VIRTUAL EXPERIENCE
OCTOBER 11-14

2021 Fall Technical Forum
SCTE • NCTA • CABLELABS

OFDMA Predistortion Coefficient and OFDM Channel Estimation Decoding and Analysis

Remove the linear delay, examine the group delay

A Technical Paper prepared for SCTE by

Tom Williams
Distinguished Technologist
CableLabs
858 Coal Creek Circle, Louisville, CO 80027
303.661.9100
t.williams@cablelabs.com

Alberto Campos
Fellow
CableLabs
858 Coal Creek Circle, Louisville, CO 80027
303.661.9100
a.campos@cablelabs.com

Lin Cheng, CableLabs

James Lin, CableLabs

Jason Rupe, CableLabs

Jay Zhu, CableLabs

Table of Contents

Title **Page Number**

List of Figures

Title **Page Number**

1. Introduction

Upstream DOCSIS® carriers use predistortion data to cancel linear distortion. Equalization is a required step in a digital transmission system to remove linear distortion. In continuous DOCSIS downstream transmissions the equalization is done in the cable modem (CM) receiver, but in DOCSIS upstream the burst transmissions are pre-distorted by the CM so that they arrive at the cable modem termination system (CMTS) receiver with linear distortion removed. The predistortion is adjusted during a periodic process called ranging, which occurs about every 20 to 30 seconds.

An analogy for predistortion is corrective eyeglasses, which custom pre-distort an image so that it arrives on a wearer's retina in focus. An optometrist can measure lenses and determine what visual impairments are being corrected, such as myopia or astigmatism. Likewise, upstream predistortion can be analyzed to determine what linear distortion impairments are being cancelled, such as echoes, group delay, tilt, suck-outs, filters, etc.

DOCSIS 3.0 single carrier upstream equalization data has been mined for over 10 years to provide troubleshooting information about plant impairments. Now both DOCSIS 3.1 upstream orthogonal frequency division multiple access (OFDMA) multicarrier equalization and downstream orthogonal frequency division multiplexing (OFDM) multicarrier channel estimation data are available from CMs using management information base (MIB) objects. OFDMA is being used mostly in mid- and high-split cable plant because of spectrum availability. OFDM and OFDMA multicarrier data should be more valuable than single frequency carrier data because of a much wider potential bandwidth relative to narrowband DOCSIS 3.0 single carrier coefficients. This wider bandwidth provides high resolution time responses, enabling cable problems to be identified with high accuracy.

Two application categories have been used to date: examination of a single response for impairments and comparing multiple responses to form groups with common impairments.

OFDM and OFDMA data obtained to date needs to be pre-processed before analysis due to the addition of delay and phase from the CM-CMTS interaction during ranging. This paper discusses the processing steps, provides field and lab data, and discusses new applications which are now feasible.

Just as full band capture (FBC) came to be described as a "free spectrum analyzer in every CM," OFDM and OFMDA coefficients can be viewed as a "free pair of network analyzers in every CM." The OFDM & OFDMA equalization data are compelling because there are many millions of deployed DOCSIS 3.1 CMs, and the data have both high time resolution and large dynamic range.

2. Background

As mentioned above, predistortion analysis has been used operationally on cable upstream single-carrier transmissions for a while, and readers are directed to the bibliography for publications on this common operational practice, done under the CableLabs® umbrella of proactive network maintenance (PNM). DOCSIS version 1.1 upstream used an 8-tap finite impulse response (FIR) filter which was upgraded to a 24-tap FIR filter for DOCSIS upstream versions 2.0 and 3.0.

There is also a possibility of reading coefficients for downstream single carrier signals which are modulated with 64 or 256 QAM. Decision feedback equalization (DFE) is used for downstream equalization. Unfortunately, this practice has not been highly successful due to the large number of configurations of feed forward and feedback taps, plus inconsistent coefficient reporting. This was caused in part by an incomplete and confusing MIB description.

A common limitation for both upstream and downstream single carrier signals has been the relatively narrow bandwidth, which is 6.0 MHz in the USA for downstream and frequently 6.4 MHz in the upstream. The narrow analysis bandwidth results in a limited time accuracy resolution. The limited time accuracy implies limited distance accuracy, resulting in an imprecise location of plant damage. This limitation makes locating a point of damage on cable plant harder to identify, which can be particularly annoying when locating damage in buried cable.

In DOCSIS 3.1 specifications, both OFDM and OFDMA were standardized. Pre-distortion for upstream OFDMA carriers was continued, although made optional. On both modulation types, MIBs were specified to reveal the frequency domain (FD) complex coefficients associated with each of the potentially thousands of active subcarriers. A MIB object request is sent to the CM, and the CM performs a trivial file transfer protocol (TFTP) transfer of a file containing the requested in-phase (I) and quadrature (Q) coefficients. Note that the upstream coefficients returned are for the corrections to remove impairments, not for the impairments themselves. The two responses are inverses of each other. In the downstream, the CM can report channel estimation coefficients for OFDM, which are also complex coefficients.

With both upstream and downstream single carrier (SC) signals, the returned complex coefficients are in the time domain (TD). Alternately, the returned coefficients in the multi-carrier (MC) systems, both OFDM and OFDMA, are FD coefficients. The discrete fast Fourier transform (FFT) and inverse fast Fourier transform (IFFT) may be used to convert between time and frequency domains.

3. Past and New Applications for Equalization Data

In the past, two main application categories were developed from analysis of the SC coefficients. The first was a stand-alone path analysis. At a CM this analysis revealed echo tunnels created by two or more impedance mismatches along with other problems such as excessive (GD) group delay, suck-outs, peaking, and tilt. The second application was a matching path analysis, indicating that a group of CMs share a common impairment. This information, along with a strand map of the plant, reveals likely locations of plant damage. Figure 1 illustrates upstream single carrier groupings done within a node, revealing homes with common impairments, which are color coded on the map and spectral plots.

Figure 1 - Upstream SC groupings showing common plant impairments.

Applications for equalization data previously mentioned are single response analysis for impairments and matching common impaired responses to assist in the location of line damage. This second application can be expanded to a third application where matching nonimpaired (normal) responses are compared to determine if subscribers are neighbors sharing a common cable line. This can be done to validate records, or as a source of information for locating CMs in the network when location data are not available.

Another application for equalization data is for plant stability analysis. Intermittent connections will cause re-adaption of coefficients to the new echoes. Connections may change infrequently, such as daily in the case of thermal-caused intermittent problems, or several times per second if intermittent connections are experiencing wind or traffic vibrations. Comparing responses with earlier stored coefficients reveals the presence of intermittent connections. If the changes are large, or frequent, the problem may be severe.

Equalization can also be affected by interference, such as band-limited ingress LTE signals or wider bandwidth interference such as common path distortion (CPD) interference or passive inter-modulation (PIM). An examination of the "smoothness" of an equalization solution can reveal noise causing a poor equalization setting. These data can be complementary to MER-per-subcarrier data available on both OFDM and OFDMA. Figure 2 shows a lab equalization response with a poor equalization solution in an LTE band.

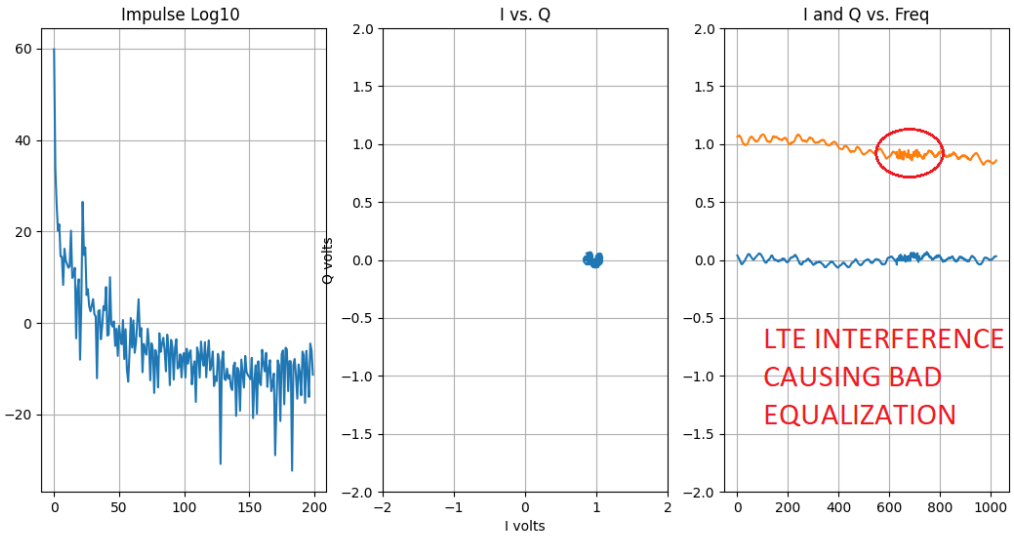

Figure 2 - LTE interference causing poor equalization shows up in the right pre-equalization plot.

Another application is determining if a selected cyclic prefix (CP) is sufficiently long to accommodate the longest significant echoes that are being corrected. Figure 3 shows an example of the strength of echoes vs. delay time. The longer a signal travels in a cable, the greater the attenuation. Knowing the cable type allows this attenuation vs. time curve to be corrected down to near the noise floor. Another application is determining if the programmed cyclic prefix length is long enough for the longest expected echo, which is shown in green in Figure 3. Any echo longer than the CP must be sufficiently weak to cause no harm.

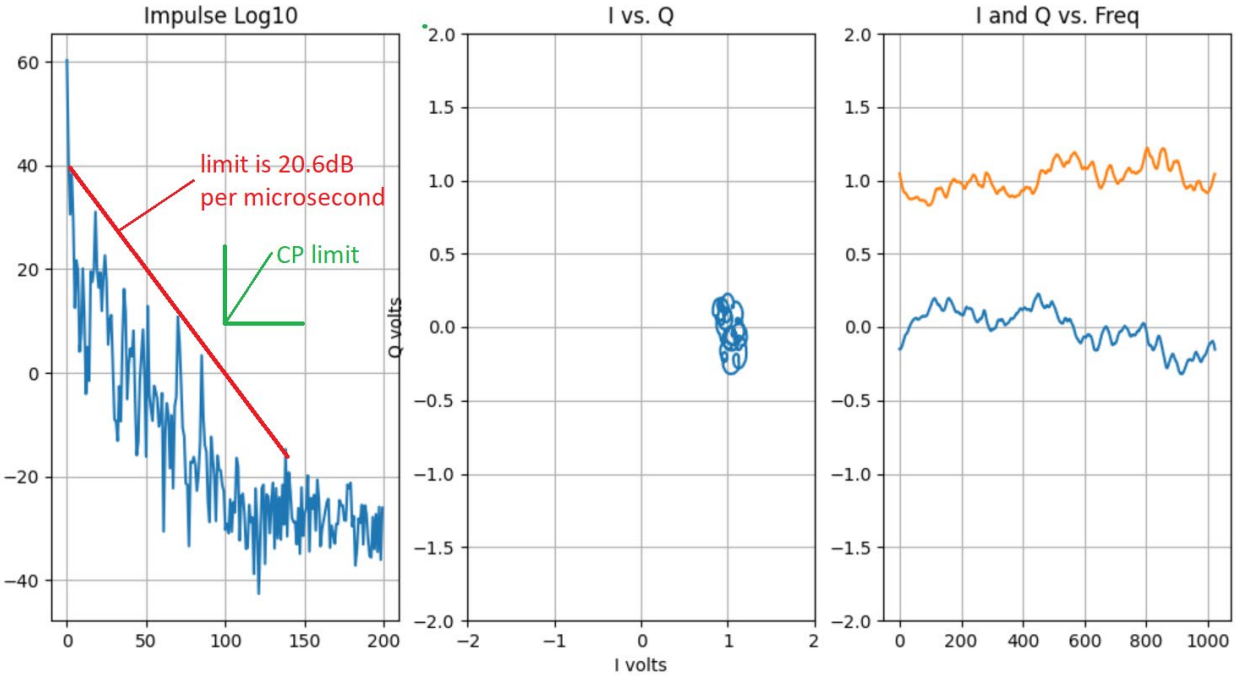

Figure 3 - See the impulse response on the left with the added red limit of 20.6 dB per microsecond, showing no unexpected echo tunnels as peaks above the line. Also no echoes appear over the green corner threshold, indicating sufficient CP is being allocated.

4. Pre-Processing OFDM and OFDMA Coefficients

Presented next are the processing steps for correcting raw (direct from CM) phase so we can make use of it for the described use cases. The basic idea to correct a frequency response is to remove added uniform time delay, which manifests itself in the frequency domain as phase rotating linearly with frequency. We assume that there is a direct current (DC) component to the frequency response, and it can be used as a single "pilot" subcarrier. In this method, there is a coarse delay removal step done in the frequency domain followed by a phase rotation done in the time domain. The phase rotation is done to make the DC term real-only, so the imaginary component of the DC term is forced to zero. In other words, the DC "pilot" is calibrated. With this assumption, the remaining signal reflects the plant condition, which allows us to process these data for our described use cases.

At a high level, the steps are as follows.

1. Import the complex FD data.
2. Estimate linear delay ($d\varphi/d\omega$) – calculate ω using arc tangent to get phase angle.
3. Coarse delay removal – Flatten the angle vs. frequency plot for the middle portion of the band. Using only the middle portion of the band prevents group delay (GD) from effecting the calculation.
4. Convert corrected response to time domain using IFFT, and measure phase on the DC term.
5. Correct phase on all time samples so Im[0]=0 by a time-domain rotation
6. View response in FD by a FFT calculation.
7. Optionally normalize coefficients. For example, make the total power in all coefficients equal to unity, or make the DC term 1.0.

8. (Optional) compare with another response using FD division

The next few figures show the transformation of the processing steps. Appendix 1 contains C++ computer code to remove the undesired (random, linear) time delay and Python code to plot the results as a set of 3 graphs per CM.

As mentioned above, I and Q coefficients for both upstream OFDMA and downstream OFDM arrive raw from a CM with a random time delay inserted that must be removed before processing the coefficients for estimating the channel-plant. In the TD, a delay manifests itself as the main tap being moved from its desired position, which is ideally the 0th or DC term. In the FD, the time delay appears as a constant rotation of phase angle with frequency. Figure 4 illustrates a raw downstream FD polar response with rotation, and Figure 5 illustrates that same response as I and Q plots vs. frequency.

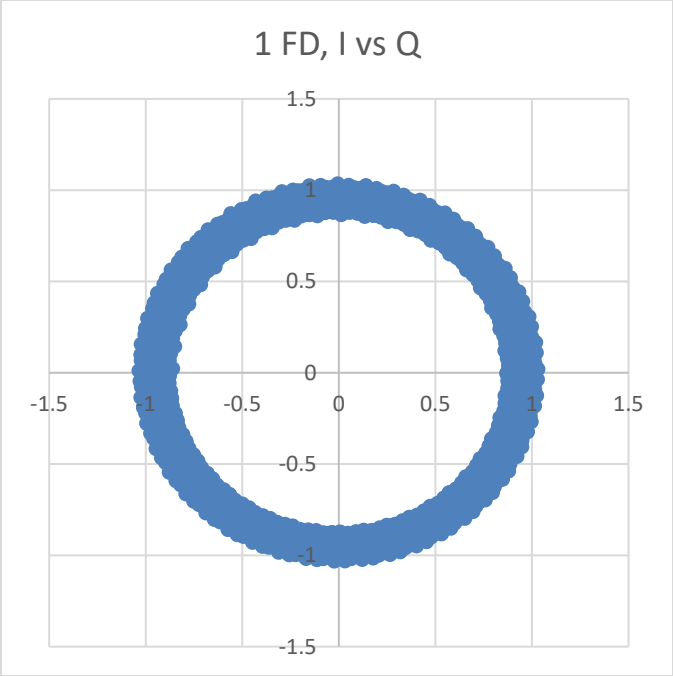

Figure 4 - Raw downstream equalization coefficients, no echo or other impairment.

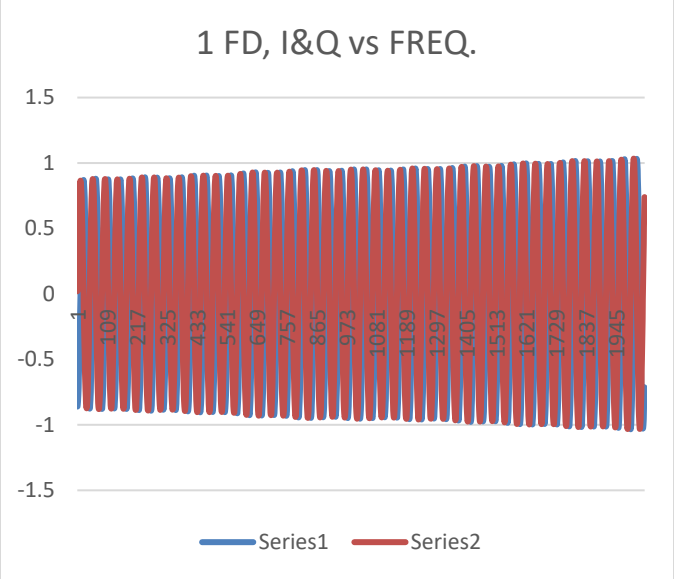

Figure 5 - Raw downstream equalization coefficients versus frequency, no echo or other impairment.

To heuristically explain and demonstrate the removal of delay in the frequency domain, an Excel spreadsheet program was made with two slide controls. It is available from CableLabs. The first slide bars adds or subtracts linear delay, and the second slide bar adjusts the phase angle. The pre-equalization data in the example was obtained from upstream lab testing.
The slide bars operate in the frequency domain. One slide bar changes the rate of change of phase vs. frequency:

$$\Theta = \Theta_0 + k_1\omega$$

where Θ is the new phase angle in radians for every frequency, Θ_0 is the starting phase, and ω in the angular rate of change in radians per second. The value of k_1 is determined by a first slide bar position. Magnitude values for each frequency are not altered; only the phase angle is adjusted.

The second slide bar rotates the response phase equally for all frequencies:

$$\Theta = \Theta_0 + k_2$$

The value of k_2 is determined by a second slide bar position.

See Figure 6 through Figure 12. Fig. 6 is a raw I-Q upstream response in FD polar form, having approximately 450 degrees of unwanted rotation before delay removal. Figure 7 is a plot of angle versus frequency associated with Figure 6. The angle in radians is obtained by taking the arctangent of the Q coefficient value divided by the I coefficient value. The object of the slide processing is to remove delay by adjusting the phase response to be flat in the middle of the spectrum, and then adjust phase angle of the DC term in the time domain to be real-only with an imaginary value of zero. Figure 8 is the angle versus frequency plot with delay (rotation in FD) removed. Observe that the middle of the band has been flattened, but there is still movement at the low and high frequencies. This is caused by group delay associated with upstream high-pass filtering on the low end, and upstream low-pass filtering on the high end.

Figure 9 is an I-Q polar plot showing the removal of rotation. Observe that the plot still has some residual rotation due to group delay, which is not uniform with frequency.

Figure 10 is a plot of group delay vs. frequency. Group delay is the derivative of the slope of phase vs. frequency.

$$GD = -\frac{d\varphi}{d\omega}$$

If the frequency steps are 50 kHz, a group delay value of 0.035 from the graph would be 111ns = 0.035/(2*π*50,000).

Figure 11 is corrected real and imaginary responses vs. frequency. Finally, Figure 12 shows the TD impulse response which is obtained by taking the IFFT of the coefficients in Figure 10. Phase angle has been adjusted to produce a maximum DC real value and zero DC imaginary value.

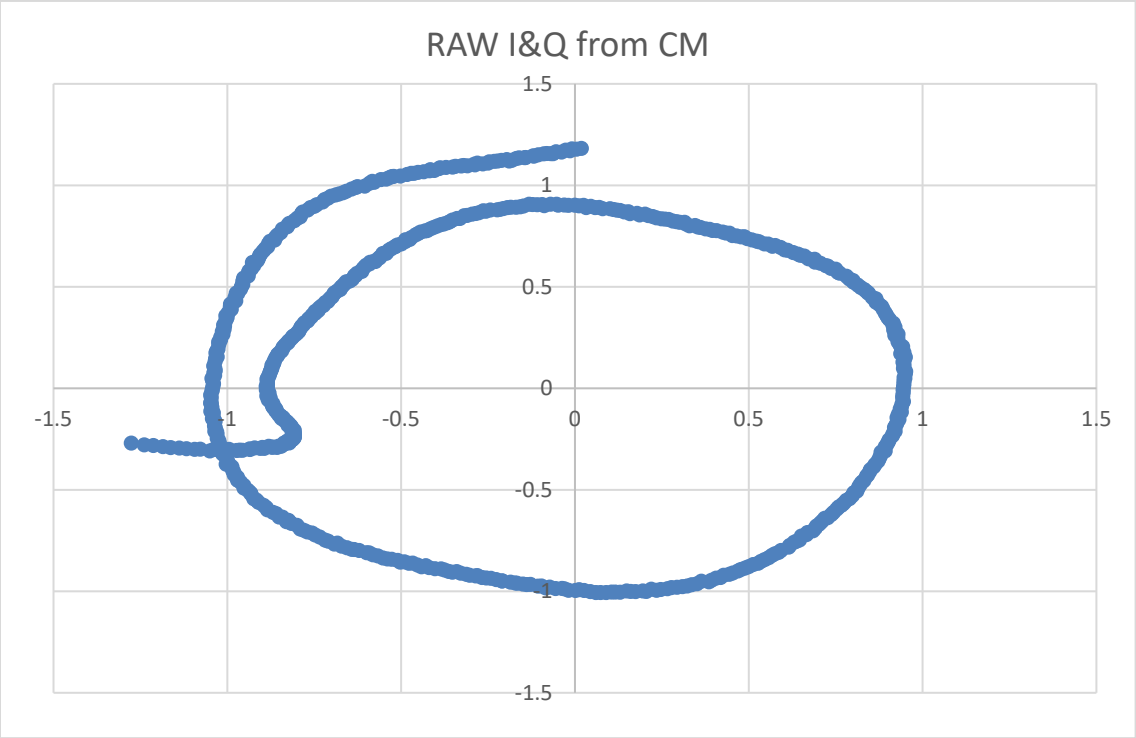

Figure 6 - In-phase (I) versus quadrature (Q) coefficients as supplied by a CM, contaminated with a time delay, which manifests itself as a uniform rate of phase rotation with frequency.

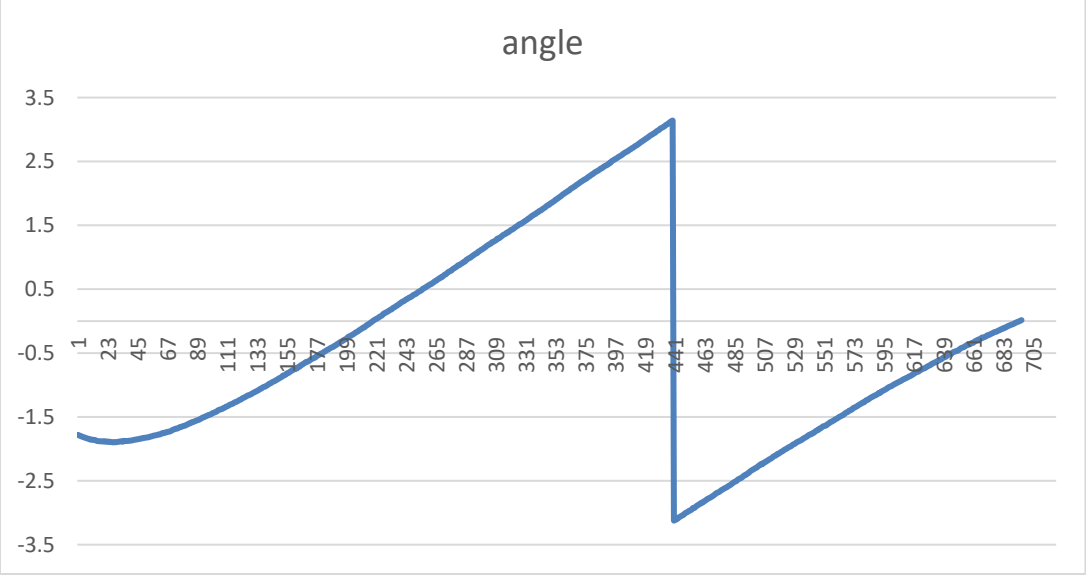

Figure 7 - Angle plot of the data from Figure 6. The rate of change of angle (in radians) versus frequency is delay.

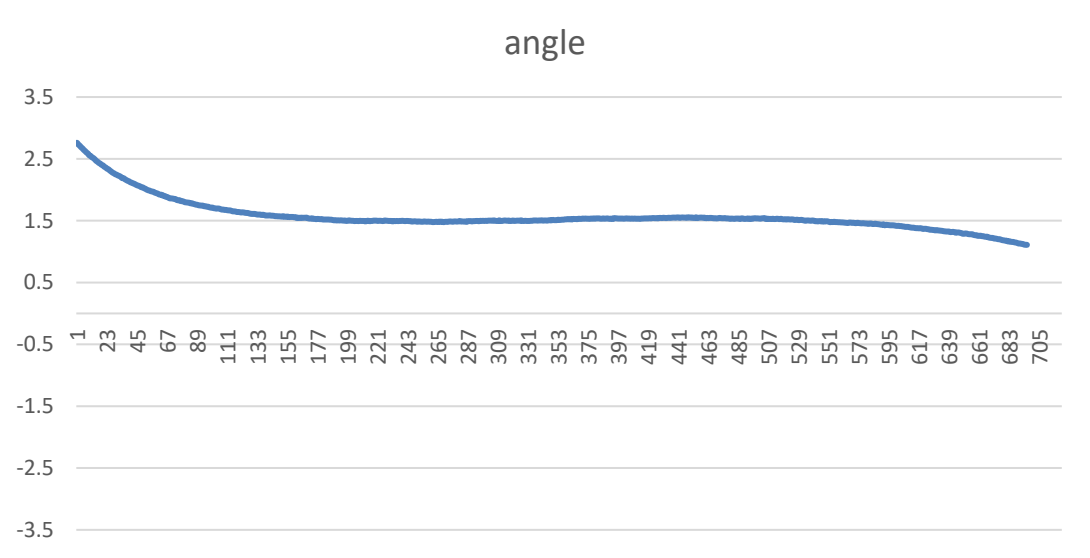

Figure 8 - Phase angle of I and Q coefficients after delay removal. Phase is adjusted to be flat in spectral band without excessive group delay.

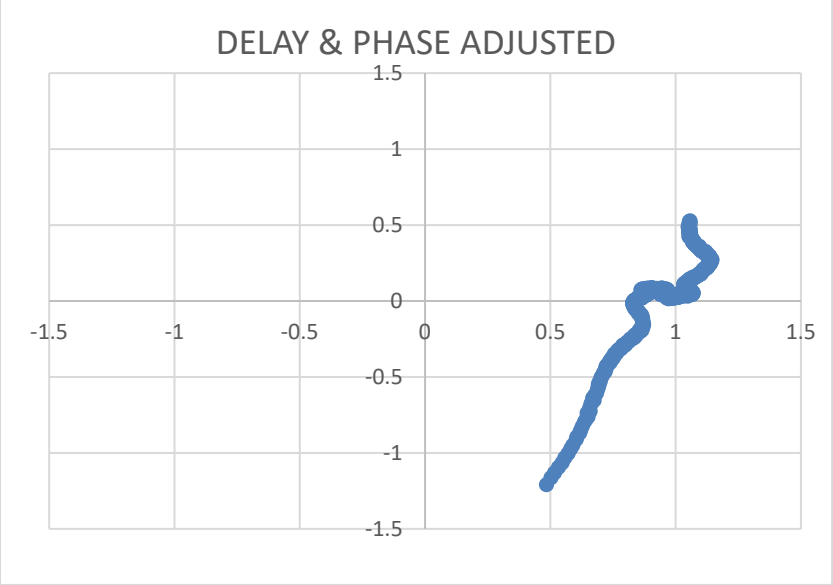

Figure 9 - I and Q coefficients as a polar plot, after delay removal.

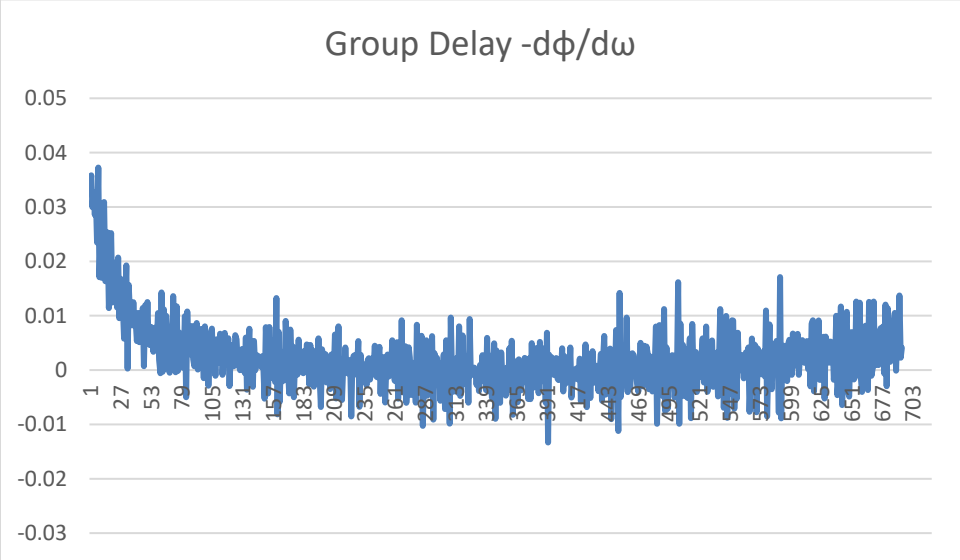

Figure 10 - Group delay, created by taking the derivative of the phase response with respect to frequency. Group delay is a "bending" of the phase response, and is caused by a high-pass filter at 5 MHz.

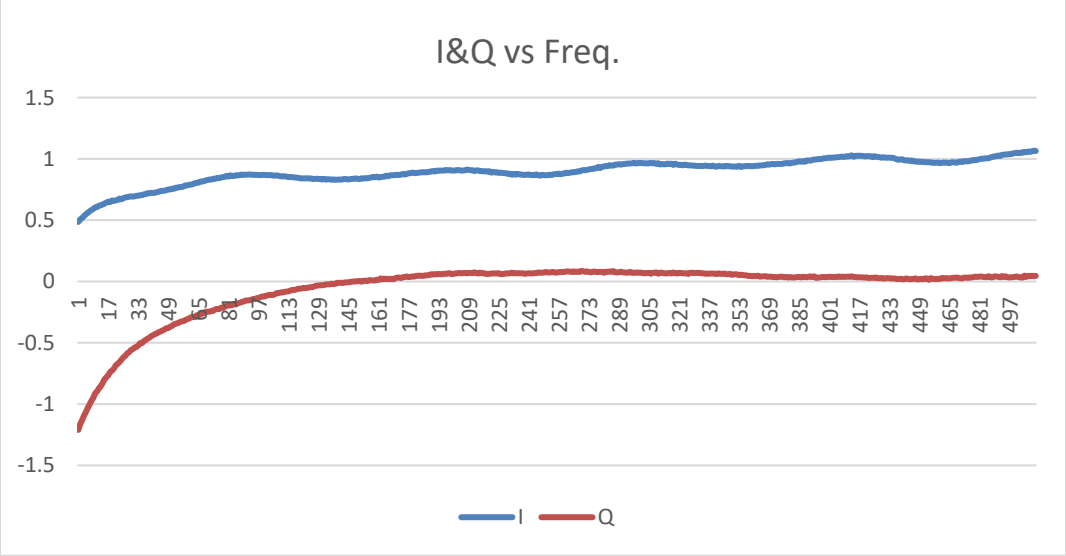

Figure 11 - De-rotated I and Q values.

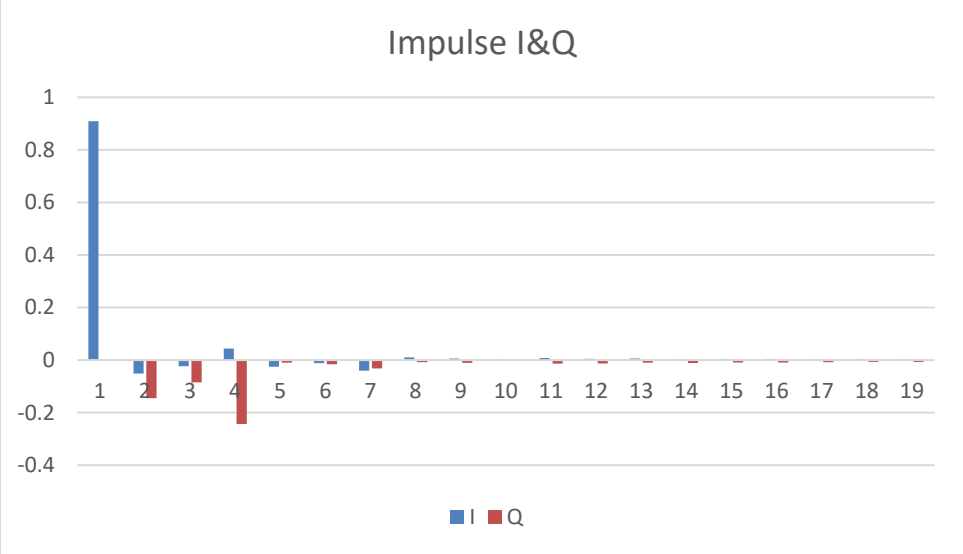

Figure 12 - Impulse (time domain) response that was obtained by taking the IFFT of the re-rotated response. It shows a maximum DC real value (I) and zero DC imaginary (Q) value.

Because of group delay at the low end of the frequency band, only the middle portion of the band was flattened. As mentioned above, computer code has been written to automate delay removal and angle adjustment with a target of maximum energy in the real DC coefficient and zero energy in the DC imaginary coefficient. The code for executing that process is in Appendix 1 of this paper.

A question arises: why is there a time delay (same as FD rotation) addition added to the frequency domain coefficients? A best guess is that delay is an adjustment method that allows an optimal selection of time domain samples on which to perform an FFT. Symbols selected for FFT processing come from an OFDM frame preceded by a CP (or cyclic extension, or guard interval). Lab vector network analyzers (VNA) also add delay to responses as a result delay from connecting cables.

5. Downstream Lab Results

In the lab, tests were performed to compare transmission (S21) on a Keysight VNA with a CM's downstream equalization coefficients after passing through a created echo. 1800 complex coefficients were analyzed with a subcarrier spacing of 50 kHz, starting at 612 MHz.

A plot of raw VNA downstream I and Q values with an echo and accompanying delay is illustrated in Figure 13. The same delay-corrected data are shown in Figure 14, which illustrates a downstream single-recursion echo created with a pair of splitters and a long 100' cable with a 12dB attenuator, and a short 1' cable with no attenuator (see Appendix 2 on types of echoes). These data are shown as I and Q coefficients plotted versus frequency. On the left of Figure 14 is a response from the VNA and on the right is a response from a CM. Figure 15 shows a time plot showing the magnitude impulse response after both responses were transformed with IFFTs. Note that the delayed echo was about 16.5 dB due to an additional 4.5 dB of cable loss.

An interesting VNA test result is that the coefficients were produced by the instrument with an accompanying electrical delay, which needed to be removed for analysis and comparison. The delay in S21 in Figure 13 was caused by cable length.

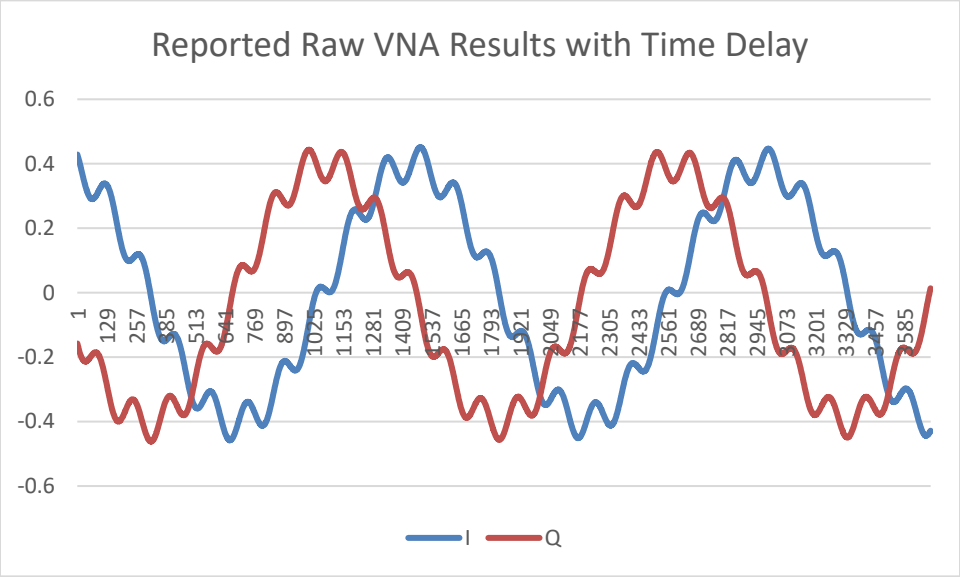

Figure 13 - Raw VNA data showing echo (fine ripple) and time delay (large coarse ripple).

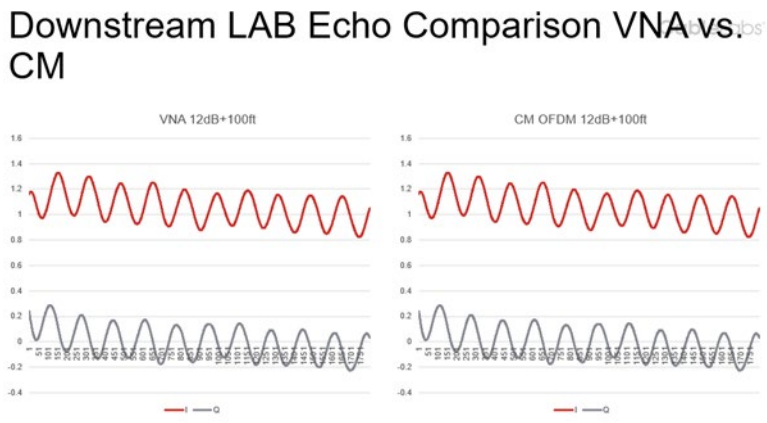

Figure 14 - A downstream single-recursion echo created with a pair of splitters and a long 100' cable with a 12dB attenuator, and a short 1' cable with no attenuator, as I and Q coefficients plotted versus frequency.

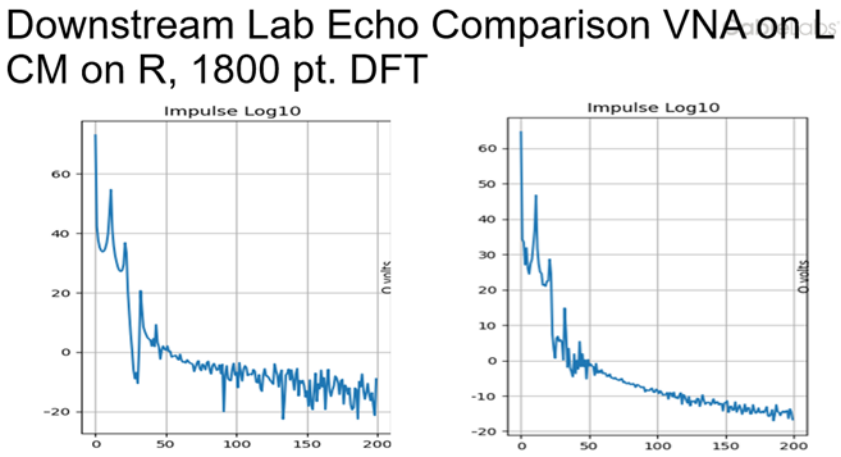

Figure 15 - Time plots showing the magnitude impulse responses of the VNA (left) and CM (right) after both responses were transformed with IFFTs.

Another experiment that was done in the lab was to measure reflection (S11) using a CM and then compare test results with those obtained by the VNA. See Figure 16 for test plant wiring. It illustrates a fiber node feeding signals to a string of taps. The output cable of a fiber node was tapped with a high impedance -20dB probe, and a cable modem (CM METER) connected to the probe was allowed to range and register on a downstream OFDM channel. The standing wave of the OFDM received signal was captured in the complex frequency domain as coefficients, and then inverse Fourier transformed. The red lines show one path the reflections can take to make a standing wave. The green graph in Figure 16 shows an expected impulse response with tap reflections. There should have been an impulse response returned from each tap, and possibly an indication of plant damage, if there was any. Figure 17 shows the OFDM's transformed coefficients (impulse response) from the lab string of taps, and Figure 18 shows the impulse response from a Keysight vector network analyzer. Other than the VNA having more dynamic range than a CM, the results are comparable.

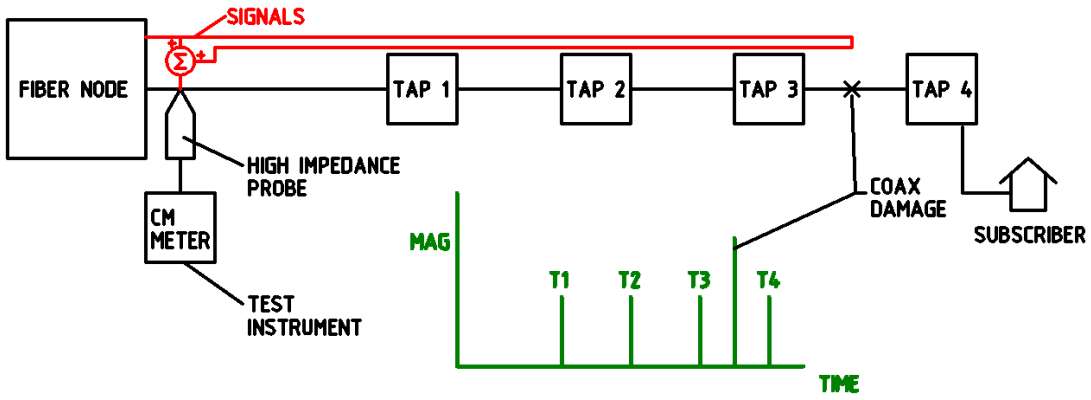

Figure 16 - Test wiring for the second experiment.

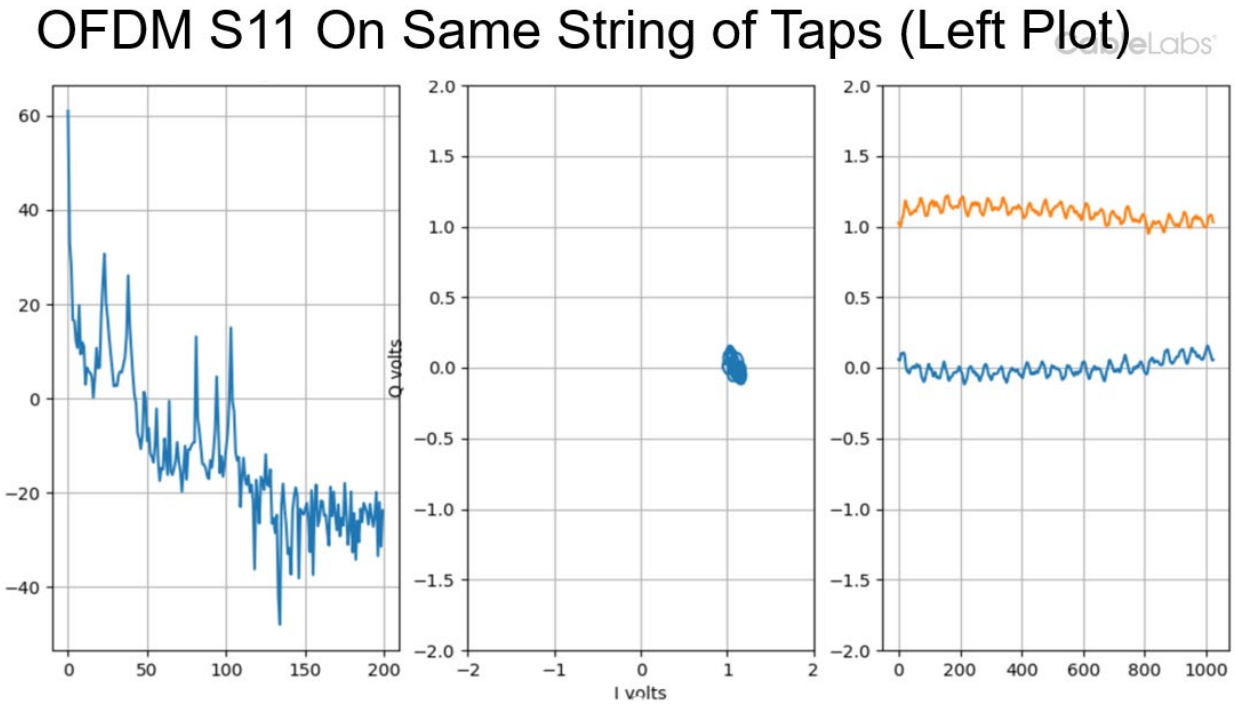

Figure 17 - Impulse response from the lab string of taps in Figure 16, obtained using a CM.

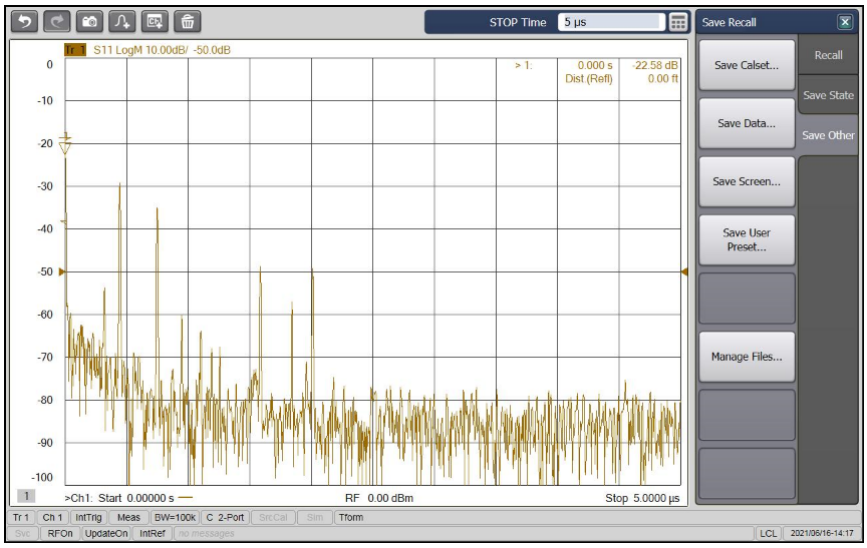

Figure 18 - Impulse response from the lab string of taps in Figure 16, obtained using a Keysight VNA.

Generally, the downstream responses contain much more uniform delay relative to upstream responses.

Lab testing has revealed that if a CM is rebooted, the delay value in the coefficients will probably change to a new value. Likewise, a different CM will likely obtain new delay values when it comes online.

Another observation that has been made is that if there is an exclusion band programmed into the OFDM(A) spectrum, the delay values "jump" over the vacant band. That is, different delay values are applied on occupied bands on either side of an exclusion band. Authors are not sure why. But this does complicate the correction work.

A lab confirmation was that, while OFDMA coefficients are commonly reported as the applied correction (demonstrated in the next section), CMs report OFDM coefficients as the channel's actual response, as expected from the specifications. The channel's correction and its response are frequency domain reciprocals of each other.

The horizontal axes of spectral data are index units, which for these tests were 50 kHz frequency steps.

On the horizontal axes of impulse responses, the units are also indices, which can be converted to time. For example, assume the downstream FFT size is 4096 and symbol rate is 204.8 Mega symbols per second. That results in a symbol period of 4.88 ns per symbol. OFDM frame time is 20 µsec. But because 4096 coefficients are not available, the FFT transform size used is reduced to 1024 (time and frequency symbols). This increases the time per symbol by a factor of 4. Therefore, there are 19.53 ns per index number.

6. Upstream Lab Results

The downstream comparison tests were repeated on the upstream OFDMA channel, which ran 10-85 MHz and used 50kHz channel spacing. This gave a 1500 point discrete Fourier transform. The network analyzer results on the echo are shown in Figure 19, and the CM results are shown in Figure 20. Note

that the VNA is showing an echo, but the CM coefficients are showing the channel's correction, which is the inverse, and infinitely recursive as described in Appendix 2. In the frequency domain, if the ripple goes up for the VNA at some frequency, it is reported as going down from the CM to make a correction. In the frequency domain the inverse solution is infinitely recursive, but in the time domain the measurement is a single recursion.

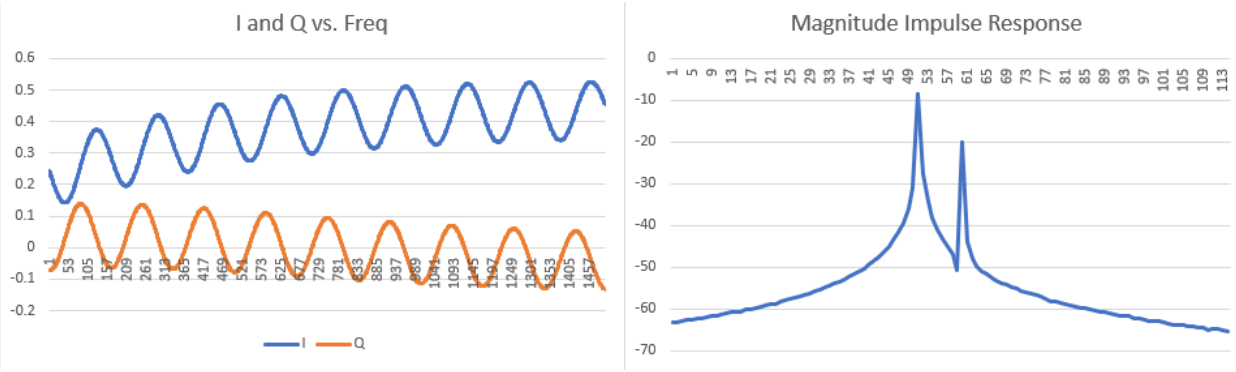

Figure 19 - Network analyzer upstream results in time and frequency.

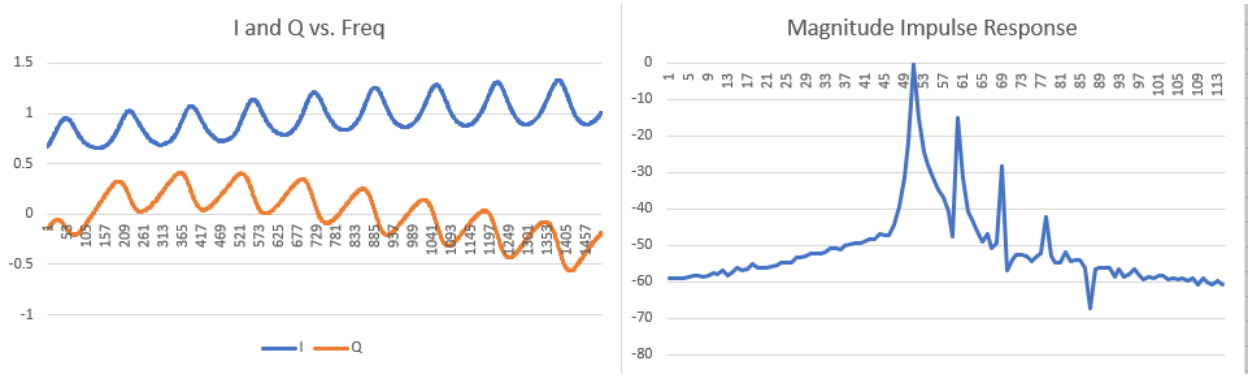

Figure 20 - Cable Modem upstream results in time and frequency. Notice the results are the inverse of the network analyzer results because the CM is returning the correction, not the channel response.

7. Field Data

A browsing tool was created in Python to display equalization coefficients. It is included in Appendix 1. The data are preprocessed for removal of the added delay. Two example plots are illustrated in Figure 21 and Figure 22. The center plot is FD linear I vs. Q. The right plot is FD linear I and Q vs frequency. The left plot is TD magnitude log impulse response. The time scale is IFFT index numbers and is 19.53125 ns per point. On the impulse response, echoes appear to the right of the main tap (delayed) and group delay effects will appear to the left of the main tap.

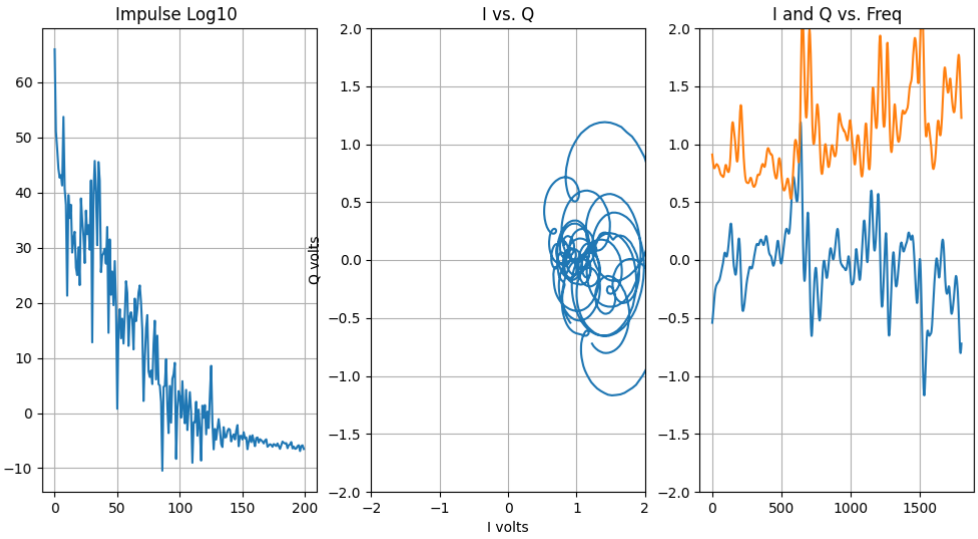

Figure 21 – Downstream example shows an unknown big problem, maybe water in coax.

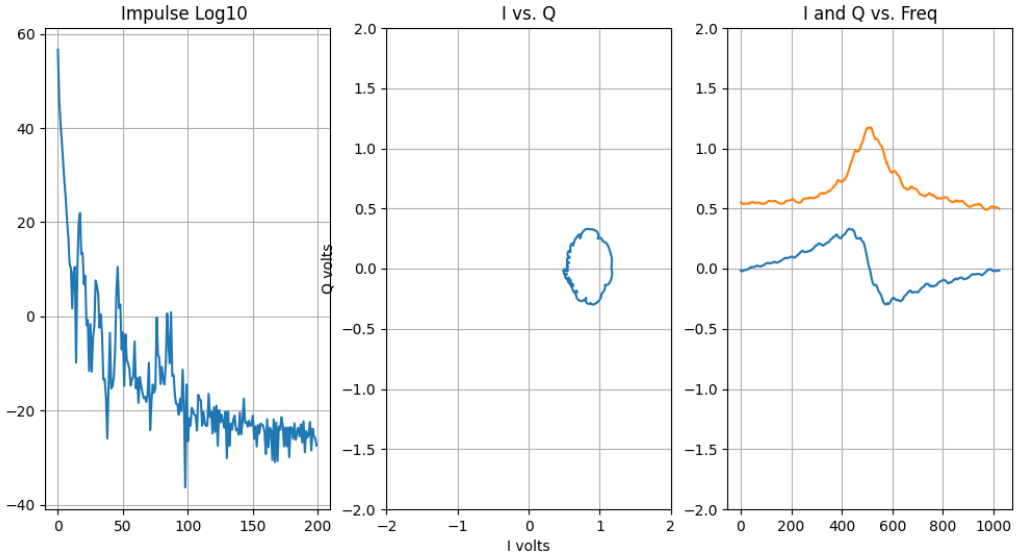

Figure 22 – Second downstream response showing resonant peaking response.

The most common impairment observed was a standing wave created by an echo tunnel. Taps, amplifiers, and other network components generally have a return loss around 16-18dB. This implies, if the coaxial cable was lossless, the strongest observed echo would be 32dB from two reflections between any two components. Of course, cable is not lossless, but has predictable loss as a function of diameter and frequency. For every nanosecond a reflected signal travels in the coax predicted attenuation should occur given by:

dB/ns = A dB/meter * VoP * 0.3meters/ns

Example: 0.5 cable at 750MHz has 7.09dB per 100m, or 0.079dB/1m

The speed of light is 3e8m/sec, and the velocity of propagation (VoP) is 87%.

So, every nanosecond an echo is attenuated by 0.0206dB, or 20.6dB per microsecond.

This allows a mask to be applied to an impulse response, as illustrated in Fig. xx. Any impulse crossing a threshold signifies an out of spec. component. This simple test allows cable damage producing less than a 16dB discrete reflection to pass undetected unless echo timing is considered.

Figure 23 is a similar plot to Figure 3. It shows a CM downstream response with an added threshold red line of some number of dB of attenuation per microsecond slope. Because the cable diameter (loss) was not known, the line's correct slope is a guess. This line can be used as a limit for finding any echo that peaks above the line, indicating of a potential plant problem. A problem is shown where the echo limit is crossed. The far-right FD plot indicates echoes (ripple) that align with the peaks in the left plot. The green CP limit line shows that the selected CP length is sufficient.

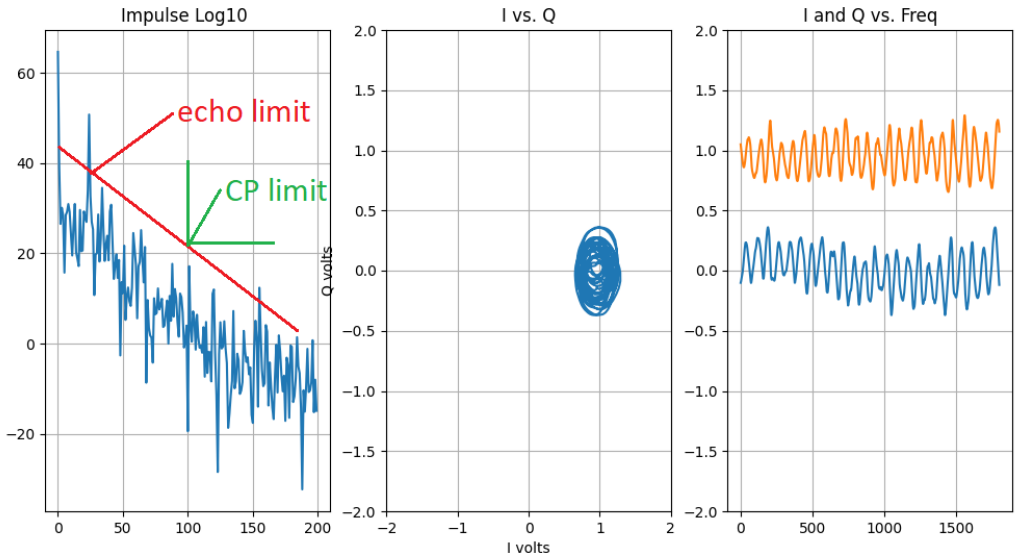

Figure 23 – Adding a line that reflects the CP, so that any peak above the line indicates a signal problem.

Appendix 4 holds a gallery of interesting downstream and upstream data plots from field data with comments in the figure captions.

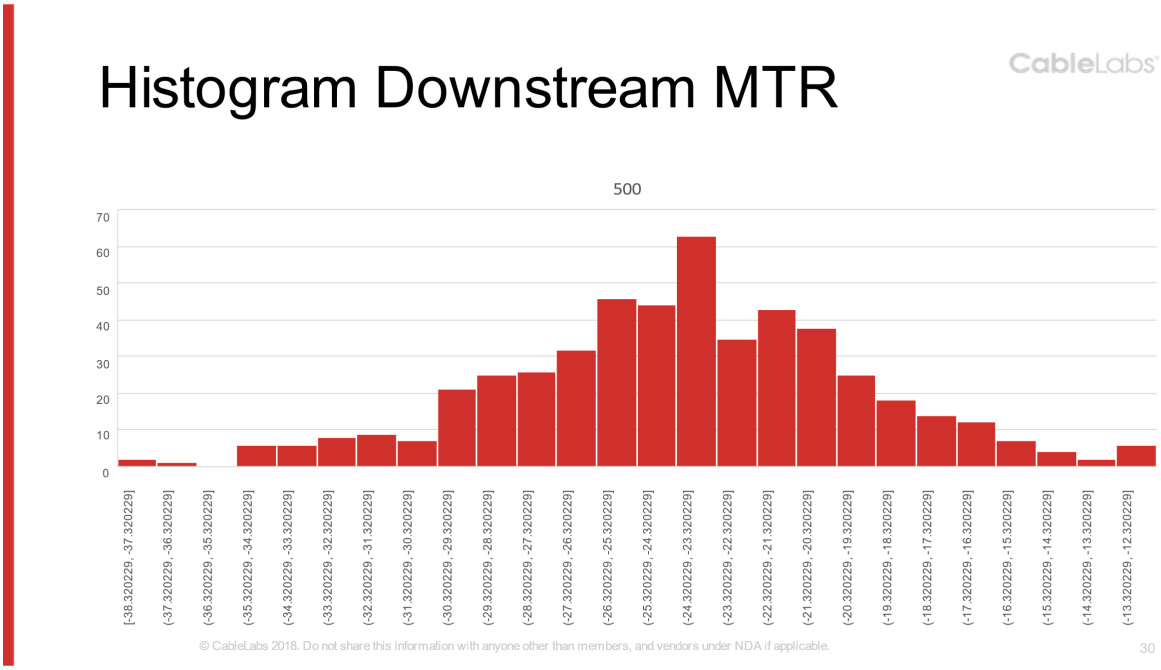

Figure 24 - A histogram of DS main tap ratio values from about 500 CMs from the field.

Fig. 24 is a histogram of downstream main tap ratio (MTR) values for a field population of about 500 CMs. The values range between 12dB and 37dB with a peak around 24db. MTR is the measure of the energy in the main tap to the energy in all other taps combined. If residual delay is not removed, the MTR values will be lowered.

8. Response Matching Algorithms

For many years PNM engineers have been discovering which field responses are similar by using frequency domain division, followed by an IFFT. Similar responses may indicate that modems share a common plant impairment, as illustrated in Figure 1. This makes for easier and quicker plant troubleshooting when a strand map is available. More recently, unimpaired normal responses are being matched, allowing discovery of which common cable lines are used, indicating CMs are in the same neighborhood. More details on this in Appendix 3.

In this matching process, a set of coefficients from one cable modem is divided into another CM's coefficients at each subcarrier frequency using complex frequency domain division. The result is a set of quotients which will be all 1.0 at 0 degrees, if responses are identical. This set of quotients is processed with an IFFT to make a time response. A temporal power measurement can be made of the DC term (coefficient zero) relative to the power in all other coefficients combined. A large ratio of main tap to all other taps combined indicates a good complex response match. This is illustrated in Figure 25, where two I-Q polar responses look very similar. Their quotient is illustrated in Figure 26 as both I and Q vs. frequency and I vs. Q. When this spectral response is transformed with an IFFT, almost all of the transform's power will be in the temporal DC term.

To form clusters, processing is generally only done on CM data from inside a fiber node. Each CM's response is used as a reference and compared with each other CM response, and all matches are noted and formed into groups.

In the past this has been done on 24 coefficient single carrier upstream responses. Now it can be done on both upstream OFDMA and downstream OFDM responses with hundreds of coefficients. Trials on downstream field data show it works well. Furthermore, the effects of different house wiring can be masked by eliminating (zeroing) low value time domain coefficients when performing the power calculation.

Using complex coefficient matching generally produces noticeably better results than using magnitude-only coefficient response matching.

Figure 25 - Two CM's responses, one used in a numerator and the other (reference) used in the denominator.

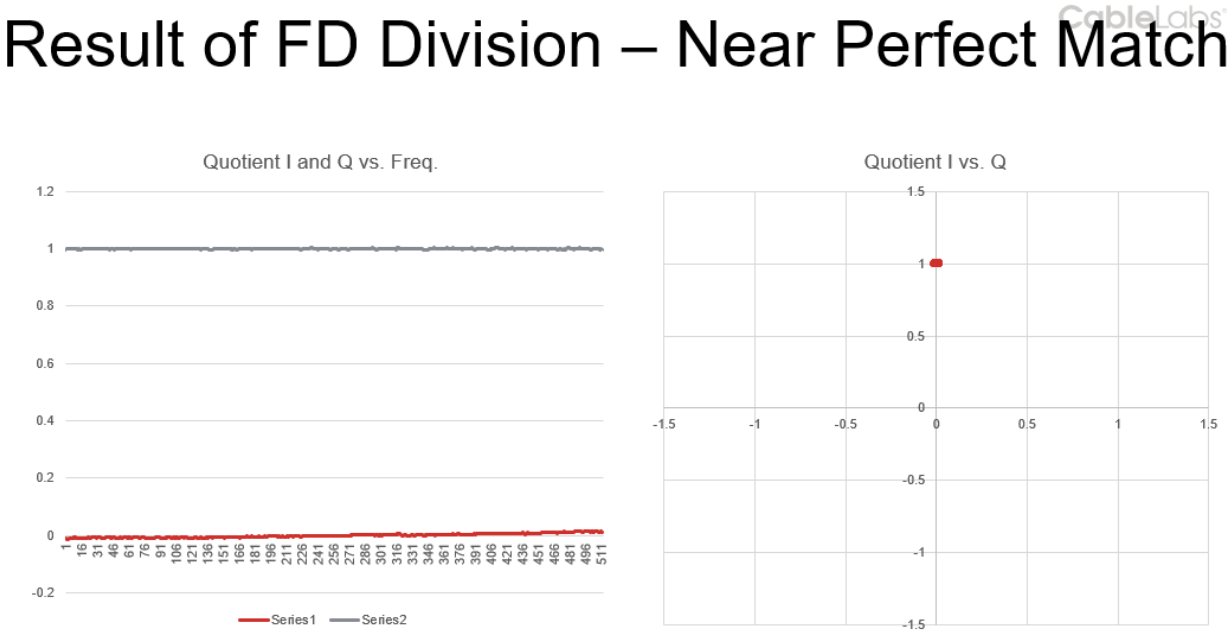

Figure 26 - Two nearly identical responses produce quotient coefficients of 1.0 at an angle of 0 degrees. Nearly all the energy is in the DC term.

9. New Applications for Wideband Equalization Data

Existing applications of impaired response matching and individual response matching work better with multicarrier than single carrier because the equalization data has more dynamic range, in part due to using elevated pilots. The wider bandwidth of the signals also provided for more accurate time resolution.

New possible applications are:

1. Wideband downstream and upstream response matching.
2. Investigation of correct sizing of OFDM(A) CP length.
3. Intermittent detection from unstable equalization responses.
4. Interference or ingress causing band-limited wrong equalization values.
5. In-house wiring problems, such as broken coaxial shields, identified by energy in low index coefficients.
6. Water detection in coaxial cable.
7. Locating subscribers connected to the same coaxial line by matching responses or identifying amplifiers in a cascade.
8. Field test gear, including time domain reflectometer (TDR) functionality.

Experience from using PNM objects has taught us that operational value is gained by learning what physical problems are manifested in observed channel responses. Unfortunately, we have not been provided feedback from our operators on what problems have created the unusual responses, partially due to time and distance, and partially due to the global pandemic.

An operational disservice occurs when time and resources are wasted on chasing non-problems. An impairment that is time-variable in nature is more suspicious than one that is static. Likewise, an impairment that increases with time, such as water slowly entering a cable, should be flagged for inspection and possible repair.

DOCSIS equalization is powerful, and many impairments can be tolerated. For example, if a tap's return loss drops from a specified 18dB to 15dB, the equalizers can normally fix that issue.

10. Conclusions

OFDM and OFDMA equalization coefficients both contain useful complex frequency response data, but the data need to be pre-processed to remove added random, linear delay which is phase rotation. Code to do this processing is in the CableLabs PNM GIT repository C3, and a verbose version is provided in the first appendix of this paper. The first appendix also has Python code for plotting results. This equalization analysis should work for water wave detection and have field test equipment applications. The wide measurement bandwidth obtainable from OFDM and OFDMA provides detail in the time domain, which allows more precise distance measurements. The impulse response can show if your CP (or Guard Interval) is the correct length or if echoes are above expected limits.

While spectrum data, which is magnitude only, has been proven to be very valuable for impairment detection, the complex data from pre-equalization and channel estimation data may be more useful for determining more precisely the location and cause of the issue, better informing maintenance decisions.

Many thanks to the members who contributed data for this analysis.

Abbreviations

CM	Cable modem
CMTS	Cable modem termination system
CP	Cyclic prefix
CPD	Common path distortion
DC	Direct Current
DFE	Decision feedback equalization
FBC	Full band capture
FD	Frequency domain
FFT	Fast Fourier transform
FIR	Finite impulse response
GD	Group delay
I	In-phase
IFFT	Inverse Fast Fourier Transform
MC	Multi-carrier
MIB	Management information base
MTR	Main tap ratio
OFDM	Orthogonal frequency division multiple
OFDMA	Orthogonal frequency division multiple access
PIM	Passive inter-modulation
PNM	Proactive network maintenance
Q	Quadrature
SC	Single carrier
SCTE	Society of Cable Telecommunications Engineers
TD	Time Domain
TFTP	Inverse Fast Fourier Transform
TDR	Time Domain Reflectometer
SCTE	Society of Cable Telecommunications Engineers
VNA	Vector network analyzer

Bibliography & References

Alan V. Oppenheim, Alan S. Willsky, S. Hamid Nawab, *Signals and Systems, second edition,* Prentice Hall, 1996. (See page 53 for a discussion on differences between linear and nonlinear.)

Alberto Campos, Bruce Currivan, Charles Moore, and Tom Williams titled “Upstream Cable Echoes Come in Two Flavors,” CED Magazine, 2010.

Data Over CableService Interface Specification Proactive Network Maintenance “Primer for PNM Best Practices in HFC Networks (DOCSIS® 3.1),” CM-GL-PNM-3.1-V02-210114 (Cable Television Laboratories).

Data-Over-Cable Service Interface Specifications DOCSIS® 3.1 Cable Modem Operations Support System Interface Specification CM-SP-CM-OSSIv3.1-I16-190917 (Cable Television Laboratories).

DOCSIS® Best Practices and Guidelines PNM Best Practices: HFC Networks (DOCSIS 3.0) CM-GL-PNMP-V03-160725 (Cable Television Laboratories).

Data-Over-Cable Service Interface Specifications DOCSIS® 3.1 Physical Layer Specification CM-SP-PHYv3.1-I17-190917 (Cable Television Laboratories).

Data-Over-Cable Service Interface Specifications DOCSIS® 3.1 CCAP™ Operations Support System Interface Specification CM-SP-CCAP-OSSIv3.1-I18-200610 (Cable Television Laboratories).

Appendix

1. Removing the delay in the TD, which is rotation in FD.

The computer code explained in Section 4 of the paper is provided in this Appendix 1. First, we provide C++ code for delay removal from raw responses, followed by Python browsing code which plots responses in 3 graphs.

1.1. C++ code

The following code is meant to be copied and pasted into your software editor for direct use.

```
//PaperCodeR3.cpp a program to process OFDM equalization coefficients

//R3 added choosing a freq band where to flatten angle, s1 to s2

//R2added normalization, group delay

//copyright Cable Television Laboratories Inc. 2021

/*

Legal notice

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated

 documentation (the "Software") the rights to use, copy, modify, merge, publish, distribute, sublicense,

 and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject

to the following conditions:

The above copyright notice and this permission notice shall apply to such person and  be included in all copies

 or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED

 TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT.

 IN NO EVENT SHALL CABLELABS, ITS MEMBERS, OR ITS SUBSIDIARIES BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY,

 WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM OR RLEATED TO THE USE OF THE SOFTWARE  OR

 OTHER EXERCISE OF THE RIGHTS AS LICENSED HEREIN.

*/

#include <iostream>

#include <math.h>

#include <stdio.h>
```



```
#include <stdlib.h>
#include <malloc.h>
#include <time.h>
#include <string.h>
#include <unistd.h>
#ifdef _WIN32
#include <conio.h>
#endif
typedef struct {float real, imag;} COMPLEX;
extern void correct(COMPLEX *,int,int,int,int,int );
extern void idft(COMPLEX *, COMPLEX *, int);
extern void dft(COMPLEX *, COMPLEX *, int);
void FDinvert(COMPLEX *,int);
float normalizePower(COMPLEX *,int);
float PI = 3.141592653, MTR;
int fftSize , cnt, good;//int fftSize = 404, cnt, good;
//e.g. from command line: main.exe 1 1880 1 1880
int main(int argc, char *argv[] )//int main()
{
        int nrIQ,nrLines=1,nrSamples=1880,s1=0,s2=1880;
                fftSize = nrSamples;
        if((int)argc >1){
        nrLines = atoi(argv[1]);//command line argument
                printf("argc = %d\t %s\t%s\n",argc, argv[1],argv[2]);
                nrSamples = atoi(argv[2]);
                nrLines = atoi(argv[1]);//command line argument
                s1=atoi(argv[3]);
                s2=atoi(argv[4]);
                printf("you entered %d arguments\t fftSize=%d\tnrLines=%d\n",argc-1,fftSize,nrLines );
        }
  printf("running batchOFDM\n");
```

```
COMPLEX *w;
w=(COMPLEX*) calloc(4096, sizeof(COMPLEX));
if(!w) { printf("\n Unable to allocate input memory.\n"); printf("\x7"); exit(1); }
        FILE *input, *output,*outputR;
int i,j,offset=0,lineLength=404;
char *pch;
char delim[] = "    ";//delimiter is a tab or a space
char data[80000] = {0};
        float A[4096],B[4096];
if( (input = fopen("input.txt", "r") ) == NULL)
{   //read data
                printf("could not open file.\n");
                system("pause");
   free(w);
                exit(0);
        }
if( (output = fopen("output.txt", "w") ) == NULL)
{   //write results to a file
                printf("could not open file.\n");
                system("pause");
   free(w);
   fclose(input);
                exit(0);
        }
if( (outputR = fopen("outputR.txt", "w") ) == NULL)
{   //write results to a file
                printf("could not open file.\n");
                system("pause");
   free(w);
   fclose(output);
   fclose(input);
```

```
                    exit(0);
            }
for(cnt=0;cnt<nrLines  ;cnt++)
{
   //if less than cnt lines, data will be empty (except for newline),
   // strtok will return NULL, and atof will crash
   fgets(data,80000,input);
   pch = strtok(data,delim);
   if (pch == NULL)
   {
      //printf("No data\n");
      continue;
   }
   A[0]=atof(pch);
   pch = strtok(NULL,delim);
   if (pch == NULL)
   {
      //printf("No data\n");
      continue;
   }
   B[0]=atof(pch);
   i=1;
   while( pch !=NULL)
   {
      pch = strtok(NULL,delim);
      if (pch == NULL)
         break;
      A[i]=atof(pch);
      pch = strtok(NULL,delim);
      if (pch == NULL)
         break;
```

```
        B[i]=atof(pch);

        i++;

    }

    lineLength = i;//how many numbers were read in

    for(i=0;i<fftSize;i++)

    {//all lines set to this length

        if(i >= offset && i<fftSize*2+offset)

        {

            w[i-offset].real=A[i];

            w[i-offset].imag=B[i];

        }

    }

    //data looks like I value followed by Q value followed by next I value followed by next Q value

    //eg: -1.343994141      -0.490112305      1.233276367      -0.723999023      -0.2265625      1.406616211
          -0.934570313      -1.072509766      1.419677734      -0.033813477

    //offset is used to pick a start point not at the start of the read file.  This can reduce effects of GD. But you must make sure

    //you have enough I-Q samples on the line or an error will occur

    fprintf(output,"file = %d\tLine Length=%d\tRead Offest=%d\n",cnt,lineLength, offset);//SC=subcarriers

    printf("\nfile = %d\tLine Length=%d SC\tRead Offest=%d SC\n",cnt,lineLength, offset);

    for(i=0;i<fftSize;i++)

    {//input data

        fprintf(output,"F%d\t%f\t%f\n",i,w[i].real,w[i].imag);

    }

    fprintf(output,"\n");

                s1=100;s2=1800;//set these for widest possible band without GD from band edge filters// probably want to do
some limit checking here

    correct(w,fftSize,s1,s2,cnt,0);//this function is used to process both pre and post equalziation data

    //do a second time to remove residual rotation

    correct(w,fftSize,s1,s2,cnt,1);

                //void correct(COMPLEX *w,int np, int s1, int s2,int fn,int iteration)

    //FDinvert(w,fftSize);  //optional step to look at what a VNA would see, not the inverse

    normalizePower(w,fftSize);

    for(i=0;i<fftSize;i++)
```



```
        {
            fprintf(outputR,"%f,",w[i].real);
            if(w[i].imag >= 0)
            {
                fprintf(outputR,"+%f,",w[i].imag);
            }
            else
            {
                fprintf(outputR,"%f,",w[i].imag);
            }
        }
                fprintf(outputR,"\n");
            }
    fclose(outputR);
    fclose(output);
    fclose(input);
    return(0);
}//end of MAIN////////////////////////////////////////////////////////////////////////////////////////////
float normalizePower(COMPLEX *w,int np){
            int i;
            float a,pow=0;
            for(i=0;i<np;i++){
                        pow += w[i].real*w[i].real +  w[i].imag*w[i].imag;
            }
            pow = pow/(float)fftSize;
printf("power = %f\n",pow/(float)fftSize);
            for(i=0;i<np;i++){
                        w[i].real = w[i].real/sqrt(pow);
                        w[i].imag = w[i].imag/sqrt(pow);
            }
            return pow;
```

```
}


void FDinvert(COMPLEX *w,int np)
{//take FD reciprocal
        int i;
        float a,b,DENOM;
        for(i=0;i<np;i++)
  {
                a=w[i].real;
                b=w[i].imag;
                DENOM = a*a + b*b;
                w[i].real = a/DENOM;
                w[i].imag = -b/DENOM;
        }
}
void correct(COMPLEX *w,int np, int s1, int s2,int fn,int iteration)
{
        FILE *output;
        int i,sp=1;//0 sp supresses fprintf, 1 prints inline
        float th1=0,mag=0,angle[4096],dphi=0,theta,delta=0,pow=0,DCpow=0,oldang,gd;
        double ang;
        char readme[16];
  sprintf(readme,"output%d.txt",0  );//to supress making a file for every response
        COMPLEX *s;
        s=(COMPLEX*) calloc(4096 , sizeof(COMPLEX));
        if(!s)
  {
    printf("\n Unable to allocate input memory.\n");
    printf("\x7");
    // add cleanup here
    exit(1);
```

```
}

        COMPLEX *r;
        r=(COMPLEX*) calloc(4096 , sizeof(COMPLEX));
        if(!r)
{
   printf("\n Unable to allocate input memory.\n");
   printf("\x7");
   // add cleanup here
   exit(1);
}

        COMPLEX *t;
        t=(COMPLEX*) calloc(4096 , sizeof(COMPLEX));
        if(!t)
{
   printf("\n Unable to allocate input memory.\n");
   printf("\x7");
   // add cleanup here
   exit(1);
}

        COMPLEX *u;
        u=(COMPLEX*) calloc(4096, sizeof(COMPLEX));
        if(!u)
{
   printf("\n Unable to allocate input memory.\n");
   printf("\x7");
   // add cleanup here
   exit(1);
}
```

```
        COMPLEX *ws;
        ws=(COMPLEX*) calloc(4096 , sizeof(COMPLEX));
        if(!ws)
{
   printf("\n Unable to allocate input memory.\n");
   printf("\x7");
   // add cleanup here
   exit(1);
}

        COMPLEX *x;
        x=(COMPLEX*) calloc(4096, sizeof(COMPLEX));
        if(!x)
{
   printf("\n Unable to allocate input memory.\n");
   printf("\x7");
   // add cleanup here
   exit(1);
}

        COMPLEX *y;
        y=(COMPLEX*) calloc(4096, sizeof(COMPLEX));
        if(!y)
{
   printf("\n Unable to allocate input memory.\n");
   printf("\x7");
   // add cleanup here
   exit(1);
}
```

```
        if( (output = fopen(readme, "w") ) == NULL)
{
            printf("could not open file.\n");
   system("pause");
   // add cleanup here
   exit(0);
}

        printf("\n");
//middle frequency is np/2
th1 = atan2(w[np/2].imag,w[np/2].real);
for(i=0;i<fftSize;i++)
{
   ws[i].real = w[i].real; //copy and store  the input
   ws[i].imag = w[i].imag;
}
printf("delay correction pass #%d\tFFT Size=%d\tAnalysis between FFT sample %d and sample %d\n",iteration, fftSize,s1,s2);
//printf("freq=%d\txR=%f\txI=%f\tang=%f\n", np ,w[np/2].real,w[np/2].imag,th1,th2);

if (sp==1)
   fprintf(output,"P1_1 input FD data: I, Q, angle[]\tLine # = %d\n", fn );
// /* begin rotation removal function
for(i=0;i<np;i++)
{
   angle[i]=atan2(ws[i].imag,ws[i].real);
   ang=atan2(ws[i].imag,ws[i].real);
   if(sp==1)fprintf(output, "~%d\t%f\t%f\t%f\n",i,ws[i].real,ws[i].imag,angle[i]); //raw data  Dan bug fix 2
}
float phi = 0;
```

```
        for(i=s1 ;i<s2;i++)
 {
    dphi = angle[i]-angle[i-1];
    phi += dphi;
    if(dphi > PI) {phi -= 2*PI;}//  printf("going clockwise");}
    if(dphi < -PI) {phi += 2*PI;}//printf("going counter clockwise");}
}
printf("fftSize = %d\t s1=%d\ts2=%d\n", fftSize, s1,s2);
delta = -phi/(float)(s2-s1);//delay estimate
printf("Radians rotation =%f\tSlope=%f radians per subcarrier\n",phi,delta );//reduce the delta by number of -pi to pi jumps
//end rotation removal function
if(sp==1)
    fprintf(output,"2 rotation removed in FD\n");
for(i=0;i<np;i++)
{
    ang = -(float)i*delta;//(float)np;
    u[i].real = ws[i].real*cos(-ang) - ws[i].imag*sin(-ang);//w is impaired
    u[i].imag = ws[i].real*sin(-ang) + ws[i].imag*cos(-ang);
    ang=atan2(u[i].imag,u[i].real);
    if(sp==1)
        fprintf(output, "2C%d\t%f\t%f\t%f\n",i,u[i].real,u[i].imag,ang);
}
//now split fd data into sidebands for using ifft
for(i=0;i<np/2;i++)
{//upper sideband
    s[i].real = u[i+np/2].real;
    s[i].imag = u[i+np/2].imag;
}
for(i=0;i<np/2;i++)
{//lower sideband
    s[fftSize-np/2+i].real = u[i].real;
```

```
    s[fftSize-np/2+i].imag = u[i].imag;
}
if(sp==1)fprintf(output,"3 sidebands swapped\n");
for(i=0;i<fftSize;i++)
{
    t[i].real = s[i].real;
    t[i].imag = s[i].imag;
}
for(i=0;i<fftSize;i++)
{
mag = sqrt(t[i].real*t[i].real + t[i].imag*t[i].imag) ;
if(sp==1) fprintf(output,"3Sf%d\t%f\t%f\t%f\n",i,t[i].real,t[i].imag,mag);
}
idft(s,r,fftSize);//tom
for(i=0;i<fftSize;i++)
{
    s[i].real=r[i].real;
    s[i].imag=r[i].imag;
}
if(sp==1)
    fprintf(output,"4 TD with angle err.\n");
for(i=0;i<fftSize;i++)
{
    mag=sqrt(s[i].real*s[i].real + s[i].imag*s[i].imag);
    if(sp==1)
        fprintf(output,"4St%d\t%f\t%f\t%f\n",i,s[i].real,s[i].imag,mag);
}
for(i=0;i<32;i++)
{
    mag=sqrt(s[i].real*s[i].real + s[i].imag*s[i].imag);
    if(sp==1)
```

```
        fprintf(output,"4St%d\t%f\t%f\t%f\n",i,s[i].real,s[i].imag,mag);
}
theta = atan2(s[0].imag,s[0].real);//rotate the DC term
printf("DC term rotation = %f radians\n",theta);
ang = theta;
if(sp==1)fprintf(output,"5 TD angle = 0 deg\n");
for(i=0;i<fftSize;i++)
{
   x[i].real = s[i].real*cos(-ang) - s[i].imag*sin(-ang);//w is unimpaired
   x[i].imag = s[i].real*sin(-ang) + s[i].imag*cos(-ang);
   mag=sqrt(x[i].real*x[i].real + x[i].imag*x[i].imag);
   if(sp==1)
      fprintf(output, "5@CT%d\t%f\t%f\t%f\t%f\n",i,x[i].real,x[i].imag,mag,20*log10(mag));
}
for(i=0;i<64;i++)
{
   mag=sqrt(x[i].real*x[i].real + x[i].imag*x[i].imag);
   if(sp==1)
      fprintf(output, "5@CT%d\t%f\t%f\t%f\t%f\n",i,x[i].real,x[i].imag,mag,20*log10(mag));
}
for(i=0;i<fftSize;i++)
{
   y[i].real = x[i].real;
   y[i].imag = x[i].imag;
}
dft(y,r,fftSize);
for(i=0;i<fftSize;i++)
{
   y[i].real=r[i].real;
   y[i].imag=r[i].imag;
}
```

```
//y has the correct answer
for(i=0;i<fftSize/2;i++)
{//swap sidebands back
   w[i].real = y[i+fftSize/2].real;
   w[i].imag = y[i+fftSize/2].imag;
}
for(i=0;i<fftSize/2;i++)
{
   w[i+fftSize/2].real = y[i].real;
   w[i+fftSize/2].imag = y[i].imag;
}
if(sp==1)fprintf(output,"6 FD corrected\n");
for(i=fftSize/2;i<fftSize;i++)
{
   mag=sqrt(y[i].real*y[i].real + y[i].imag*y[i].imag);
                        ang = atan2(y[i].imag,y[i].real);
                        gd = -(ang-oldang)/(6.28*50000); //assuming a 50kHz subcarrier spacing
   if(sp==1)
   fprintf(output, "6CF%d\t%f\t%f\t%f\t%f\t%f\t%e\n",i,y[i].real,y[i].imag,mag,20*log10(mag),ang,gd);
                                oldang = ang;
}

for(i=0;i<fftSize/2  ;i++)
{
  mag=sqrt(y[i].real*y[i].real + y[i].imag*y[i].imag);
                        ang = atan2(y[i].imag,y[i].real);
                        gd = -(ang-oldang)/(6.28*50000); //assuming a 50kHz subcarrier spacing
   if(sp==1)
   fprintf(output, "6CF%d\t%f\t%f\t%f\t%f\t%f\t%e\n",i,y[i].real,y[i].imag,mag,20*log10(mag),ang,gd);
                        oldang = ang;
        }
```

```
    pow=0; //initialize
    for(i=0;i<fftSize;i++)
    {
        pow += x[i].real*x[i].real + x[i].imag*x[i].imag;
    }
    DCpow =  x[0].real*x[0].real + x[0].imag*x[0].imag;//imag component should be zero
    printf("total power = %f\tDCpower= %f\n",pow,DCpow);
    printf("power correction to unity power is %f\t%f dB\n",1/pow, 10*log10(1/pow) );
    float Vcorr = sqrt(1/pow);
    printf("voltage correction is %f\n", Vcorr );
    for(i=0;i<fftSize;i++)
    {
        x[i].real *= Vcorr;
        x[i].imag *= Vcorr;
    }
    DCpow =  x[0].real*x[0].real + x[0].imag*x[0].imag;//imag component should be zero
    MTR = 10*log10(1-DCpow)/1.0;
    printf("DC power  =%f\t Other power =%f\t MTR=%f\n",DCpow,1.0 - DCpow, MTR) ;
    fclose(output);
} //END Correct ////////////////////////////////////////////////////////
/*****************************************************************************
dft - Discrete Fourier Transform
This function performs a straight DFT of N points on an array of
complex numbers whose first member is pointed to by Datain.  The
output is placed in an array pointed to by Dataout.
*****************************************************************************/
void dft(COMPLEX *Datain, COMPLEX *Dataout, int N)
{
    int i,k,n,p;
    static int nstore = 0;      /* store N for future use */
    static COMPLEX *cf;         /* coefficient storage */
```

```
    COMPLEX *cfptr,*Dinptr;
    double arg;
/* Create the coefficients if N has changed */
    if(N != nstore) {
        if(nstore != 0) free((char *) cf);    /* free previous */
        cf = (COMPLEX  *) calloc(N, sizeof(COMPLEX));
        if (!cf) {
            printf("\nUnable to allocate memory for coefficients.\n");
            // add cleanup here
            exit(1);
        }
        arg = 8.0*atan(1.0)/N;
        for (i=0 ; i<N ; i++) {
            cf[i].real = (float)cos(arg*i);
            cf[i].imag = -(float)sin(arg*i);
        }
    }
/* Perform the DFT calculation */
    printf("\n");
    for (k=0 ; k<N ; k++) {
        Dinptr = Datain;
        Dataout->real = Dinptr->real;
        Dataout->imag = Dinptr->imag;
        Dinptr++;
        for (n=1; n<N; n++) {
        p = (int)((long)n*k % N);
            cfptr = cf + p;         /* pointer to cf modulo N */
            Dataout->real += Dinptr->real * cfptr->real
                        - Dinptr->imag * cfptr->imag;

            Dataout->imag += Dinptr->real * cfptr->imag
```

```
                    + Dinptr->imag * cfptr->real;
            Dinptr++;
        }
        if (k % 32 == 31) printf("*");
        Dataout++;          /* next output */
    }
    printf("\n");
}
/*****************************************************************************
idft - Inverse Discrete Fourier Transform
This function performs an inverse DFT of N points on an array of
complex numbers whose first member is pointed to by Datain.  The
output is placed in an array pointed to by Dataout.
It returns nothing.
*****************************************************************************/
void idft(COMPLEX *Datain, COMPLEX *Dataout, int N)
{
    int i,k,n,p;
    static int nstore = 0;      /* store N for future use */
    static COMPLEX *cf;         /* coefficient storage */
    COMPLEX *cfptr,*Dinptr;
    double arg;
/* Create the coefficients if N has changed */
    if(N != nstore) {
        if(nstore != 0) free((char *) cf);    /* free previous */
        cf = (COMPLEX  *) calloc(N, sizeof(COMPLEX));
        if (cf == 0) {
            printf("\nUnable to allocate memory for coefficients.\n");
            // add cleanup here
            exit(1);
        }
```

```
/* scale stored values by 1/N */
        arg = 8.0*atan(1.0)/N;
        for (i=0 ; i<N ; i++) {
           cf[i].real = (float)(cos(arg*i)/(double)N);
           cf[i].imag = (float)(sin(arg*i)/(double)N);
        }
    }
/* Perform the DFT calculation */
    printf("\n");
    for (k=0 ; k<N ; k++) {
        Dinptr = Datain;
        Dataout->real = Dinptr->real * cf[0].real;
        Dataout->imag = Dinptr->imag * cf[0].real;
        Dinptr++;
        for (n=1; n<N; n++) {
        p = (int)((long)n*k % N);
           cfptr = cf + p;        /* pointer to cf modulo N */
           Dataout->real += Dinptr->real * cfptr->real
                        - Dinptr->imag * cfptr->imag;
           Dataout->imag += Dinptr->real * cfptr->imag
                        + Dinptr->imag * cfptr->real;
           Dinptr++;
        }
        if (k % 32 == 31) printf("*");
        Dataout++;         /* next output */
    }
    printf("\n");
}
```

1.2. Python code

```
#makesGraphs from outputR.txt C++ program
#ver 15
import matplotlib.pyplot as plt
import numpy as np
```

```
from numpy.fft import fft, ifft
import math
import cmath
line =0
k=int(line)
s=int(0) #unit to display
dl=int(1600)#data length this value needs to be the number of complex I-Q
samples#fdl = float(dl)
with open('outputR_1600.txt') as f: #data file written by C++ code that
performed delay correction
    for line in f:
        data = line.split(",")
        re = []
        im = []
        fre = []
        fim = []
        y = []
        cx = []
        LX = []
        LY = []
        NLY = []
        TNLY = []

        for cnt in range(0,2*dl,2):
            re.append(data[cnt])
            im.append(data[cnt+1])

        for i in range(dl):
            v=re[i]+im[i]+"j"
            cx.append(v)

        for item in re:
            fre.append(float(item))
        for item in im:
            fim.append(float(item))

        for c in range(0,dl,1):
            x= complex(fre[c],fim[c])

        for item in range(dl):
            y.append(x)

        X=fft(cx)

        LX = np.array(X)

        for i in range(dl):
            LX[i] = LX[i].real*LX[i].real + LX[i].imag*LX[i].imag
            LX[i]=cmath.log10(LX[i]) #log values

        LY = np.append(LX,LX[0])
        NLY = LY[::-1]
        TNLY = NLY[:200]

        fig, (ax0,ax1, ax2,) = plt.subplots(nrows=1, ncols=3,)
```

```
ax0.plot(10*TNLY)# normally 20, not 10.  mag is squared
ax0.grid()
ax0.set_title('Impulse Log10 ')
k += 1
print("line ",k)
t = np.arange(0, dl, 1)

fig.set_figwidth(12)
fig.set_figheight(6)
ax1.plot(fre,fim) #X-Y plot
ax1.set_ylabel('Q volts')
ax1.set_xlabel('I volts')
ax1.set_title('I vs. Q')
ax1.grid(True)
ax1.set_xlim([-2,2])
ax1.set_ylim([-2,2])

ax2.plot(fim) # plot vs freq
ax2.plot(fre) # plot vs freq
ax2.grid(True)
ax2.set_title('I and Q vs. Freq')
ax2.set_ylim([-2,2])
plt.show()
```

2. Upstream Cable Echoes Come In Two Flavors

A first type of echo can be compared to a person standing within two walls of a canyon and yelling "hello". They hear their call repeated several times, each time getting weaker. The signals are bouncing off the 2 walls.

A second type of echo can be compared to a person standing in front of a big single wall and yelling "hello". They hear their call repeated only once.

Cable systems have both types, although the first type is by far more common. The first type, "infinitely recursive", is created by a pair of impedance discontinuities created on a cable line. It is of interest as the impedance mismatch may be caused by cable line damage.

The second type, "single recursive", may be caused by a signal finding two paths upstream. This condition was discovered from upstream transmissions that took the easy upstream path from a home through a high value tap port to the fiber node, and a slightly harder and longer path from a home, through the tap port to the tap's output port, and then off an impedance mismatch downstream to head upstream to the fiber node.

Figure A1 shows the first type of channel (top left) and how it is canceled with the top right impulse response. Essentially, each time an echo is canceled, a tap makes another smaller echo that also needs to be canceled until it is too weak to matter. The infinitely recursive solution impulse response is shown on the upper right.

Figure A2 shows the second type of channel (top left) and how it is canceled with the top right single recursion impulse response. Essentially, each time a multi-recursive echo is encountered, a second tap makes an inverse echo to cancel the next recursion. The solution impulse response is shown on the upper right.

Thus, the solution to an infinitely recursive echo is single recursion set of tap coefficients.

The solution to a single recursive echo is an infinitively recursive set of taps, up to the point when you run out of taps. Hopefully at that point the echo has died out.

This same principle works when observed in the frequency domain for both types of echoes because of time-frequency duality. In the frequency domain, when a frequency response ripple goes up, its inverse response ripple goes down.

OFDM/OFDMA does not employ time domain taps per se but multiplies each frequency domain subcarrier with a complex coefficient for correction. An IFFT can reveal the associated impulse response.

The Math

A single recursion echo can be modeled by: 1 + a where 1.0 is the main signal amplitude and 'a' is the echo's amplitude in linear terms. That is, for a -3dB echo a = 0.707. In a baseband channel 'a' is real, but in an RF channel 'a' may be complex. To be more precise,

$$a = Ae^{j2\pi fT}$$

where A is the amplitude of the echo, f is carrier frequency (MHz) and T is the delay of the echo (usec). However, the equations are easier to present if we simply use 'a'.

The equalizer solution is for a single recursion echo is:

$\frac{1}{1+a} = 1 - a + a^2 - a^3 + a^4 -$ So, the result is what was expected: infinitely recursive. The first term '1' represents the main tap of the equalizer. The second term '-a' acts to cancel the echo by subtracting it. However, in doing so, it causes another, smaller echo in the response. This smaller echo requires the third term to cancel it. This produces another, yet smaller echo, which requires the 4th term to cancel it, and so on until we reach the end of the equalizer delay line. After that, any remaining echo energy (hopefully very small) is not canceled and shows up as reduced RxMER (received modulation error ratio), essentially a noise floor, in the receiver.

If 'a' is a relatively big number such as 0.707 (-3 dB), and the delay T of the echo is several symbol periods, we might run out of taps in the pre-equalizer before we get an accurate solution. DOCSIS 2.0 and later pre-equalizers have 24 taps, with 7 taps normally assigned ahead of the main tap, leaving 16 taps to cancel the echo. For a small value of 'a' such as 0.1 (-20 dB), with a short echo delay, 16 taps is normally sufficient to cancel the echo with minimal residual energy.

Note that the recursions in the above equation have alternating signs. To check for this effect, we can examine the real and imaginary parts of the equalizer taps. If the response is alternating in sign, it is a hint that this type of solution may be present.

A multiple recursion echo may be modeled as:

$$1 + a + a^2 + a^3 + a^4 - .....$$

So its solution can be computed as:

$$\frac{1}{1+a+a^2+a^3+a^4-....} = 1-a$$

Which shows that an infinitely recursive echo can be canceled by a single recursion. So, you should be able to cancel a multiple recursion echo with an adaptive equalizer with only two taps – one for the main signal and one for the echo. To be precise, this example applies to an ideal case where the echo delay T equals a multiple of the symbol period, which for a 5.12 Msps DOCSIS upstream symbol rate is T_s = 195 ns. So, the pre-equalizer can exactly cancel the echo with a single tap if the single echo has delay T = 195 ns, 390 ns, or 585 ns, etc. If the echo lies between these multiples, the equalizer will activate additional taps to provide interpolation. In that case it will be more difficult to see the pattern of a single main recursion.

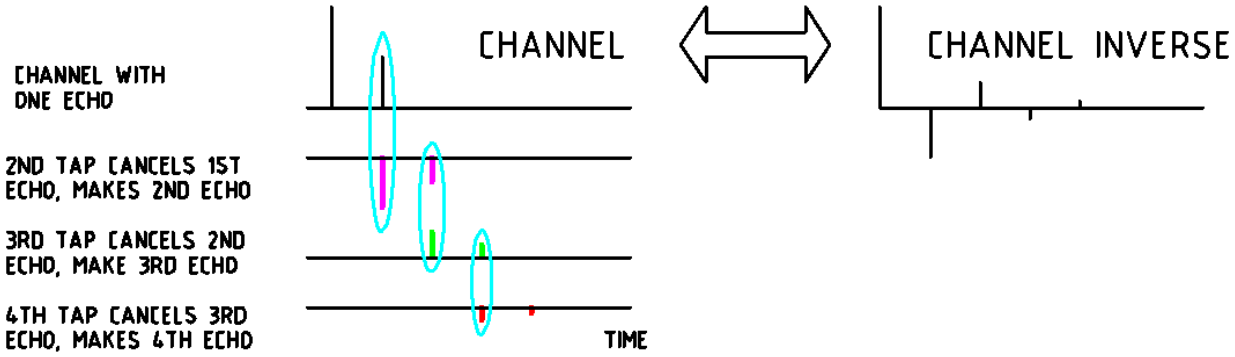

Figure 27 – A single recursive echo needs an infinite number of taps for cancelation. Plots are linear voltage vs time. Blue ovals show cancelation.

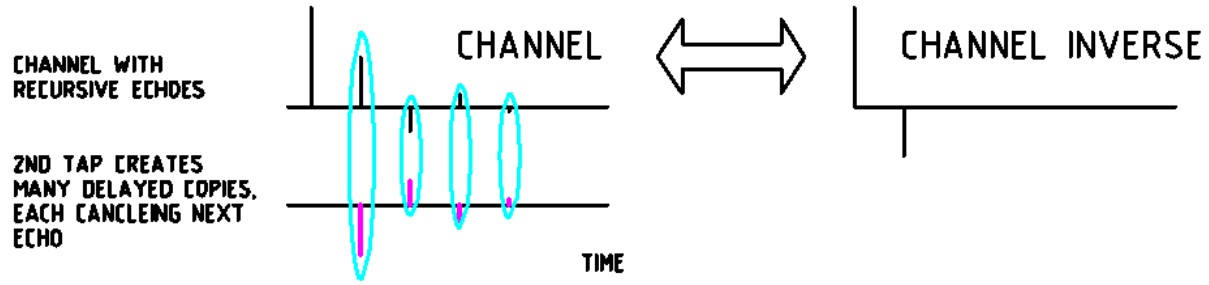

Figure 28 – An infinitely recursive echo is candeled with a two-tap equalizer. Each echo encountered cancels the next echo with a single tap.

3. Linear Distortion Analysis and Network Discovery

Since early days of PNM, we found that by analyzing the characteristics of equalization coefficients, we were able to identify characteristics and features of our HFC network. Many times, these characteristics are apparent when accompanied by a noticeable impairment or condition in the plant such as an impedance mismatch due to a loose connector or an amplifier malfunction. Prior to the DOCSIS 3.1 spec version, we relied on 6.4 MHz or narrower channels in the upstream leveraging up to 24 pre-equalization coefficients. In the downstream direction, although number of coefficients have not been specified, we have seen implementations using 32, 48 or higher number of equalization coefficients to compensate for distortion within the 6 MHz downstream channel. The upstream uses the ranging process to iteratively adjust the coefficients until they converge to a value that compensates channel distortion. In the downstream there is no handshaking with the CMTS so the CM has the sole responsibility in distortion compensation. We have introduced techniques to correlate common impairments within these narrow channels and figure out which CMs share the same impairment and infer the portion of the network that is common to them. One technique consisted of performing complex division of the FFTs of time domain equalization coefficients of the two CMs under comparison. High correlation results in the complex division approaching 1, while if uncorrelated values the division would be much different than 1.

The same approach is valid in DOCSIS 3.1 systems except that the coefficients are already in the frequency domain so one just has to divide the coefficients of CMs under comparison. The normalized equation is shown below.

$$DistortionMatchingMetric = \frac{\sum_{i=1}^{N} \left| \frac{Coef_i CM_k}{Coef_i CM_l} \right|}{N}$$

Where each of the i equalization coefficients of CM_k and CM_l, are complex divided with N being the total number of subcarriers.

Because in DOCSIS 3.1 systems, we are talking about thousands of coefficients rather than tens of coefficients, a much higher resolution is expected. The fact that you are leveraging a complex division process, the likelihood of false matches is much smaller compared to an amplitude only comparison operation. This greater sensitivity enables grouping of cable modems that have more subtle conditions in common, thereby enabling network discovery even on healthy portions of the network.

Another useful tool leveraging complex equalization coefficients analysis is group delay distortion. In DOCSIS 3.0 and earlier versions of the specifications, by analyzing the energy in the pre-main tap coefficients, we had a good indication of the amount of group delay distortion present, and we could infer whether the CMs where sitting behind 0, 1, 2 or higher numbers of amplifiers in cascade. One challenge was that very short micro-reflections would also cause pre-main tap energy to rise, so careful analysis is needed. Figure 32 shows the tap values from an example DOCSIS 3.0 CM.

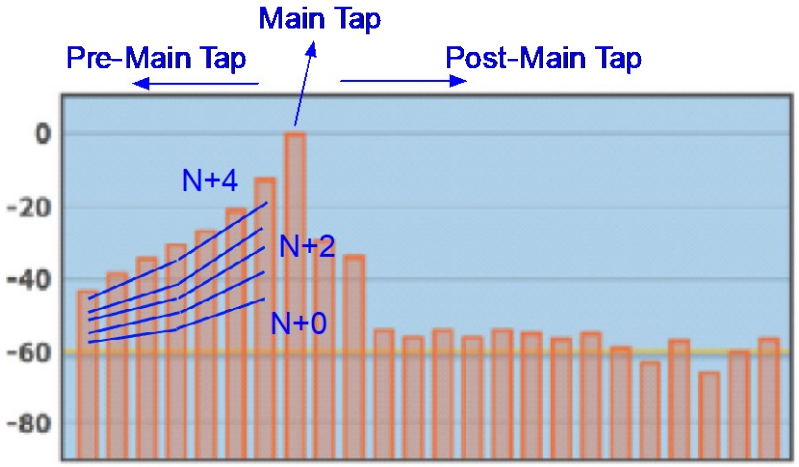

Figure 29 - DOCSIS 3.0 Pre-main tap coefficients and group delay.

In DOCSIS 3.1 systems, we have a view of not just 6 .4 MHz but we can cover the entire US and by obtaining group delay from phase response we can have an accurate view of how group delay changes at the band edges as the number of actives in cascade changes; see Figure 33.

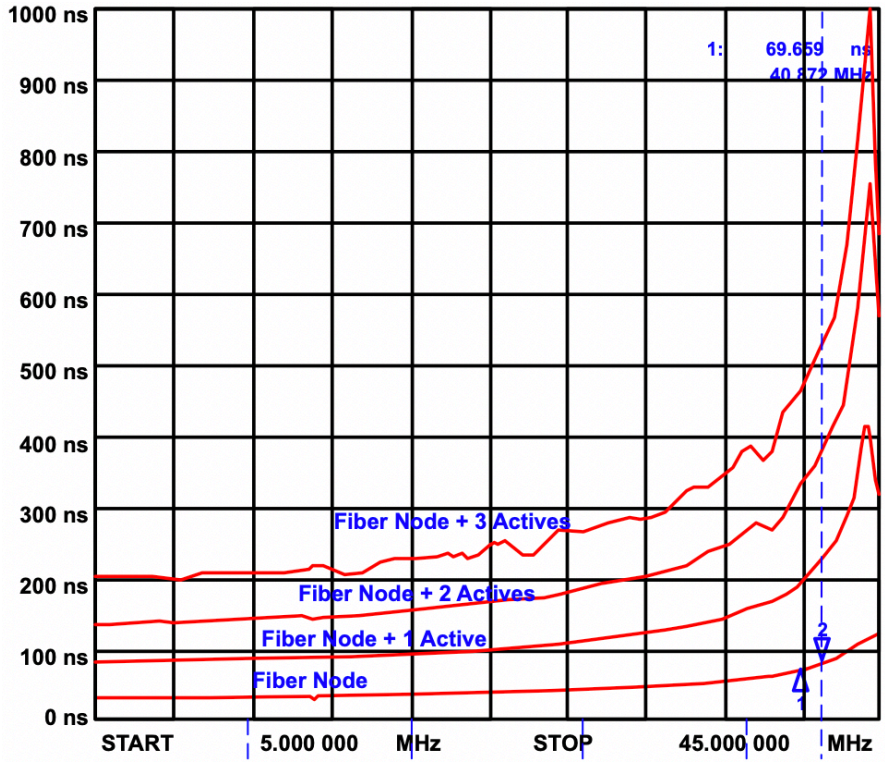

Figure 30 - Group delay and cascade value.

When reflections occurred at tap-coax interfaces, we have been able to deduce coaxial lengths between taps and amplifiers or between taps. This has been useful in determining plant characteristics. As DOCSIS 3.1 specifications came along, the channel became much wider, up to 192 MHz in the downstream and 96 MHz in the upstream. DOCSIS 3.1 specifications evolved from single carrier QAM to multiple subcarriers using either 50KHz or 25KHz, resulting respectively in up to 3800 or 7600 subcarriers in the downstream. In DOCSIS 3.1 specifications, the approach to distortion compensation is the same as with DOCSIS 3.0 and earlier specification versions, in the upstream pre-equalization leverages the ranging handshaking process between CMTS and CM and in the downstream only CM equalization takes place.

A change between DOCSIS 3.0 SC-QAM and DOCSIS 3.1 multicarrier OFDM/OFDMA is that while in DOCSIS 3.0 and earlier versions we had equalization coefficients represented in the time domain across several symbols, in DOCSIS 3.1 we are dealing with a frequency domain equalization representation with one coefficient per subcarrier. This difference is easily overcome through a translation from time to frequency domain using Fourier transformation.

A key difference is that instead of 24/32/48 coefficients in DOCSIS 3.0 and earlier specifications you have up to 3800 or 7600 coefficients in DOCSIS 3.1 specifications. The level of granularity that is possible is more than two orders of magnitude. The techniques discussed here enable the use of equalization coefficients' magnitude and phase for all processes.

Knowing magnitude and phase allows us to determine whether the reflection due to an open a short or a specific complex impedance. This becomes very useful to discriminate whether the distortion impacting CMs is the same or not and will play a key role in grouping devices with common features with high level of sensitivity. The goal is to correlate and group CMs not only when there is a noticeable event or change in channel conditions but also to detect subtle changes even when the network is not impaired, thereby making the HFC network discovery process not just impairment driven but feasible on a healthy network.

Regarding the wider bandwidth coverage, it is worth mentioning that the band-edges in the narrow SC-QAM channels introduce uncertainties as the nature of the FFT requires same values at both edges. The much wider DOCSIS 3.1 channel and frequency equalization operation circumvents that issue. The wider channel view allows us to better describe the group delay distortion in the upstream that is indicative of cascade value.

4. Several upstream (first operator, OFDMA Pre Equalization) and downstream (second operator, OFDM Channel Estimation) results in figures.

For this appendix section, we provide several field data examples in plots. The data come in as I-Q coefficients, one per subcarrier, and typically every 50kHz. The plotted spectral coefficients have linear delay removed before viewing. The FFT size is typically 1024 points for downstream, while the DFT size is 224 points for upstream. In each figure in this section, the center plot is polar I vs. Q, the right plot is I and Q vs. frequency, and the left plot is magnitude of time (impulse response) in dB vs. time. The horizontal scale is the index number. For downstream the time is 19.53ns. per index (4/204.8e6) with a 1024 sample FFT. The bandwidth is 50 kHz times the index, or about 11.2 MHz for upstream and 51.2 MHz for downstream. Note: not all the downstream data provided were plotted.

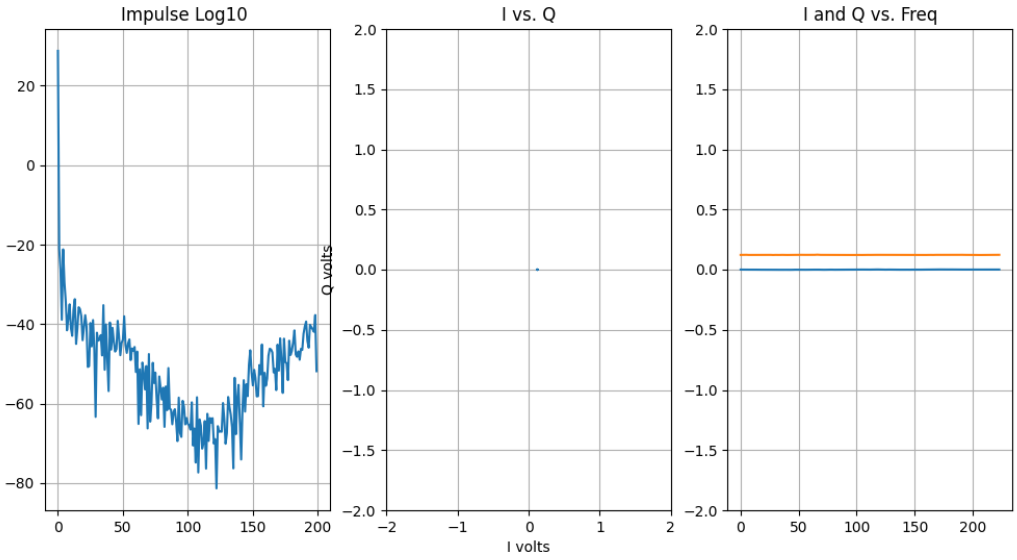

Figure 31 - Pre-equalization data that appears too clean to be cored.

Figure 34 above, shows an upstream response that is too good to be real. Half of the CMs in the data set we received have this response. We guess the result is possibly related to having pre-distortion turned off.

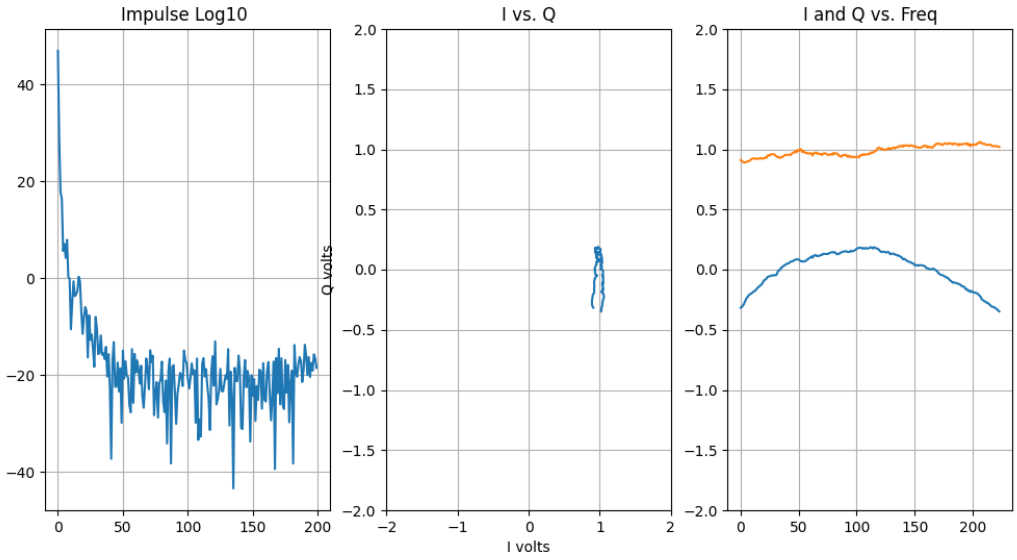

Figure 32 - A typically good pre-equalization result.

Figure 35 above shows a more believable upstream response. A typical upstream good response. Most CMs in our data set look similar to this one. Group delay variation is shown at the top and bottom of the band.

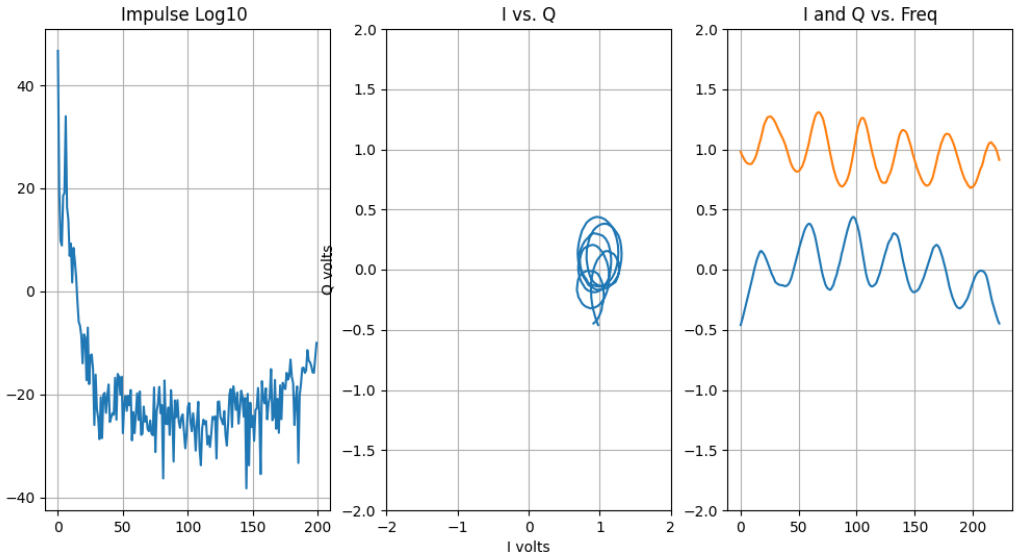

Figure 33 - Pre-equalization showing a standing wave.

Figure 36 above shows an upstream response with a standing wave caused by an infinite recursion echo, which we know because the solution appears as a single recursion.

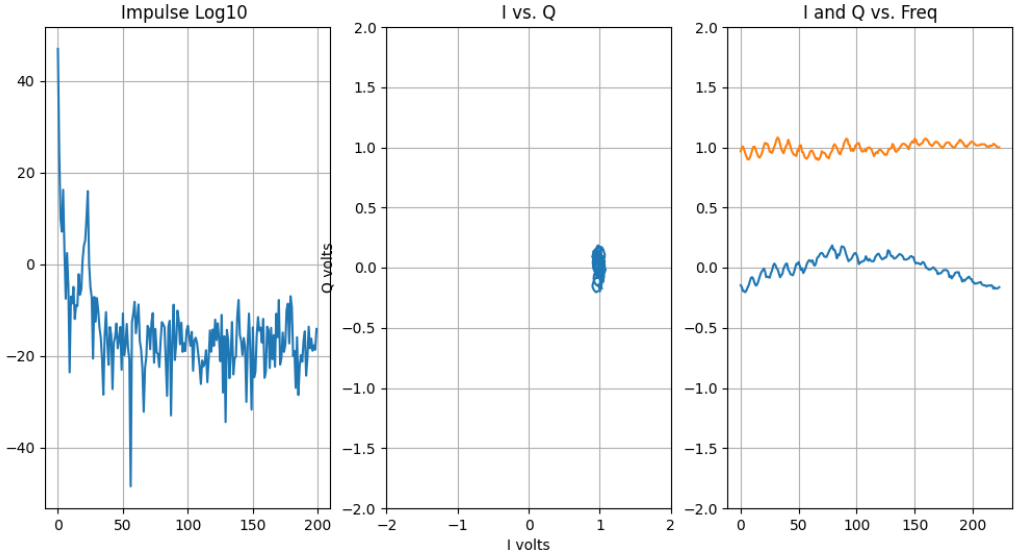

Figure 34 - Upstream pre-equalization data showing a relatively long echo.

Figure 37 above shows an example of an upstream response with a long echo. Note the tight ripple response in the far-right graph.

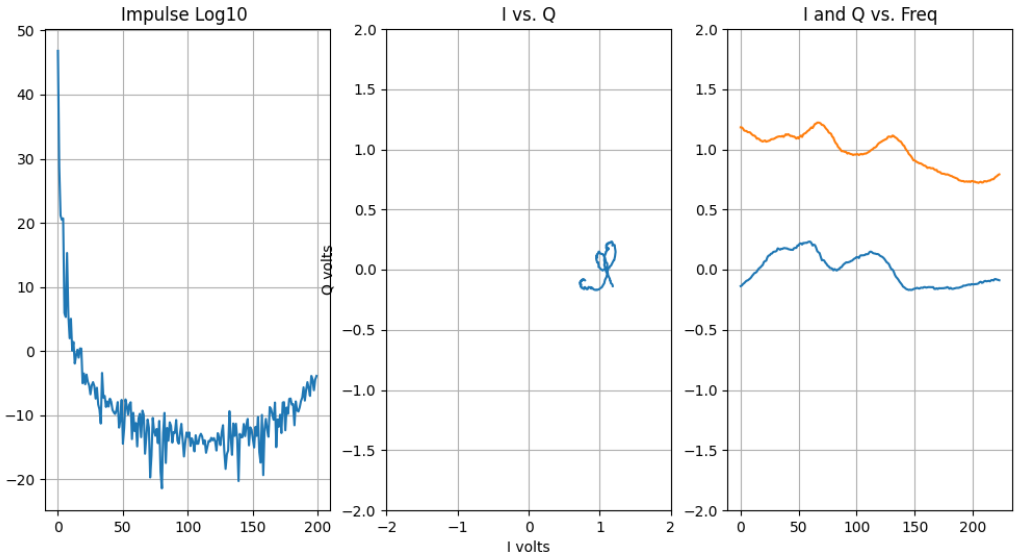

Figure 35 - Upstream pre-equalization, problem probably in house.

Figure 38 shows an upstream data plot of a close-in echo which is probably in the home.

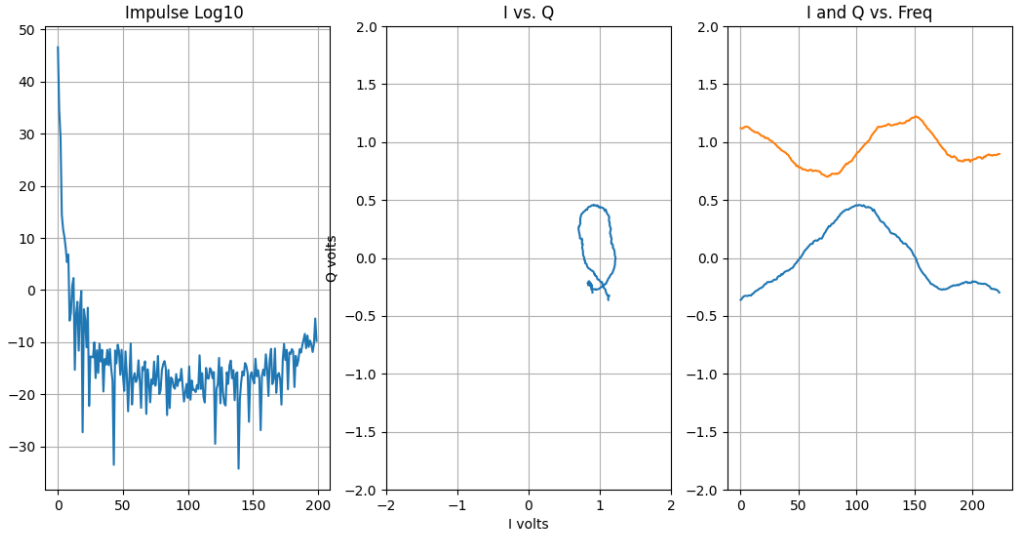

Figure 36 - Upstream pre-equalization data, in-home wiring, maybe open coax shield.

Figure 39 shows pre-equalization data from a CM where there is likely an open shield or other in-home wiring problem.

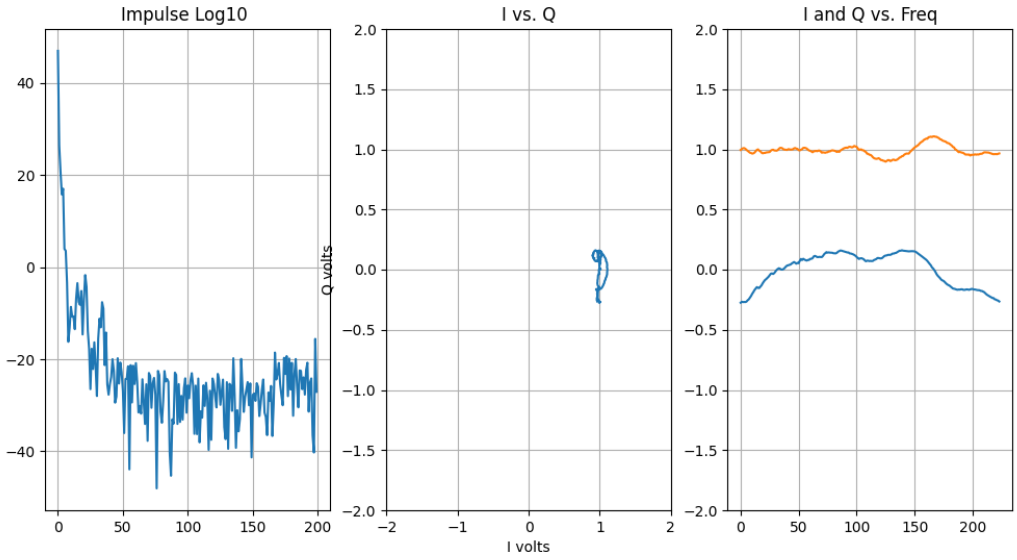

Figure 37 - Upstream pre-equalization, echo only in low frequency bands, so defect probably in home.

Figure 40 above shows an echo but only in the lower part of the band, which suggests an echo cavity, likely in the home.

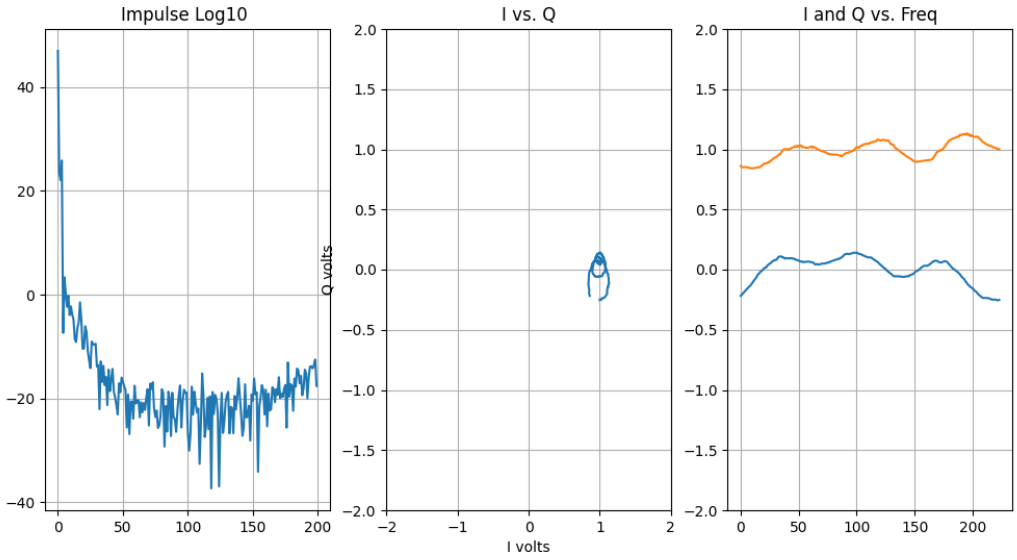

Figure 38 - Upstream pre-equalization data, standing wave.

Figure 41 above shows an example of a small standing wave. Echo is again probably in the house.

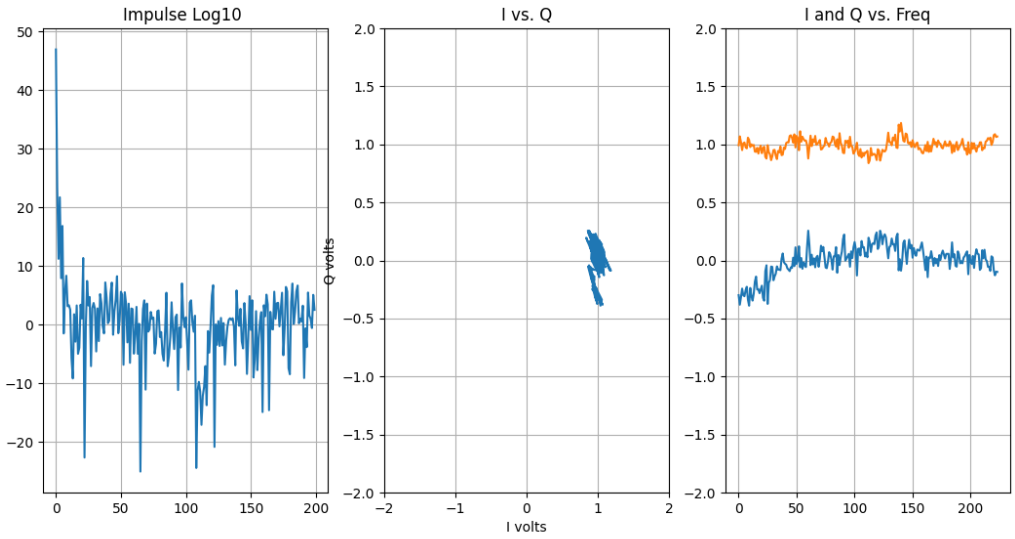

Figure 39 - Upstream pre-equalization, guessed to be water in coax.

Figure 42 above is rather rare in the data set we received. A guess is that it is water in the coax cable from the jagged frequency response. No field tests confirmed this conjecture.

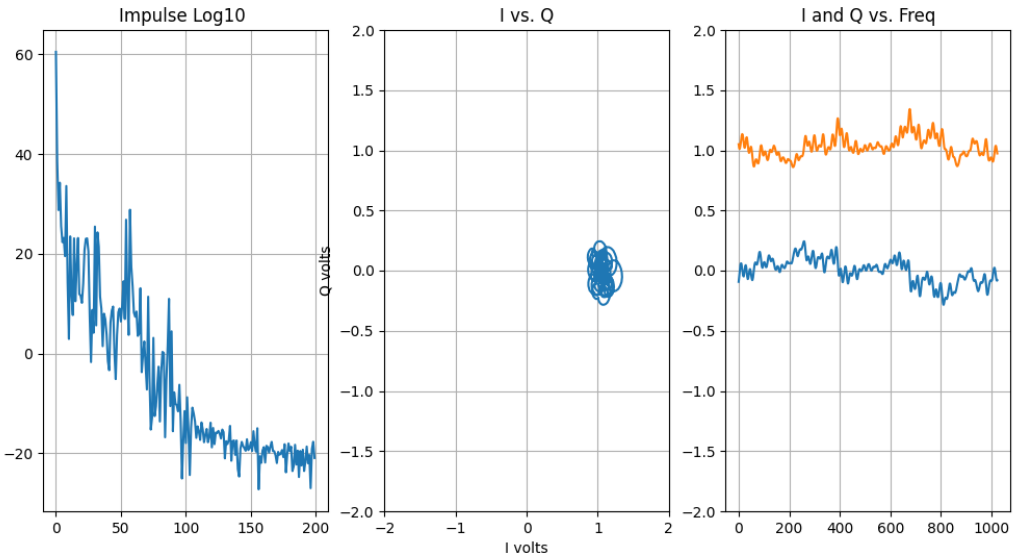

Figure 40 - Downstream, guessed to be water in coax.

Moving now to downstream data, in Figure 43 we see our first example which is guessed again be a case of water in the coax plant. Note the jagged plot on the right, and echo energy close in time on the far-left impulse response plot.

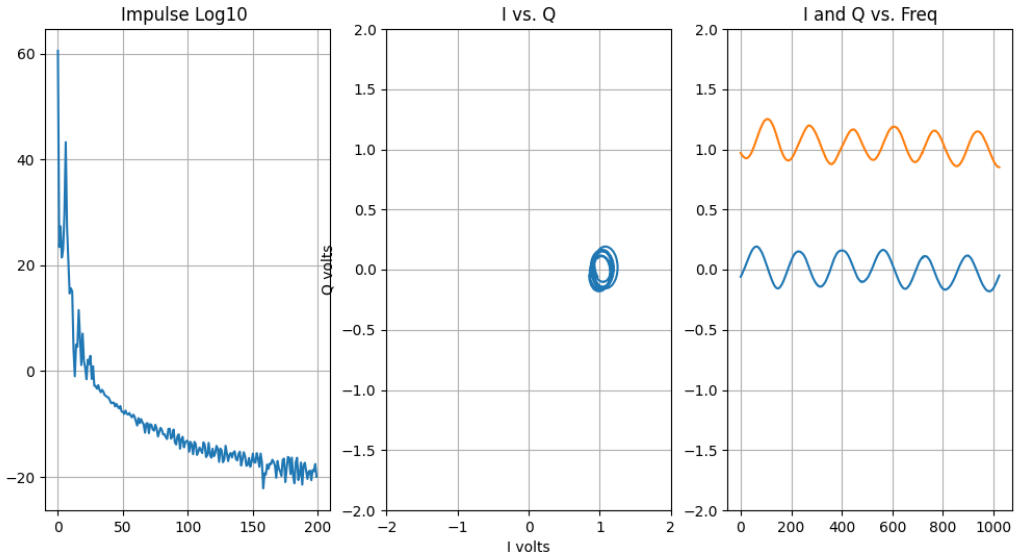

Figure 41 - Downstream lab test. CM is reporting channel response.

Figure 44 shows a confirming lab test, which we used to prove that the CM here was providing channel estimation data in the downstream, clearly showing the echo we inserted.

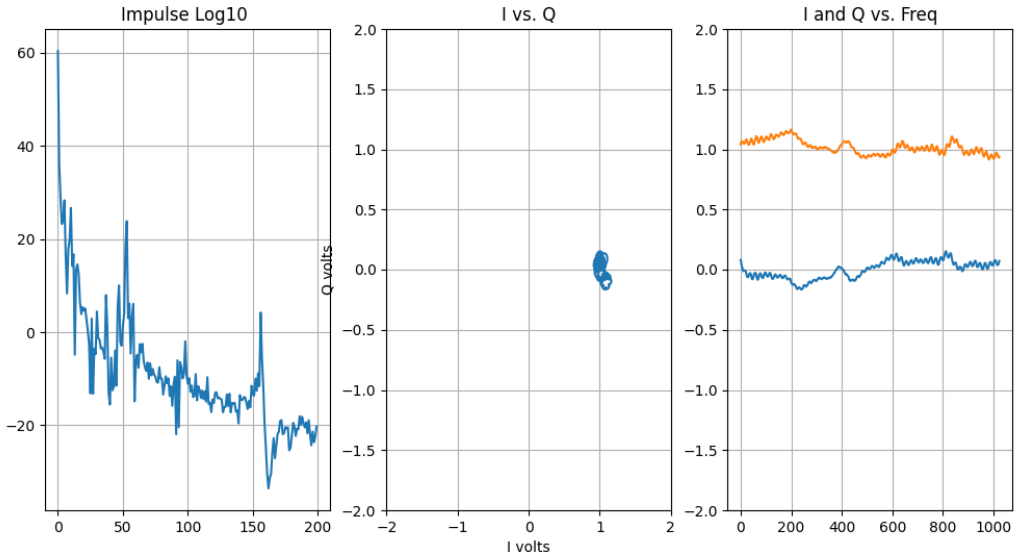

Figure 42 - Downstream, long echo tunnel so echo was weak. Source unknown.

Figure 45 is a channel estimation data plot of a long echo tunnel, which shows a weak response.

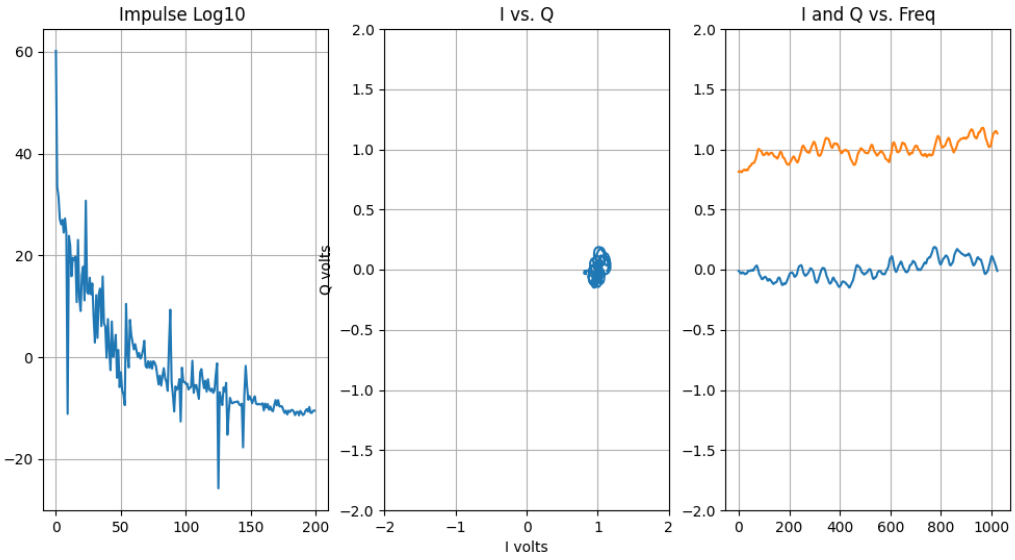

Figure 43 - Downstream, pair of long echoes.

Figure 46 shows a pair of long echoes in the channel estimation data.