LASER INTERFEROMETER GRAVITATION WAVE OBSERVATORY
-LIGO-
CALIFORNIA INSTITUTE OF TECHNOLOGY
MASSACHUSETTS INSTITUTE OF TECHNOLOGY

Technical Report LIGO-T010125-01-D
LineMonitor
S.Klimenko, G.Mitselmakher, A.Sazonov

Distribution of this draft:

This is an internal working note
of the LIGO Project

LIGO Hanford Observatory
P.O. Box 1970;Mail Stop S9-02
Richland, WA 99352
Phone (509) 372-2325
Fax (509) 372-2178
E-mail: info@ligo.caltech.edu

LIGO Livingston Observatory
P.O. Box 1970;Mail Stop S9-02
Livingston, LA 70754
Phone (225)686-3100
Fax (225) 686-7189
E-mail: info@ligo.caltech.edu

California Institute of Technology
LIGO Project – MS 51-33
Pasadena, CA 91125
Phone (626) 395-2129
Fax (626) 304-9834
E-mail: info@ligo.caltech.edu

Massachusetts Institute of Technology
LIGO Project – MS 20B-145
Cambridge, MA 01239
Phone (617) 253-4824
Fax (617) 253-7014
E-mail: info@ligo.mit.edu

WWW.http://www.ligo.caltech.edu

Abstract

In this note we describe a design, implementation and usage of the DMT Line Monitor. The LineMonitor can be used for real-time monitoring of narrow resonances in LIGO data or can be run off-line on frame data.

1 Introduction

Narrow resonances in the LIGO detector are one of the noise sources that may affect the detection of gravitation waves. Thus, a detailed study of this noise (or line noise) is important for understanding of the LIGO data. It's also essential for the commissioning and operation of the LIGO detectors.

The narrow resonances can be roughly divided into two groups: the mechanical resonances (MR) and the environmental resonances (ER). The mechanical resonances are intrinsic for the LIGO interferometers and associated with the internal resonances of the optics, the optics suspension (pendulum and violin modes) and the mount and support structure of the optics. A list of expected mechanical resonances in the LIGO interferometers can be found elsewhere [1]. The environmental resonances are not intrinsic for the LIGO interferometers, but enter through electrical equipment connected to the instrument or through the optics suspension system. They could be resonances generated by pumps, motors or air handling equipment. A particular example of the ER is the AC power interference (60Hz and its harmonics).

To monitor the parameters of narrow resonances we have developed a DMT monitor, which

2 Algorithm

2.1 *Line Interference Signal*

We consider a resonance to be narrow if its width is much less compare to the fundamental frequency f_0. Then in time domain the resonance can be described with a monochromatic harmonic signal. Given a data segment of length T, we assume that the harmonic signal parameters (amplitude and frequency) do not change much during the time interval T. It means that the line width should be much less the $1/T$. Then the line interference signal, which is a sum of signals produced by the line and its harmonics, can be characterized as follows

$$I(t) = \sum_n a_n \cdot \cos(\Psi_n(t)) = \sum_n a_n \cdot \cos(2\pi n f_0 t + \phi_n) \quad \textbf{(2.1)}$$

where a_n is the harmonic amplitude, $\Psi_n(t)$ is the harmonic's phase approximated with a linear function. We assume that all harmonics have the same fundamental frequency and they are characterized by an arbitrary phase shift ϕ_n. Given a consecutive set of data segments, the LineMonitor finds the parameters of the line interference signal for each segment and produces a trend data for f_0, a_n and ϕ_n.

2.2 Quasi-Monochromatic Line Removal

To estimate the parameters of the line interference signal we use the Quasi-Monochromatic

Line Removal (QMLR) algorithm. It uses the advantage of the basis of orthogonal Fourier functions used for discrete Fourier transform

$$F_n(m) = \exp\left(-i\frac{2\pi nm}{N}\right),\ n,m = 0,...,N-1,$$

where N is the number of samples in the data segment. For a sampled harmonic signal

$$L(t_m) = L(m/f_s) = \exp\left(-i\frac{2\pi m f_0}{f_s}\right)$$

where f_s is the sampling rate and f_0 is the signal frequency, $L(m)$ is one of the basis Fourier functions if $f/f_s = n/N$. Then if narrow lines with fundamental frequency f_0 are present in the data, we use the following algorithm to estimate *I(t)* :

1. Resample data at new sampling rate $\widetilde{f}_s$, which satisfies $\widetilde{f}_s / f_0 = \mathrm{int}(f_s / f_0) + 1$.
2. Select data segment with number of samples $N = k \cdot \widetilde{f}_s / f_0$, where k is an integer number.
3. Do Fourier transform of the data. Then select the line harmonics by applying a comb filter

   $$h(f \neq nf_0) = 0,\ h(f = nf_0) = h_n$$

   where nf_0 is the frequency of the n[th] harmonic. The optimal filter coefficients h_n can be estimated as

   $$h_n = (S_n - N_n)/S_n$$

   where S_n is the spectral density at the harmonics frequency and N_n is the noise spectral density in the vicinity of the harmonic. Usually the coefficients h_n are very close to unity. The complex Fourier coefficients give the harmonics amplitudes a_n and phases ϕ_n.
4. The *I(t)* signal can be reconstructed as the inverse Fourier transform of filtered data.

Since the resampled data contains the integer number of the fundamental line cycles, we can reconstruct

2.3 Fundamental Frequency

To reconstruct the interference signal with the algorithm described above, an accurate measurement of the fundamental frequency f_0 is required. The discrete Fourier transform gives a rough estimate of the line frequency with resolution ~*1/T*. To measure the line frequency with better precision we use the following algorithm. First, we resample date for some seed value of f_0 (specified by input parameter) and find the amplitudes a_n. Then we vary the fundamental frequency to maximize the intensity of the interference signal $E = \sum a_n^2$. Figure 1 shows E as a function of the fundamental frequency variation for the Caltech 40m interferometer. With this method the line frequency can be measured with the accuracy of 0.1-10 mHz depending on intensity of the line.

The measured value of frequency is used then on the next data segment as a new seed value of the frequency. If the line is too weak, so the ratio of the line and noise spectral densities is less the some limit, the measured value of frequency is not used as a new seed value.

3 LineMonitor description

3.1 Design

The LineMonitor module has a standard for the DMT monitors design. Along with other standard objects, it has a list of LineFilter objects, one per line, which are used to monitor the line parameters.

3.1.1 LineFilter

The QMLR algorithm is implemented in the LineFilter class, which is a part of the DMT signal processing software. It has the following member functions:

1. LineFilter::LineFilter(
   - double **F,[60]** - approximate (seed) line frequency (Hz). If a negative value of seed frequency is specified, the LineFilter will not remove the line interference signal from the data.
   - double **T,[1]**- time stride in seconds to estimate the line parameters
   - int **id,[1]** – optimal filter ID (0/1).
   - int **nT,[1]** - number of subdivisions of the time stride w. Subdivisions are used to monitor weak and wide lines. The line width should be less then **nT/T**.

)

   examples:
   - LineFilter F(60.,64,1,8) // - create LineFilter object with the base frequency 60Hz. Use filter 1. Process data in segments of 64 sec long with 8 subdivisions. It means that the LineFilter will estimate the line parameters for each sub-segment (8sec long) and then average all 8 measurements.

2. void LineFilter::setFilter(
   - int **nF** - first harmonic, **[1]**.
   - int **nL** - last harmonic, **[0]**. If nL=0, all harmonics are processed.
   - int **nS** - skip harmonics, **[1]**. Process harmonics nF, nF+abs(nS), nF+2*abs(nS), etc. If nS is negative, set the fundamental frequency to be f/abs(nS) and set the first harmonic to be nFirst*abs(nS). It is used for processing of lines with frequencies greater then $F_s/4$, where F_s is a data sampling rate.
   - int **nLPF** - number of decimation steps (by 2) of input time series, **[1]**.If nLPF<0-do up-sampling of input time series by 2 to estimate the interference.
   - int **nBand** - number of frequency bins in the vicinity of the line to estimate the noise spectral density, **[5]**.

)

   examples:
   - F.setFilter(1,10); //- select harmonics 1-10.
   - F.setFilter(1,0,2); //- select odd harmonics .
   - F.setFilter(2,20,2,1); //- select even harmonics 2-20 and decimate the input time series by 2 before processing.

3. void LineFilter::setFScan(
   - double **F** - update seed frequency with value of abs(F), **[0]**. If F<=0 - don't scan (measure) frequency. If F=0 - don't update and

don't scan the base frequency. [0]

- double **SNR, [2.]** - limit on signal to noise ratio, which is calculated as ratio of the line spectral density and noise spectral density. If this ratio is less then SNR, then filter doesn't use the measured value of frequency from this run as a seed frequency for the next run. Also, in this case, the negative value of estimated base frequency is saved in the database.
- double **fBand, [0.45]** - frequency band in units of **nT/T** to scan the base frequency.
- int **nScan, [1]** - limit on number of iteration steps during the frequency scan

)

examples:

- F.setFScan(); // - don't measure frequency
- F.setFScan(60.1,3.) // - update seed frequency to be 60.1 Hz and set signal to noise ratio to be 3.

4. void apply(const Tseries& ts)
   Apply the QMLR algorithm to time series ***ts*** and store line parameters in the local database. If requested, the ***apply***() method will remove the line interference signal from the input time series.

5. double fScan(const Tseries& ts)
   Measure and return the line fundamental frequency.

6. double Interference(Tseries & ts)
   Estimate the interference signal for input time series ts. Replace ts with the estimated line interference signal and return its total intensity.

3.2 Input data

The LineMonitor has the following input data:

- Time series for specified interferometer channel.
- Input parameters specified in the command line.
- Input parameters specified in the configuration file

3.2.1 Input time series

Currently the LineMonitor doesn't support multi-channel operation. Time series from only one interferometer channel are processed by a single LineMonitor task. But several LineMonitor tasks can be launched to monitor different channels.

3.2.2 Parameters

The input parameters are used to configure the LineMonitor for a specific monitoring task. The parameters can be entered in the Unix command line or in the configuration file (see next section).

-i <input config file name>:
The LM configuration file with the LineMonitor input parameters. The configuration file should be used if run the LineMonitor to track several lines or if run by the DMT process manager.

-c <channel name>:
The channel name is mandatory. The LineMonitor can be run on one channel only. However several LineMonitor tasks can be launched to monitor different channels.

-infile <input frame file name>:
If **-infile** option is specified, the LineMonitor will read data from the input frame file rather then from the onLine buffer.
<< "
-l <lock condition>:
Track lines if some lock condition is specified. Only one lock condition can be used.
For example:
-l H2:Both_arms_locked
-l H1:Mode_cleaner_locked
-l L1:X_arm_locked

-H <file name to dump html table>:
Specify file name to dump the html summary. The output directory should be specified in the global variable DMTHTMLOUT. If DMTHTMLOUT is not defined, then the local directory is used to store the html output file.

-f <seed frequency (Hz)> [default=60.]:
To track a line, the LineMonitor should be given its approximate fundamental frequency. Typically the seed frequency uncertainty should be less than **n/t**, where **t** is a time stride and **n** is the number of stride subdivisions (input parameter of the LineFilter::LineFilter() function in section 3.1.1).

-t <time stride (sec)> [default=1]:
The time series length requested by the LineMonitor. It sets the sampling rate of the trend data (input parameter of the LineFilter::LineFilter() function in section 3.1.1).

-n <number of stride subdivisions> [default=1]:
This parameter is needed to track weak and wide lines. In this case the time stride **t** (or the measurement integrated time) should be long enough to distinguish the lines from the noise. The long data segments should be divided on sub-segments to keep **n/t** greater then the line width (n is input parameter of the LineFilter::LineFilter() function in section 3.1.1).

-I <filter ID (0/1)> [default=1]:
The LineFilter ID. 0 - build the LineMonitor filter ignoring the noise around the line, 1 - find the LM filter with the noise taken into account.

-s <no frequency scan> [default=scan]:
Do not perform frequency scan for lines with exactly known frequency (for example, calibration lines). No value should be entered.

-F <first harmonic index> [default=1]:
Specify the first harmonic index.

-L <last harmonic index> [default=0]:
Specify last harmonic index. If L=0, track all harmonics available for given data sampling rate.

-S <step to skip harmonics> [default=1]:

Along with **-F** and **-L** options the **-S** option allows select a subset of harmonics.
For example,

```
-F 1 -S 1   // select odd harmonics
```

-R <limit on signal to noise ratio (SNR)> [default=2.]:
The lines may not be strong enough to be monitored by the LineMonitor. The LineMonitor calculates the ratio of the line and noise spectral densities and compare it with the SNR. If the ratio is greater then SNR, the LineMonitor measures and updates the seed frequency and measures the line parameters. If the ratio is less then SNR, the LineMonitor doesn't update the seed frequency, however it places a negative value of measured fundamental frequency in the database.

-d <number of strides> [default=1]:
Dump trend data into file every <number of strides>.

-b <length of the DMTVIEWER buffer> [default=1024]:
This parameter defines the length of time series (in units of time strides) served to the dmtviewer.

-W <number of decimation by 2 steps> [default=-1]:
This option is used to monitor lines with frequency much less then the sampling rate. To increase the LineMonitor performance the data can be down-sampled by the built-in wavelet filter before processing. It is recommended to keep the data sampling rate by at least factor of two greater than the line frequency.

-m <max number of iterations during frequency scan> [default=10]:
Usually there is no need to specify this parameter.

-N <number of frequency bins to estimate the noise SD> [default=5]:
To build the optimal Weiner filter, the detector noise spectral density is estimated in the vicinity of the line. The "vicinity" is defined in units n/T Hz, where T is the stride time and n is the number of stride subdivisions.

-B <bandwidth in units of fft bins> [default=0.45]:
Usually there is no need to specify this parameter.

3.2.3 Configuration file

The configuration file should have the following structure. First the LineMonitor input parameters, like channel name, are specified. Then the configuration parameters of the LineFilter objects should be entered. All options except **-i** and **-infile** can be used to build an entry (a string) in the file. Several parameters can be specified in each string. If string starts with **-f F** option, then a LineFilter entry for a line with frequency **F** is created. There should be only one entry per line. All other LineFilter parameters (options: **-t, -n, -L, -F, -S, -s, -R, -I, -W**) should be specified in the same string. It's recommended to put LineFilter entries in the end of the configuration file

For example,

```
-c L0:PEM-LVEA_V1  // track LLO power monitor  channel
-l L1:X_arm_loced  // when x arm locked
-t 4 -d 10         // stride 4sec, dump trend data every 10 strides
```

```
-f  60. -L 8          // track first 8 harmonics of 60Hz line
-f 272. -L 1 -s       // calibration line at 272Hz , one line, no frequency measurement
-f  35. -t 1          // monitor 35Hz line with 1sec stride
```

Note, the meaning of –t option is different if it is used in the beginning of the configuration file and in the lines starting with –f option. In the first case, the –t option tells the LineMonitor how much data to request from the data server. In the example above the LineMonitor gets data with the stride time of 4 seconds. In the second case the –t option defines what is the data segment length to measure the monitored line parameters. In the example the LineMonitor will do 4 measurements of the 60Hz line for each chunk of data obtained from the server.

3.3 Output data

The LineMonitor has several types of output data:

1. Trend data served to the DMTVIEWER. The fundamental frequencies, signal to noise ratios, amplitudes and phases of all harmonics can be viewed with the DMTVIEWER as a function of time. The length of the data buffer send to the DMTVIEWER can be specified with the –b option.
2. Trend data files. The LineMonitor trend data can be saved in a file. There are two types of trend data files:
    - The DMT trend data frame files. The data served to the DMT viewer can be saved in a file.
    - The LineMonitor data files. The LineMonitor saves the line parameters in files, one line (including harmonics) per file. These data files have more detail information about the lines, like the coefficients of the optimal filter, raw data for frequencies and phase, etc. The line monitor data files can be processed in ROOT and they are for expert use only.
3. The LineMonitor summary in form of html file.
   The html summary has three sections:
    - General information (start time, elapsed time, etc)
    - Summary of monitored line (average amplitude, average frequency, signal to noise ratio, etc. The parameters of monitored lines are served to the DMTVIEWER and they are saved in the trend and LineMonitor data files.
    - Summaru of all detected lines. The LineMonitor looks through the data for lines strong enough to exceed the signal to noise ratio specified with the –R option.

4 Usage and performance

The LineMonitor executable is distributed with the DMT library. If the DMT is installed, the LineMonitor can be executed as follows: LineMonitor <list of parameters>.
If no input parameters are specified, the LineMonitor print out a help message. There are two

5 Conclusion

TBD.

Acknowledgments

TBD

References