LASER INTERFEROMETER GRAVITATIONAL WAVE OBSERVATORY

LIGO Laboratory /LIGO Scientific Collaboration

LIGO-T070191-00-R *LIGO* Aug 14th, 2007

Report: Coherence2.exe

John Draskovic

California Institute of Technology
LIGO Project – MS 18-34
1200 E. California Blvd.
Pasadena, CA 91125
Phone (626) 395-2129
Fax (626) 304-9834
E-mail: info@ligo.caltech.edu

Massachusetts Institute of Technology
LIGO Project – NW17-161
175 Albany St
Cambridge, MA 02139
Phone (617) 253-4824
Fax (617) 253-7014
E-mail: info@ligo.mit.edu

LIGO Hanford Observatory
P.O. Box 1970
Mail Stop S9-02
Richland WA 99352
Phone 509-372-8106
Fax 509-372-8137

LIGO Livingston Observatory
P.O. Box 940
Livingston, LA 70754
Phone 225-686-3100
Fax 225-686-7189

http://www.ligo.caltech.edu

This material is based upon work supported by the US National Science Foundation under Cooperative Agreement No. PHY-0107417

CONTENTS

ABSTRACT

As part of the ongoing collaboration between the VIRGO and LIGO projects, undergraduate students from American and European universities are exchanged annually for a Summer-term research program. Assigned to Dr. Francesco Fidecaro, of the Noise Studies Group, I visited the primary VIRGO facility in Cascina, IT for a ten-week period from 28 May to 1 August 2007, participating in the 2nd Annual Virgo-Ego Scientific Forum (VESF) Gravitational Wave School and working with digital signal processing (DSP) software. My software work focused primarily on the implementation of a C-language code for computing coherence data between the dark-fringe channel (the primary data stream from the interferometer) and environmental channels corresponding to various microphones, magnetometers, seismometers, and accelerometers. In addition, I collaborated with another American student, Kiel Howe, on the creation of a data storage and access system for LineMonitor, a C-language code currently in-use at VIRGO. Finally, these two tools were integrated via a web-based php applet, providing researchers with a simple and powerful tool for tracking noise signals.

VESF GRAVITIATIONAL WAVE SCHOOL

The program of the school was a daily series of lectures by professors and students from various international universities, as well as VIRGO researchers and technicians. Content of the lectures fell into three disciplines: gravitational theory and wave sources, in-depth description of the VIRGO interferometer systems, and signal processing techniques for gravitational wave signal identification.

BACKGROUND

I: Overview

VIRGO is an observatory which utilizes a combination of Michaelson-Morley and Fabry-Perot interferometers to detect small (on the order of 10^-20 m) displacements associated with the propagation of gravitational waves. Injection is provided by a small source laser, which drives a larger slave laser to a power of roughly 20 watts. After passing through a beam splitter, the laser light enters two Fabry-Perot cavities oriented at 90 degrees, the combination of which comprises the Michaelson-Morley apparatus. Displacement of the terminal mirrors is measured by the movement of the first dark fringe of the interference pattern registered by photo diodes in the detection bench. For the purpose of energy conservation, as well as enhanced sensitivity, a power recycling mirror is used. The mirror is positioned in front of the injection bench such that, in the event of no signal, the laser light is reflected back into the arm cavities, rather than simply allowed to leave the instrument. Use of the power recycling mirror boosts the laser power in the instrument to the to the order of kilowatts.

II: Noise

Given the large sensitivity necessary to detect the displacements associated with gravitational wave activity, it is inevitable that an interferometric gravitational wave detector will encounter noise in the dark fringe signal. At the most fundamental level, the nature of photon behavior introduces noise due to quantum uncertainty. This “shot noise” is a design consideration which limits the ultimate sensitivity of the detector in the high frequency range. At the low frequency range, the sensitivity of the instrument is limited by a combination of seismic noise and thermal noise in the mirrors and optical benches. The combination of these noise limitations creates a “sensitivity curve” in the detection frequency spectrum-- an effective background against which the information-containing signals may be measured.

While noise is a design consideration regarding sensitivity, the signal contains other unforeseen noises. Manifested as large peaks within the detection spectrum, these can be associated with a variety of sources. A simple example is the peak found at 50 Hz-- the cycling frequency of European AC power. The noise analysis group at VIRGO attempts to study and understand such systematic sources of noise in the observatory. The goal is to catalog as many sources as possible, such that these noises may be removed, either physically or by way of analysis, thereby leaving a signal which may contain gravitational wave information. Fourier analysis provides the tools for this pursuit, implemented by DSP in the way of Fast Fourier Transforms (FFT's). The task is great, as FFT's of VIRGO data produce data in a wide band from 0 to 10 000 Hz, given the photo diode sampling rate of 20 000 Hz. Noise peaks in the detection spectrum may migrate temporally on the order of seconds or minutes, or more slowly, drifting in frequency over the course of months. When viewed in a spectrogram of detector data, these noise peaks create visible lines against the background of the detector.

III: Lines and LineMonitor

LineMonitor is a DSP code currently in-use at VIRGO. Written in C, and utilizing the frame-based data stream from the interferometer, it isolates signal peaks by computing FFT's of the raw data stream, averaging the effective noise background, and identifying peaks found above this background. These peaks, which become lines in the time domain as the code executes sequentially, are stored by the program in frames as well as an on-line database which interfaces with a web-based php applet.

IV: Environmental Monitors

Critical to noise analysis at VIRGO is environmental monitoring. Microphones, seismometers, accelerometers, photo diode instruments, magnetometers, and temperature probes are present throughout the instrument, providing valuable data for finding the sources of noise in the primary signal. Standard sampling rates for these channels are 20 kHz, 10 kHz, 5 kHz, and 1 kHz, depending on the nature of the sensor.

V: Coherence

A tool used by the noise group, coherence is a mathematical concept that measures the linear response between two signals. A commonly used coherence process at VIRGO is written:

$$C\ \ f^{2}=\frac{X\ \ f\ \ Y\ \ f^{\ 2}}{X\ \ f^{\ 2}\,Y\ \ f^{\ 2}}$$ [1]

The convolution of the two signals, squared, is divided by the product of the two power spectra. The coefficient C runs from 0 (which means no linearity between the signals), to 1 (which indicates an exact linear relationship between the signals. Coherence, when computed between the VIRGO environmental channels and the dark fringe signal, allows VIRGO researchers to develop connections between environmental occurrences and noise signals in the interferometer signal.

1 This is the procedure used in `compute.c`, by Gabriele Vajente

DOCUMENTATION OF `coherence2.c`

This is intended as a rough "walkthrough" of the algorithm. For more complete information, consult the source code comments (attached).

I: Main

The main function first establishes a connection with a MYSQL database. As of my last modification of the program, the database was located on the "slvtf" machine and called "test". The table used by the program is called "coherence" and contains 7 fields:

- id : a unique ID is assigned to each point, for internal use only
- gps_start: the start time of the lock containing the processed coherence data
- gps_end: the end time of the lock containing the processed coherence data
- frequency: the frequency of the given point
- value: the value of the coherence coefficient (between 0 and 1)
- channel: a string naming the environmental channel used to compute coherence with the dark fringe
- threshold: the set minimum cutoff of the "value" entry

After connecting, the main function begins the main processing loop, governed by two GPS times given by the user (labeled "USER DEFINED PARAMETERS"). First, the `findnextlock` function is called. This function searches the `trend.ffl` data file for data tagged as "science mode" data. When a lock is found, with a definite beginning and end time, the function returns the time of the end, and sets the values of variables which are needed for the processing functions.

Next, the function searches a text file, named `channels.txt`, which is provided by the user and contains a list of channels. The first entry in the file must be the dark fringe channel selected by the user. Below, any number of environmental channels may be listed.

NOTE: at this point, the environmental channels must match the sampling rate of the dark fringe channel.

The program stores the first entry of the text file as the dark fringe data, and begins a loop through the environmental channels. The end of the `channels.txt` file breaks this loop. For each channel, the process function is called. When all channels are processed for each lock found between the start and end times specified, the main function closes the connection with the database and terminates.

II: findnextlock Function

This function opens the `trend.ffl` file for lock-finding. Using a function from the frame library, it obtains a vector for the search time which contains a flag for each point of data. Next, a loop searches this vector for data above the acceptable value. When a good data point is found, the start time of the lock is recorded. Next, another loop begins, looking for a bad data point. When found, the time is recorded and used to calculate the duration of the lock. The function terminates, returning the end time of the found lock and passing pertinent values back to the main function.

NOTE: Though the function passes the actual length of the lock back to main, it also returns a user-defined length for the FFT to be computed. Currently, it may be set up to 2 hours with stability.

III: process Function

This program performs the necessary FFT's, coherence computations, and database output for the specified FFT length returned by the `findnextlock` function. The function first opens the `raw.ffl` file for interferometer data. It also opens a `.txt` file for debugging output. The text files are automatically named with the channel identification of the the dark fringe channel and the GPS time of the lock beginning. This program processes data buffered by the individual frames. Before the processing loop, the program grabs the first frame of Analog-to-Digital Converter (ADC) data of the specified process time. From this frame data, it extracts the sampling rate of the data, so that memory allocation can be performed. The program allocates three arrays: time series data for the dark-fringe, time series data for the environmental data, and an array to contain the processed coherence data.

The primary loop fills the time series data for the environmental and dark fringe channels. It executes until the time series data is filled, freeing and accessing subsequent frames as necessary. When complete, the time series data arrays are passed to `computeCoherence`, a function found in the code `compute.c`, written by Gabrielle Vajente. This code divides the time series into segments of length specified by the user (must be a power of 2 to optimize execution time). It then computes FFT's of these segments and averages the results, for reasons of speed and cleanup of computational noise. A Hanning window is used. The actual FFT computation is done by a freely-available algorithm called FFTW[2]. Returned from the `computeCoherence` function is an array containing the frequency-domain data from the coherence computation outlined above.

After the primary loop terminates, output is performed by another loop which searches through the coherence array. A simple filter checks each point against the set threshold value. If greater than the threshold, this value is checked against the average of the 20 surrounding points. While rather crude, this filter works to reduce the data output to the database without sacrificing information about coherence peaks.

2 "The Fastest Fourier Transform in the West". Documentation of this code is found at www.fftw.org.

WEB INTERFACE APPLET

Access to the database containing the coherence2.c output is facilitated by a multi-functional web interface, written in php by Kiel Howe[3]. This interface allows for graphing of data and tab-separated text file downloads. The graphing window displays noise peak data computed by LineMonitor. A line-seeking algorithm is used to identify individual noise lines. Once identified, either by the algorithm or user specification, the interface can automatically query the database for correlating entries in the VIRGO Known Lines Database, which is a logbook-style repository for noise lines cataloged by researchers, and the coherence2 database. The user simply specifies a frequency delta about the noise line to query these databases. The advantage of the web interface over other tools available to VIRGO researchers is convenience. With this web applet, a researcher simply specifies a data range of interest and quickly receives information entered by other researchers in the Known Lines Database, as well as a list of channels which exhibit high coherence values near the frequency of the noise line.

3 https://slvtf.virgo.infn.it/vtf/wwwDev/lmTest/stable/lmView.php

REMARKS

`Coherence2.c` was written with a rudimentary knowledge of C programming. In its current state, user-defined parameters must be entered into the source code, and the code rebuilt before use. If this tool is to become frequently utilized by a variety of users, a configuration file must be implemented to streamline input.

Furthermore, the code works only with environmental channels utilizing a 20 kHz sampling rate, matching the sampling rate of the dark fringe channel. To use other environmental channels, a down-sampling process must be added. In a previous (nonfunctional) version of the `coherence2.c` code, a down-sampling function was used, but this proved difficult to implement in the latest code. A code containing a down-sampling function, `xcorr.c`, is located in the `virgoDev/Coherence/Old_codes` directory.

Regarding execution time, it was difficult to find a reliable measure of program speed, due to frequent revisions to the code. On semi-reliable figure emerged with repeatability: with the complete list of 20 kHz channels and an FFT length of 65536 points, the code computed 1 hour of data in about 75 min. Increasing the FFT resolution to 196608 points slowed the execution, but the extent of this was not recorded reliably.

On a personal note, this program was by far the most ambitious code I have attempted in C. It was written without knowledge of Object-Oriented Programming (OOP) techniques, and as a result, may be difficult to follow or modify. I would have liked to rewrite the code using OOP, but chose instead to work on implementing the database and web functionality before my time at VIRGO concluded.