Coherent Line Removal

–in LAL C code–

Alicia M. Sintes
sintes@aei-potsdam.mpg.de
Max-Planck-Institut für Gravitationsphysik
Albert-Einstein-Institut
Golm, Germany
http://www.aei-potsdam.mpg.de

Hanford LSC meeting. August 15-17, 2000

LIGO-G000260-00-D

Coherent Line Removal (CLR)

- CLR is an algorithm able to remove interference present in the data while preserving the stochastic detector noise.
- CLR works when the interference is present in many harmonics, as long as they remain coherent with one another.
- CLR can remove the external interference without removing any 'single line' signal buried by the harmonics.
- CLR can be used to remove all harmonics of periodic or broadband signals (e.g., those which change frequency in time).
- CLR requires little a priori knowledge of the signals we want to remove (e.g., no external reference).

The principal of CLR is the following:

We assume that the interference has the form

$$y(t) = \sum_n a_n m(t)^n + (a_n m(t)^n)^*$$

where a_n are complex amplitudes and $m(t)$ is a nearly monochromatic function near a frequency f_0.
The idea is to use the information in the different harmonics of the interference to construct a function $M(t)$ that is as close a replica as possible of $m(t)$ and then construct a function close to $y(t)$ which is subtracted from the output of the system cancelling the interference.

• CLR can be used to remove lines corresponding to the electricity supply frequency and its harmonics.

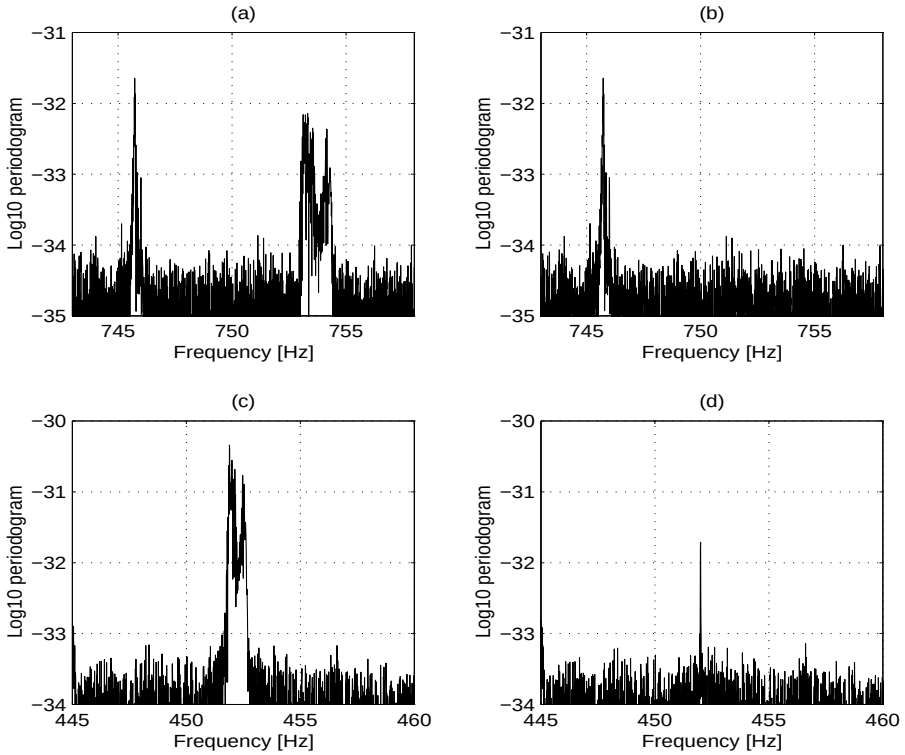

Figure 1: Decimal logarithm of the periodogram of 2^{19} points of the prototype data. (a) One of the harmonics near 754 Hz. (b) The same data after the removal of the interference using CLR. (c) The same experimental data with an artificial signal added at 452 Hz. (d) The data in (c) after the removal of the interference, revealing that the signal remains detectable.

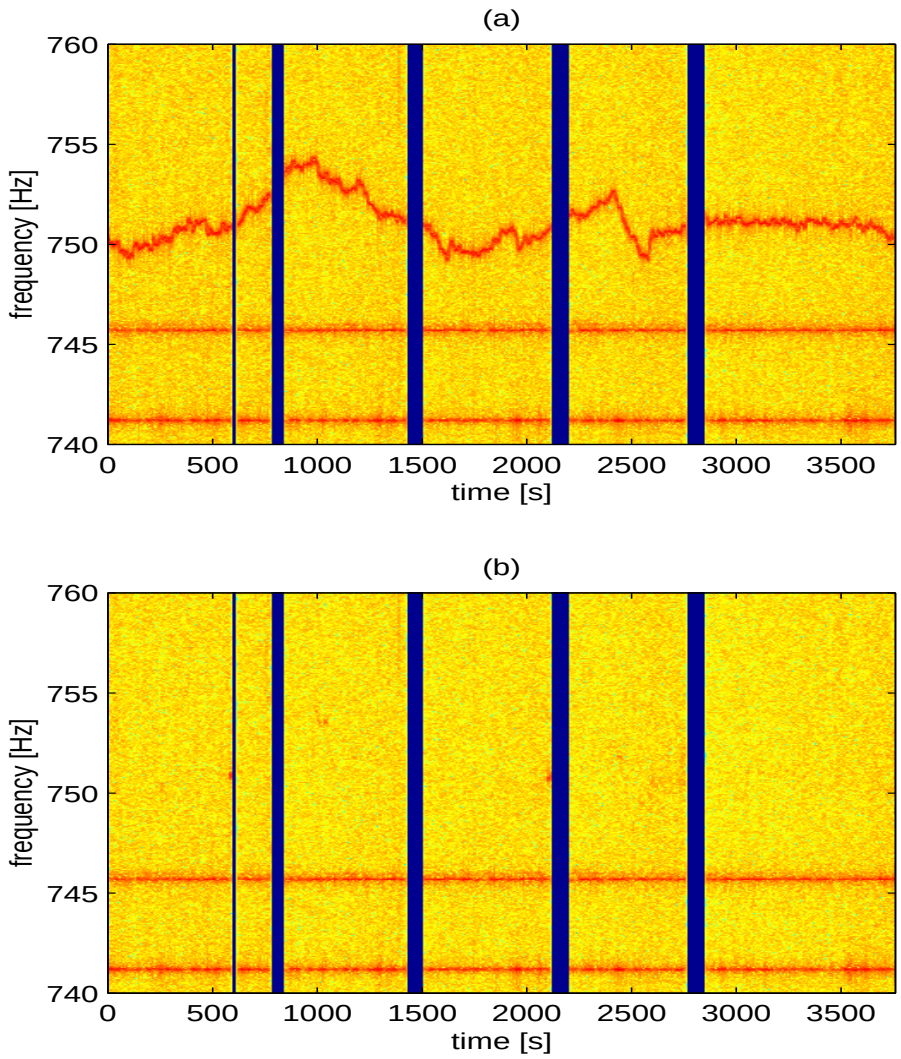

Figure 2: Comparison of a zoom of the spectrogram. (a) is obtained from the prototype data. We can observe the wandering of the incoming electrical signal. The other two remaining lines at constant frequency correspond to violin modes. (b) The same spectrogram as in (a) after applying CLR, showing how the electrical interference is completely removed.

Package cohlineremoval

This package provides routines for finding line harmonics, generating a reference interference signal, and removing all the interference harmonics.

The package is organized under the header CLR.h and the modules HarmonicFinder.c, RefInterference.c and CleanAll.c.

The code is written following the last LAL specifications (LIGO-T990030-08, from 07.18.2000) and also includes the automatic documentation system.

The code was compiled and tested using a previous version (function names without the prefix LAL) i.e., linked with the last release lal-0.4-beta and with the fftw library.

Structures

`struct REAL4TVectorCLR`

It contains time domain data and other information.
The fields are:

`UINT4 length` The number of elements in data.

`REAL4 *data` The (real) time domain data.

`REAL4 deltaT` The sample spacing in time (in seconds).

`REAL4 fLine` The interference fundamental frequency f_0 (in Hz), e.g., 60 Hz.

`struct REAL4FVectorCLR`

It stores the spectrum, $|\tilde{x}(\nu)|^2$, and other information needed to find the location of the line harmonics.
The fields are:

`UINT4 length` The number of elements in data.

`REAL4 *data` The (real) frequency domain data, e.g., $|\tilde{x}(\nu)|^2$.

`REAL4 deltaF` The ΔF offset between samples (in Hz).

`REAL4 fLine` The interference fundamental frequency f_0 (in Hz), e.g., 60 Hz.

HarmonicFinder.c

```
void LALHarmonicFinder (
     LALStatus          *status,
     INT4Vector         *out,
     REAL4FVectorCLR    *in2,
     INT4Vector         *in1 );
```

Given certain harmonic indices $\{k\}$ finds the frequency interval location (in bins) of the interference (around $k \times f_0$) from the power spectrum.

*in1: The harmonic indices

in1->length Number of harmonics.

in1->data List of harmonics to consider, e.g., $\{k\} = \{3, 5, 9, 11 \ldots\}$

*in2: The power spectrum, $|\tilde{x}(\nu)|^2$, f_0 and Δf

in2->length The number of elements in in2->data.

in2->data The spectrum, $|\tilde{x}(\nu)|^2$.

in2->deltaF The ΔF offset between samples (in Hz).

in2->fLine The interference fundamental frequency f_0 (in Hz), e.g., 60 Hz.

*out: a vector whose length is out->length $= 3 \times$ in1->length, and contains for each considered harmonic, in the following order, its index k and the bin location of ν_{ik} and ν_{fk}.

out->length The number of elements in out->data.

out->data $\{k, \nu_{ik}, \nu_{fk}\}$, e.g., {3, 9868, 9894, 5, 16449, 16487, 9, 29607, 29675. . . }.

RefInterference.c

```
void LALRefInterference (
    LALStatus          *status,
    COMPLEX8Vector     *out,
    COMPLEX8Vector     *in1,
    INT4Vector         *par );
```

Generates a reference interference signal.

Given the complex vector *in1 of length $n/2+1$, containing the Fourier transform of the data $\tilde{x}(\nu)$,

in1->length The number of elements in in1->data $= n/2+1$.

in1->data The data $\tilde{x}(\nu)$,

and given another vector *par containing the information related to the harmonics from which we want to construct the reference signal (i.e., indices, and initial and final frequency bin locations),

par->length The number of elements in par->data. This is equal to three times the number of harmonics that will be used to construct the reference signal $M(t)$.

par->data $\{k, \nu_{ik}, \nu_{fk}\}$, e.g., {3, 9868, 9894, 5, 16449, 16487, 9, 29607, 29675. . . },

it generates the time domain $M(t)$ reference interference signal, *out. This is a complex vector of length n.

out->length The number of elements in out->data $= n$.

out->data $M(t)$ complex data.

$M(t)$ corresponds to a nearly monochromatic function near the frequency f_0, that is implicit in the information given in *par.

CleanAll.c

```
void LALCleanAll (
      LALStatus         *status,
      REAL4Vector       *out,
      COMPLEX8Vector    *in2,
      REAL4TVectorCLR   *in1 );
```

Gets data cleaned from line harmonic interference given a time domain reference signal.

`*in1`: the time domain data of type `REAL4TVectorCLR`, containing f_0 and the sampling spacing.

`in1->length` The number of elements in `in1->data` $= n$.

`in1->data` The (real) time domain data, $x(t)$.

`in1->deltaT` The sample spacing in seconds.

`in1->fLine` The interference fundamental frequency f_0, e.g., 60 Hz.

`*in2`: the time domain reference signal (a complex vector).

`in2->length` The number of elements in `in2->data` $= n$.

`in2->data` The $M(t)$ complex data.

`*out`: a real vector containing the clean data.

`out->length` The number of elements in `out->data` $= n$.

`out->data` The clean (real) time domain data.