HasKAL Reference Manual

Edition 0.1 alpha

March 17, 2016

Contents

1 Monitor Tools

1.1 RayleighMonitor

1.1.1 Introduction

RayleighMonitor is a tool for calculated a quantile value of normalized spectrum of $x(t)$. The deviation of the calculated quantiles from the expected one in Gaussian noise case shows deviation of the detector noise from Gaussian distribution.

Normalized spectrogram, $w(t, f)$, of input signal, $x(t)$, is calculated

$$w(t_i,\ f_j) = \frac{|\ \mathrm{STFT}[x(t)]\ |}{S_0(f)},$$

where $1 \leq i \leq N,\ \ 1 \leq j \leq M$ and $S_0(f)$ is a normalization factor. Normalization factor can be estimated

$$S_0(f) = |\ \mathrm{FFT}[x(t)]\ |.$$

P-quantile value of input signal is calculated from normalized spectrogram as the function of time and frequency, $Q(P;\ f_l)$ where $1 \leq l \leq M/m$, $m(l-1) - 1 \leq j \leq ml$ and $m = \mathrm{d}f/\mathrm{d}f_{\mathrm{fft}} = \mathrm{d}f\ \mathrm{d}t_{\mathrm{fft}}$

1.1.2 Function: rayleighMonWaveData

```
rayleighMonWaveData p secfft df x0 xt
```

This function compute p-quantile value, $Q(p;\ f)$, of the input signal, $x(t)$, as the function of frequency, f. The arguments are:

- `p`: Input. The list of dimensionless p-values ($0 \leq p \leq 1$).
- `secfft`: Input. The data length for short time Fourier transform in seconds.
- `df`: Input. The frequency resolution of $Q(p;\ f)$ in Hertz
- `x0`: Input. The time series signal for estimating averaged spectrum
- `xt`: Input. The time series for calculating quantile value $Q(p;\ f)$
- `q`: Output. The quantile value of input signal $Q(p;\ f)$.

1.1.3 Example: rayleighMon

This program calculates the $Q(p;\ f)$ of the input signal.

Typical usage: `rayleighMon param.conf file.lst`

```
import Data.Maybe (catMaybes)
import System.Environment (getArgs)

import HasKAL.DetectorUtils.Detector (Detector(..))
import HasKAL.FrameUtils.Function (readFrameWaveData')
import HasKAL.Misc.ConfFile (readFileList, readConfFile)
import HasKAL.MonitorUtils.RayleighMon.RayleighMon (rayleighMonWaveData)
import HasKAL.PlotUtils.HROOT.PlotGraph
import HasKAL.WaveUtils.Data (WaveData(..))
import HasKAL.WaveUtils.Function (catWaveData)

main = do
  {-- arg check --}
  args <- getArgs
  (conf, lst) <- case length args of
                  2 -> return (args!!0, args!!1)
                  _ -> error Usage: rayleighMon conffile filelist"
```

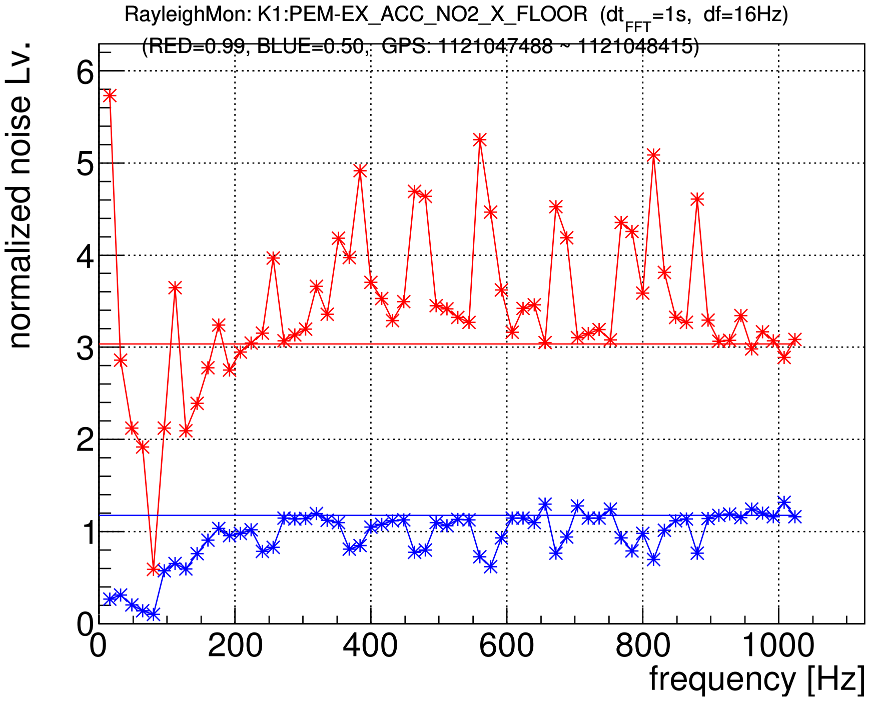

Figure 1: sample plot of rayleighMonitor

```
{-- read param --}
filelist <- readFileList lst
([ch, dtfft, df], [qs]) <- readConfFile conf ["channel", "dtfft", "df"] ["quantile"]

{-- read data --}
mbWd <- mapM (readFrameWaveData’ KAGRA ch) filelist
let wd = case catMaybes mbWd of
          [] -> error "Can’t find file."
          xs -> catWaveData xs

{-- main --}
let result = rayleighMonWaveData (map read qs) (read dtfft) (read df) wd wd
    lineType = concat $ replicate (length qs) [LinePoint, Line]
    colors = concatMap (replicate 2) [RED, BLUE, PINK, GREEN, CYAN, YELLOW, BLACK]
    title = ch ++ ": " ++ (show . fst . startGPSTime $ wd) ++ " ~ " ++ (show . fst . stopGPSTime $ wd)
oPlotV Linear lineType 1 colors ("frequency [Hz]", "normalized noise Lv.") 0.05
title "X11" ((0,0),(0,0)) $ concatMap (\(x,y) -> [x,y]) result
```

Param file format: `param.conf`

```
channel: X1:HOGE-XX   # channel name
quantile: 0.5 0.95    # list of dimensionless p-value
dtfft: 1              # data length for STFT in seconds
df: 16                # frequency resolution of Q(p;~f) in Hertz
```

List file format: `file.lst`

```
/path/to/framefile/a.gwf
/path/to/framefile/b.gwf
```

contact person: Takahiro Yamamoto (`yamamoto@yukimura.hep.osaka-cu.ac.jp`)

1.2 StudentRayleighMonitor

1.2.1 Introduction

StudentRayleighMonitor is a tool for investigating stationarity non-Gaussianity of input signal $x(t)$ by assuming detector noise distributed the Student-t distribution. In this assumption, non-Gaussianity is represented by only one parameter, ν, which shows weight of tail of the distribution. Non-Gaussianity, ν, is computed as the function of time, t, and frequency, f, from normalized spectrum of $x(t)$.

Normalized spectrogram, $w(t, f)$, of input signal, $x(t)$, is calculated

$$w(t_i,\ f_j) = \frac{\mid \mathrm{STFT}[x(t)] \mid}{S_0(f)},$$

where $1 \le i \le N,\ \ 1 \le j \le M$ and $S_0(f)$ is a normalization factor. Normalization factor can be estimated

$$S_0(f) = \mid \mathrm{FFT}[x(t)] \mid.$$

P-quantile value of input signal is calculated from normalized spectrogram as the function of time and frequency, $Q_P(t_k, f_l)$ where $1 \le k \le N/n$, $1 \le l \le M/m$, $n(k-1)+1 \le i \le nk$, $m(l-1)-1 \le j \le ml$, $n = \mathrm{d}t/\mathrm{d}t_{\mathrm{fft}}$ and $m = \mathrm{d}f/\mathrm{d}f_{\mathrm{fft}} = \mathrm{d}f\ \mathrm{d}t_{\mathrm{fft}}$

On the other hand, theoretical quantile value in the Student-t noise case can be described

$$Q_{\mathrm{sr}}(\sigma, \nu; P) = \sigma \sqrt{\frac{\nu(1-(1-P)^{2/\nu})}{(1-P)^{2/\nu}}}$$

Degree of non-Gaussianity ν is calculated from P-quantile value of data and theoretical quantile value.

$$\nu(t_k, f_l) = \arg\min_{\nu} |Q_{P=P_0}(t_k, f_l) - Q_{\mathrm{sr}}(\sigma, \nu; P = P_0)|$$

1.2.2 Function: studentRayleighMonWaveData

```
studentRayleighMonWaveData p secfft chunck dt df x0 xt
```

This function compute the non-Gaussianity, ν, of the input signal, $x(t)$, as the function of time, t, and frequency, f. The arguments are:

- `p`: Input. The dimensionless p-value ($0 \le p \le 1$).
- `secfft`: Input. The data length for short time Fourier transform in seconds.
- `chunck`: Input. The data length for estimating $\nu(f)$ in seconds. ($\texttt{secfft} \le \texttt{chunck}$)
- `dt`: Input. The time resolution of $\nu(t,\ f)$ in seconds.
- `df`: Input. The frequency resolution of $\nu(t, f)$ in Hertz
- `x0`: Input. The time series signal for estimating averaged spectrum
- `xt`: Input. The time series for estimating $\nu(t,\ f)$
- `nu`: Output. The dimensionless non-Gaussian parameter $\nu(t,\ f)$.

1.2.3 Example: studentRayleighMon

This program calculates the $\nu(t, f)$ of the input signals.

Typical usage: `studentRayleighMon param.conf file.lst`

```
import Data.Maybe (catMaybes)
import System.Environment (getArgs)

import HasKAL.DetectorUtils.Detector (Detector(..))
import HasKAL.FrameUtils.Function (readFrameWaveData’)
import HasKAL.Misc.ConfFile (readFileList, readConfFile)
import HasKAL.MonitorUtils.SRMon.StudentRayleighMon (studentRayleighMonWaveData)
import HasKAL.PlotUtils.HROOT.PlotGraph3D
import HasKAL.WaveUtils.Data (WaveData(..))
import HasKAL.WaveUtils.Function (catWaveData)

main = do
  {-- arg check --}
  args <- getArgs
  (conf, lst) <- case length args of
                  2 -> return (args!!0, args!!1)
                  _ -> error "Usage: rayleighMon conffile filelist"

  {-- read param --}
  filelist <- readFileList lst
  ([ch, q, dtfft, dt, lap, df], _) <- readConfFile conf ["channel", "quantile", "dtfft"
                                                         , "dt", "overlap", ‘‘df"] []

  {-- read data --}
  mbWd <- mapM (readFrameWaveData’ KAGRA ch) filelist
  let wd = case catMaybes mbWd of
            [] -> error "Can’t find data"
            xs -> catWaveData xs

  {-- main --}
  let result = studentRayleighMonWaveData (read q) (read dtfft)
                 (read dt) (read dt - read lap) (read df) wd wd
      title = ch ++ ": " ++ (show . fst . startGPSTime $ wd)
                ++ " ˜ " ++ (show . fst . stopGPSTime $ wd)
  histgram2dM Linear COLZ ("time [s]", "frequency [Hz]", "nu")
     title "X11" ((0,0),(0,0)) $ result
```

Param file format: `param.conf`

```
channel: X1:HOGE-XX  # channel name
quantile: 0.95       # dimensionless p-value
dtfft: 1             # data length for STFT in seconds
dt: 128              # time resolution of \nu(t,f) in seconds
overlap: 124         # data overlap in seconds
df: 16               # frequency resolution of \nu(t,f) in Hertz
```

List file format: `file.lst`

```
/path/to/framefile/a.gwf
/path/to/framefile/b.gwf
```

contact person: Takahiro Yamamoto (`yamamoto@yukimura.hep.osaka-cu.ac.jp`)

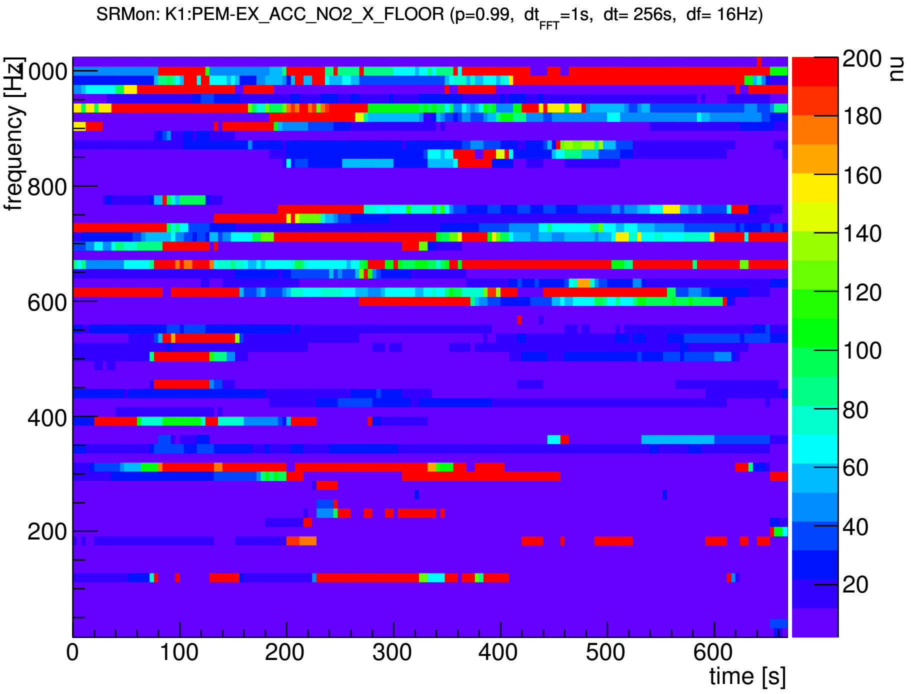

Figure 2: sample plot of studentRayleighMonitor

1.3 RMSMonitor

1.3.1 Introduction

The RMSMonitor is a tool to investigate the non-stationary behavior of the input time series $x(t)$ by calculating the band-limited root-mean square(for short RMS) values.

The RMS values, $\rho_{RMS}(t)$, of input time series, $x(t)$, are calculated as

$$\rho_{RMS}(t) = \sqrt{\int_{f_1}^{f_2} |\tilde{x}(f)|^2 df} \tag{1}$$

where f_1 and f_2 are the frequency band and $\tilde{x}(f)$ is the input frequency domain signal calculated by FFT as,

$$\tilde{x}(f) = \mathrm{FFT}[x(t)].$$

1.3.2 Function: rmsMon

```
rmsMon nmon fs ys freq
```

This function compute the RMS, $\rho_{RMS}(t)$, of the input time series $x(t)$. The time series $x(t)$ are divided into small chunks. The RMS value is calculated from each chunk. The number of chunks are calculated by $N_{ys}/$`nmon`, where N_{ys} is the number of samples of input time series.
The arguments are:

- `nmon`: [Input] The number of samples in one chunk.
- `fs`: [Input] The sampling frequency of input time series.
- `ys`: [Input] The input time series.
- `freq`: [Input] The frequency bands $[f_1 : f_2]$ described in Eq. (1).
- `RMS`: [Output] The calculated RMS values.

1.3.3 Function: rmsMonWaveData

`rmsMonWaveData nmon freq wd`

This function compute the RMS, $\rho_{RMS}(t)$, of the input time series $x(t)$. The difference between `rmsMon` and `rmsMonWaveData` is the type of input time series. `rmsMonWaveData` uses WaveData type instead of the time series $x(t)$. The other arguments are same as `rmsMon`.
The arguments are:

- `nmon`: [Input] The number of samples in one chunk.
- `wd`: [Input] The input data (`WaveData` type).
- `freq`: [Input] The frequency bands $[f_1 : f_2]$ described in Eq. (1).
- `RMS`: [Output] The calculated RMS values.

1.3.4 Example: rmsMon

This program calculates the ρ_{RMS} of the input time series. Examples of RMSMon are described in Fig. 3 and 4.

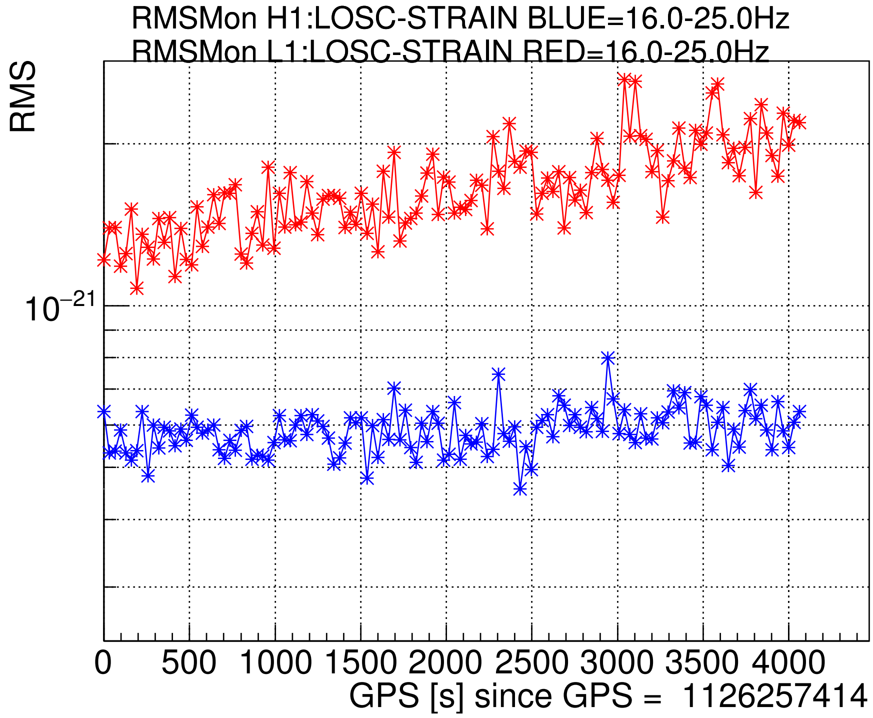

Figure 3: Sample plot of RMSMon. The duration of one chunk is 32s. The frequency bands is [16:25]Hz

contact person: Hirotaka Yuzurihara (`yuzurihara@yukimura.hep.osaka-cu.ac.jp`)

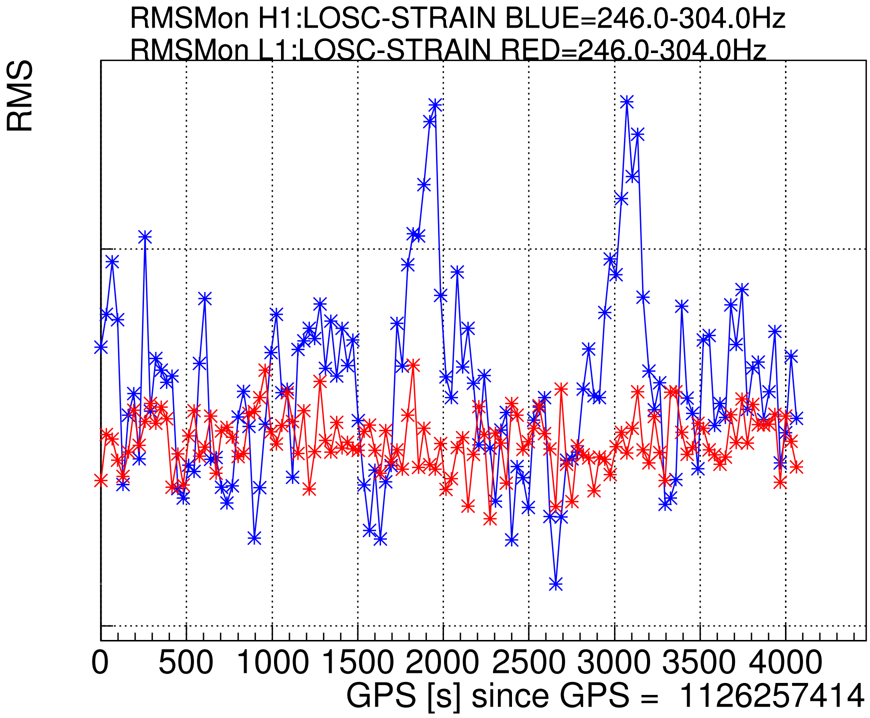

Figure 4: Sample plot of RMSMon. The duration of one chunk is 32s. The frequency bands is [246:304]Hz

1.4 LineTrackingMonitor

1.4.1 Introduction

LineTrackingMonitor is a tool to monitor various lines included in the strain data. The amplitude and frequency are tracked.

1.4.2 Function: butterBandPass

```
butterBandPass dat fs flow fhigh order
```

This applies a Butterworth bandpass filter of given cutoff frequencies and filter order to given data.

The arguments are:

- `dat`: Input. data
- `fs`: Input. Sampling frequency [Hz]
- `flow`: Input. Lower cutoff frequency [Hz]
- `fhigh`: Input. Higher cutoff frequency [Hz]
- `order`: Input. Filter order

It should be noted that this filter is a zero-phase filter (so called filtfilt), so the effective filter order is twice your input value (For example, if you set order = 5, the actual filter order becomes ten).

1.4.3 Function: nha

```
nha dat fs nsig nframe nshift nstart nend tlength
```

The arguments are:

- `dat`: Input.
- `fs`: Input. Sampling frequency [Hz]
- `nsig`: Input. The number of signals (lines) to extract
- `nframe`: Input. Frame length
- `nshift`: Input. Shift length
- `nstart`: Input. The start point
- `nend`: Input. The end point
- `tlength`: Input. Time length of data

1.4.4 Sample plots of LineTrackingMonitor

contact person: Koh Ueno (`ueno@gwv.hep.osaka-cu.ac.jp`)

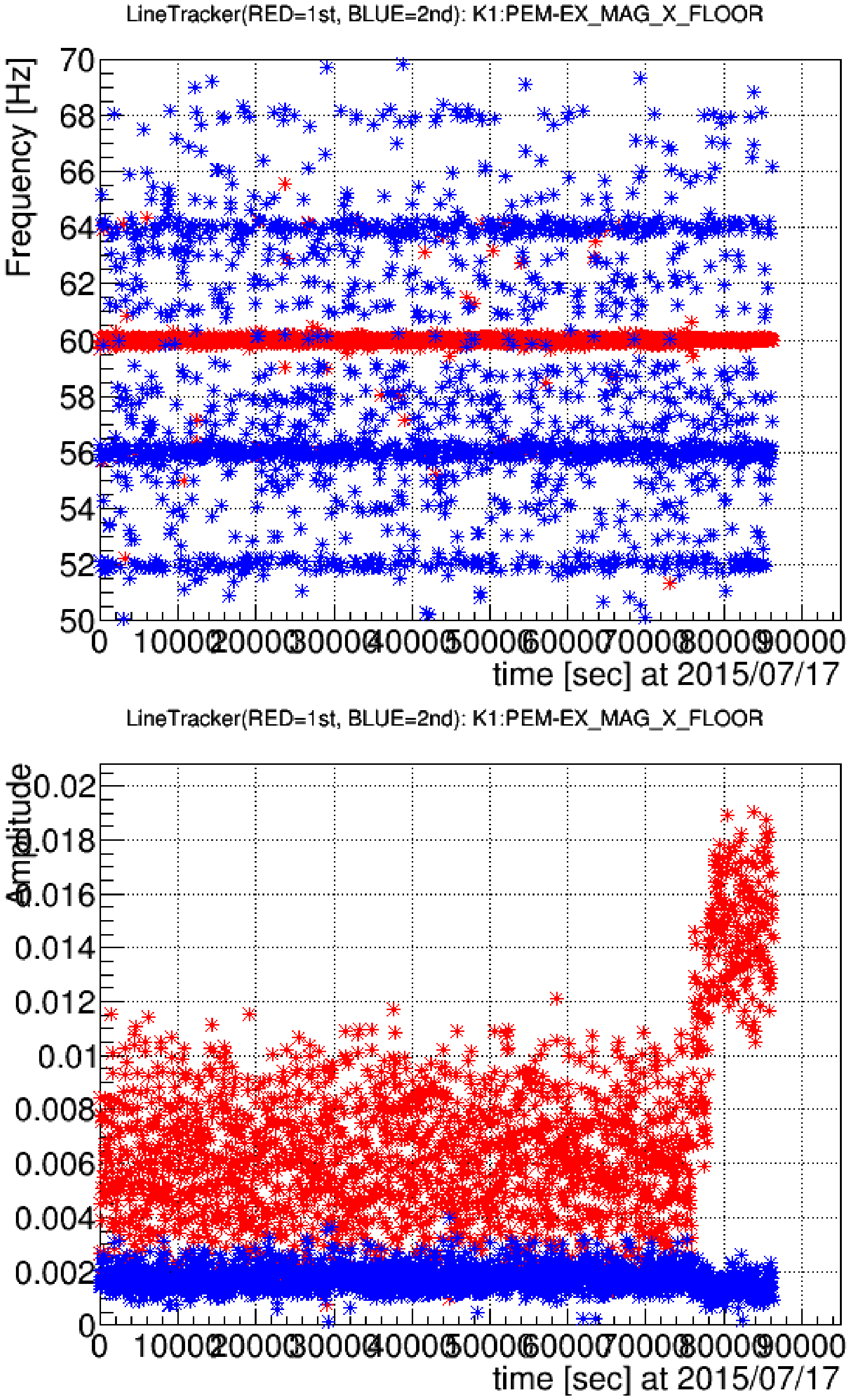

Figure 5: sample plots of frequency (top) and amplitude (bottom) of LineTrackingMonitor