Hello Leo,

Monday is fine for discussion.

There is no material links to the problem. The best way to think

about it is from a security standpoint. Someone (a malicious

character) wishes to hack a system and figures an approach would be to

embed malicious code in the radio astronomy recorded data from a a

single dish. It could be M5B formatted data, or VDIF formatted data.

Being the recorded data is Gaussian white noise, is it possible to

determine the embedded (non-Gaussian) data within a frame or multiple

frames and how can you detect it.

The key is to verify if data is bad, you would have to correlate the

data and get a report back that bad frames existed, is there another

way and then?

1) can an algorithm be determined,

2) how long would it take to determine it from the recorded channels

w/o running it thru correlation (so how long algorithmically would it

take per scan, the order)

There is no data that exists to even start with or memos.

Kind regards, Chet.

=======================

Hi Chet,

(1) I have created a demonstration script to show the method to test the

hypothesis

"With confidence 95%, the data set is sampled from a normally

distributed data source".

This is a standard Pearson chi^2 test, and the confidence level can be

varied. The data set numbers are binned to form a distribution to be fit

to the standard normal PDF. The total deviation from standard is

assessed by еру chi^2 criterion.

The script generates an array of normally distributed float numbers

(say, N=10000), tests it for normality, and saves its plot with testing

result in a multipage pdf file. This is done in a loop with randomly

chosen means and std-s in each iteration. Here I attach several pdf

files with the results, both positive and negative, for the three bin

sizes, 0.5, 0.25, and 0.125. I also attach a couple of svg files with

individual plots.

(2) To continue the work, I need metadata for the m5b file you gave me.

I cannot read it correctly with the python baseband package without

knowing exact date, channel number etc. of the m5b file.

I need access to many M5B or/and VDIF files to make fast normality

testing on multi-core graphic processors (GPU).

Have a nice weekend!

Leo.

==============================

Hi Chet,  
  
As we discussed earlier, I read an m5b file in 2504-word frames, 4-word header + 2500-word data array, np.array(2500, dtype=uint32). For example, from each uint32 word I extract two-bit numbers, say, the first two bits for the XX chan. 0. It forms the 2-bit signal stream, which I need to check for the normal distribution.  
  
(Question 1:) With only 2 thresholds (say, -V and V) of quantization, how can I test for the Gaussianity of the data? Well, I can assume two levels around the zero, -0.5V and +0.5V. But what level can I assume for the intervals (-Inf .. -V) and (V .. +Inf) ? There must be some specialized methods of which I do not know. Can you suggest some literature, or, maybe, people who could know?  
  
This is a histogram for frame 0, chan 0, horiz. pol, from file

rd1910\_wz\_268-1811.m5b. I voluntarily assigned levels as:

0b00 -> -1.5, 0b01 -> -0.5, 0b10 -> 0.5, 0b11 -> 1.5.

But I doubt it is a correct way.  
  
  
  
  
(Question 2:) What is the meaning of the four possible states, 00, 01, 10, and 11?  
  
While writing this email, I found the answer in the documents. Please, just say, should I follow the tables below?  
  
I tried to find a description of the signal coding ("quantization") in the literature on Mark-5B. I found a document on Mark-5C:  
  
Mark 5C Data-Frame Specification  
<https://www.haystack.mit.edu/wp-content/uploads/2020/07/memo_mark-5_058.pdf>  
  
In Section 4. Mark 5B-Emulation Data Frame Format, at page 5 I found  Table 13: Sample codes for Mark 5B emulation mode:  
  
  
  
Just now, in the document "Mark 5B design specifications",  
  
I found the table:  
  
  
So I know now that the threshold is 220 mV.  
  
  
(Question 3:) By the way, they write: "The Mark 5B fill-pattern word is (0x11223344)". Probably, you made a mistake giving me the fill-pattern 0x00112233. Is it the case? For me it is important -- not to accidentally treat it as valid data.  
  
Best,  
  
Leo.