# PNMP-基于预均衡的主动故障维护

**方案预研-作者胡江昳**

**感谢在方案调研过程中吕海艇、程翔的耐心指导**

## Pre-Equalization的原理

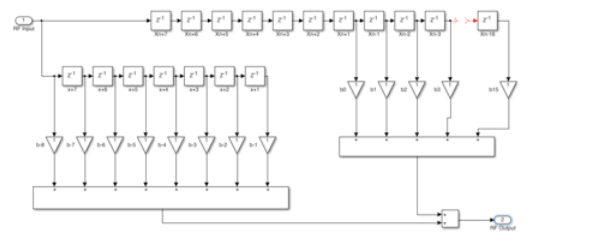
CMTS根据CM的测距消息，评估CM的信号质量，生产预均衡参数，下发给CM，由CM的预均衡器进行信号增益，从而抵消上行线路上的损耗。

## PNM using Pre-Equalization的原理

由于CM的预均衡机制，导致在有可以被预均衡掉的故障的情况下，传统的故障发现机制中SNR,接收电平等机制已经比较难发现问题。

而预均衡参数实际上是可以反馈出为了抵消线路损耗，而做出的频率响应，通过分析频率响应可以分析出上行故障类型，上行故障严重程度，甚至可以分析出故障点与CM之间的距离

## 预均衡器的工作原理



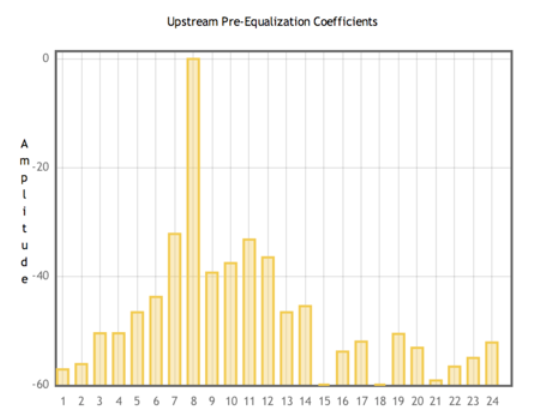
从DOCSIS2.0开始，CM上的预均衡器就有24个抽头，其中8号抽头是主抽头，前7个叫前主抽头，后16个叫后主抽头，前主抽头的作用是用于均衡群延时，后主抽头是用于抵消微反射。

每个抽头都有一个复数的系数Fn，叫预均衡系数

其中方块表示的是一个延时模块，每个方块延时一个T，三角是放大模块，整个预均衡器最终的效果就是通过调整预均衡参数，让信号在16个T里面每个周期进行一个幅度的增益，从而抵消掉反射回波

## DOCSIS预均衡还能做什么

DOCSIS预均衡除了可以抵消上行链路损伤导致的损耗外，通过分析预均衡系数，可以更多地了解DOCSIS上行链路的故障情况



预均衡系数

我们可以看到，8号抽头具有最大能量，用于传输主信号

我们可以根据前主抽头、主抽头和后主抽头得出一些排障用的分析参数

### F8主抽头 传输主信号

### F1-F7 前抽头 群时延补偿

### F9-F24后抽头 微反射补偿

### Frequency Response 频率响应

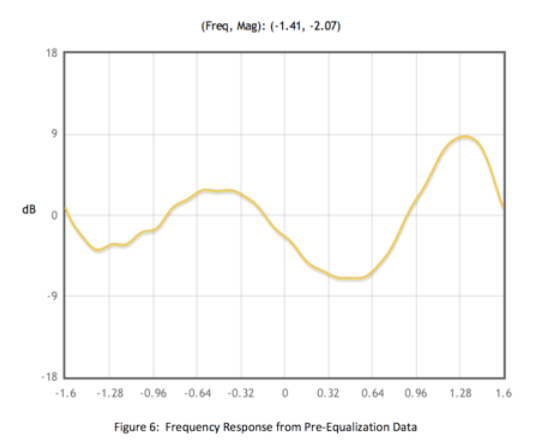
由于抽头系数描述了一个在时域内的增益幅度变化的关系，通过复数傅里叶转换公式可以通过24个抽头系数计算出频率响应，频率响应可以用于将CM进行分类，相同物理拓扑的相同故障CM应该拥有相似的频率响应。

由于给定的是24个复数系数，故考虑使用32元快速傅里叶变换来从时域转换为频域，主抽头需要与第16个复数重合，以保持响应在频率上的相对位置

映射到FFT的参数就是

Fin1-Fin8=0

Fin9=F1，Fin10=F2......Fin31=F23，Fin32=F24

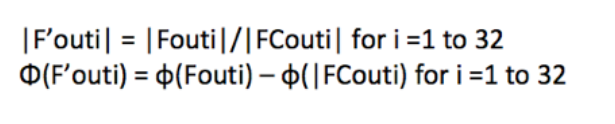


### 频率响应的校准

另外可以建立一个校准基线，用于消除CMTS和CM的失真在频率响应中的影响

其做法是可以建立一个参考CM，计算出基准频率响应

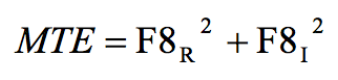
使用公式



最终可以得出校准后的频率响应值

参考6.5.2章节

### MTE-Main Tap Energy 主抽头能量 其实也可以看作一个功率，



### Pre MTE-Pre – Main Tap Energy前抽头能量



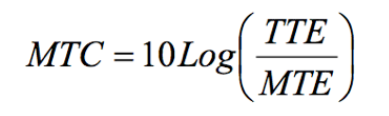
### Post MTE – Post Main Tap Energy 后抽头能量



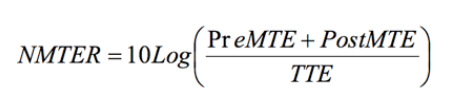
### TTE – Total Tap Energy 总抽头能量



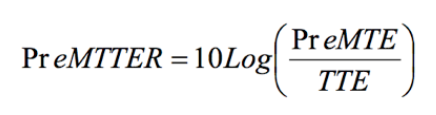
### MTC – Main Tap Compression主抽头压缩 MTC 大于2dB预均衡器达到极限无法成功补偿失真，均衡失效，线路故障十分严重



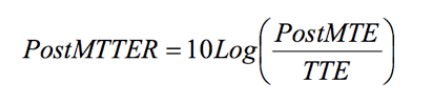
### NMTER- Non Main Tap to Total Energy Ratio 非主抽头与总能量比 线路补偿要消耗的能量比，反应失真的严重性



### PreMTTER -Pre Main Tap to Total Energy Ratio

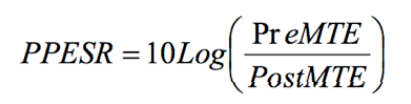


### PostMTTER – Post Main Tap to Total Energy Ratio



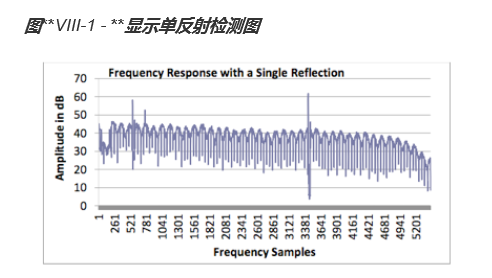
### PPESR – Pre –Post Energy Symmetry Ratio

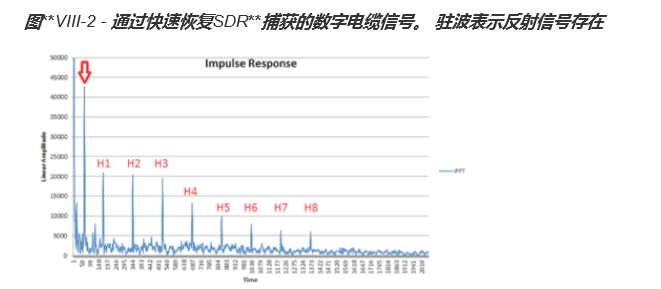
前后抽头之间的能量特性，反应不同的故障类型



### TDR -Time Domain Reflectometry

通过频率响应可以通过反向傅里叶转换，推算出一个时域内的时间幅度图，通过分析图中的驻波间的时间间隔可以计算出一个时间差，再通过时间差乘以光速除以二可以得到一个初略的故障距离

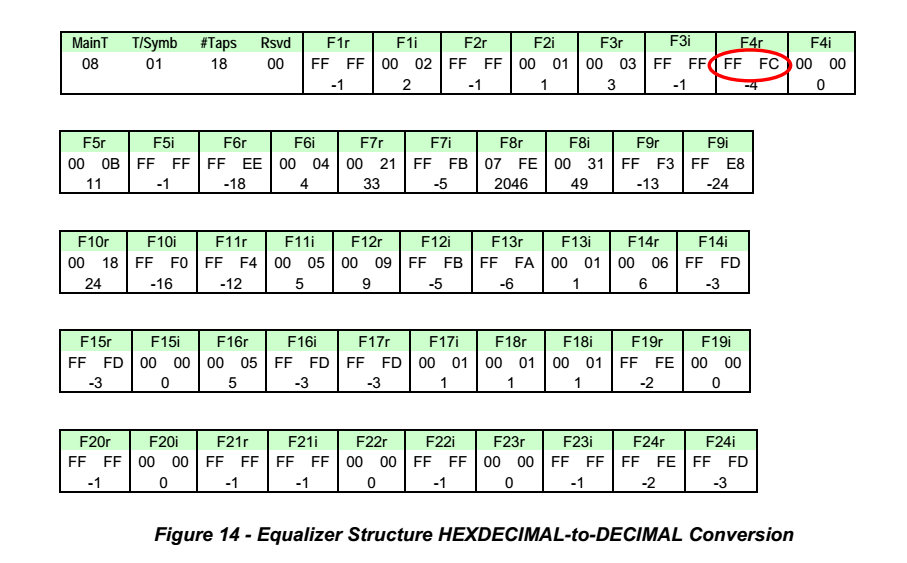




### MRLevel Micro reflection level

反应故障级别

## MIB格式解释



### 格式讲解

MainT 第一个字节 表示主抽头的位置

T/Symb 表示周期每符号，与抽头延时模块有关

Taps 总抽头数 为16进制

Rsvd 保留位

Fxr，Fxl ，x =1 ~24 表示24个抽头的预均衡系数，每个系数分实部和虚部两个参数，每个参数两个字节，一个系数4个字节

### 系数解码规则

有两种解码规则 4个半字节解码和3个半字节解码，由于市场上流通的CM，主抽头最大振幅为2047，不会超过三个半字节的解码规则，故为了兼容性，统一采用三个半字节解码

解码规则如下

a）000D = 0000 0000 0000 1101→消除第一个半字节→0000 0000 1101 = 13

b）0FFE = 0000 1111 1111 1110→消除第一个半字节→1111 1111 1110之后，反向和加载1→000000000001 + 1 = 000000000010 = 2 → -2

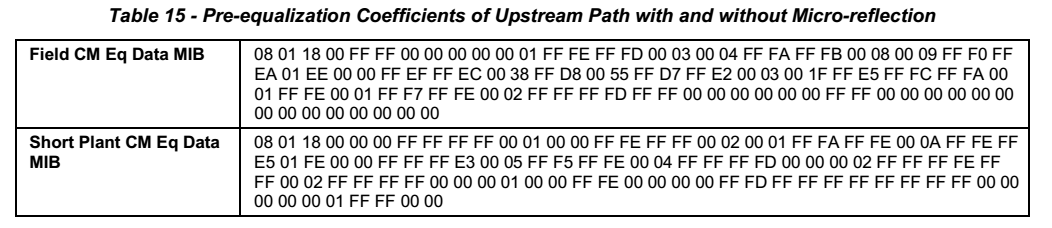
## 频率响应计算方法

通过mib计算出24个预均衡系数，采用32元快速傅里叶变换，将主抽头放在第16位，对1-8位空出的全部补0，傅里叶转换后的数据中心点的位置在最左边，需要通过滚动，将中心点移到16号位上，通过傅里叶变换后的复数，计算出每个频率上的振幅，再通过归一运算，计算出每个频点的增益，形成最终的频率响应图

实验

参考CM-GL-PNMP-V03-160725.pdf 6.5.3章节

Mib数据



下面是我的用python实现的一套预均衡系数转频率响应的代码

还留有的疑问：由于归一算法不清楚，只能靠朦，经过多次实验，程翔提供的 10\*log(x/n)的算法绘制出的图比较接近文档中的效果，故如何归一是现在遗留的一个待讨论的问题

#! /usr/bin/env python

# -\*- coding: utf-8 -\*-

# vim:fenc=utf-8

#

# Copyright © 2014 jay <hujiangyi@dvt.dvt.com>

#

# PNMP Example

import matplotlib.pyplot as pl

import numpy as np

def fftResult(param,n) :

result = np.fft.fft(param);

realResult = np.real(result);

imagResult = np.imag(result);

sRealResult = np.square(realResult);

sImagResult = np.square(imagResult);

addResult = np.add(sRealResult,sImagResult);

sqrtResult = np.sqrt(addResult);

logResult = 10 \* np.log(sqrtResult /n);

return logResult;

freqResult = np.fft.fftfreq(32);

freqShiftResult = np.fft.fftshift(freqResult);

param1 = np.array([

0 +0j,

0 +0j,

0 +0j,

0 +0j,

0 +0j,

0 +0j,

0 +0j,

0 +0j,

-1 +0j,

0 +1j,

-2 -3j,

3 +4j,

-6.0 -5.0j,

8 +9j,

-16 -22j,

494 +0j,

-17 -20j,

56 -40j,

85 -41j,

-30 +3j,

31 -27j,

-4 -6j,

1 -2j,

1 -9j,

-2 +2j,

-1 -3j,

-1 +0j,

0 +0j,

-1 +0j,

0 +0j,

0 +0j,

0 +0j

]);

value1 = fftResult(param1,494);

redLineValue = np.fft.fftshift(value1);

pl.plot(freqShiftResult,redLineValue,'r')

param2 = np.array([

0.0 +0.0j,

0.0 +0.0j,

0.0 +0.0j,

0.0 +0.0j,

0.0 +0.0j,

0.0 +0.0j,

0.0 +0.0j,

0.0 +0.0j,

0.0 -1.0j,

-1.0 +1.0j,

0.0 -2.0j,

-1.0 +2.0j,

1.0 -6.0j,

-2.0 +10.0j,

-2.0 -27.0j,

510.0 +0.0j,

-1.0 -29.0j,

5.0 -11.0j,

-2.0 +4.0j,

-1.0 -3.0j,

0.0 +2.0j,

-1.0 -2.0j,

-1.0 +2.0j,

-1.0 -1.0j,

0.0 +1.0j,

0.0 -2.0j,

0.0 +0.0j,

-3.0 -1.0j,

-1.0 -1.0j,

-1.0 +0.0j,

0.0 +1.0j,

-1.0 +0.0j

]);

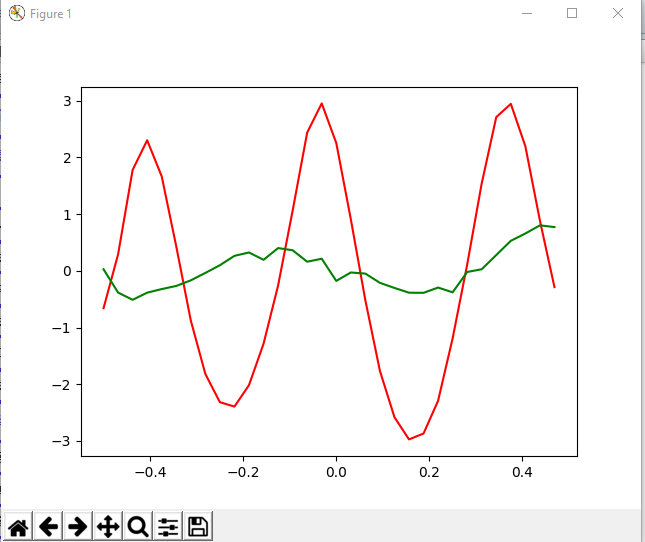
value2 = fftResult(param2,510);

greenLineValue = np.fft.fftshift(value2);

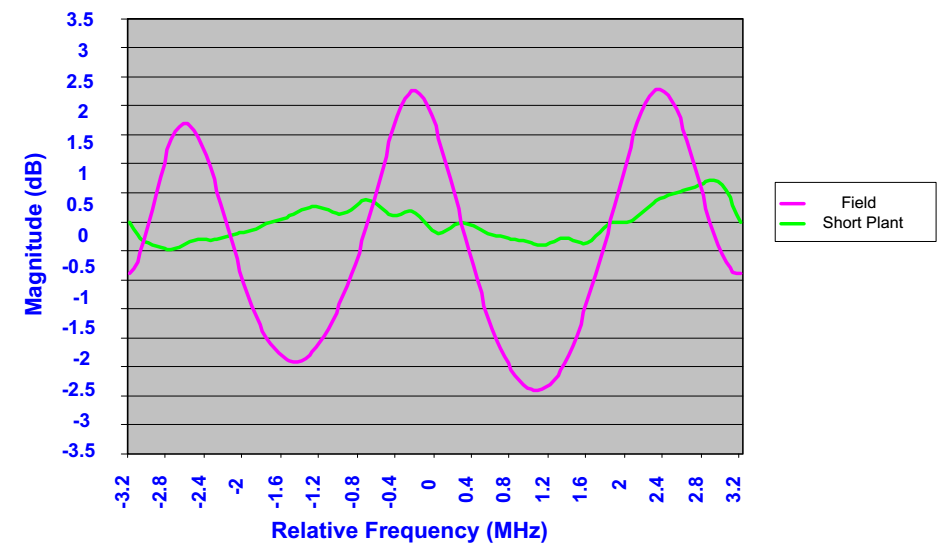
pl.plot(freqShiftResult,greenLineValue,'g')

pl.show()

最终效果如图



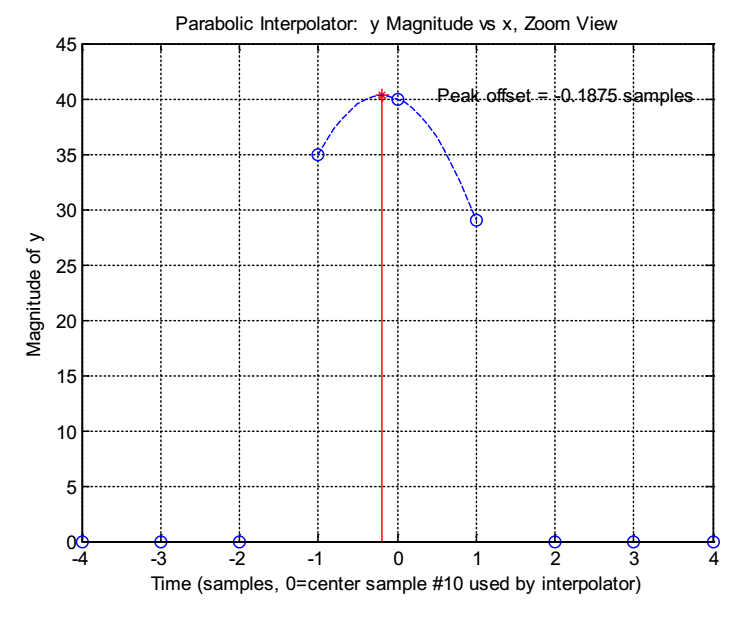
对比CM-GL-PNMP-V03-160725.pdf中的结果图



没有再继续研究校正后的图，算法都在那，应该不难换算

### 计算TDR，通过TDR估算故障距离

通过抛物线插值法计算微反射导致的延迟



先计算每个抽头的幅度值找到最大那个抽头以及左右两个抽头，建立一个抛物线，通过

抛物线方程

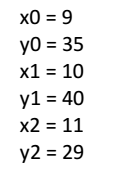
y=ax2+bx+c

和顶点式

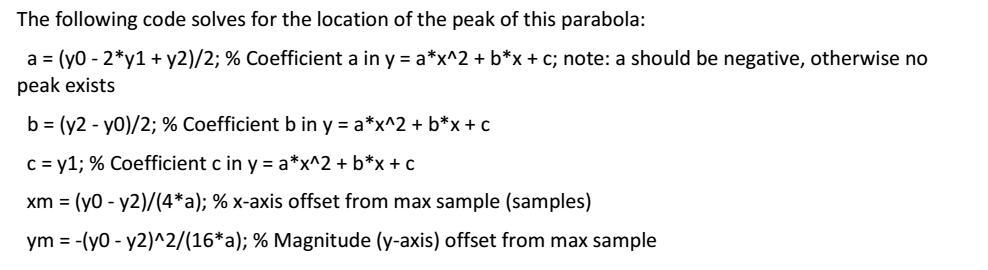
Xm=-b/2a

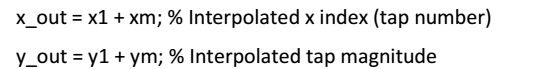
Ym=（4ac-b^2）/4a

加上三个点的xy坐标

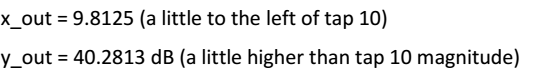


可以求解出如下方程



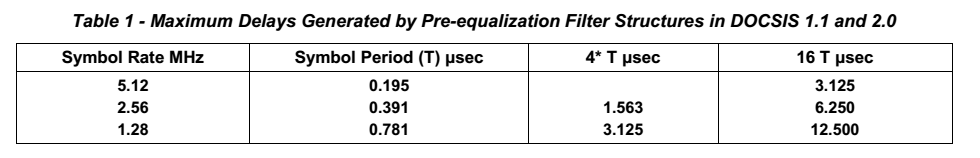


其中x\_out，y\_out就是抛物线顶点坐标



由于微反射 导致了延迟 本来最大信号应该在 8号抽头上现在发生了偏移 偏移到了x\_out这个虚拟抽头上，根据x\_out-8可以得到偏移的抽头差，由于一个抽头延迟一个symbol，从而推算出延迟了多少个symbol

通过读取CM的上行信道带宽，根据下表，获得每个symbol延迟的具体时长



从而得到微反射导致的时延

信号的传输速度可以认为是光速，故

故障距离=时间差\*光速/2

从而求得故障距离

以文档中的为例，假设是6.4M带宽

故障距离=(9.8125 - 8)\*3.125/16 \*0.000001 \* 299792458 = 106.1277011962890625≈106米

## 频率相应匹配方法



1、排除掉无法获取PNM

## 我们能做什么

我们可以实现一套基于采集CM预均衡参数，通过分析预均衡参数来找出由于预均衡技术而掩盖下来的物理故障

通过分析抽头的能量比，可以找到对应的故障类型，故障等级

通过对CM的频率响应进行分类，结合CM的物理拓扑，可以分析出故障大致区域

通过求TDR，较精确的告知工程人员，进一步缩小排查范围

## 主要参考文档

[主动式故障维护技术在HFC网络中的应用](http://xueshu.baidu.com/s?wd=paperuri:(46468d5acdf348d2c2e11b2cd562239c)&filter=sc_long_sign&tn=SE_xueshusource_2kduw22v&sc_vurl=http://cpfd.cnki.com.cn/Article/CPFDTOTAL-EUGS201410001006.htm&ie=utf-8&sc_us=14867582223593667729" \t "http://xueshu.baidu.com/_blank) 华为

CM-GL-PNMP-V03-160725.pdf CableLabs

Pre-Equalization-Final.pdf ZCorum