega i)\sum_{i=0}^{N} ic_i \sin(\omega i)\right) \\ /\left\{\left[\sum_{i=0}^{N} c_i \cos(\omega i)\right]^2 + \left[\sum_{i=0}^{N} c_i \sin(\omega i)\right]^2\right\} \tag{4.5}$$

Based on above all-pass filter group delay characteristic, Gregorian's further approach is to get a close match between a desired response and the actual one and try to obtain equality between the two responses at the largest possible number of frequencies.

His final relations are shown in following:

$$\sum_{i=0}^{N} c_i \sin\left[\frac{\beta_s(\omega_k)}{2} + i\omega_k\right] = 0, \quad k=1,2, \dots, N \qquad (4.6)$$ And,

$$\frac{dN(\omega, c_i)}{d\omega} = \sum_{i=0}^{N} c_i \left[\frac{\tau_s(\omega)}{2T} + i\right] \cos\left[\frac{\beta_s(\omega)}{2} + i\omega\right] = 0, \text{ where } \omega = \omega_1, \omega_2, \dots, \omega_{N/2} \qquad (4.7)$$

Above two equations represent the N linear equations for $c_1, c_2, \dots, c_N$, where $c_0=1$ is assumed. Relation (4.6) and (4.7) are the basic relations in Gregorian's direct match all-pass filter design method.

It should be noted that, although the group delays are matched only at N/2 points, an additional N/2-1 matching points are automatically obtained. Hence, the total number intersections between τ and τ_s is at least N-1.

5. All-pass filter's application in digital transmission system ----- Smearing Technique:

In this section, we will study All-pass filter's application in transmission system. The principle of the smearing filter has been discussed in many papers [2][3]. The design of the smearing filters in these classical papers is based on the following criteria:
1) Both the smearing and de-smearing filter have all-pass characteristics,
2) The smearing (de-smearing) filter have a linearly increasing (decreasing) group delay in the working frequency band and both filters are complementary (i.e., the couple of the filters is equivalent with a flat delay).

An intuitive understanding for the effect of smearing technique could be shown in following.

Considering a typical smearing transmit system show in following graph:

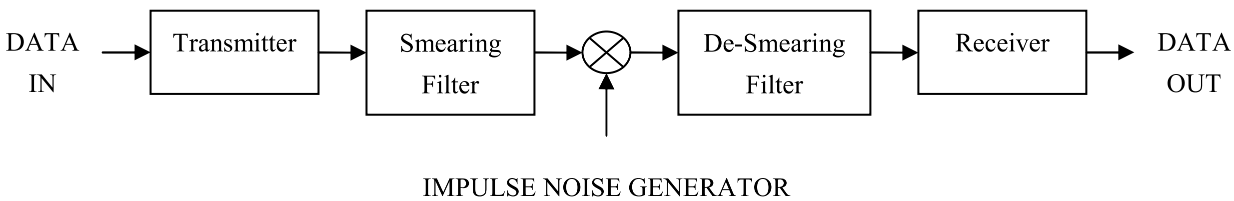


Figure 5.1 Smearing transmission system block graph

In the system, the smearing filter is a raising delay all-pass filter, so the de-smearing filter is the same characteristic falling delay all-pass filter. We could demonstrate the effect of the smearing and de-smearing filters as following graph [2]:

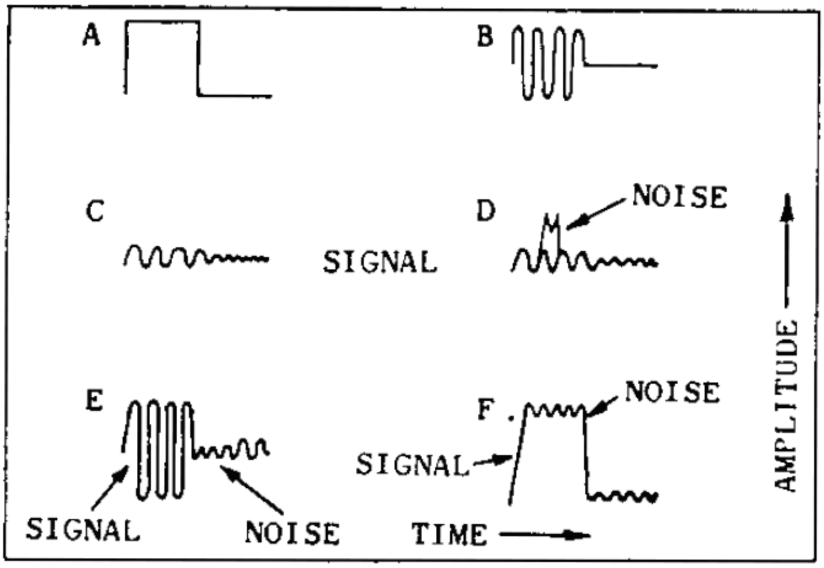


Figure 5.2 Signal and noise at different point [2]

Where:

(A) is the data-input signal to the transmitter;

(B) is the amplitude-modulated output of the transmitter;

(C) shows the effect of the smear filter to the signal shown in (B), note that, in(C) the lower-order components occur first in time and are followed by the higher-order components at a later time for the delay caused by smearing filter. The total-time spread in this case is dependent on the slope and the extent of the time-delay function of the smear filter;

(D) shows the smeared signal and noise on the transmission line;

(E) shows the effect of the de-smearing filter in recovering the smeared data signal and of smeared noise pulse. In this case, the lower-order components of the noise pulse are delayed more than the higher-order component because the de-smearing filter is a falling delay all-pass filter. And how much lag for the lower-order components after the filter is determined by the all-pass filter's group delay.

(F) shows the demodulated, band-limited data signal, along with the smeared noise. It shows that with the smearing technique, the impulse noise spreads its energy into a wider time length and thus reduces the peak level of the impulse noise.

From above analysis, we could see that it is the de-smearing filter in receiver that performs the function of smearing the impulse noise. And we could easily find that the spread time length is determined by the filter group delay. The smearing filter in transmitter is used to compensate this smearing effects for the transmit signal. Because the smearing technique is to spread the narrow high-peak impulse into a wider time duration but cannot reduce the total energy of the noise, so it is most effective at the low impulse repetition rate.

Part II: Characteristic and Implementation of All-Pass Filter

6. Characteristic and Factorization of ALLPS2 Filter:

We have given a brief review for the digital filter and the Gregorian's method for all-pass filter design in previous sections. In this section, we will study and factorize CCITT ALLPS2 all-pass filter.

ALLPS2 filter is derived from CCITT 16th order all-pass filter ALLPS1. When we doubled the zeros and poles of ALLPS1 filter, we could get the zeros and poles of the ALLPS2 filter and could further get its transfer function and characteristic.

The coefficients of 16th order C.C.I.T.T. ALLPS1 raising delay filter are:
C = [1.0000, 3.24837, 5.27556, 5.83603, 5.03954, 3.68737, 2.42122, 1.49040, 0.88413, 0.50638, 0.27250, 0.13289, 0.05723, 0.02149, 0.00652, 0.00153, 0.00031]

This is the starting point of our all-pass filter studying. The study and design are based on Matlab. We will factorize the zeros and poles of ALLPS2 filter, study its impulse response and group delay, and finally get its second order section coefficients.

6.1 Factorization of ALLPS2 filter:

The first step of the design is to find the zeros and poles of the filter. i.e. change the filter transfer function from the form: $H(z) = \dfrac{Y(z)}{X(z)} = \dfrac{b_0 + b_1 z^{-1} + \cdots + b_M z^{-M}}{1 + a_1 z^{-1} + \cdots + a_N z^{-N}}$

into the zero-poles form: $H(s) = K \dfrac{(s - z_1)\cdot(s - z_2)\cdots(s - z_n)}{(s - p_1)\cdot(s - p_2)\cdots(s - p_n)}$

This could be implemented with following Matlab codes:

```
a=[1.0000,3.24837,5.27556,5.83603,5.03954,3.68737,2.42122,1.49040,0.88413,0.50638,0.27250,
0.13289,0.05723,0.02149,0.00652,0.00153,0.00031];
b=fliplr(a);    [zeros,poles,k]=tf2zp(b,a);
```

The given coefficient is stored in variable *a*, this will be the denominator coefficient in filter transfer function.

Function *fliplr()* inverse the sequence of this vector and stored it into the variable *b*. From the property of all-pass filter, we know this is the numerator coefficients in the transfer function.

Matlab function *tf2zp()* is the core function in this step. It will transform the filter transfer function into the zero-poles form. The converted results are stored in the variables *zeros*, *poles* and *K*.

By now, we have obtained the zeros, poles and the gain for the ALLPS1 raising delay filter which stored in variables *zeros*, *poles* and *K*.

As mentioned above, double the zeros and poles of ALLPS1, we could get the ALLPS2 filter. This could be performed with the following codes:

```
zeros2=[zeros;zeros];    poles2=[poles;poles];    k2=k^2;
```

From above zeros and poles of ALLPS2, we could further get the ALLPS2 transfer function coefficients with following code:

```
[b2,a2]=zp2tf(zeros2,poles2,k2);
```

The function *zp2tf()* get the polynomial form transfer function from the zero and pole value.

By now, we have obtained the zeros, poles, gain and coefficients of polynomial form transfer function form of the ALLPS2 raising delay filter as following:

Zeros,

1.32767+1.10839i   1.32767-1.10839i   0.483763+2.00127i   0.483763-2.00127i   0.183813+2.07175i
0.183813-2.07175i   -0.0672577+1.85845i   -0.0672577-1.85845i   -0.685265+1.34841i   -0.685265-1.34841i
-1.05227+0.955574i   -1.05227-0.955574i   -1.26376+0.533419i   -1.26376-0.533419i   -1.39443+0.103471i
-1.39443-0.103471i   1.32767+1.10839i   1.32767-1.10839i   0.483763+2.00127i   0.483763-2.00127i
0.183813+2.07175i   0.183813-2.07175i   -0.0672577+1.85845i   -0.0672577-1.85845i   -0.685265+1.34841i
-0.685265-1.34841i   -1.05227+0.955574i   -1.05227-0.955574i   -1.26376+0.533419i   -1.26376-0.533419i
-1.39443+0.103471i   -1.39443-0.103471i

Poles,

0.443854+0.370547i   0.443854-0.370547i   -0.713212+0.0529225i   -0.713212-0.0529225i
-0.671634+0.28349i   -0.671634-0.28349i   -0.520824+0.472962i   -0.520824-0.472962i
-0.299531+0.589392i   -0.299531-0.589392i   0.114118+0.472096i   0.114118-0.472096i
0.0424909+0.478913i   0.0424909-0.478913i   -0.0194479+0.53738i   -0.0194479-0.53738i
0.443854+0.370547i   0.443854-0.370547i   -0.713212+0.0529225i   -0.713212-0.0529225i
-0.671634+0.28349i   -0.671634-0.28349i   -0.520824+0.472962i   -0.520824-0.472962i
-0.299531+0.589392i   -0.299531-0.589392i   0.114118+0.472096i   0.114118-0.472096i
0.0424909+0.478913i   0.0424909-0.478913i   -0.0194479+0.53738i   -0.0194479-0.53738i

K, 9.61e-008

Coefficients of polynomial form transfer function

b,numerator

0.000000096100000, 0.000000948600000, 0.000006383300000, 0.000033275000000, 0.000143752400000,
0.000537745200000, 0.001783692700000, 0.005340436600000, 0.014637968500000, 0.037155320400000,
0.088205150100000, 0.197515130799999, 0.420077413999998, 0.852739026799995, 1.656850770799991,
3.084106183999983, 5.497709091899971, 9.376709369799952, 15.291204371999923, 23.838809099599885,
35.528325506199828, 50.576365018199780, 68.559786745299718, 87.908034316999675,
105.433751356199640, 116.438533429599620, 116.030361619499640, 101.691973953199690,

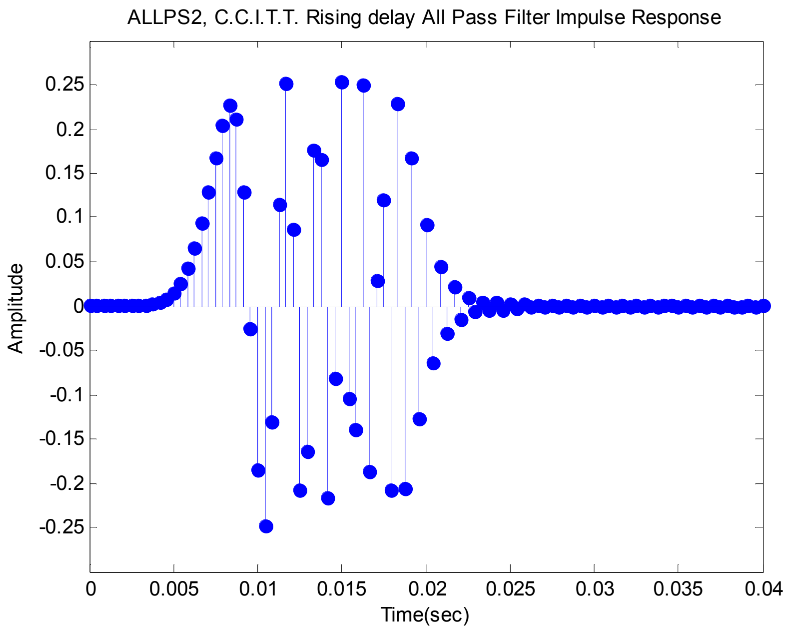
75.825782855799758, 45.946001674399838, 21.103027656899911, 6.496739999999967, 0.999999999999993,

a,denominator

1.000000000000000, 6.496739999999999, 21.103027656900000, 45.946001674399980, 75.825782855799986, 101.691973953199980, 116.030361619499980, 116.438533429599970, 105.433751356199980, 87.908034316999988, 68.559786745299988, 50.576365018199986, 35.528325506199984, 23.838809099599985, 15.291204371999990, 9.376709369799997, 5.497709091899999, 3.084106184000000, 1.656850770800000, 0.852739026800000, 0.420077414000000, 0.197515130800000, 0.088205150100000, 0.037155320400000, 0.014637968500000, 0.005340436600000, 0.001783692700000, 0.000537745200000, 0.000143752400000, 0.000033275000000, 0.000006383300000, 0.000000948600000, 0.000000096100000,

6.2 Impulse response:

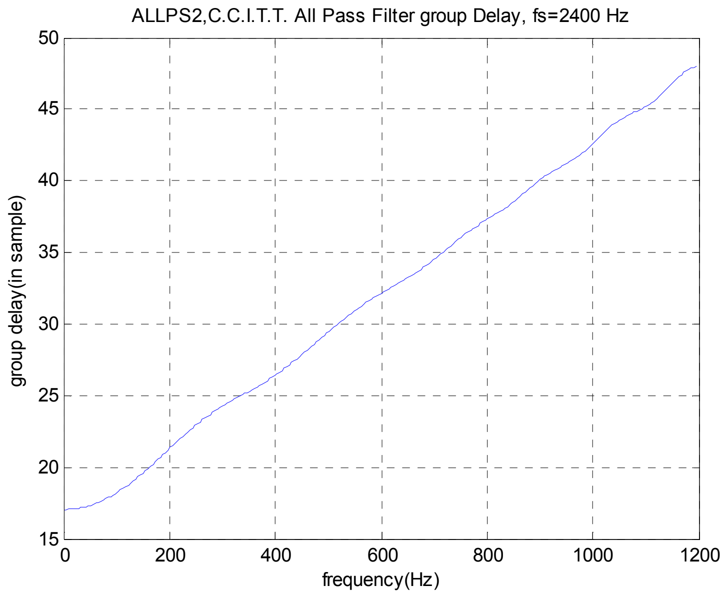
The impulse response could get with Matlab code: `impz(b,a,2.^8,fclock);`
Where *fclock* is the sample frequency, which is 2400 in this case. The impulse response plot is:



6.3 Group delay:

One of the most important properties for ALLPASS filter is the group delay. This could be got with Matlab code: `grpdelay(b,a,2.^8,fclock);`

The result group delay plot is:



The Nyquist frequency is 1200Hz at the 2400Hz sample frequency. In above group delay plot, the overall gradient of the group delay is approximately a constant with a few local 'rippling' deviations from the ideal straight line. The gradient for ALLPS2 is about 40samples/1000Hz. The delay starts from almost 0Hz. We will see later that this gradient is the twice of the gradient of group delay of ALLPS1. This could be understandable from the phase response of all-pass filer which is:

$$\theta(\omega) = -2\tan^{-1}\left[\frac{\sum_{i=0}^{N} c(i)\sin(i\omega)}{\sum_{i=0}^{N} c(i)\cos(i\omega)}\right]$$

and the definition of group delay which is: $\tau(j\omega) = -[d\theta(j\omega)]/d\omega$.

For every zero and pole is duplicated in ALLPS2 filter, the above $\theta(\omega)$ will be

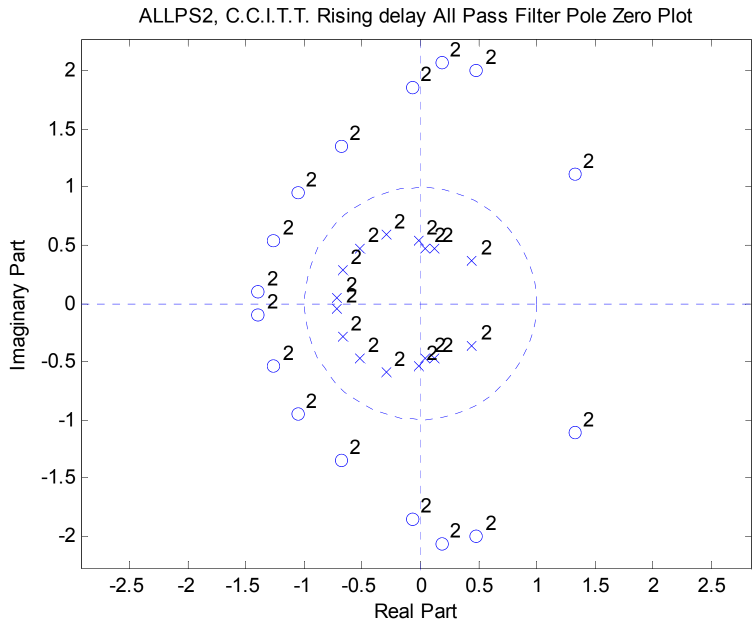
doubled, group delay will become to: $\tau_{ALLPS2}(j\omega) = -[d2\theta(j\omega)]/d\omega = 2\tau_{ALLPS1}(j\omega)$.

The small gradient deviations from constant come from that we just use the 16th order filter. From previous mentioned Gregorian filter design method, for the 16th order filter, the design just match 16 sample points. If we need a smooth rising slope, we need a higher order filter.

6.4 Zero-Pole Plot:

The zero-pole graph could get from: `zplane(zeros,poles);`

The result is:



From above figure, we could get that every zero and pole point has been doubled. All the poles are located in unit circle, so the filter is stable.

6.5 Second Order Coefficients :

We will implement the final filter with the cascade bi-quadratic digital filter sections. So we need change the transfer function of the filter into second order form. The second order form is the following form:

$$H(z) = g\prod_{k=1}^{k=l}\frac{b_{0k}+b_{1k}z^{-1}+b_{2k}z^{-2}}{1+a_{1k}z^{-1}+a_{2k}z^{-2}}$$

With the following Matlab code, we could get the second order section coefficients:

```
sos=zp2sos(zeros,poles,k),
```

Where variable *sos* is a L by 6 matrix which storing the converted second order coefficient. It has the following form:

$$\mathrm{SOS} = \begin{bmatrix} b_{01} & b_{11} & b_{21} & 1 & a_{11} & a_{21} \\ b_{02} & b_{12} & b_{22} & 1 & a_{12} & a_{22} \\ \cdots & & & & & \\ b_{0L} & b_{1L} & b_{2L} & 1 & a_{1L} & a_{2L} \end{bmatrix}$$

From above codes, we could get ALLPS2 second order coefficients as following:

Second Order Section Coefficient

0.23116330, -0.08498175, 1.00000000, 1.00000000, -0.08498175, 0.23116330

0.23116330, -0.08498175, 1.00000000, 1.00000000, -0.08498175, 0.23116330

0.23589759, -0.22823696, 1.00000000, 1.00000000, -0.22823696, 0.23589759

0.23589759, -0.22823696, 1.00000000, 1.00000000, -0.22823696, 0.23589759

0.28915555, 0.03889589, 1.00000000, 1.00000000, 0.03889589, 0.28915555

0.28915555, 0.03889589, 1.00000000, 1.00000000, 0.03889589, 0.28915555
0.33431174, -0.88770844, 1.00000000, 1.00000000, -0.88770844, 0.33431174
0.33431174, -0.88770844, 1.00000000, 1.00000000, -0.88770844, 0.33431174
0.43710230, 0.59906221, 1.00000000, 1.00000000, 0.59906221, 0.43710230
0.43710230, 0.59906221, 1.00000000, 1.00000000, 0.59906221, 0.43710230
0.49495076, 1.04164826, 1.00000000, 1.00000000, 1.04164826, 0.49495076
0.49495076, 1.04164826, 1.00000000, 1.00000000, 1.04164826, 0.49495076
0.51147168, 1.42642334, 1.00000000, 1.00000000, 1.42642334, 0.51147168
0.51147168, 1.42642334, 1.00000000, 1.00000000, 1.42642334, 0.51147168
0.53145863, 1.34326744, 1.00000000, 1.00000000, 1.34326744, 0.53145863
0.53145863, 1.34326744, 1.00000000, 1.00000000, 1.34326744, 0.53145863

There are 16 second order sections in ALLPS2 filter. The coefficients are duplicated for every two sections.

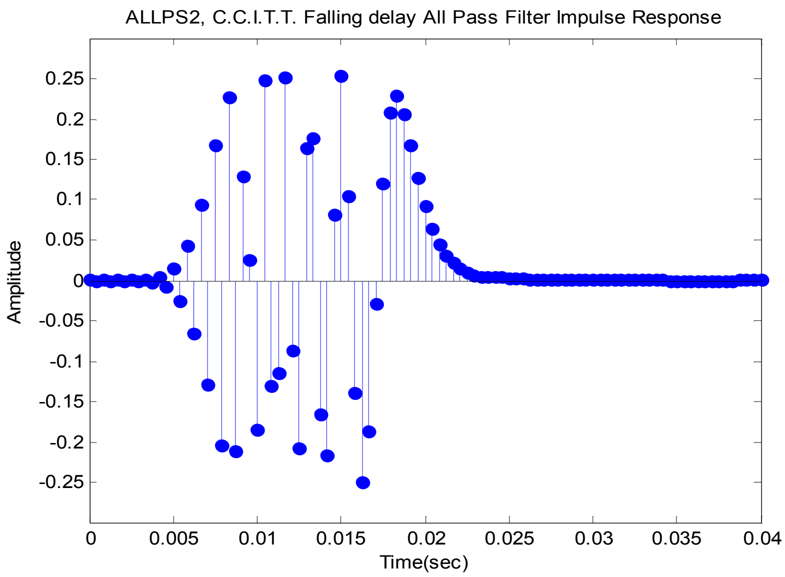
6.6 Falling Delay ALLPS2 Filter Implementation and Characteristic:

Above are the ALLPS2 raising delay filter's characteristic analysis and parameterizations procedure. As we mentioned above, in transmission system, the smearing filter is coupled with a de-smearing filter. If we use raising delay filter as the smearing filter, the de-smearing should be a falling delay filter with the same characteristic. So we need transform above ALLPS2 raising delay filter into falling delay filter. This transformation could be made by changing the sign of zeros and poles in above raising delay ALLPS2 filter. The following Matlab code could perform this transformation:

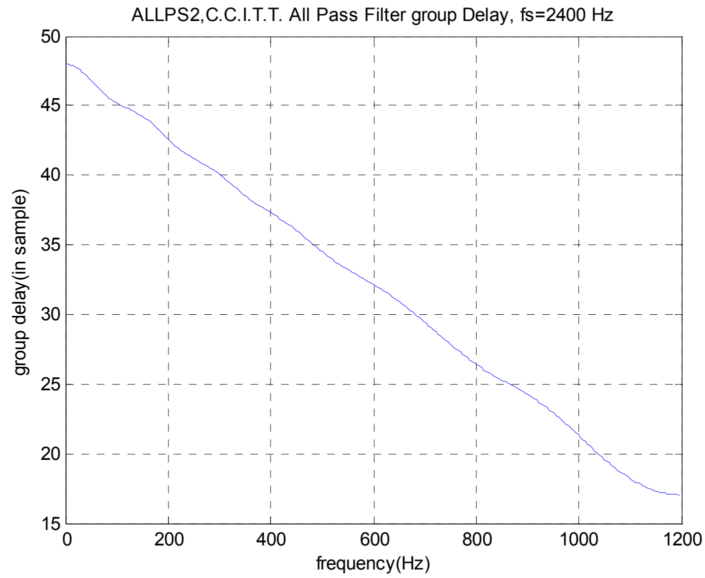
```
zeros=zeros.*-1;     poles=poles.*-1;     [b,a]=zp2tf(zeros,poles,k);
```

It inverse the sign of zeros and poles of the raising delay ALLPS2 filter, then transform the zeros and poles into polynomial form transfer function. Then we could follow the same process procedure in raising delay ALLPS2 filter to get the characteristic of falling delay filter in following:

Impulse response:

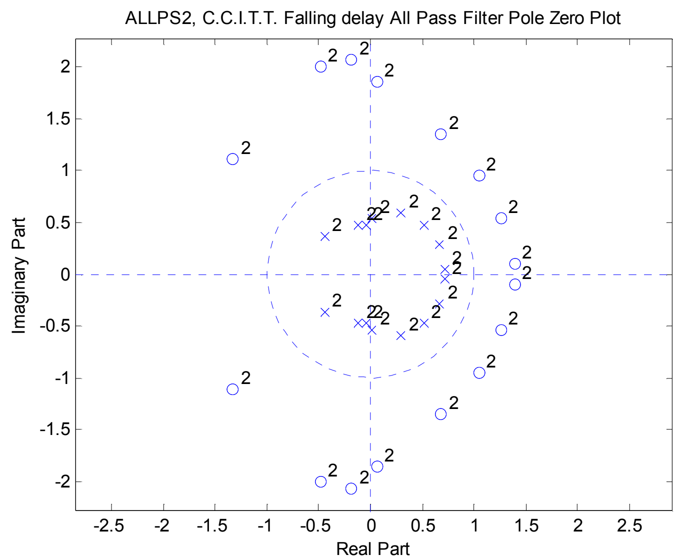


Group delay plot:



From above group delay graph, could get that the gradient of falling delay ALLPS2 filter is about -40samples/1000Hz. So it could compensate the group delay in transmission system caused by the raising delay ALLPS2 filter. In real transmission system, if we use a raising delay filter as the smearing filter, we could use the same characteristic falling delay filter as the de-smearing filter.

Zeros-poles plot:



Comparing to the raising delay filter, the locations of zeros and poles in falling filter are opposite to the locations of zeros and poles in raising delay filter and they are symmetry with imaginary axis.

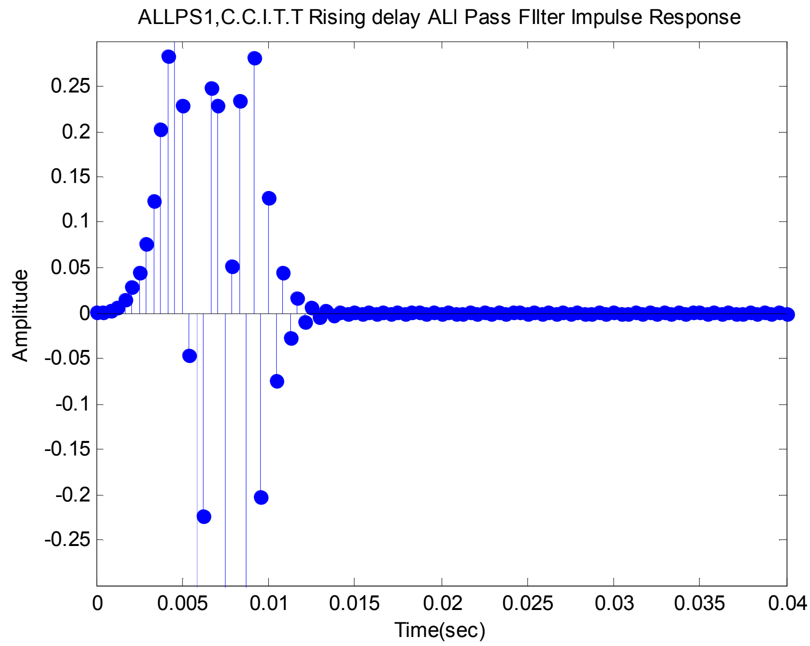
7. Characteristic of ALLPS1 and ALLPS3 Filter:

The process procedure for CCITT ALLPS1 and ALLPS3 filter are same as ALLPS2, so in here we just list the characteristic graph for ALLPS1 and ALLPS3. Because the characteristic of falling delay filter are same as the raising delay filter, only opposite to the direction and sign, so we just list the characteristic graph for raising delay filter and compare them with ALLPS2.

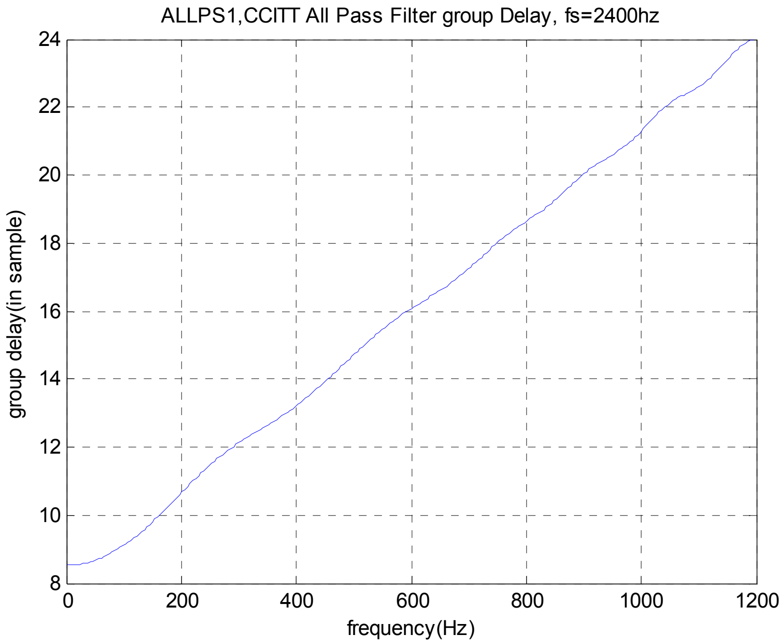
7.1 CCITT ALLPS1 raising delay all-pass filter:

When we process the characteristic in ALLPS2 raising delay filter, if we delete the following Matlab code: zeros2=[zeros;zeros]; poles2=[poles;poles]; k2=k^2;
We could get the characteristic of ALLPS1 raising delay all-pass filter. The characteristic of ALLPS1 raising delay is in following:

Impulse response:



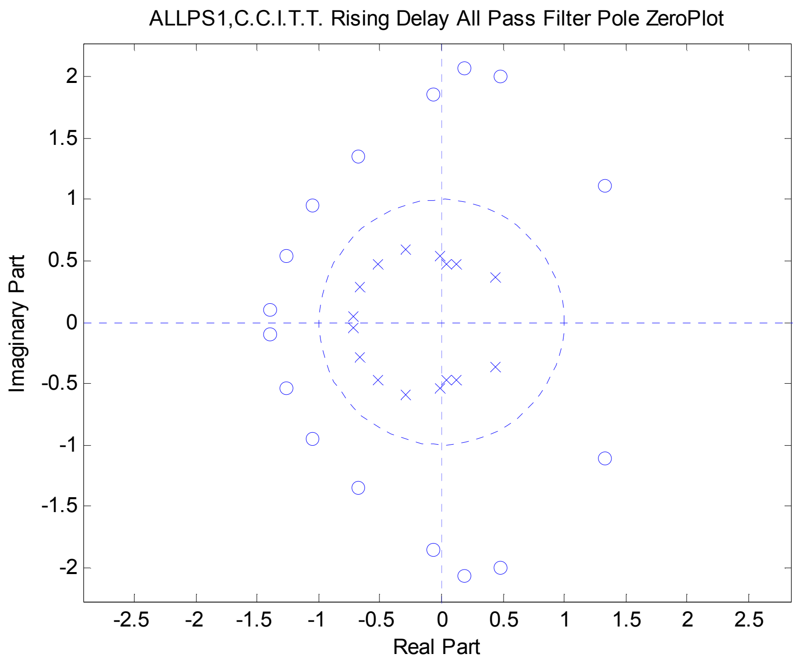
Group delay plot:



From above group delay plot we could find the smoothness of group delay curve is the same as ALLPS2 for they have the same sample point number.

The gradient value is about 20 Samples/1000Hz, which is one half of the value in ALLPS2. The reason for it is that ALLPS1 filter has one half of zeros and poles in ALLPS2 filter.

Zeros-poles plot:



From above zero-poles plot, we could see that ALLPS1 has only single density zeros and poles.

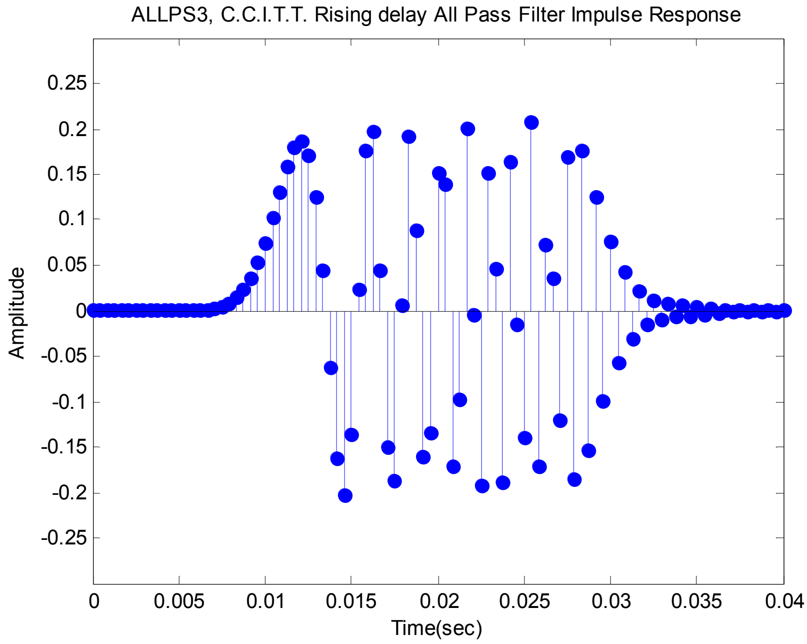
7.2 CCITT ALLPS3 Raising Delay All-pass Filter:

Similar to ALLPS2 filter, if we tripled the zeros and poles of ALLPS1 instead of doubled them, we could get the ALLPS3 filter. The Matlab code is shown in following:

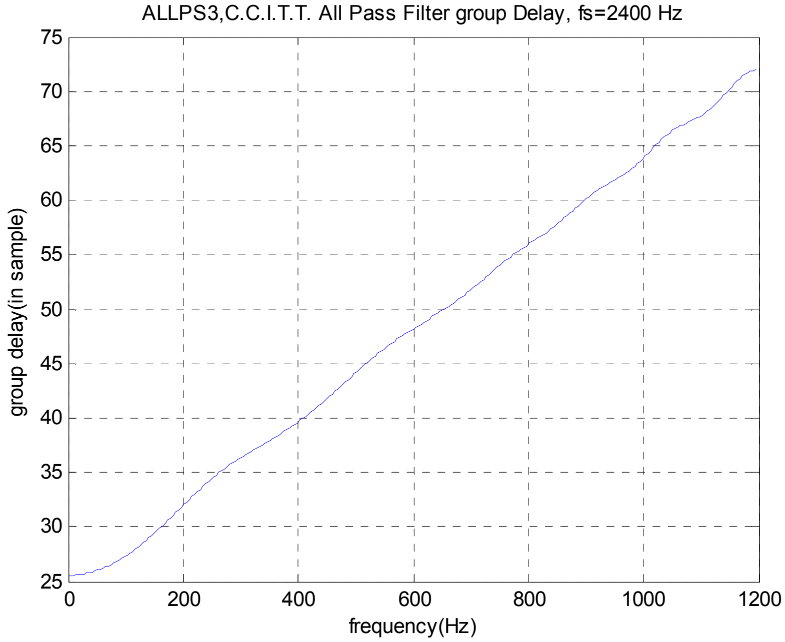
```
zeros3=[zeros;zeros;zeros];    poles3=[poles;poles;poles];    k3=k^3;
[b3,a3]=zp2tf(zeros3,poles3,k3);
```

The characteristic of ALLPS3 filter show in following:

Impulse response:



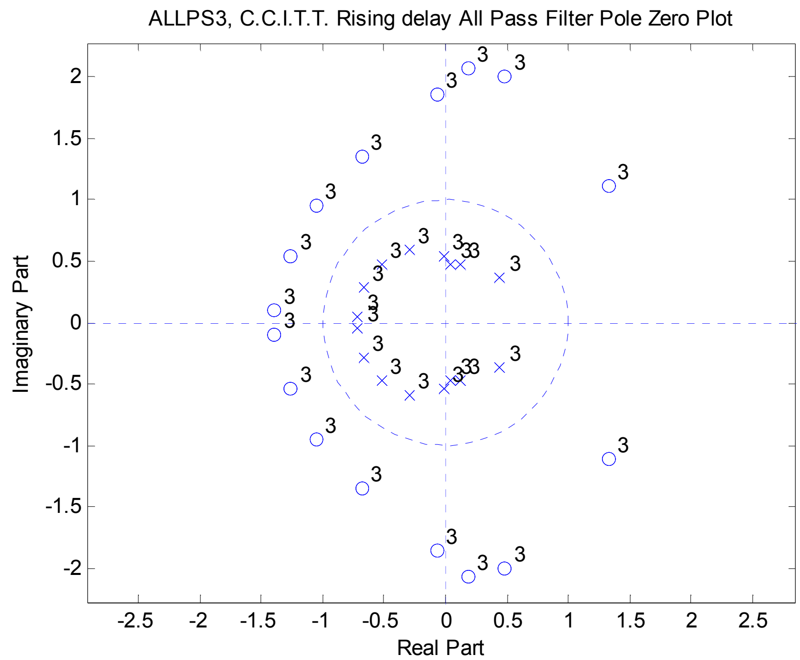
Group delay plot:



From above group delay plot, we could find the same smoothness in group delay curve as in ALLPS1 and ALLPS2 for they have the same sample point number.

The gradient value is about 60 Samples/1000Hz, which is the triple times value in ALLPS1 for ALLPS3 has the triple density zeros and poles.

Zeros-poles plot:



From above zero-poles plot, we could see that there are triple density zeros and poles in ALLPS3 filter.

8. Characteristic and Implement of Gregorian ALL10 Filter:

Gregorian filter is another kind of all-pass filter. Instead of the transfer function coefficient in CCITT 16th all-pass filter, the starting point for Gregorian filter studying is its zeros and poles. The zeros and poles of the Greegorian filter are given directly in the following:

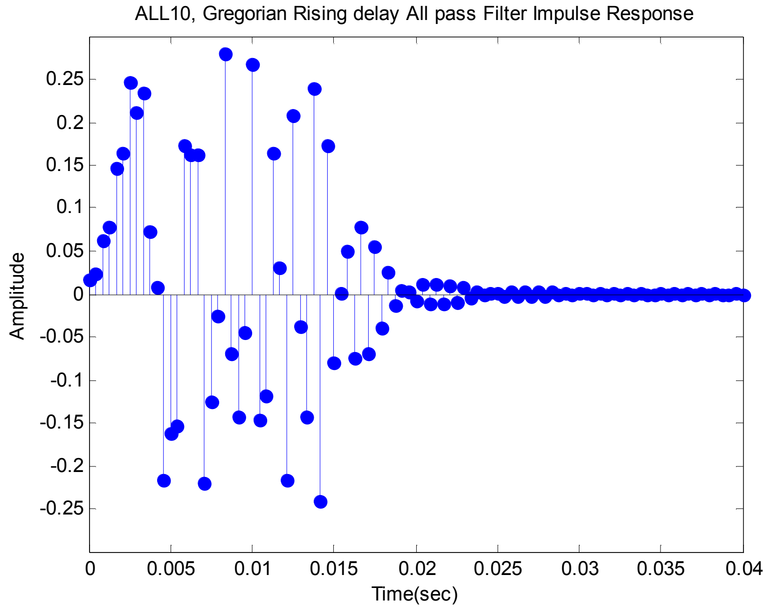
```
zeros=[ -0.461648887+1.25246385i, -0.461648887-1.25246385i, 0.973952849+0.63467108i,
        0.973952849-0.63467108i, -0.960710951+1.04881853i, -0.960710951-1.04881853i,
        -0.039003468+1.28446062i, -0.039003468-1.28446062i, 0.714604385+0.96711655i,
        0.714604385-0.96711655i,   1.054490209+0.30186918i, 1.054490209-0.30186918i,
        0.256100351+1.22419589i,   0.256100351-1.22419589i, 1.038390812+0.46496734i,
        1.038390812-0.46496734i,   0.866692305+0.80473736i, 0.866692305-0.80473736i,
        0.513129283+1.11163293i,   0.513129283-1.11163293i ];

poles=[ -0.2590804248+0.7029370198i, -0.2590804248-0.7029370198i, 0.7207036740+0.4696426373i,
        0.7207036740-0.4696426373i,   -0.4748975184+0.5184507522i, -0.4748975184-0.5184507533i,
        -0.0236189926+0.7778197051i,   -0.0236189926-0.7778197051i,   0.4942028425+0.6688340535i,
        0.4942028425-0.6688340535i,     0.8764961062+0.2509147692i,   0.8764961062-0.2509147692i,
        0.1637215923+0.7826123637i,     0.1637215923-0.7826123637i,     0.8021868165+0.3592006676i,
        0.8021868165-0.3592006676i,   0.6196157205+0.575322887i,      0.6196157205-0.575322887i,
        0.3423076782+0.7415684489i,   0.3423076782-0.7415684489i ];
```

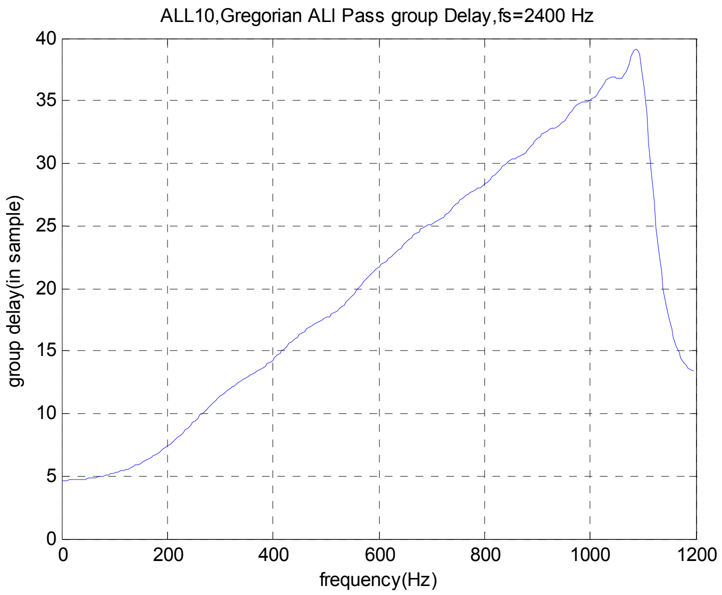
So the first step is to find its polynomial form transfer function coefficients. This is achieved with Matlab code: `[b,a]=zp2tf(zeros,poles,k);`

The other procedures in Matlab processing are the same as in CCITT ALLPS1/2/3 all-pass filter. So we just list its characteristic plot to compare them with CCITT filter:

a. Raising Delay Gregorian filter:
Impulse response:

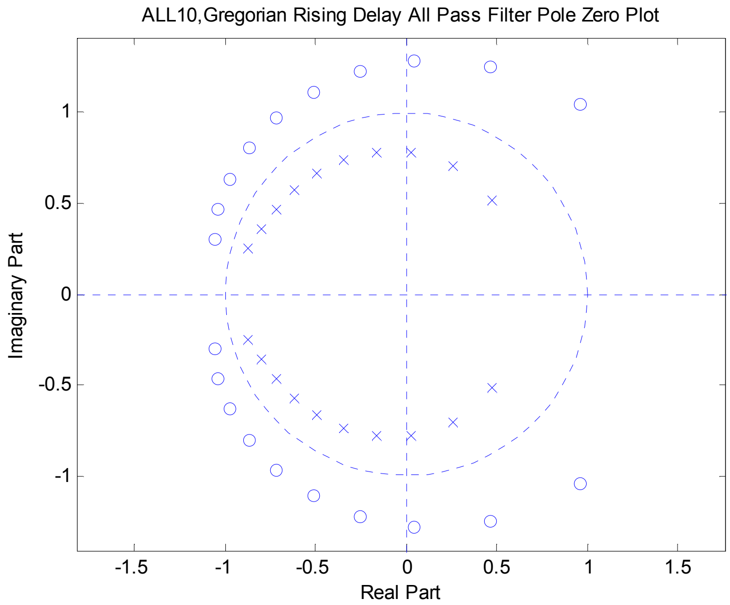


Group delay plot:



The group delay curve in above graph starts from about 100Hz and then sharply decrease from about 1080Hz. In its increasing part, it has almost const gradient as 35samples/1000Hz. The deviations from the const come from the limitation of match point for this is a 20th order filter. For higher order filter, the smoothness may increase.

Zeros-poles plot:

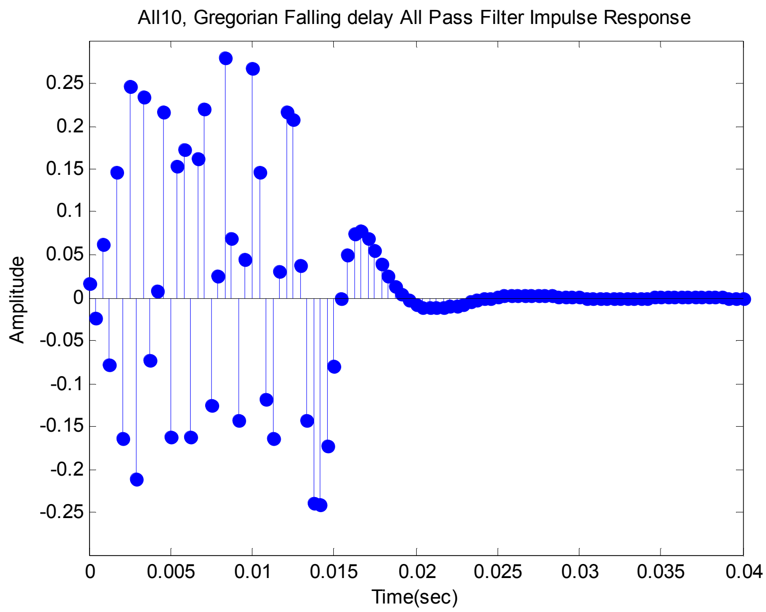


Above is the zero-pole plot of Gregorian ALL10 filter. All the poles are located inside the unit circle, so it is the stable filter.

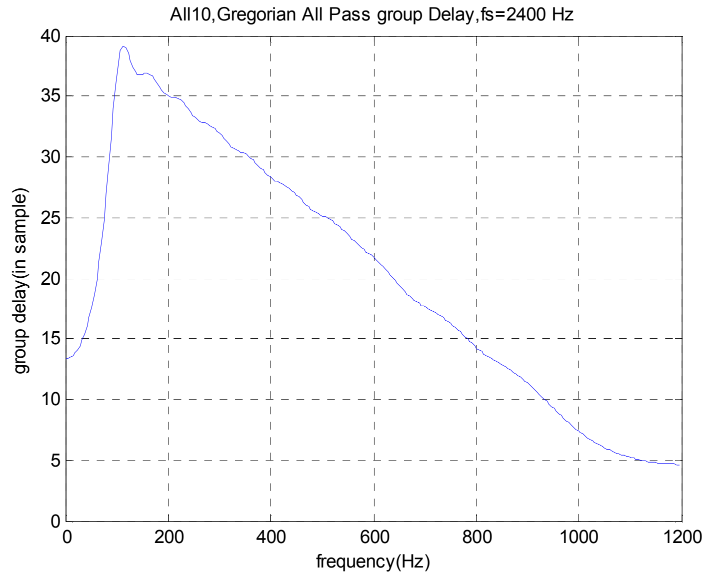
b. Falling Delay Gregorian ALL10 filter:

Similar to CCITT filter, if we inverse the sign of zeros and poles of raising delay Gregorian ALL10 filter , we could transform it into the same type Falling Delay filter. The characteristic of falling delay Gregorian filter shows in following:

Impulse response:

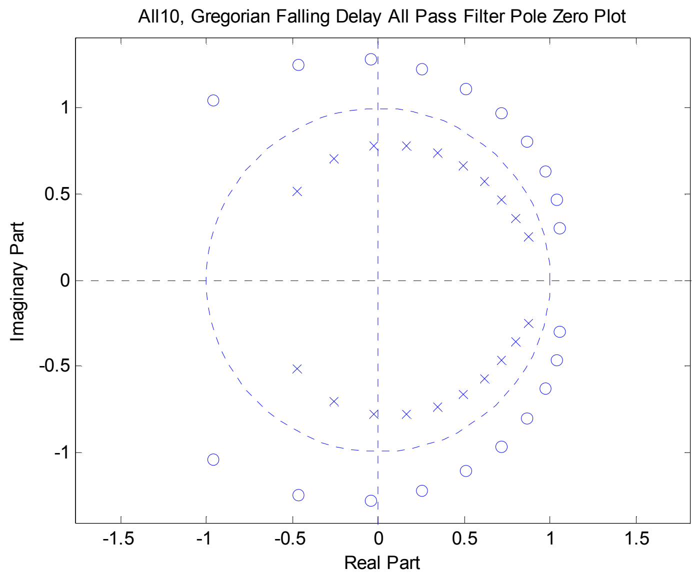


Group delay plot:



The group delay curve in above graph shows an opposite trend as in raising delay filter, so it could compensate the delay caused by raising delay filter and coupled using in transmission system.

Zeros-poles plot:



The locations of zeros and poles in falling filter are opposite to zeros and poles in raising delay filter and symmetry with imaginary axis.

9. The Second Order Section Realization of All-Pass Filter:

All the All-Pass filters will be implemented with cascade bi-quadratic digital filter sections (ALLPASS TYPE). The coefficients of the sections are the Second Order Coefficient get with Matlab which we mentioned above. The section structure is same for all filters only the coefficient is different. For convince, we take the CCITT ALLPS1 raising delay filter as the example to show it.

Firstly, we list ALLPS1 raising delay filter second order coefficients as following:

0.23116330,	-0.08498175,	1.00000000,	1.00000000,	-0.08498175,	0.23116330
0.23589759,	-0.22823696,	1.00000000,	1.00000000,	-0.22823696,	0.23589759
0.28915555,	0.03889589,	1.00000000,	1.00000000,	0.03889589,	0.28915555
0.33431174,	-0.88770844,	1.00000000,	1.00000000,	-0.88770844,	0.33431174
0.43710230,	0.59906221,	1.00000000,	1.00000000,	0.59906221,	0.43710230
0.49495076,	1.04164826,	1.00000000,	1.00000000,	1.04164826,	0.49495076
0.51147168,	1.42642334,	1.00000000,	1.00000000,	1.42642334,	0.51147168
0.53145863,	1.34326744,	1.00000000,	1.00000000,	1.34326744,	0.53145863

From above ALLPS1 raising delay all-pass filter second order section coefficients, we could get the following cascade all pass filter realization, where the coefficient is the second order coefficients we listed in above:

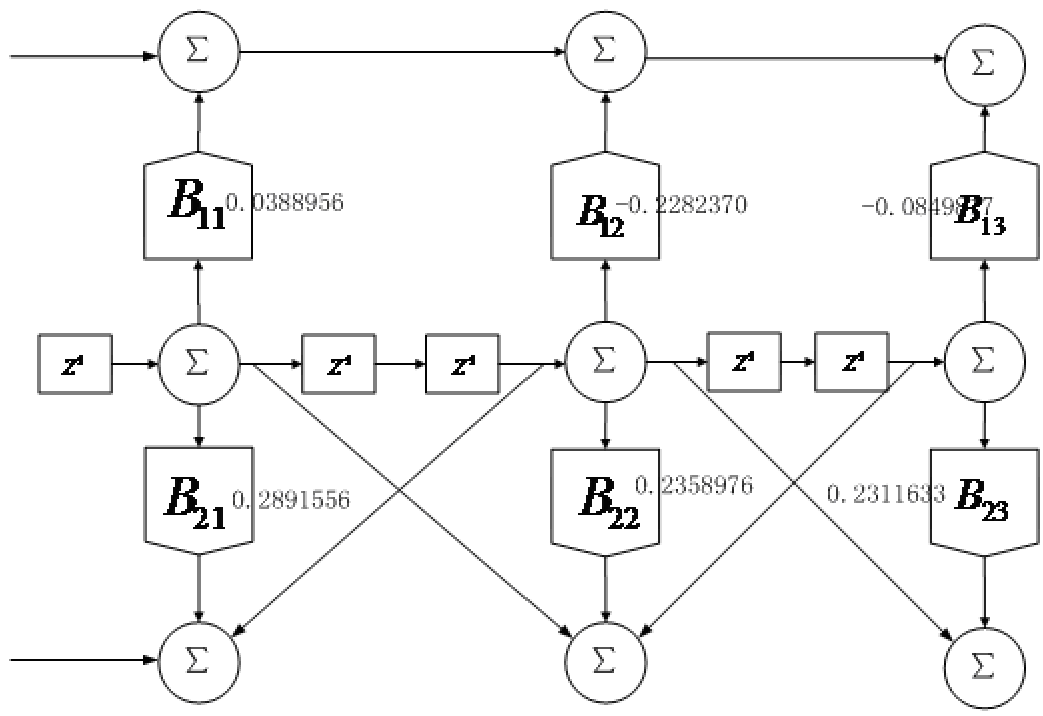


Figure 9.1 AllPS1 Raising Delay Filter Cascade Structure

If the raising delay filter is used as the smearing filter, the same type falling delay all-pass filter could be used as the de-smearing filter. The structure of the receiver filter is the same as above except the coefficient in above structure has an opposite sign. For the double and triple density implementation, each stage above is duplicated or tripled.

Part III: Experiment Results and Discussion

10.Experiment Configuration:

From this section, we will analysis and discuss the experimental results for smearing filters application in transmission system. These filters were designed to cover the modem bandwidth for the 4800I and 9600C modem.

The experiment configuration is shown in following:

As we mentioned in Introduction part, the experiment is set as the high count mode and the low count mode. With the high count test mode, the filters are connected back to back as show in figure 1:

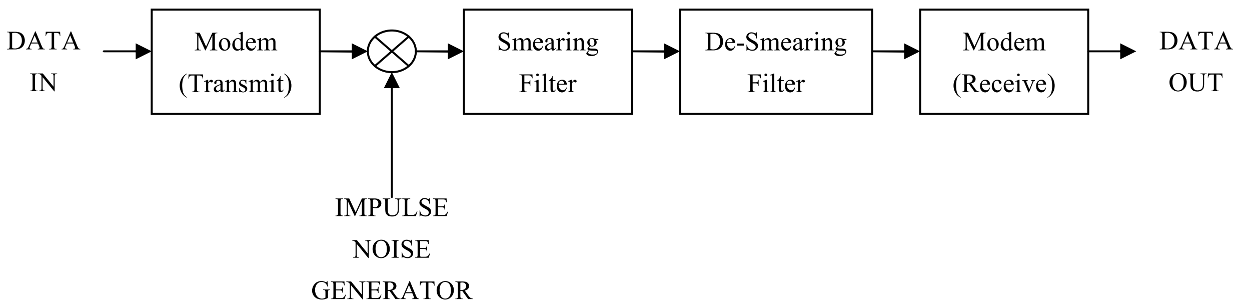


Figure 10.1 Configuration for filter in high count mode

The impulse noise is introduced between the transmit Modem and the filters.

In the low count test mode, the impulse noise generator is put into the transmission path between the smearing/de-smearing filter as shown in figure 2:

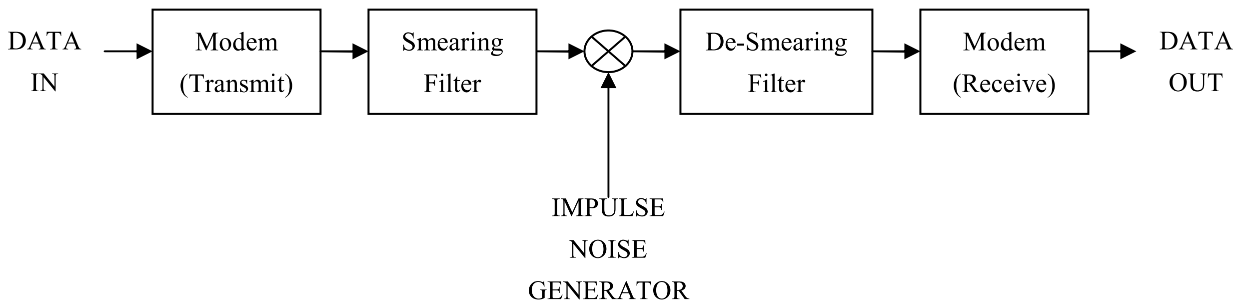


Figure 10.2 Configuration for filter in low count mode

All the test modes used periodic impulse noise. Interval between pulses is 0.15 second, the pulse width is 100 μ s as depict in figure 3,

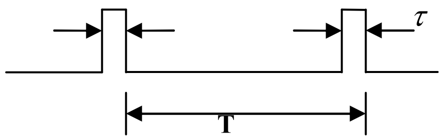


Figure 10.3 Impulse Noise Waveform

From our analysis in Part I, we could get that the low-count mode is the typical smearing transmission system. In high count mode tests, the system perform the same process for the impulse noise and transmit data. So the smearing filters have no effect in this mode. The error counts are due to the modem only. This means that high count data are interchangeable, and high count tests from one filter combination can be used in another test as long as the same modem has been used. The modems used in tests are:

CODEX 9600C (16 QAM 9600,7200,4800)
CODEX 4800I (8 PSK CONSTANT SPEED)

The smearing filters used were from two basic sources which were a 16th order polynomial (CCITT)&filters based on a Gregorian software design process. All filters were implemented in software using cascade bi-quadratic digital filter sections (ALLPS TYPE) which has discussed in Part II. All filters relied on double (or triple) density poles to give steeper delay/frequency responses(i.e. more relative delay increase/decrease in modem channel bandwidth). The smearing filters from the 16th order polynomial were originally baseband in design and since tests were done in the modulated channel clock frequency had to be increased to allow the filters to work over a larger bandwidth. The Gregorian filters were also designed for the Modem channel bandwidth.

11.Experiment results and discussion:

11.1 Test Results and Discussion for ALLPS2 filter:

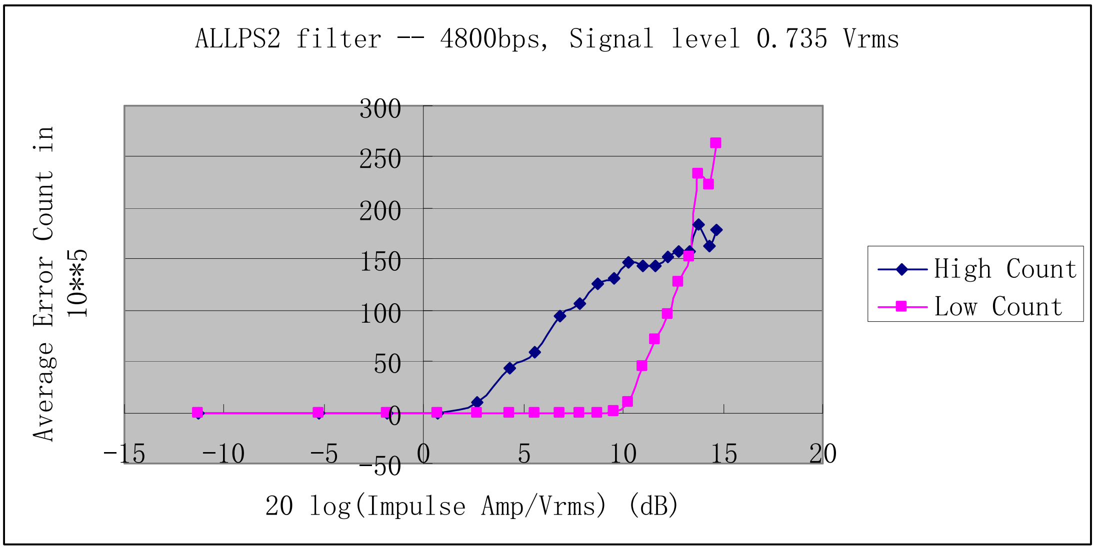
a) Average Error V.S. Amplitude of Noise Impulses

CODEX 9600 Modem, 4800bps, Signal level 0.735 Vrms

100 μ s noise impulses occurring every 100ms.

Impulse Amplitude (Volts)	20 log(Impulse Amp/Vrms) (dB)	Average Error Count in 10^5	
		HIGH COUNT	LOW COUNT
0.2	-11.3	0	0
0.4	-5.3	0	0
0.6	-1.8	0	0
0.8	0.7	0	0
1.0	2.7	9.2	0
1.2	4.3	44	0

1.4	5.6	59.6	0
1.6	6.8	93.6	0
1.8	7.8	106.2	0
2.0	8.7	126.4	0
2.2	9.5	131	1.6
2.4	10.3	146.4	9.6
2.6	11	142.6	45.8
2.8	11.6	143.2	71.2
3.0	12.2	152	96.6
3.2	12.8	156.8	128.4
3.4	13.3	157.4	151.8
3.6	13.8	184.4	232.6
3.8	14.3	162.2	221.8
4.0	14.7	177.8	263.6



In above ALLPS2 filter test results at 4800bps, we could get that when the Impulse Noise Amp/Signal Level Vrms smaller than 13.3 dB, the error rate of the High count is higher than the Low count. When the impulse noise level is small, the error rate difference is big. With the increasing of the impulse noise level, the error rate difference become smaller gradually. When noise level come to 13.3 dB, both error rates become equality. When the noise level becomes larger than 13.3 dB, the error rate in low count becomes larger than the high count.

Considering our discussion about smearing technique in Part I, we could explain this phenomenon. As we mentioned in Part I, the principle of the smearing technique to reduce the error rate is to spread the narrow impulse pulse into a wider time duration length, i.e., the smearing filter is to decrease the noise peak value with the cost increase the noise time duration length. But it doesn’t reduce the total energy of the impulse noise.

When the impulse noise level is small, the total energy of the noise is also small. The smearing filter spread this limited energy into wider time duration, the impulse noise peak value will decrease noticeable. So the smearing filter could decrease the error rate noticeable.

When the impulse noise level become higher and higher, the noise energy also become larger. The smearing filter still spread this noise into the same time length (this is determined by the filter's group delay, not dependent on noise level), so the smeared noise level may not be negligible compare to signal level. The error rate in low count thus will increase with the increasing of noise level.

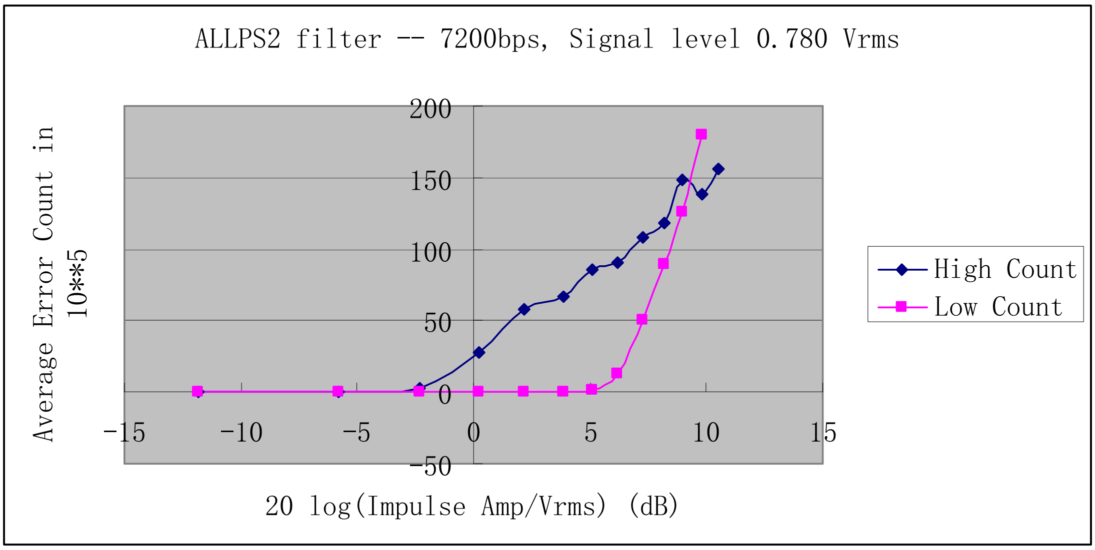
When the noise level comes to a certain level, the error rate in low count mode will comes to the error rate level in high count mode, we may call this noise level as the **impulse noise threshold level**. In this test, this impulse noise threshold level is 13.3dB. Above this threshold level, the smearing filter still spread the noise into a wider time length, but the error rate in unit noise duration time will increase further. Not only this, because the smeared noise has a longer duration time, the total error rates caused by smeared noise pulse will become higher than the original noise pulse. This is the phenomenon that we observed in this test.

b) Average Error V.S. Amplitude of Noise Impulses

CODEX 9600 Modem, 7200bps, Signal level 0.780 Vrms

100 μ s noise impulses occurring every 100ms.

Impulse Amplitude (Volts)	20 log(Impulse Amp/Vrms) (dB)	Average Error Count in 10^5	
		HIGH COUNT	LOW COUNT
0.2	-11.8	0	0
0.4	-5.8	0	0
0.6	-2.3	2.8	0
0.8	0.2	27.4	0
1.0	2.2	57.8	0
1.2	3.9	66.4	0
1.4	5.1	85.2	1.4
1.6	6.2	91	12.4
1.8	7.3	108.4	50
2.0	8.2	118.6	90
2.2	9	148.8	126
2.4	9.8	138.6	180.2
2.6	10.5	156	



Above is the test results for the same Modem, same filter, same impulse noise conditions but the transmit speed increase to7200bps. It could find that the basic plot trend is the same as in the case of 4800bps, only the impulse noise threshold level lower to about 9.2dB. The reason for it is that:

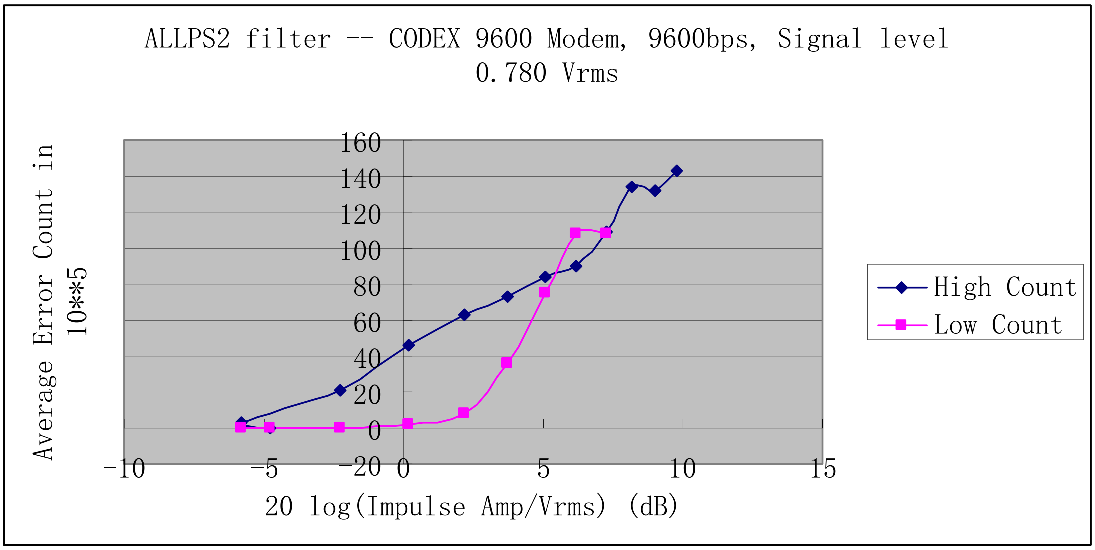
In this test, we use the same type filter (ALLPS2), so it will spread the noise into the same time length as in the case of 4800bps. But Modem speed has raise to 7200bps, which means there are more signal bits affect by smeared noise during this time duration. So the transmit signal error rate is higher than 4800bps.

c) Average Error V.S. Amplitude of Noise Impulses

CODEX 9600 Modem, 9600bps, Signal level 0.780 Vrms

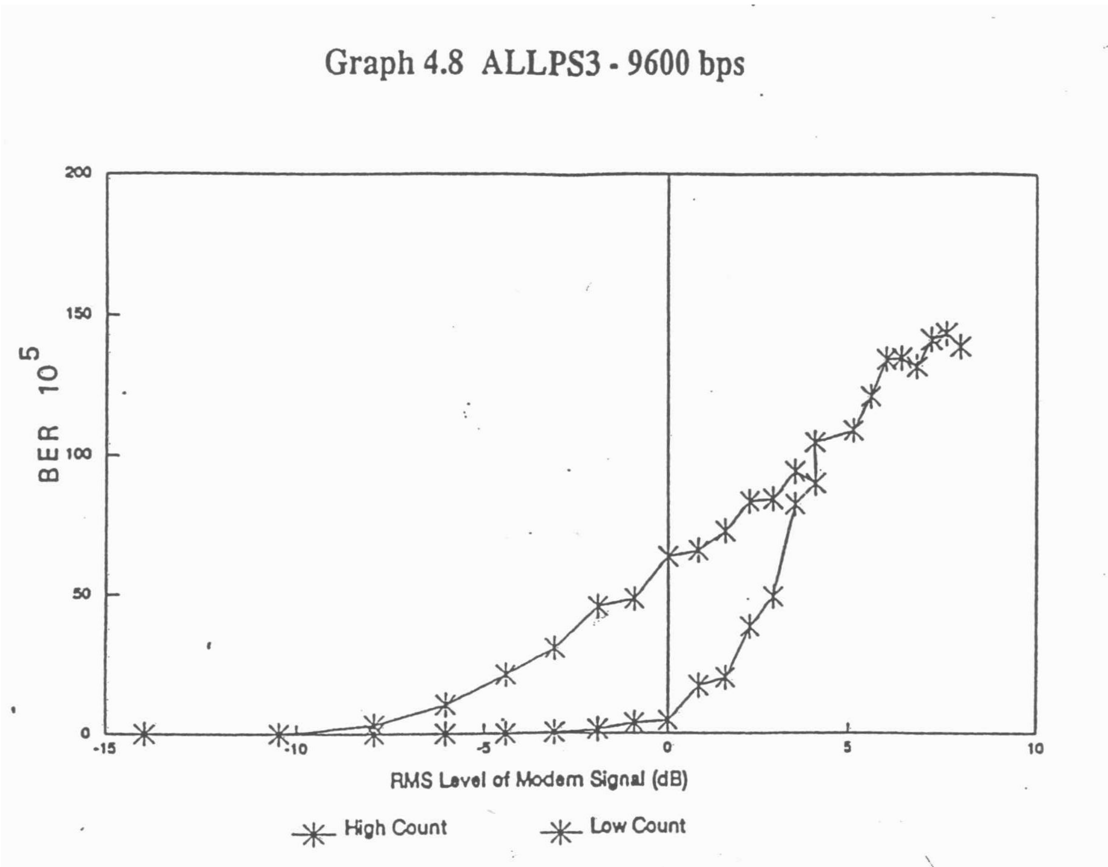
100 μs noise impulses occurring every 100ms.

Impulse Amplitude (Volts)	20 log(Impulse Amp/Vrms) (dB)	Average Error Count in 10^5	
		HIGH COUNT	LOW COUNT
0.2	-4.8	0	0
0.4	-5.8	3	0
0.6	-2.3	21	0
0.8	0.2	45.8	1.8
1.0	2.2	63.4	8.2
1.2	3.7	72.6	35.6
1.4	5.1	84.2	75.4
1.6	6.2	89.8	108
1.8	7.3	108.8	108.2
2.0	8.2	134.4	
2.2	9.0	131.6	
2.4	9.8	143.4	



Above is the result in 9600bps. It could find that the impulse noise threshold level has down to about 5.2dB. As we mentioned the reasons in 7200bps, with the raising of the Modem speed, the affected transmit signal also raising, so the transmit error rate also increasing.

11.2. Test Results and Discussion for ALLPS3 filter:



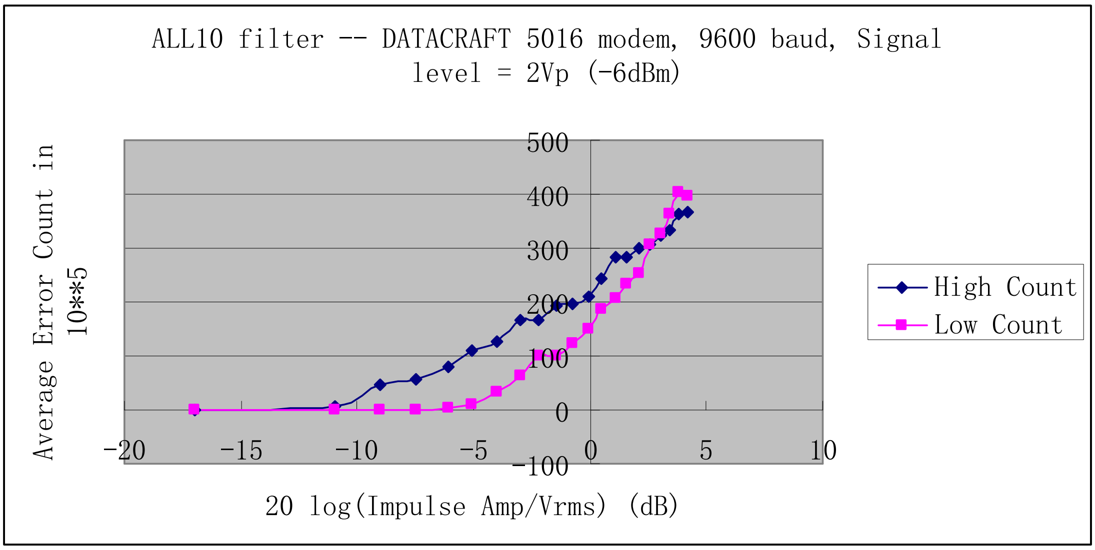
Above is the comparison test result with ALLPS3 filter at the 9600bps. It shows the impulse noise threshold level is about 4.7 dB. Comparing it with the result from ALLPS2 at the 9600bps, they don't have noticeable difference. It seems that when the group delay increase to a certain level, the benefit from the noise pulse spread is not increase noticeable.

11.3. Test Results and Discussion for ALL10 filter:

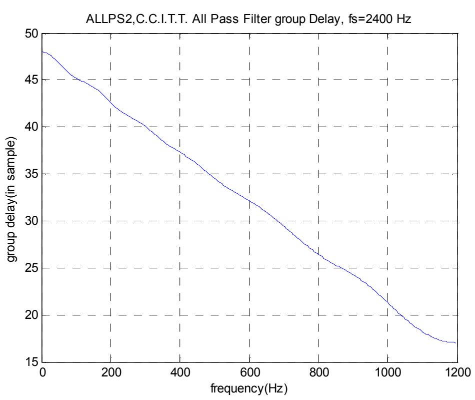
Average Error V.S. Amplitude of Noise Impulses
DATACRAFT 5016 modem, 9600 baud, Signal level = 2Vp (-6dBm)

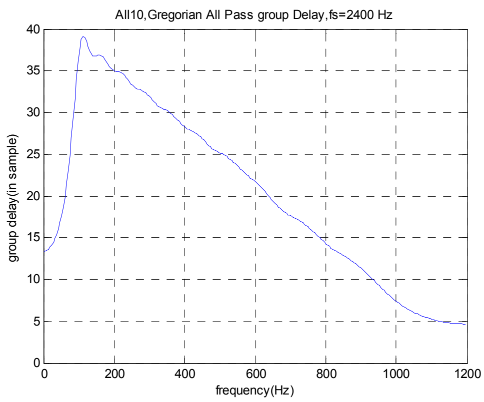
Repetition: 0.15 sec, Pulse width: 100 μ s, Fs: 13.6 kHz.

Impulse Amplitude (Volts)	20 log(Impulse Amp/Vrms) (dB)	Average Error Count in 10^5	
		HIGH COUNT	LOW COUNT
0.2	-16.98	0	0
0.4	-10.97	5.6	0
0.5	-9.03	47	0
0.6	-7.45	55.2	0
0.7	-6.11	80.33	4
0.8	-9.95(-5.04)	109.8	10.3
0.9	-3.96	128.2	33.2
1.0	-3.01	166.2	62.6
1.1	-2.18	167.3	100
1.2	-1.43	192.8	99
1.3	-0.73	196.2	124.8
1.4	-0.09	209.3	148.6
1.5	0.51	241.8	186.2
1.6	1.07	283.7	205.3
1.7	1.59	283.5	233.6
1.8	2.09	300.6	254
1.9	2.56	307.5	308.2
2.0	3.01	324.4	326.5
2.1	3.43	332.8	361.7
2.2	3.84	363.6	403
2.3	4.22	366.4	395.7



Using ALL10 filter do the same test, could get the similar result. The impulse noise threshold level in this test is about 2.56dB at the 9600 baud rate. This level is lower the value of ALLPS2 and ALLPS3 at the same speed. This may understandable from two filters' group delay which we discussed in Part II, we list them again:





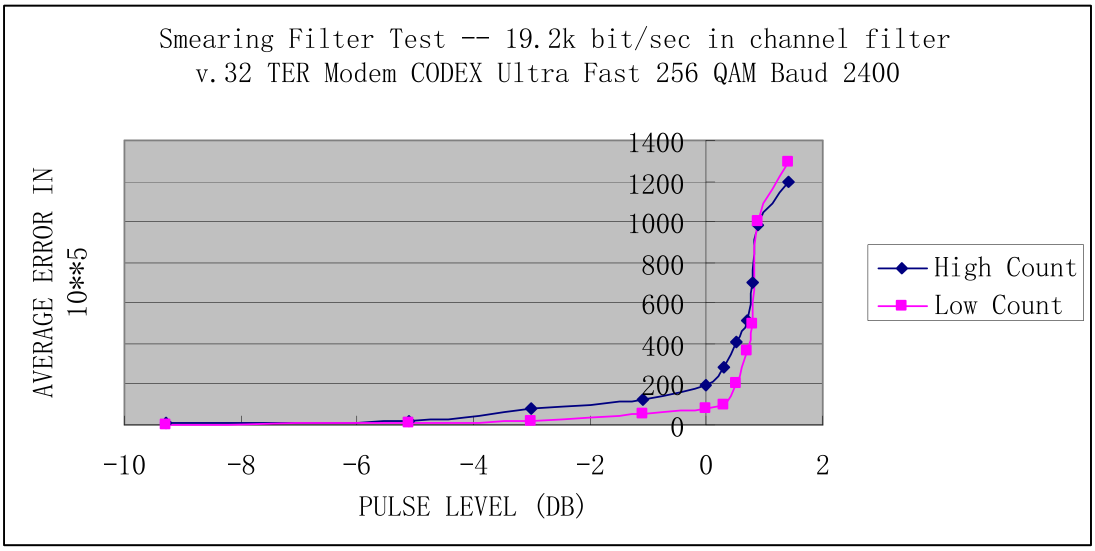
Comparing above ALLPS2 and ALL10 falling delay filter's group delay, we could find **(a.)** the ALLPS2 filter has a bigger gradient value than ALL10, and **(b.)** at the zero Hz, ALLPS2 has the maximum group delay value, but the ALL10 filter's group delay is gradually increase to its maximum value from zero frequency. So All10 filter's delay to the low frequency component may not as good as the ALLPS2 filter. That means the ALLPS2 filter may have a better pulse spread effect.

11.4 Smearing Filter Test Results:

19.2k bit/sec in channel filter
v.32 TER Modem
CODEX Ultra Fast 256 QAM
Baud 2400
Pulse 100 Microseconds REP. Rate 0.15 Seconds

PULSE LEVEL DB	AVERAGE ERROR IN 10^5

	H.C.	L.C
-9.3	11	0
-5.1	20	7
-3.0	80	20
-1.1	121	50
0	191	80
0.3	280	100
0.5	407	200
0.7	515	361
0.8	698	499
0.9	981	997
1.4	1200	1295



This test also has the similar result and verifies our discussion in previous sections again.

Part IV: Conclusion

12.Conclusion:

This paper consists of three parts.

In part I, we discussed the principle of All-pass filter and its Gregorian design method and its application in transmission system, i.e. smearing technique. When we discussed the smearing technique, we reached to the following conclusion which becomes the basis of our analysis in section III:

In smearing system, it is the de-smearing filter in receiver part that performs the smearing function to impulse noise. The smeared pulse noise spreading time length is determined by the filter group delay. The smearing filter in transmitter is used to compensate for transmit signal. Because the smearing technique is to spread the narrow high-peak impulse into a wider time duration but cannot reduce the total energy of the noise, so it is most effective at the case of low impulse repetition rate.

In Part II, we studied some all-pass filter's characteristic and their implementation, which including CCITT AllPS1/2/3 and Gregorian ALL10 filter. The analysis and design are based on the Matlab software. The analysis starting point for CCITT filter is its polynomial transform coefficients, the staring point for the Gregorian filter is its zeros-poles value. The ALLPS2/3 filter has the doubled and tripled density of zeros and poles for ALLPS1, so they have the doubled and tripled group delay gradient value than ALLPS1. All the filters are implemented as the second order section form. And the second order section coefficients could get with the Matlab function:

```
sos=zp2sos(zeros, poles, k)
```

From these second order section coefficients, we could implement the filter with ALLPASS type second order section structure. All the filters have the same structure, only the coefficients value and sign may different with the difference of the filter.

In Part III, we studied some Modem transmission system's test results with the smearing filter we discussed in Part II. The experiment configuration is set as the high count mode and the low count mode. The low count mode is the typical smearing transmission system, while in high count mode, the smearing filter don't play an effective role. From these test, we could get the following results:

a) When the impulse noise level belows a certain level, which we called it as impulse noise threshold level in this paper, the error rate in low count mode is lower than the high count mode.
b) When noise level is above the threshold value, the error rate in low count mode become higher than in high count mode.
c) With the increasing of the transmit speed, the noise threshold value is lowering gradually.

d) For the filter which has a lower group delay in low frequency components and a smaller gradient value, they have a lower noise threshold level.

All the above test results could be explained with our analysis conclusion in Part I, that is:

Given the smearing filter is to just decrease the noise peak value, it doesn’t reduce the total energy of the impulse noise.

When the impulse noise level is small, the total energy of the noise is also small. The smearing filter spread this limited energy into wider time duration, the impulse noise peak value will decrease noticeable. So the smearing filter could decrease the error rate noticeable at this noise level range.

With the increasing of the impulse noise level, the noise energy also increase. The smearing filter still spread this noise into the same time length , so the smeared noise level may not be negligible compare to signal level. The error rate in low count thus increase with the increasing of noise level.

When the noise level comes to the threshold noise level, the error rate in low count mode will be equal with error rate level in high count mode. Above this threshold level, with the error rate in unit noise duration time increasing further, the total error rates caused by smeared noise pulse will become higher than the original noise pulse for the smeared noise has a longer duration time.

For the higher transmit speed, the transmit signal error rate is also increasing because there are more signal bits are affect by smeared noise during this time duration.

When we compared the results of ALLPS2 filter with ALL10 filter, we could find the ALL10 has a lower noise threshold level. The reasons may come from:
(a) ALLPS2 filter has a bigger gradient value than ALL10, and
(b) At 0Hz, ALLPS2 has the maximum group delay value, but the ALL10 filter’s group delay gradually increases to its maximum value from zero frequency. So the ALLPS2 filter may have a better pulse spread effect for low frequency signal.

13.References

[1]R. Gregorian, G. C. Temes, "Design Techniques for Digital and Analog All-pass Circuits", IEEE Transactions on Circuits and Systems, VOL.CAS-25, No.12, Dec. 1978. pp981-988.
[2]R. A. Wainwright, "On the potential Advantage of a Smearing-Desmearing Filter Technique in Overcoming Impulse-Noise Problems in Data Systems", IRE Transactions on Communications Systems, Dec. 1961, pp362-366
[3]W. J. Richter, T. I. Smiths, "Signal Design and Error Rate of an Impulse Noise Channel", IEEE Transaction on Communication Technology, Aug.1971, pp496-458

14.Appendices:

14.1. APPENDIX I:

File: Allps2.m

This is the main program. It will invoke the second file: Data.m during the running to give the required results.

```
%*************************************************************************
%16th order polynomial C.C.I.T.T.
%*************************************************************************
a=[1.0000,3.24837,5.27556,5.83603,5.03954,3.68737,2.42122,1.49040,0.88413,0.50638,0.27250,0.13289,0.05723,0.02149,0.00652,0.00153,0.00031];
b=fliplr(a);
fclock=2400;
strings={'ALLPS1,C.C.I.T.T Rising delay ALl Pass FIlter Impulse Response' 'ALLPS1,CCITT All Pass Filter group Delay, fs=2400hz' 'ALLPS1,C.C.I.T.T. Rising Delay All Pass Filter Pole ZeroPlot' 'ALLPS1, C.C.I.T.T. Raising Delay All Pass Filter Pole Zero Plot' 'AllPS1_rd.txt' 'AllPS1 Rising Delay Filter' };
[zeros,poles,k]=tf2zp(b,a);
data(strings,a,b,zeros,poles,k,fclock);

zeros=zeros.*-1;
poles=poles.*-1;
[b,a]=zp2tf(zeros,poles,k);
strings={'ALLPS1,C.C.I.T.T Falling delay ALl Pass FIlter Impulse Response' 'ALLPS1,CCITT All Pass Filter group Delay, fs=2400hz' 'ALLPS1,C.C.I.T.T. Falling Delay All Pass Filter Pole ZeroPlot' 'ALLPS1, C.C.I.T.T. Falling Delay All Pass Filter Pole Zero Plot' 'AllPS1_fd.txt' 'AllPS1 Falling Delay Filter'};
data(strings,a,b,zeros,poles,k,fclock);
%--------------------------------------------------------
% ALLPS2(double density filter)
%--------------------------------------------------------
zeros2=[zeros;zeros];
poles2=[poles;poles];
k2=k^2;
[b2,a2]=zp2tf(zeros2,poles2,k2);

strings={'ALLPS2, C.C.I.T.T. Falling delay All Pass Filter Impulse Response' 'ALLPS2,C.C.I.T.T. All Pass Filter group Delay, fs=2400 Hz' 'ALLPS2, C.C.I.T.T. Falling delay All Pass Filter Pole Zero Plot' 'AllPS2_fd.txt' 'ALLPS2 Falling Delay Filter'};
data(strings,a2,b2,zeros2,poles2,k2,fclock);

%rising
zeros2=zeros2.*-1;
poles2=poles2.*-1;
[b2,a2]=zp2tf(zeros2,poles2,k2);
```

```
strings={'ALLPS2, C.C.I.T.T. Rising delay All Pass Filter Impulse Response' 'ALLPS2,C.C.I.T.T. All Pass Filter group Delay, fs=2400 Hz' 'ALLPS2, C.C.I.T.T. Rising delay All Pass Filter Pole Zero Plot' 'AllPS2_rd.txt' 'ALLPS2 Rising Delay Filter'};
data(strings,a2,b2,zeros2,poles2,k2,fclock);

%--------------------------------------------------------
% ALLPS3(triple density filter)
%--------------------------------------------------------
zeros3=[zeros;zeros;zeros];
poles3=[poles;poles;poles];
k3=k^3;
[b3,a3]=zp2tf(zeros3,poles3,k3);

strings={'ALLPS3, C.C.I.T.T. Falling delay All Pass Filter Impulse Response' 'ALLPS3,C.C.I.T.T. All Pass Filter group Delay, fs=2400 Hz' 'ALLPS3, C.C.I.T.T. Falling delay All Pass Filter Pole Zero Plot' 'AllPS3_fd.txt' 'ALLPS3 Falling Delay Filter'};
data(strings,a3,b3,zeros3,poles3,k3,fclock);

%rising
zeros3=zeros3.*-1;
poles3=poles3.*-1;
[b3,a3]=zp2tf(zeros3,poles3,k3);
strings={'ALLPS3, C.C.I.T.T. Rising delay All Pass Filter Impulse Response' 'ALLPS3,C.C.I.T.T. All Pass Filter group Delay, fs=2400 Hz' 'ALLPS3, C.C.I.T.T. Rising delay All Pass Filter Pole Zero Plot' 'AllPS3_rd.txt' 'ALLPS3 Rising Delay Filter'};
data(strings,a3,b3,zeros3,poles3,k3,fclock);
fclock=2400;

%greegorian filter
zeros=[
        -0.461648887+1.25246385i,
        -0.461648887-1.25246385i,
        0.973952849+0.63467108i,
        0.973952849-0.63467108i,
        -0.960710951+1.04881853i,
        -0.960710951-1.04881853i,
        -0.039003468+1.28446062i,
        -0.039003468-1.28446062i,
        0.714604385+0.96711655i,
        0.714604385-0.96711655i,
        1.054490209+0.30186918i,
        1.054490209-0.30186918i,
        0.256100351+1.22419589i,
```

```
        0.256100351-1.22419589i,
        1.038390812+0.46496734i,
        1.038390812-0.46496734i,
        0.866692305+0.80473736i,
        0.866692305-0.80473736i,
        0.513129283+1.11163293i,
        0.513129283-1.11163293i
        ];

poles=[
      -0.2590804248+0.7029370198i,
      -0.2590804248-0.7029370198i,
      0.7207036740+0.4696426373i,
      0.7207036740-0.4696426373i,
      -0.4748975184+0.5184507522i,
      -0.4748975184-0.5184507533i,
      -0.0236189926+0.7778197051i,
      -0.0236189926-0.7778197051i,
      0.4942028425+0.6688340535i,
      0.4942028425-0.6688340535i,
      0.8764961062+0.2509147692i,
      0.8764961062-0.2509147692i,
      0.1637215923+0.7826123637i,
      0.1637215923-0.7826123637i,
      0.8021868165+0.3592006676i,
      0.8021868165-0.3592006676i,
      0.6196157205+0.575322887i,
      0.6196157205-0.575322887i,
      0.3423076782+0.7415684489i,
      0.3423076782-0.7415684489i
      ];

k=16.832E-3;
[b,a]=zp2tf(zeros,poles,k);
strings={'All10, Gregorian Falling delay All Pass Filter Impulse Response' 'All10,Gregorian All Pass group Delay,fs=2400 Hz' 'All10, Gregorian Falling Delay All Pass Filter Pole Zero Plot' 'All10_fd.txt' 'All10 Gregorian Falling Delay Filter'};
data(strings,a,b,zeros,poles,k,fclock);

%rising
zeros=zeros.*-1;
poles=poles.*-1;
[b,a]=zp2tf(zeros,poles,k);
strings={'ALL10, Gregorian Rising delay All pass Filter Impulse Response' 'ALL10,Gregorian ALl Pass group
```

```
Delay,fs=2400 Hz' 'ALL10,Gregorian Rising Delay All Pass Filter Pole Zero Plot' 'All10_rd.txt' 'All10 Gregorian Rising Delay Filter'};
    data(strings,a,b,zeros,poles,k,fclock);
```

14.2. APPENDIX II:

File: Data.m

```
function result =data2(strings,a,b,zeros,poles,k,fclock)
%to plot the impulse response
impz(b,a,2.^8,fclock);
title(strings(1));
ylabel('Amplitude');
xlabel('Time(sec)');
axis([0,0.04,-0.3,0.3]);
pause;

%to plot the group delay
grpdelay(b,a,2.^8,fclock);
title(strings(2));
ylabel('group delay(in sample)');
xlabel('frequency(Hz)');
pause;

%plot the pole zero plot
zplane(zeros,poles);
title(strings(3));
pause;

%save all results to file
fid=fopen(char(strings(4)),'wt');
buffer=num2str(zeros.',6);

%fprintf(fid,'%s\n\n',char(strings(5)));
fprintf(fid,'Zeros,\n%s\n',buffer);
buffer=num2str(poles.',6);
fprintf(fid,'POles,\n%s\n',buffer);
buffer=num2str(k.',6);
fprintf(fid,'K,\n%s\n',buffer);

% %second order sections
sos=zp2sos(zeros,poles,k);
fprintf(fid,'\nSecond Order Section Coefficient\n');
fprintf(fid,'%2.8f,%2.8f,%2.8f,%2.8f,%2.8f,%2.8f\n',sos.');
```

```
%difference equation coefficients
fprintf(fid,'\nDifference Equation Coefficients\n');
fprintf(fid,'b,numerator\n');
fprintf(fid,'%2.15f,%2.15f,%2.15f,%2.15f\n',b');
fprintf(fid,'\na,denominator\n');
fprintf(fid,'%2.15f,%2.15f,%2.15f,%2.15f\n',a');
fclose(fid);
```

14.3. APPENDIX III:

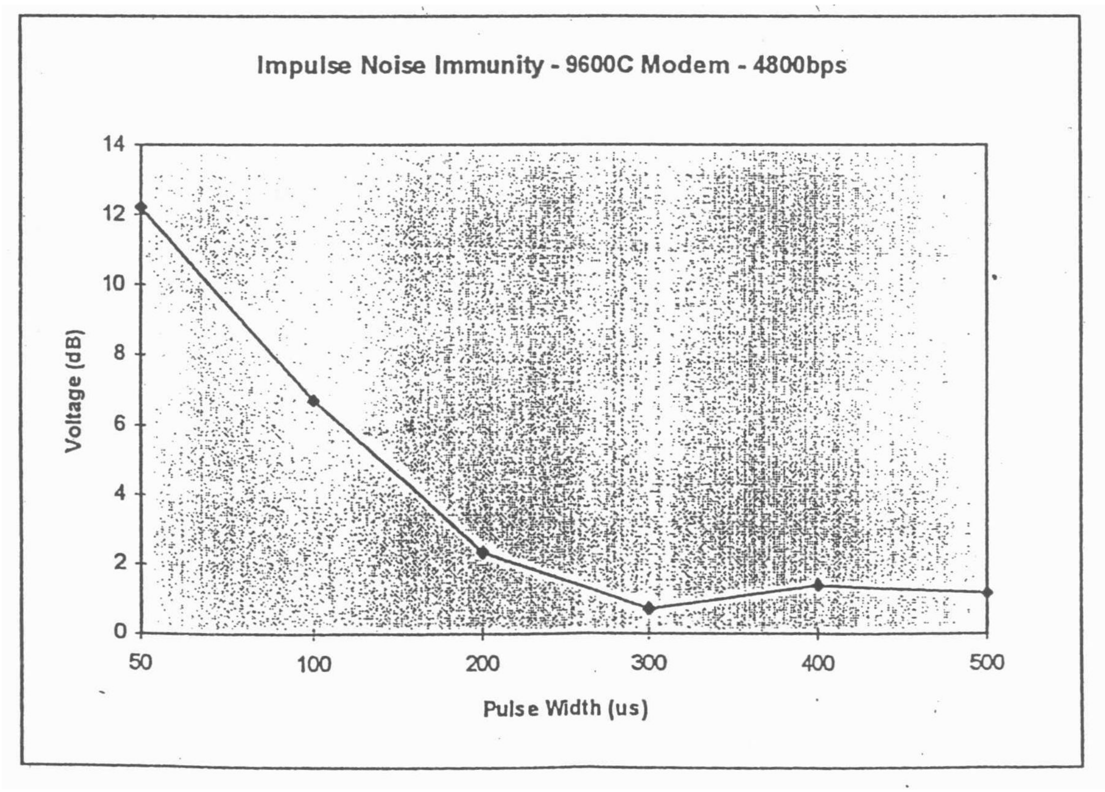
Modem Noise Immunity Data:

The following shows the impulse noise immunity for the 9600C codex modem at 4800,7200, 9600 bps and the impulse noise immunity for the 4800I modem at 4800bps.

Modem noise Immunity is a test of the Modem only. An interfering pulse is injected into the channel and the height and width are varied. The error free region is below the curve in each case.

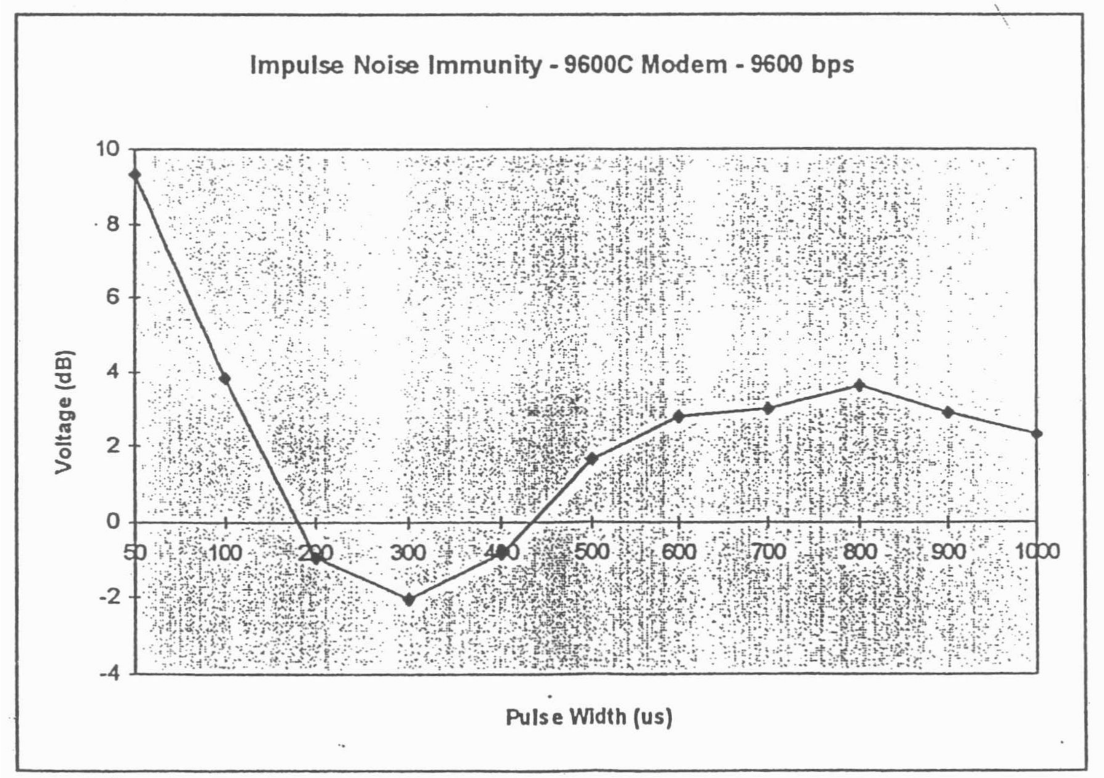
a) 9600C Codex Modem – 4800bps:

Pulse (μ s)	Amplitude (Volts)	20 log(Amp/Vrms) (dB)
50	3.01	12.25
100	1.59	6.70
200	0.96	2.32
300	0.80	0.74
400	0.86	1.37
500	0.84	1.16



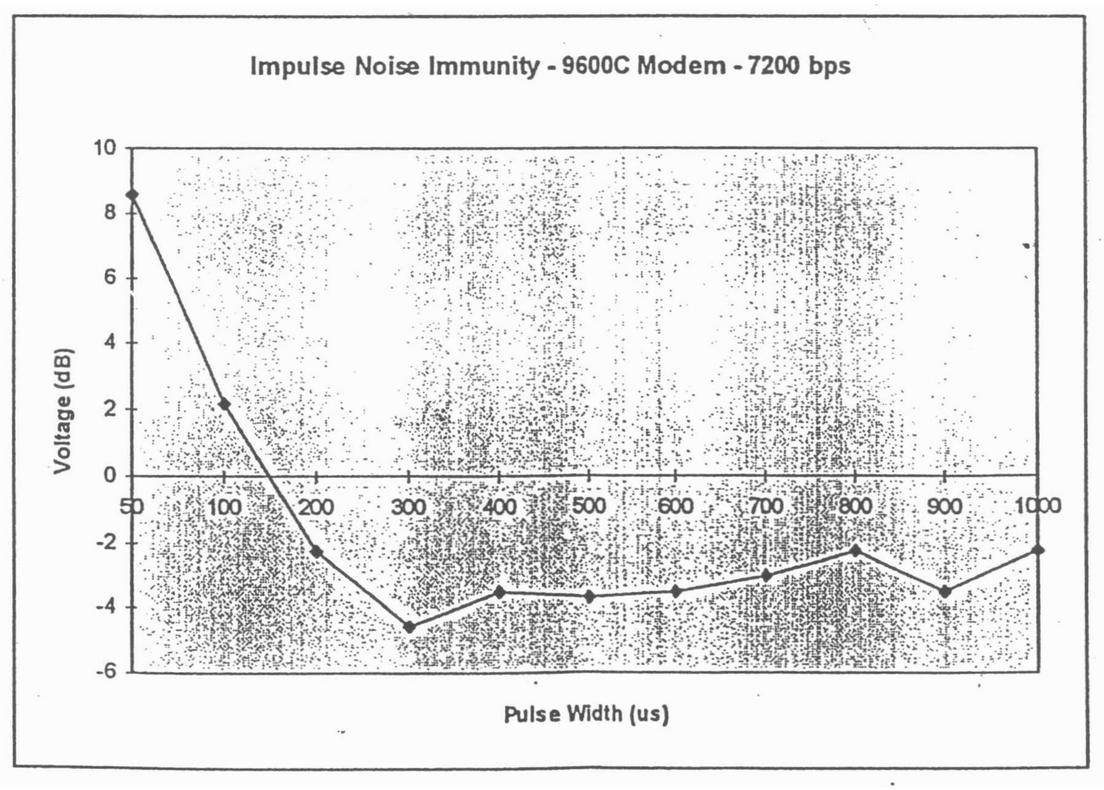
b) 9600C Codex Modem – 9600bps:

Pulse (μ s)	Amplitude (Volts)	20 log(Amp/Vrms) (dB)
50	1.7	9.34
100	0.9	3.82
200	0.52	-0.95
300	0.46	-2.01
400	0.54	-0.82
500	0.70	1.63
600	0.82	2.79
700	0.82	3.00
800	0.88	3.62
900	0.81	2.90
1000	0.76	2.34



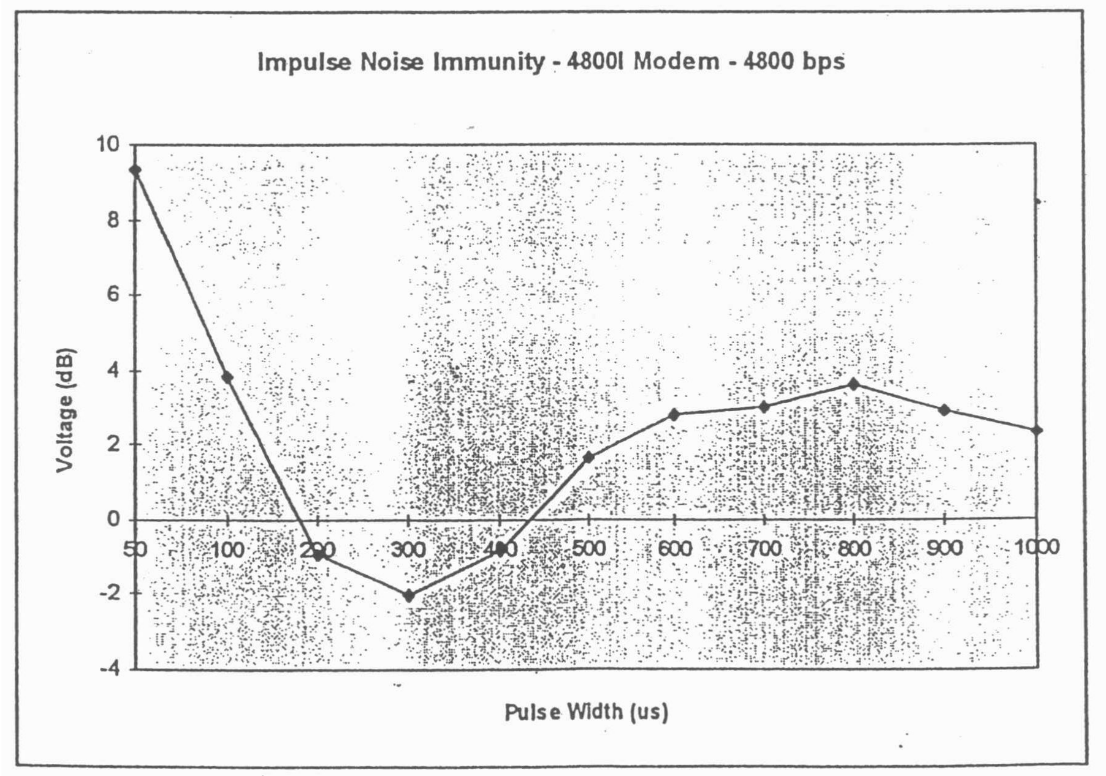
c) 9600C Codex Modem – 7200bps:

Pulse (μ s)	Amplitude (Volts)	20 log(Amp/Vrms) (dB)
50	2.1	8.6
100	1.0	2.16
200	0.60	-2.28
300	0.46	-4.59
400	0.52	-3.52
500	0.51	-3.69
600	0.52	-3.52
700	0.55	-3.03
800	0.60	-2.28
900	0.52	-3.52
1000	0.60	-2.28



d) 4800I Modem – 4800bps:

Pulse (μ s)	Amplitude (Volts)	20 log(Amp/Vrms) (dB)
50	1.7	9.34
100	0.9	3.82
200	0.52	-0.95
300	0.46	-2.01
400	0.54	-0.82
500	0.70	1.63
600	0.82	2.79
700	0.82	3.00
800	0.88	3.62
900	0.81	2.90
1000	0.76	2.34


Impulse Noise Immunity - 4800I Modem - 4800 bps
Voltage (dB)
10
8
6
4
2
0
-2
-4
50
100
200
300
400
500
600
700
800
900
1000
Pulse Width (us)