

/snr\_ser:

/direct: 存放了不同情况下通过 ruo\_main.m 跑 🎧 来的 snr 和 ser。

/direct/11.16: 存放了经过信道发送数据补正后的 snr-ser, (见 /vol\_save/11.12\_test)。

/direct/11.23: 存放了在更正传输错误前后的 snr-ser

/direct/11.25: 存放了用来 debug 的数据, 见 ruo\_debug.m。

/direct/12.1: 将两次接收信号 (补正信号和被补正信号) 重新同步后, 跑 🎧 来的 snr-ser。

/direct/12.2: 在发送信号幅度很大时 ser 会上升, 因此储存下来一些数据用来看 ser 上升的原因, 对应程序 ruo\_debug。

/direct/12.3: 在发送信号前加了一段幅度很大的导频用来定位同步点, 并且将两次接收信号 (补正信号和被补正信号) 重新同步, 跑 🎧 来的 snr-ser。

/vol\_save:

11.11\_test: 1 路信道为实验组, 4 路为对照组, 两路信道发送同样的数据。尝试通过对比实验组和对照组, 来删除通过 1 路信道的 160M 信号中 🎧 错的点。其中, 1 路信道发送两次同样的数据, 分为实验组和实验组 1。如果对照组无法删除实验组的全部错误点, 那么尝试通过实验组 1 来对实验组进行补正。

unquit\_num\_amp40\_loc10.txt 中存放的是对照组的长度、两个实验组的长度以及两个实验组中未能删除点的个数, 以及两个实验组重合的错误点的个数, 最后一行是 20 次 exp\_time 中有多少次两个实验组有重合点。

corrindex\_save\_amp40\_loc10.mat 中存放的是两个实验组重合点的坐标。  
errloc\_save\_amp40\_loc10.mat 中存放的是实验组中未删除点在 160M 数据中的位置。errloc\_save1\_amp40\_loc10.mat 中存放的是实验组 1 中未删除点在 160M 数据中的位置。对应程序：ruo\_test.m。

11.12\_test、2、5：与 11.11\_test 相同，三个文件夹对应的发送数据长度不同。txt 文件里存放了发送错误的数量以及未能补正的数量，以及实验组 1 和 2 相关错误点的数量。

ruo\_calculate\_ser.m：集合了计算 snr、均衡、计算错误点个数的功能。

ruo\_channel\_coefficient.m：用于仿真生成信道参数。

ruo\_debug.m：用于往 mat 里存用来 debug 的数据。

ruo\_debug2.m：用于从 mat 里读数据，并用这些数据 debug。

ruo\_debug3.m：先在这个 m 文件里写程序，没问题再放进 main2.m 中。

ruo\_filter\_gen.m：如果 origin\_rate 为不规则的速率（如 1.17e6），那么在速率转换时会因为公倍数过大而占满计算机内存。为了解决此问题，编写了 ruo\_filter\_gen 程序。在该程序中，假如 origin\_rate 设为 1.23456e6，那么会通过计算，将该速率转换为相差不超过一定范围的新的 origin\_rate，新的速率的公倍数不会很大，规避占满内存。

ruo\_gen\_light\_data.m：用来生成并保存给神经网络用的数据。

ruo\_gen\_newsend.m：12.23 发现，由于电路板隔直流的原因，发送信号的低频部分会被信道滤掉，即接收信号比发送信号少了低频分量。如果用少了低频分量的接收信号进行均衡，再与发送信号进行对比计算误码率时，会增加不必要的错

误。因此，选择将发送信号过一个高通滤波器，在进入信道前滤掉低频部分，并将滤掉低频分量的信号作为新的发送信号、原信号。但是，新的发送信号由于经过了滤波器，不再是 pam 信号。因此，需要用该函数将新的发送信号还原成 pam 信号。

`ruo_load_data.m`: 用来从 mat 里加载数据，用在 `ruo_plot.m` 里。

`ruo_main.m`: 主程序，用于在平台上发送数据、接收数据、速率转换、更正传输错误、同步、均衡。

`ruo_main2.m`: `ruo_main.m` 的版本备份，是 `ruo_main.m` 的上个版本。

`ruo_not_replace.m`: 用来测试没有更改传输错误时的 ser。

`ruo_pam4_testsend.m`: 用于 `ruo_test.m`。

`ruo_pam4_testsend2.m`: 用于 `ruo_test.m`。

`ruo_pam4_volsend.m`: 用于发送从 .mat 中导出的数据，用在 `ruo_main_voltest.m` 里。

`ruo_pamdemod.m`: 用来给 pam 信号解调，解调结果是  $0 \sim M-1$  的整数。

`ruo_pilot_gen.m`: 用于生成导频。

`ruo_sam_rate_con.m`: 用于速率转换。

`ruo_send.m`: 用于发送数据。

`ruo_send_correct.m`: 用于在参考信道发送补正数据。

`ruo_signal_equal.m`: 用于均衡。

`ruo_signal_equal_ls.m`: 用于进行 ls 均衡,和 `ruo_signal_equal.m` 中不同的是, `equal.m` 需要的参数是粗同步点, `equal_ls.m` 需要的参数是精同步点。

`ruo_signal_syn.m`: 用于同步。

`ruo_signal_syn_recorrect.m`: 信道 1 两次发送的信号进行精同步的时候会有一两

个相位点的差距，用这个程序来修正这个差距，让这两个信号的图像可以完全重叠。

ruo\_test.m: 用于生成/vol\_save/11.12\_test 中的数据。

发送和接收端口：

```
function X = rue_txDataSort(Tx)
% Tx cell数组
ch_num = length(Tx);
odd_zeros = 1407; %奇数补零
even_zeros = 1408; %偶数补零
if ch_num==1
    Xa = Tx{1};
    Xb = Tx{1};
    Xc = Tx{1};
    Xd = Tx{1};
elseif ch_num ==2
    Xa = Tx{1}; % DA2 in hardware
    Xb = Tx{2}; % DA3 in hardware
    Xc = Tx{1}; % DA4 in hardware
    Xd = Tx{2}; % DA1 in hardware
elseif ch_num ==3
    Xa = Tx{1};
    Xb = Tx{2};
    Xc = Tx{3};
    Xd = Tx{1};
elseif ch_num ==4
    Xa = Tx{1};
    Xb = Tx{2};
    Xc = Tx{3};
    Xd = Tx{4};
elseif ch_num ==0
```

在ch\_num=2的时候，Tx{1}的数据从DA2、DA4口发送 🎧 去；Tx{2}的数据从DA1、DA3口发送 🎧 去

```
function Rx = rue_rxDataReceive(ch_num)
%%
% ch_num 1 (AD6 in hardware)
% ch_num 2 (AD8 in hardware)
% ch_num 3 (AD2 in hardware)
% ch_num 4 (AD4 in hardware)
%%
Pid = fopen('/data_bin/rxwire_test.bin','r');
Resig = fread(Pid, 'int16');
fclose(Pid);
Resig = reshape(Resig,4,length(Resig)/4);
Rx = Resig(ch_num,:);
Rx = Rx';
```

Ch\_num=1 的时候从 AD6 口接收数据；Ch\_num=2 的时候从 AD8 口接收数据；Ch\_num=3 的时候从 AD2 口接收数据；Ch\_num=4 的时候从 AD4 口接收数据；

