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

| | |
|---|---|
| **Document Type: Technical Report** | **LIGO-T990084-00-E** 9/10/99 |
| **SURF 1999 Report: Use of Higher Order Statistics [HOS] to Detect and to Characterize Non-Gaussian Noise** | |
| Denis Petrovic<br>*University of Belgrade, Belgrade, Yugoslavia*<br>Albert Lazzarini [SURF Mentor] | |

*Distribution of this draft:*

LIGO

This is an internal working note of the
LIGO Laboratory and the
LIGO Scientific Collaboration.

**California Institute of Technology**
**LIGO Project - MS 18-34**
**Pasadena CA 91125**
Phone (626) 395-3064
Fax (626) 304-9834
E-mail: info@ligo.caltech.edu

**Massachusetts Institute of Technology**
**LIGO Project - M/S NW17-161**
**Cambridge, MA 01239**
Phone (617) 253-4824
Fax (617) 253-7014
E-mail: info@ligo.mit.edu

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

# Use of Higher Order Statistics to Detect and Characterize Non-Gaussian Noise

Denis Petrovic
Student, School of Electrical Engineering, University of Belgrade, Yugoslavia

Mentor: Dr. Albert Lazzarini
Member of the Professional Staff in Physics, Caltech, Pasadena, CA, USA

## Abstract

The goal of the Laser Interferometer Gravitational Wave Observatory (LIGO) is the detection and study of gravitational waves from astrophysical sources.
The detection of gravitational waves is limited by a number of noise sources including thermal noise, shot noise, seismic noise and excess instrument noise. The intent of this project is to develop techniques based on Higher Order Statistics (HOS) to identify non-Gaussian instrumental noise in LIGO digitized time series data from the 40m prototype. On the basis of the power spectrum of the original signal, simplified models of a signal are generated in order to understand characteristic patterns of certain non-Gaussian processes present in LIGO 40m data.

The principal task is to develop techniques to identify frequency and phase coherent effects, such as narrowband features and their harmonics; non-linear upconversion processes which produce frequency modulation effects which are not easily or immediately discovered in a simple power spectrum.

## Introduction

The goal of the Laser Interferometer Gravitational Wave Observatory (LIGO) is the detection and study of gravitational waves from astrophysical sources. The detection of the gravitational waves is limited by a number of noise sources [1] including physical noise sources (fundamental limits) and technical noise sources which can be in principle designed out.
In physical noise sources we include seismic noise at frequencies ($f<50$ Hz), shot noise at ($f>1$ KHz) and thermal noise at ($50$ Hz$<f<1$KHz) [2], while in technical noise sources we include 60 Hz power line harmonics and excess instrument noise.
There are two ways of eliminating or at least reducing noise: 1. Choose different design choice for the instrumentation and 2. Apply digital signal processing techniques.
In this SURF project we will concentrate on applying Higher Order Statistics (HOS) measures which contain information that conventional spectral analysis cannot provide. [3]. “Contains”, does not necessarily mean that we are able to extract that information. To

exploit HOS, we must first obtain a complete picture of what HOS measures can provide and then identify signal processing techniques based on HOS in order to filter out noise from the signal.
Motivation for using HOS techniques for characterizing LIGO data is HOS ability to identify non-Gaussian signals by suppressing additive Gaussian noise, extract information due to deviations from Gaussianity, detect and characterize non linear properties in a signal [3].

Method of investigation:
1. Use the power spectrum to identify features that are present in the power spectrum of LIGO data
2. Generate simplified models of a signal with these "features"
3. Generate bispectral representation of the signal
4. Identify and understand characteristic patterns in bispectrum
5. Generalize to look for similar features in LIGO data

**Results and discussion**

Application of HOS to the LIGO 40m data stream requires understanding HOS of much simpler random processes. Thus, having the power spectrum and some apriori knowledge about LIGO 40m data, we generated LIGO 40m simplified models.

However, after considering some results we obtained, we concluded that when estimating HOS we need to consider problems known in power spectrum estimation such as: number of samples, number of segments, number of samples per segment, windowing methods, type of estimation used, all in order to get unbiased and consistent estimates.
One problem with bispectrum is that it has a large variance and that its variance is frequency dependent [9]. In order to avoid this problem we explored the use of the bicoherence which is actually a normalized bispectrum.

We began to explore using the bicoherence when we obtained some unexpected results while estimating the bispectrum of a harmonic process (process consisting of only harmonic signals). It can be shown that the bispectrum of a harmonic process is identically zero [6]. The numerically calculated bispectrum is not zero and we sought on explanation why.
In all numerical simulations presented here, the bispectrum was computed using the direct method [5], which is an approximation of the formula:

$$B(f_1, f_2) = E\{X(f_1)X(f_2)X^*(f_1 + f_2)\}$$

where X(f) is a Fourier transform of a signal, x(t)

The Fourier transform of harmonic signal with frequency $f_0$ is:

$$X(f) \sim \delta(f - f_0)e^{j\varphi} + \delta(f + f_0)e^{-j\varphi}$$

Thus each factor $X(f_1)$,$X(f_2)$,$X^*(f_1+f_2)$ produces two parallel lines in $(f_1,f_2)$ plane. For instance $X(f_1)$ will be represented by two lines at frequencies $f_1=f_0$ and $f_1=-f_0$. The product of three such factors is theoretically zero. However spectral leakage due to finite data sets seems to be the reason a non zero bispectrum is obtained.

The first numerical simulation was done taking $f_0$ as $n*2\pi/f_s$. Where n is an integer and fs is the sampling frequency. In this case spectral leakage (as a result of the way fast fourier transform (fft) algorithm works) is minimal and we obtained very weak bispectrum (see Fig. 1). However, in the case when $f_0=(n+1/2)*2\pi/f_s$, $f_0$ is at frequency shifted by half of an fft bin from the previous case, spectral leakage occurs and one obtains much greater values for bispectrum (~145db higher).(see Fig. 2)

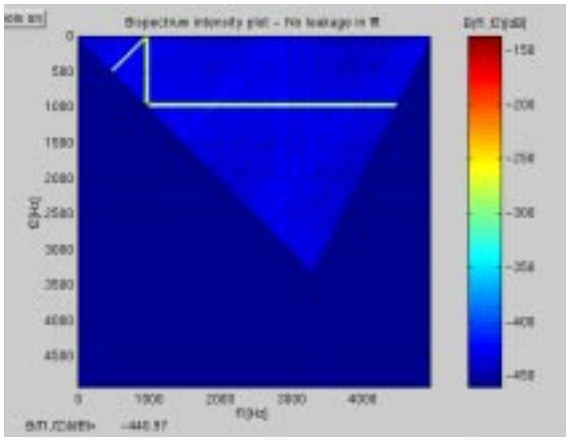

Figure 1: Bicoherence for sine wave - $f_0=n*2\pi/f_s$

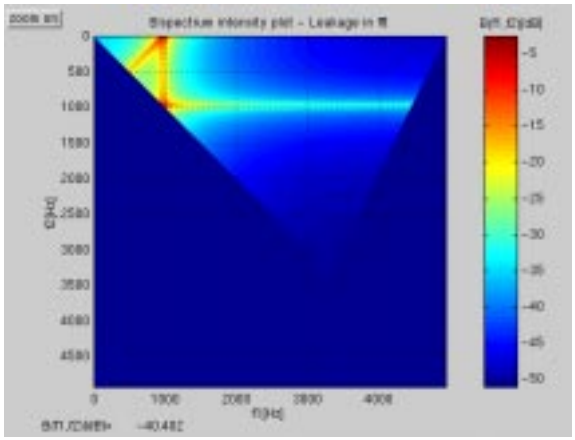

Figure 2: Bicoherence for sine wave - $f_0=(n+1/2)*2\pi/f_s$

This result means that bispectrum estimation highly depends on the details in which the ftt is performed. This is not the only problem with the bispectrum estimator. Its variance depends on the power of a signal and is in general different for each point in the $(f_1,f_2)$ plane. This is one of the reasons the bicoherence is preferred to the bispectrum. We will proceed with a discussion of the properties of bicoherence.
In order to differentiate between relevant information and numerical artifacts produced by the estimation algorithm used, we need to characterize estimator. It is known that the bicoherence is an unbiased estimator and that its variance is a non-central $\chi^2$ distributed variable [11, 12].

Let us first assume that we want to see if a process has non-Gaussian characteristics, which implies that the bicoherence is non zero.
The problem can be seen as hypotesis testing:

$H_0$: The process is Gaussian

$H_1$: The process is not Gaussian

Under the null hypothesis bicoherence is a $\chi$ distributed variable with two degrees of freedom (DOF). We will want to set a threshold to test the hypothesis. Therefore we will need probability density function (pdf) of the bicoherence assuming the null hypotesis [6]. This requires knowing the variance estimated from the data and we obtained empirical result that variance of our estimator is:

$$var\left\{\hat{bic}(f_1, f_2)\right\} = \frac{D}{N}$$

where N - number of samples and D - number of samples per segments.
The theoretical curve and a histogram of pdf for the bicoherence obtained for the simulated Gaussian data are shown in Figure 3

The threshold for the null hypotesis at confidence level of 95% can be obtained from:

$$Prob\left\{\left|\hat{bic}(f_1, f_2)\right| > \sqrt{3Var[\hat{bic}(f_1, f_2)]}\right\} < 0.05$$

A simple test for Gaussianity would be to (i) determine the number of bicoherence points in the principal domain* that exceed the threshold; (ii) if the ratio between this number and the total number of estimated bicoherence values in the principal domain is greater than 0.05 we reject null hypothesis.

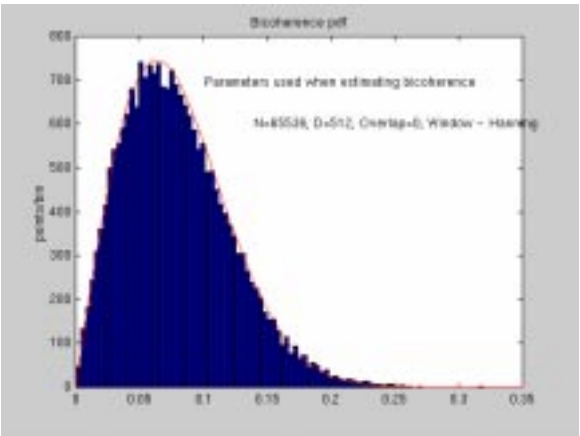


Figure 3: Histogram and theoretical curve of the magnitude of the estimated normalized bicoherence of white Gaussian noise

* Due to symmetry properties of bicoherence we concentrate on only part of the first octant in ($f_1$,$f_2$) plane. Specifically, triangle between points (0,0), (fs/2,0) and (fs/3,fs/3) which is called principal domain (PD)

Let us see whether the LIGO 40m data have non-Gaussian characteristics using such a test. We histogrammed the LIGO 40m data and the pdf looks very much like Gaussian distribution.(see Fig. 4).

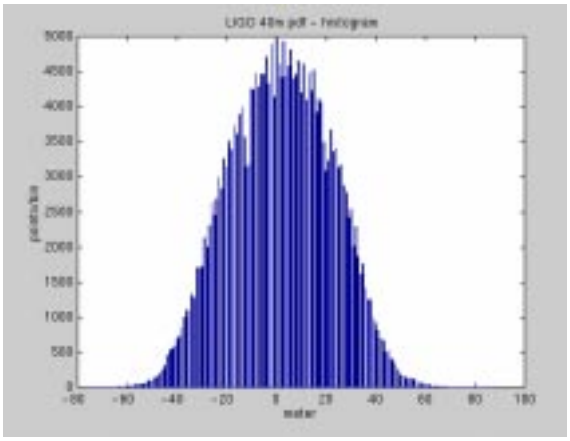

Figure 4: Histogram of LIGO 40m data

The observation consisted of 65536 samples of the LIGO data, with 512 samples per segment. Theoretical variance is 512/65536=0.0078 and thus 95% confidence threshold is 0.1531.

The number of bicoherence values exceeding threshold was 3687 and the total number of values in the principal domain was 21930.This gives 3687/21930=0.168 which is greater than 0.05 which means that there is some deviation from Gaussianity in the signal.

These results were in a way surprising taking into consideration that pdf of LIGO 40m data appears like pdf of a Gaussian distributed random variable.

Instead of comparing pdf of LIGO 40m data and Gaussian data, which seems to lead to wrong conclusions, we compared probability distribution function (pf) of the estimated bicoherence for two signals. Assuming that LIGO 40m data is Gaussian they should be the same.

We took already estimated bicoherences from the previous simulation and obtained probability distribution functions as shown on Fig. 5-6. On Fig. 5 you can see pf as a function of x in linear scale while on Fig. 6 it is plotted as a function of $x^2$ in the log scale. It can be readily seen from both figures that pf's for two signals largely differ for large values of x.

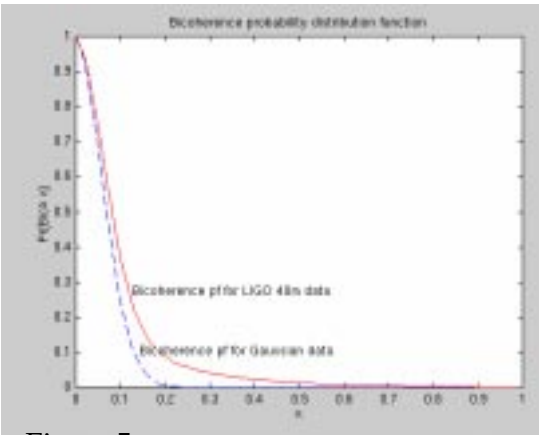


Figure 5: Bicoherence probability distribution function

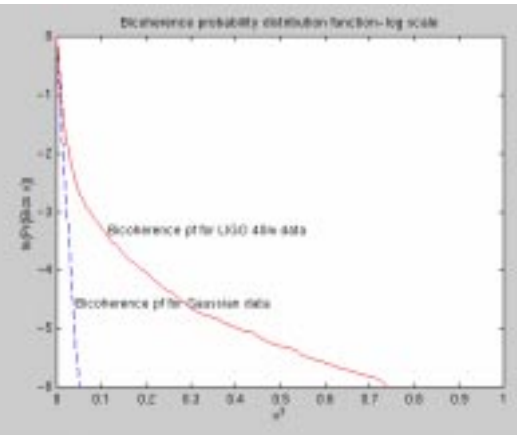


Figure 6: Bicoherence probability distribution function-log scale

So far we have two results: 1. pdf of LIGO 40m data looks very much like Gaussian 2. pf of LIGO 40m data largely differs from the pf of the Gaussian data.

This raises the question if there is anything in LIGO 40m data that may cause this contradictory results. We will try to find the answer to this question by examining power spectrum and bicoherence plots of LIGO 40m data. Figs. 7-8
It may be seen from the power spectrum of LIGO 40m data that it is rich in harmonics at 60Hz. The bicoherence is essentially zero except for peaks at [n*60Hz,m*60Hz]. Therefore such a large difference between the bicoherencies of real and Gaussian data may be caused by presence of the peaks.

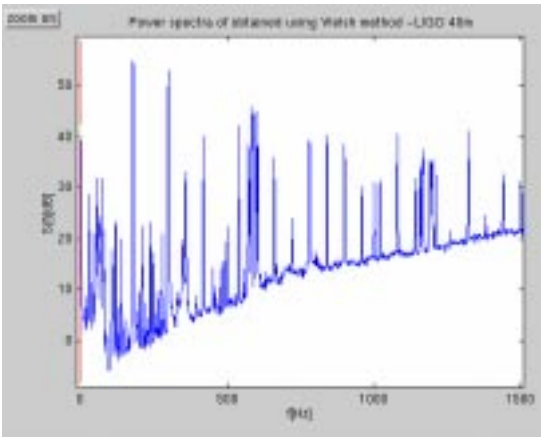

Figure 7: LIGO 40m Power

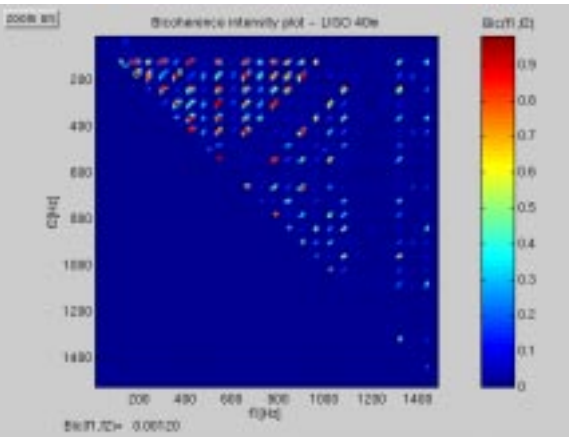

Figure 8: LIGO 40m Bicoherence

Peaks in the bicoherence occur when the interaction between two harmonic components causes contribution to the power at their sum and/or difference frequencies [9].
To show this let us consider the signals which follow the model:

$$x = x_1 + x_2 + Bx_3 + Cx_1x_2 + n_G(t)$$
$$x_1 = A\cos(2\pi f_1 t + \varphi_1)$$
$$x_2 = A\cos(2\pi f_2 t + \varphi_2)$$
$$x_3 = A\cos(2\pi f_3 t + \varphi_3)$$

where A,B,C are arbitrary constants; $n_G(t)$ - Colored 1/f Gaussian noise; $\varphi_1,\varphi_2,\varphi_3$ are all independent, uniformly distributed random variables over $(-\pi,\pi]$ and $f_3=f_1+f_2$ . We have chosen $f_1$, $f_2$ and $f_3$ in a way to decrease the influence of the fft leakage to the HOS measures as discussed at the beginning of this report. Random process x was obtained by generating 256 realizations, each realization with 1024 samples. Sampling frequency of the signal is 10KHz. $f_1=f_s*50/1024$, $f_2=f_s*115/1024$ and $f_3=f_s*165/1024$.

**Case studies:**

**Case 1:**

Let us set B=1 and C=0. In this case signal consists of three independent harmonics located at frequencies $f_1$,$f_2$ and $f_3$. Power spectrum and bicoherence of the resulting signal are shown on the Fig. 9 and Fig. 10 respectively.

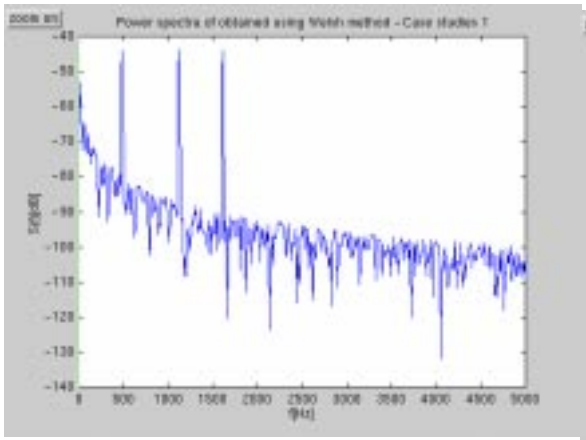

Figure 9: Case studies (case1) - Power spectrum

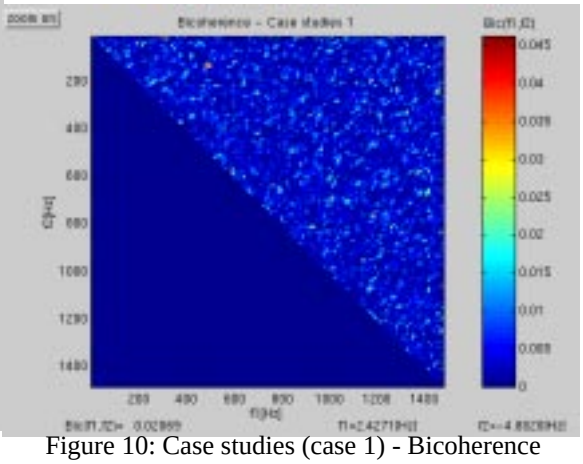

Figure 10: Case studies (case 1) - Bicoherence

Bicoherence doesn't show any features which is expected since we are dealing with harmonic signal consisting of three independent components. However, if instead of component $x_3$ which is located at frequency $f_1+f_2$, we introduce component which is not independent from $x_1$ and $x_2$ but is the result of the non-linear interaction between them we obtain different results. To see this we will consider another signal that follows general model previously discussed.

**Case 2:**

Instead of the independent component $x_3$ which is at frequency at frequency $f_3=f_1+f_2$, we introduced term $x_{1*}x_2$ to the signal. This produces two components in the power spectrum. One at frequency $f_1+f_2$ and the other at $f_1-f_2$ frequency.

This corresponds to setting B=0 and C=1 in our model. Power spectrum and bicoherence of such a signal are shown on Fig. 11 and Fig. 12. Note that the only difference between components at f1+f2 in case 1 and case 2 is that phase of the former is independent random variable while in the case 2 its phase is not independent but is equal to φ1+φ2. We may conclude that peak in the bicoherence occurs if there is phase coherency between frequency components in the sense previously stated.

From Fig. 11 we can see that another strong peak is present at bifrequency ($f_1$-$f_2$,$f_2$). This peak results from the interaction of components at $f_1$-$f_2$, $f_2$ and $f_1$. Note that the phase of the component at $f_1$-$f_2$ is $\varphi_1$-$\varphi_2$. If we say that $f_1$'=$f_1$-$f_2$, $f_2$'=$f_2$ and $f_3$'=$f_1$ then $f_3$'=$f_1$'+$f_2$' and $\varphi_3$'=$\varphi_1$'+$\varphi_2$'. Relationship between components at $f_1$', $f_2$' and $f_3$' is the same as the relationship between components at $f_1$, $f_2$ and $f_1+f_2$. Thus the second peak stems from the phase relations between frequency components in the signal as was the case for the first peak.

We have derived anaalitically that apart from the ability to detect phase coherency bicoherence can provide way to quantify the amount of power at a given frequency which is due to phase coupling as compared to the total power at that frequency.

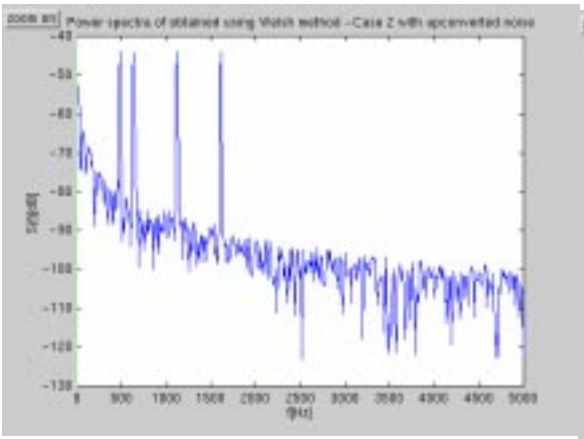

Figure 11: Case studies (case 2) - Power spectrum

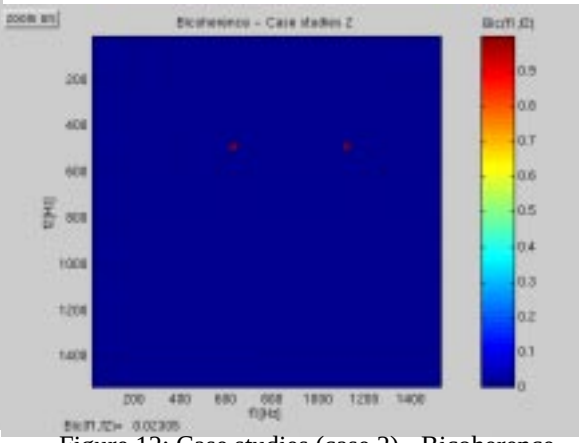

Figure 12: Case studies (case 2) - Bicoherence

**Case 3:**

We set B=0.5 and C=1 which means that both independent component and component which is the result of non-linear interaction are present.
We wish to quantify the amount of power at frequency $f_1+f_2$ that is due to the non-linear process resulting from the product of $x_1$ and $x_2$ and how much of the power is from an independently excited and uncorrelated process.
For this special case it may be shown analytically that:

$$bic(f_1, f_2)E\{|X(f_1 + f_2)|^2\} = E\{|X(f_1)X(f_2)|^2\}$$

where first term on the left is the bicoherence at bifrequency $(f_1,f_2)$, second term is the total power at frequency $f_1+f_2$, and the term on the right side is correlated power at $f_1+f_2$ due to non-linear effect.
From this result it may be seen that the case where independently excited harmonic is absent the bicoherence is unity, while if independent component exists then the bicoherence must be less than unity, moreover the bicoherence gives the ratio between the power due to the non-linearity and total power at frequency $f_1+f_2$. We tried to verify this result through simulation and the results obtained on Figs. 13-14. Note that the bicoherence of the signal (see Fig. 14) has two peaks at $(f_1,f_2)$ and $(f_1-f_2,f_2)$. Peak at the former bifrequency has value 0.8! and the peak at the latter bifrequency has value 1. This is not the expected result since at $(f_1,f_2)$ we have both independent and non-linearly excited component of equal power and therefore the ratio between the power due to non-linearity and the total power should be 0.5. At frequency $(f_1-f_2,f_2)$ we have only due to interaction between $x_1$ and $x_2$ and thus this ratio equals to one and this is expected result.
We don't understand why we obtained larger value for bicoherence at $(f_1,f_2)$ than we expected, but it is worth to mention that bicoherence may be used to detect that both

independent and non-linearly excited component are present at the same frequency since we get for bicoherence at $(f_1,f_2)$ value 0.8 instead of 1 which was the case when there was only non-linearly excited component at frequency $f_1+f_2$.

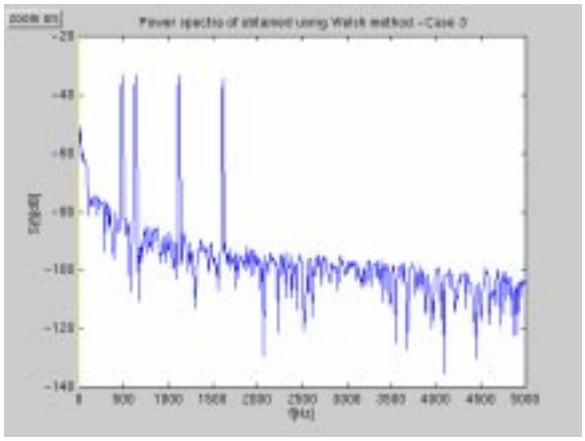

Figure 13: Case studies (case 3)-Power spectrum

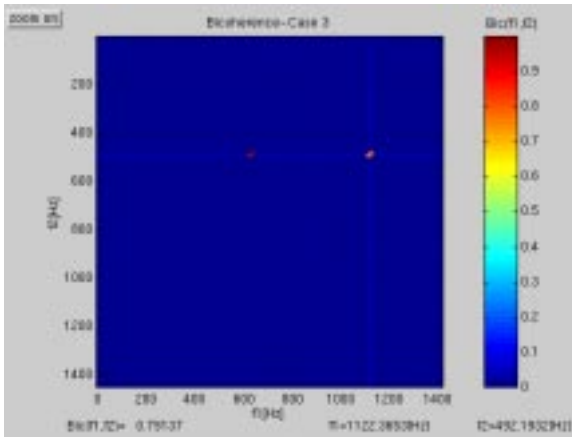

Figure 14: Case studies (case 3)-Bicoherence

If we examine LIGO 40m power spectrum more closely we notice the presence of the sidebands around some of the harmonics. We know from other measurements that the laser power level exhibited big, narrow line spikes at 180Hz and 300Hz. The apparent mirror motion that shows up at the interferometer output is the product of real mirror motion multiplied by the laser power. If we look at the neighborhood of 300Hz for example we will see a pair of sidebands below and above 300Hz line. We simulated this effect by taking the low-frequency Gaussian 1/f Colored noise spectrum and superimposed it onto the signal we used in case 2 of case studies. FM modulating a baseband noise causes appropriate frequency shifting.

Therefore the signal we consider is the following:

**Case 4:**

$$x = (x_1 + x_2 + Bx_3 + Cx_1x_2)n_G(t) + x_1 + x_2 + Bx_3 + Cx_1x_2$$

where B, C, $x_1$, $x_2$, $n_G(t)$ hold the same definition as before. Power spectrum and bicoherence of such a signal are shown on Figs. 15-16.

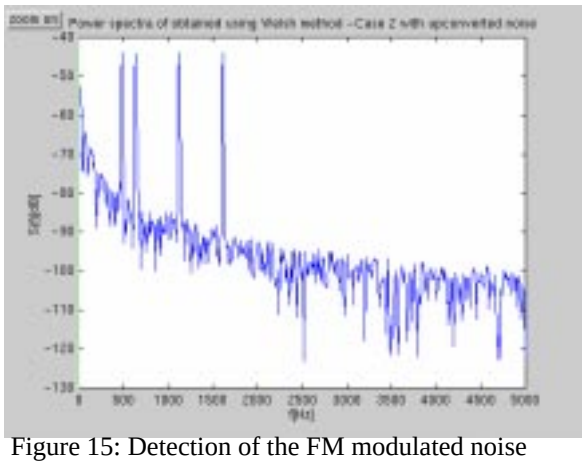

Figure 15: Detection of the FM modulated noise power spectrum

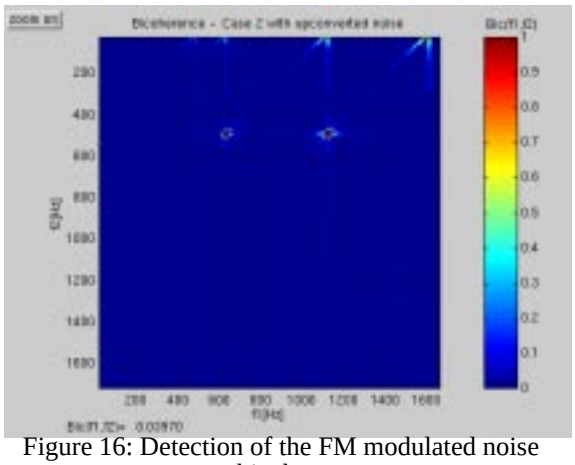

Figure 16: Detection of the FM modulated noise bicoherence

From Fig. 15 we see that presence of the modulated noise at high frequencies causes additional features to appear in the bicoherence. Such appearance of the bicoherence resembles the appearance of the bispectrum of a harmonic process. Can we find something like this in LIGO 40m bicoherence?

We have mentioned before that peaks in the bicoherence occur when the interaction between two harmonic components causes contribution to the power at their sum and/or difference frequencies. If we consider the vicinity of bifrequencies (180Hz,0) and (300Hz,0) where we expect some features we can see that bicoherence is different from zero around bifrequencies $(f_1,f_2)$[Hz]={(180,30),(300,30),(780,30),(900,30)}. We may immediately suspect that those features are the result of quadratic phase coherency. However if that was the fact, we would have coherency between harmonics at frequencies 300Hz, 30Hz, 330 Hz or 780Hz, 30Hz, 810Hz. Harmonics at frequencies 330Hz and 810Hz exist although they are very weak. The reason why we may believe that these features stem from the FM modulation is that components at 180Hz, 300Hz, 780Hz, 900Hz have expressive sidebands unlike other lines in the spectrum below where these features do not exist.
In Fig.17 we show the bicoherence of the LIGO 40m data in the vicinity of $(f_1,f_2)$=(780Hz,30Hz) as compared with the features obtained from numerical simulation in Case studies (case 2) plotted in Fig. 18. Also power spectrum of LIGO 40m data is in the vicinity of 780Hz is plotted in Fig. 19.

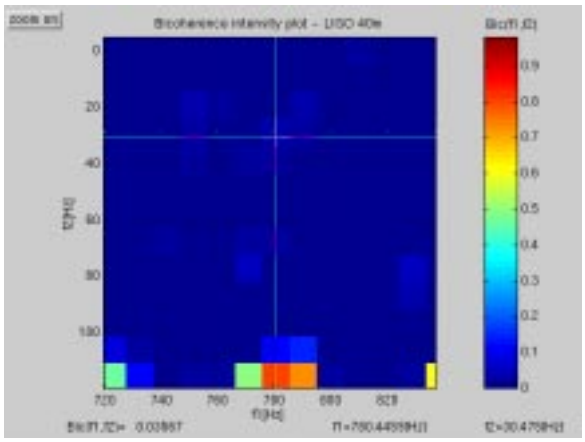

Figure 17: LIGO 40m data

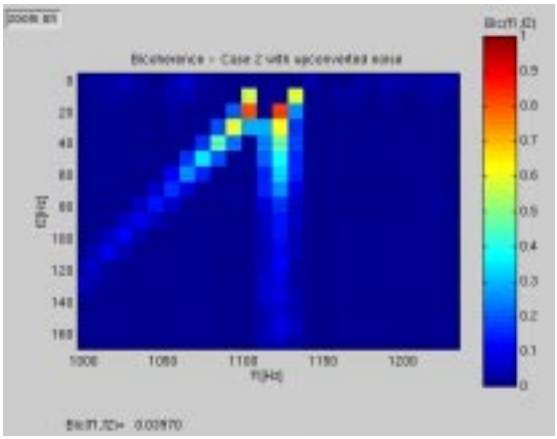

Figure 18: Detection of FM modulated signals

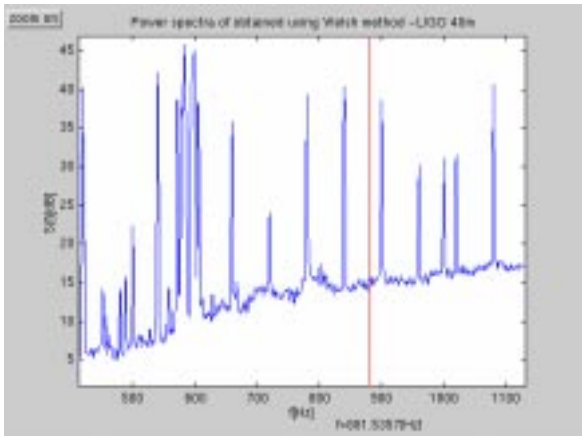

Figure 19: LIGO 40m Power

## Conclusions

HOS may be used as a tool for characterizing random processes. It can detect phase coherency between frequency components, specifically Quadratic Phase Coupling, as well as the presence of noise upconversion. Distribution of LIGO data looks to be Gaussian however its bicoherence is not zero and it features presence of peaks in the ($f_1$, $f_2$) plane. From the bicoherence of LIGO 40m data we may conclude that coherency between 60Hz harmonics is present. Bicoherence has the ability to detect FM nonlinearities such as FM upconverted noise. This kind of nonlinearity is present at several frequencies in LIGO 40m data. Estimates of HOS measures are highly dependent on parameters that are also critical when estimating power spectrum. Attention should be paid to this issues even though it is not studied extensively in the theory, yet.

**Acknowledgements**

The SURF at California Institute of Technology has been a rewarding experience. I want to thank everybody who helped me during the program.
Deep gratitude goes to my mentor Dr. Albert Lazzarini for his guidance, inspiration and support throughout my stay at Caltech. Despite his busy schedule he always had a time for raising questions and carrying discussions about this research work.
I would also like to thank the other members of the LIGO Group especially Chip Sumner, Wallid Majid and Suresh Singh for their advice, inspiration and interest in my research.
Additionally, I want to express my appreciation to Dr. Kenneth Libbrecht and members of the SFP Office-director Carolyn Merkel, Carol Casey, and Ryan Tischler for their help before and during my stay at Caltech.
Finally, I would like to thank California Institute of Technology, National Science Foundation and LIGO for financial support and making my research at Caltech possible.

**List of references**

[1] D. Sigg, “Gravitational waves”, Publication LIGO-P980007-00-D
[2] A. D. Gillespie, “Thermal Noise in the Initial LIGO Interferometers”, Ph.D. thesis, California institute of Technology, Pasadena, California
[3] C.L. Nikias and J. M. Mendel, “Signal Processing with Higher-Order Spectra”, IEEE Signal Processing Magazine, July 1993
[4] J.M. Mendel, “Tutorial on Higher-Order Statistics(Spectra) in Signal Processing and System Theory: Theoretical Results and Some Applications”, Proceedings of the IEEE, Vol. 79, no. 3, March 1991
[5] C. L. Nikias, “Bispectrum Estimation: A Digital Signal Processing Framework’, Proceedings of the IEEE, Vol 75, no. 7, July 1987
[5] M. J. Hinich and H. Messer, “On the Principal Domain of the Discrete Bispectrum of a Stationary Signal”, IEEE Transactions on Signal Processing, vol. 43, no.9, September 1995
[6] V. Chandran and S. L. Elgar, “Mean and Variance of Estimates of the Bispectrum of a Harmonic Random Process-An Analysis Including Leakage Effects”, IEEE Transactions on signal processing, Vol. 39, no. 12, December 1991
[6] A.M Richardson and W.S. Hodgkiss, “Bispectral Analysis”, Journal of the Acoustical Society of America, Vol. 96, No. 2, Pt.1, August 1994
[8] J. W. Dalle Molle and M. J. Hinich, “Trispectral analysis of stationary random time series”, Journal of the Acoustical Society of America 97 (5), Pt. 1, May 1995
[9] J. W. Fackrell, “Bispectral analysis of Speech Signals”, Ph.D. thesis, The University of Edinburgh, September 1996
[10] A. Swami, J. M. Mendel and C. L. Nikias, “Higher-Order Statistics Spectral Analysis Toolbox Tutorial”, Mathworks 1998
[11] M. J. Hinich, “Testing for Gaussianity and Linearity of a Stationary Time Series”, Journal of time series analysis, Vol 3, No. 3

[12] D. Kletter and H. Messer, “Suboptimal Detection of Non-Gaussian Signals by Third Order Spectral Analysis”, IEEE Transactions on acoustics, speech and signal processing, Vol. 38, No. 6, June 1990

# APPENDIX A: DOCUMENTATION FOR SOFTWARE TOOLS PRODUCED DURING SURF PROJECT

This document describes how to use Visualization tools that have been developed in order to examine results of the numerical simulations we performed throughout the project.
If you are novice in the HOS use this document in concert with Higher-Order Spectral Analysis Toolbox User Guide available from Matlab.
1. We provided information on how to use functions that are part of this tools as well as Reference for locating information on specific functions. You can find more information on how this functions work from the code which is also available.
2. We present application of all functions through the examples and in order that we usually practiced when performing simulations.
3. All examples here are done using LIGO40m data.
4. We used GRtool (part of GRASP) to read LIGO 40m data from the Frame libraries (see appendix for more information and installation procedure)

In this SURF project we were concentrated on applying HOS measures which contain information that conventional spectral analysis cannot provide. However power spectrum of a signal is extremely useful measure and we always examined power spectrum of a signal before examining HOS measures.
For both estimating and plotting power spectrum of a signal we used function MPSDLOG which estimates Power Spectral Density of the signal using PSD function in Matlab's Signal Processing Toolbox. Additionally it plots graph of the PSD in log scale with zooming ability and ability to read frequencies at the frequency axis using ruler. Let us plot the power spectrum of LIGO 40m data using this function. (type in the Matlab's workspace following command);

```
>> mpsdlog(ligodata, 8192, 9868.412, hanning(8192), 0,'none',
'LIGO 40 Power spectrum');
```

Power spectrum is plotted in Fig. 1
On the Figure1 you can see 'zoom on' button and the ruler on the left. Clicking on the zoom button changes the zoom mode from on to off (zooming enabled and zooming disabled respectively) and vice versa and by moving ruler (when figure is not is zoom 'on' mode) we can get exact frequencies of some features of interest in the power spectrum.
The way we realized this features is the following: We used the ability that each Graphic object (see Using Matlab Graphics available from Matlab) can be controlled by its properties.

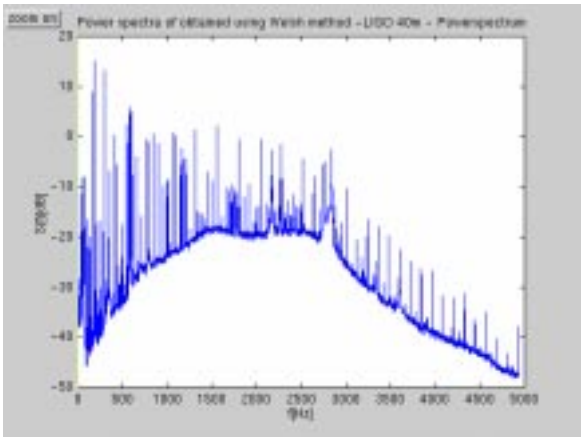

Figure 1: LIGO 40m Power Spectrum

'Zoom on' is one of the User Interface Control Objects, specifically Toggle button. Its 'Callback' (one of the properties) is executed whenever it is clicked on 'zoom on' button. It checks the current mode of the figure and changes it. You cannot move rulers while in zoom on mode. When interactive zooming is enabled in a figure, pressing a left mouse button while your cursor is within an axis (rectangle where your PSD is plotted)

zooms into the point or out of the point if right mouse button is pressed. Zooming changes the axis limits.

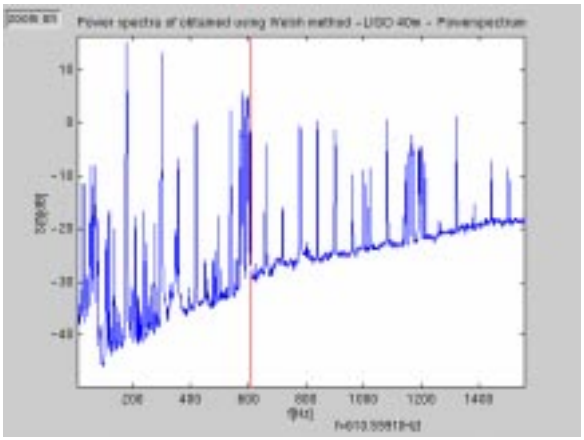

Figure 2: LIGO 40m Power Spectrum (magnified)

Clicking and dragging over an axes when interactive zooming is enabled draws a rubber-band box. When the mouse button is released, the axis zoom into the region enclosed by the rubber-band box. (see Fig.2). We zoomed region between (0-1500Hz)
When the figure is not is zoom mode we can move ruler and read current frequency at which the ruler is located.(see Fig.2) We can see that one of the suspension line harmonics is at f~610Hz
This feature is realized using animator function (see Function reference). 'ButtonDownFcn' property of the line object (ruler) is set to (animator 'start'). Clicking on the ruler calls animator function with input argument 'start'. As the ruler is dragged animator function is repeatedly called with an input argument 'move' until the button is released. IMPORTANT: If you want to zoom at some region and use ruler afterwards to read frequencies within that region, then first move the ruler to that region and then perform zooming. If you forget to move the ruler before zooming then just zoom out and drag the ruler.

When we examined power spectrum of the signal we proceed with estimating its higher order statistics of the signal. For this purpose we used the following functions:

MBISPECD Bispectrum estimation using the direct (fft-based) approach.
MBISPECI Bispectrum estimation using the indirect method.
MBICOHER - Direct (FD) method for estimating bicoherence

These three functions use mbispecd_con, mbispeci_con and mbicoher_con respectively which are slightly different from standard functions for estimating bispectrum and bicoherence (bispecd, bispeci and bicoher from HOSA Toolbox) in that, that they don't plot contour plots automatically. This makes these functions less time consuming, and prevents overwriting existing figures on the screen!
MBISPECD,MBISPECI and MBICOHER return, as the output arguments, only the values of HOS measures in first quadrant and in such a way that the values out of principal domain* are set to zero.
Let us estimate bispectrum of LIGO40m data using direct method and first 262144 samples of the time series. (type in Matlab's workspace)

```
>> [bspec,w] = mbispecd(ligodata(1:262144), 512, 5, 512, 0,
9868.412);
```

* Due to symmetry properties of bicoherence we concentrate on only part of the first octant in $(f_1,f_2)$ plane. Specifically, triangle between points (0,0), (fs/2,0) and (fs/3,fs/3) which is called Principal Domain (PD)
We can plot estimates of HOS measures in two ways.

1. 3-D plot using functions MMESH or MMESHLOG

MMESH Plots 3-D mesh surface in linear scale
MMESHLOG Plots 3-D mesh surface in log scale

2. Intensity plots - two dimensional plots with axis $f_1$ and $f_2$ where color at each bifrequency corresponds to the value of the plotted measure (which is function of $f_1$ and $f_2$). This plots are plotted using INTENSITY and INTENSITYLOG functions.

INTENSITY Plots intensity plots of HOS measures in linear scale
INTENSITYLOG Plots intensity plots of HOS measures in log scale

First, we will plot mesh plot using MMESHLOG function

```
>> mmeshlog(bspec,w,'Bispectrum of LIGO 40m data','bispectrum');
```

(see Fig. 3). Clicking with left mouse button on the axis within the figure will rotate the figure over z-axis. (see Fig. 4)

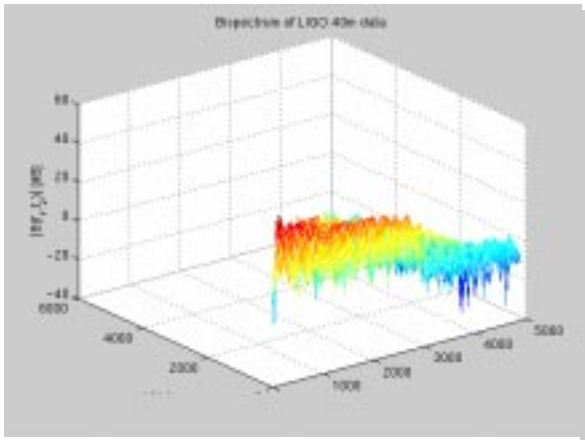

Figure 3: LIGO 40m Bispectrum mesh plot

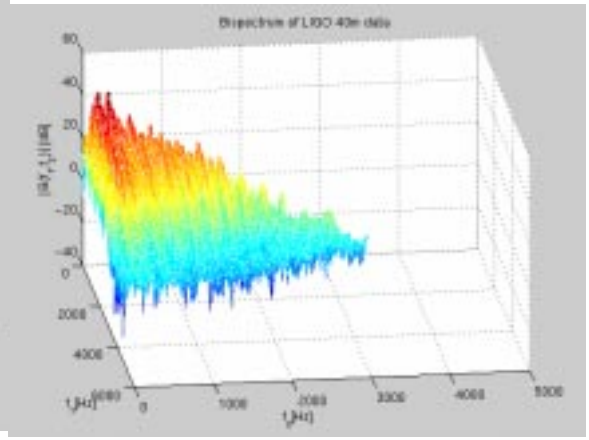

Figure 4: LIGO 40m Bispectrum mesh plot (ro

We found, however that the plots providing the most useful information on HOS measures are intensity plots. These plots have zooming option and the rulers (two rulers since all measures are function of two frequencies) and these options are the same as for psdlog function. Only rulers use functions animatorx and animatory which work similarly like function animator as explained before.
Let us plot intensity plot of the previously estimated bispectrum. Type into Matlab's workspace following command:

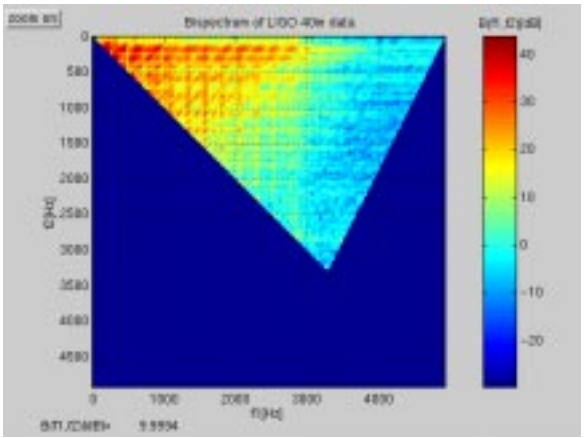

Figure 5: LIGO 40m Bispectrum isometric plot

```
>> intensitylog(bspec,w,' Bispectrum of LIGO 40m... data','
bispectrum'); (see Fig. 5)
```

This command plots bispectrum of LIGO 40m data in semilog scale. These plots have additional feature which is reading values of the estimated measure at each bifrequency. Once the plot is made, a file containing array of estimated values of the measure of interest and frequency vector is created.

This file is automatically saved in the directory set by the user. In order to ensure that these functions that use this file work on your machine you should change the name of that directory in all functions where it appears. These functions are INTENSITY, INTENSITYLOG, GETDATAFORISO, DELMATFILE, SAVEMATFILE. Note that in case of DELMATFILE and SAVEMATFILE name of the working directory should have space before the name in the string (see example below) since it is used to execute UNIX commands. For instance if the directory where you want to save these files is /home/user/ matlab/plots, change in the SAVEMATFILE should look like:

```
source=strcat('  /home/user/matlab/plots/',s,'.mat');
destination=strcat('  /home/user/matlab/plots/',name,'.mat');
```

When all this is set when you plot intensity plots file named figure*.mat is saved in../plots directory, where * is random three digit number. You don't need to know the name of that file for zooming or reading frequency on the figure. However if you want to save the figure you should execute command

```
>>savematfile(gcf,name);
```

This command sets the name for that file as desired. For instance if you have Figure 5 on the screen and you want to save that figure as LIGO40mbspec.m (NOTE: Along with m-file matlab makes LIGO40mbspec.mat automatically) you should execute SAVEMATFILE function as follows:

```
>>savematfile(gcf,'figureLIGO40mbspec');
```

Now you have figureLIGO40mbspec.mat file in your directory instead of figure*.mat

ATTENTION:
1. Do not name this file LIGO40mbspec only, because you will overwrite already existing .mat file that Matlab automatically created when saving LIGO40mbspec.m

2. Make sure that you make the figure you want to save to be current figure. The easiest way to do this is to press 'zoom on' button twice. Remember that you shouldn't save figure when zooming is enabled.

In case when you don't need figure on the screen and you want to close it you should delete created .mat file. To do this make the figure you want to close current and execute command:

```
>>delmatfile(gcf);
```

Let us assume that you saved the figure 5. In this case .mat file that corresponds to this figure contains array-bspec and vector- w. Moving the rulers calls function showamplitude (see Function reference) which reads the value of the measure from the .mat file using

function getdataforiso (see Function reference) and shows the current value on the figure. From Figure 6 we can see that bispectrum of LIGO 40m data has a peak at bifrequency ($f_1$,$f_2$)~(540,300) and which value is 32.029[dB].

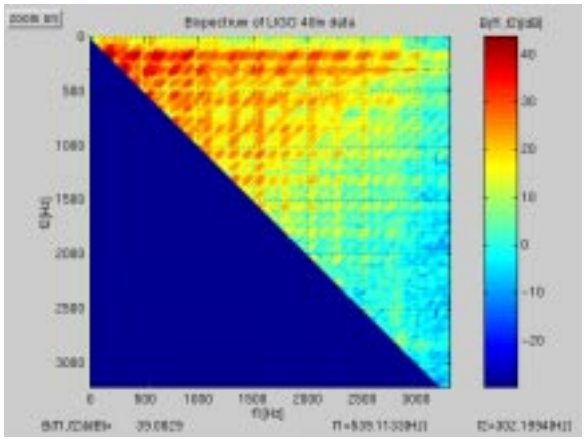

Figure 6: LIGO 40m Bispectrum isometric plot

Axis labeling of these plots depends on the measure that is plotted (bispectrum or bicoherence). Passing the argument ‘flag’ to the function which may take values ‘bicoherence’ or ‘bispectrum’ determines how the labeling is performed.

## Function reference

**ANIMATOR**(action) controls moving ruler on the plot
plotted by function mpsd
action start
move
stop

when action is set to start we set initial handles of the current object in this case line object (ruler) WindowButtonMotionFunction to animator move WindowButtonUpFunction to animator stop. This means that when the mouse button is pressed and kept this way as we move mouse this function is repeatedly called with input argument 'move'. Position and value of the current position is read by showamplitude function and this is changed on the figure. Since ButtonUpFunction is set at the beginning to animator stop, once we release the button animator function is called with 'stop' input argument and the complete procedure stops until we press button again. To enable this handle ButtonDownFcn of rulers (line object) is set to 'animator start'

see code for MPSDLOG function

**ANIMATORX**(action) controls moving rulers on plots plotted by functions INTENSITY/ INTENSITYLOG
action start
move
stop
when action is 'start' we set initial handles of the line object (ruler) WindowButtonMotionFunction animatorx move and WindowButtonUpFunction to animatorx stop.This means that when the mouse button is pressed and kept this way as we move mouse this function is repeatedly called with input argument 'move'. Position and value of the current position is read by show amplitude and this is changed on the figure. Since ButtonUpFunction is set at the beginning to aimatorx stop once we release the button this function is called with 'stop' input argument and the complete procedure stops until we press button again. To enable this handle ButtonDownFcn of rulers (line object) is set to'animatorx start'
see code for INTENSITY/INTENSITYLOG functions

**ANIMATORY**(action) controls moving rulers on plots plotted by functions INTENSITY/ INTENSITYLOG

action start
move
stop
when action is set to start we set initial handles of the current object in this case line object (ruler) WindowButtonMotionFunction to animatory move and WindowButtonUpFunction to animatory stop. This means that when the mouse button is pressed and kept this way as we move mouse this function is repeatedly called with input argument 'move'. Position and value of the current position is read by show amlitude and this is changed on the figure.

Since ButtonUpFunction is set at the beginning to animatory stop once we release the button this function is called with 'stop' input argument and the complete procedure stops until we press button again. To enable this handle ButtonDownFcn of rulers (line object) is set to 'animatory start'
see code for INTENSITY/INTENSITYLOG functions

**DELMATFILE** - Deletes the file that is automatically created
when INTENSITY/INTENSITYLOG functions are executed
This functions needs to be executed before closing
the figure you don't need anymore. You don't need
to save.
gcf - get current figure. Matlab variable.
NOTE: Not executing this function when closing
unnecessary pictures will cause pile up of files
named figure*.mat where * is three digit number.

**INTENSITY** Plots intensity plots of HOS measures - linear scale
intensity(bspecm,wm,name,flag);
bspecm - estimated HOS measures (bispectrum, bicoherence...);
wm - frequency-domain axis associated with HOS measures
- the i-th row (or column) of bspecm corresponds to f1 (or f2)
value of wm(i).
name - name of the figure type string (will be shown on the title)
flag - specifies which measure is plotted. This is used in the
function for labeling purposes

**INTENSITYLOG** Plots intensity plots of HOS measures - log scale
intensitylog(bspecm,wm,name,flag);
bspecm - estimated HOS measures (bispectrum, bicoherence...);
wm - frequency-domain axis associated with HOS measures
- the i-th row (or column) of bspecm corresponds to f1 (or f2)
value of wm(i).
name - name of the figure type string (will be shown on the title)
flag - specifies which measure is plotted. This is used in the
function for labeling purposes

**MBICOHER** Direct (FD) method for estimating bicoherence
[bicm,wm] = bicoher (x, nfft, wind, samp_seg, overlap,sampling_frequency)
x - data vector or time-series
nfft - fft length [default = power of two > samp_seg]
actual size used is power of two greater than 'nsamp'
wind - specifies the time-domain window to be applied to each
data segment; should be of length 'segsamp' (see below);
otherwise, the default Hanning window is used.
samp_seg - samples per segment [default: such that we have 8 segments]
- if x is a matrix, segsamp is set to the number of rows
overlap - percentage overlap, allowed range [0,99]. [default = 50];

- if x is a matrix, overlap is set to 0.
sampling frequency - frequency at which data is sampled. In case
you want to use normalized frequencies then sampling_frequency=1
bicm - estimated bicoherence: only first quadrant and all values
not in principal domain are set to zero
wm - vector of frequencies associated with the rows and columns of bic;

**MBICOHER_CON** Direct (FD) method for estimating bicoherence
This is equivalent to function bicoher except that it doesn't
plot contour plot and is less time consuming
[bic,waxis] = mbicoher_con (y, nfft, wind, segsamp, overlap)
y - data vector or time-series
nfft - fft length [default = power of two > segsamp]
actual size used is power of two greater than 'nsamp'
wind - specifies the time-domain window to be applied to each
data segment; should be of length 'segsamp' (see below);
otherwise, the default Hanning window is used.
segsamp - samples per segment [default: such that we have 8 segments]
- if x is a matrix, segsamp is set to the number of rows
overlap - percentage overlap, allowed range [0,99]. [default = 50];
- if x is a matrix, overlap is set to 0.
bic - estimated bicoherence: an nfft x nfft array, with origin
at the center, and axes pointing down and to the right.
waxis - vector of frequencies associated with the rows and columns
of bic; sampling frequency is assumed to be 1.

**MBISPECD** Bispectrum estimation using the direct (fft-based) approach.
[bspecm,wm] = bispecd (y, nfft, wind, samp_seg, overlap, sampling_frequency)
y - data vector or time-series
nfft - fft length [default = power of two > samp_seg]
wind - window specification for frequency-domain smoothing
if 'wind' is a scalar, it specifies the length of the side
of the square for the Rao-Gabr optimal window [default=5]
if 'wind' is a vector, a 2D window will be calculated via
w2(i,j) = wind(i) * wind(j) * wind(i+j)
if 'wind' is a matrix, it specifies the 2-D filter directly
samp_seg - samples per segment [default: such that we have 8 segments]
- if y is a matrix, segsamp is set to the number of rows
overlap - percentage overlap [default = 50]
- if y is a matrix, overlap is set to 0.
sampling frequency - frequency at which data is sampled. If you want to
use normalized frequencies then sampling_frequency=1
bspecm - estimated bispectrum: only first quadrant and all values
not in principal domain are set to zero
wm - frequency-domain axis associated with the bispectrum.
- the i-th row (or column) of bspecm corresponds to f1 (or f2)

value of wm(i).

**MBISPECD_CON** Bispectrum estimation using the direct (fft-based) approach.
This is equivalent to function bispecd except that it doesn't plot contour plot and is less time consuming
[Bspec,waxis] = bispecd_con (y, nfft, wind, segsamp, overlap)
y - data vector or time-series
nfft - fft length [default = power of two > segsamp]
wind - window specification for frequency-domain smoothing
if 'wind' is a scalar, it specifies the length of the side of the square for the Rao-Gabr optimal window [default=5]
if 'wind' is a vector, a 2D window will be calculated via w2(i,j) = wind(i) * wind(j) * wind(i+j)
if 'wind' is a matrix, it specifies the 2-D filter directly
segsamp - samples per segment [default: such that we have 8 segments]
- if y is a matrix, segsamp is set to the number of rows
overlap - percentage overlap [default = 50]
- if y is a matrix, overlap is set to 0.
Bspec - estimated bispectrum: an nfft x nfft array, with origin at the center, and axes pointing down and to the right.
waxis - vector of frequencies associated with the rows and columns of Bspec; sampling frequency is assumed to be 1.

**MBISPECI** Bispectrum estimation using the indirect method.
[bspecm,wm]=mbispeci(x,nlag,samp_seg,overlap,flag,nfft,wind,sampling_frequency)
x - data vector or time-series
nlag - number of lags to compute [must be specified]
samp_seg - samples per segment [default: row dimension of y]
overlap - percentage overlap [default = 0]
flag -'biased' or 'unbiased' [default is 'unbiased']
nfft - FFT length to use [default = 128]
wind - window function to apply:
if wind=0, the Parzen window is applied (default);
otherwise the hexagonal window with unity values is applied.
sampling_frequency - frequency at which data is sampled. If you want to use normalized frequencies then sampling_frequency=1
bspecm - estimated bispectrum; only first quadrant in (f1,f2) plane and all values not in principal domain are set to zero
wm - frequency-domain axis associated with the bispectrum.
- the i-th row (or column) of bspecm corresponds to f1 (or f2) value of wm(i).

**MBISPECI_CON** Bispectrum estimation using the indirect method.
This is equivalent to function bispeci except that it doesn't plot contour plot
[Bspec,waxis] = mbispeci_con (y,nlag,segsamp,overlap,flag,nfft, wind)
y - data vector or time-series
nlag - number of lags to compute [must be specified]
segsamp - samples per segment [default: row dimension of y]
overlap - percentage overlap [default = 0]

flag - 'biased' or 'unbiased' [default is 'unbiased']
nfft - FFT length to use [default = 128]
wind - window function to apply:
if wind=0, the Parzen window is applied (default);
otherwise the hexagonal window with unity values is applied.
Bspec - estimated bispectrum it is an nfft x nfft array
with origin at the center, and axes pointing down and to the right
waxis - frequency-domain axis associated with the bispectrum.
- the i-th row (or column) of Bspec corresponds to f1 (or f2)
value of waxis(i).

**MMESH** Plots 3-D mesh surface in linear scale
mmesh(bspecm,wm,name) uses standard Matlab function MESH to plot
the colored parametric mesh of the estimated HOS measures defined by
bspecm and wm.
bspecm - estimated HOS measure (bicoherence, bispectrum)
wm - vector of frequencies

MMESHLOG Plots 3-D mesh surface in log scale
mmeshlog(bspecm,wm,name) uses standard Matlab function MESH to plot
the colored parametric mesh of the estimated HOS measures defined by
bspecm and wm.
bspecm - estimated HOS measure (bicoherence, bispectrum)
wm - vector of frequencies

**MPSDLOG** Power Spectral Density estimate.
Estimates Power Spectral Density of the signal using PSD function
in Matlab's Signal Processing Toolbox. Additionally it plots graph in log scale of the PSD with zooming option and ability to read frequencies at the frequency axis using ruler.
mpsdlog(xn,nfft,Fs,window,noverlap,dflag,name);

xn - data vector or time series
nfft - number of points at which power is estimated
Fs - sampling frequency which doesn't affect spectrum estimate
but is used for scaling of plots
window - window that will be applied to each data section
noverlap - overlaping between sections in percents
dflag - can be 'linear', 'mean' or 'none', specifies a
detrending mode for the pre windowed sections of X.
name - name will be shown as the title of the figure
Matlab's PSD function estimates the Power Spectral Density of signal vector X using Welch's averaged periodgram method. X is divided into overlapping sections, each of which is detrended, then windowed by the WINDOW parameter, then zero-padded to length NFFT.

The magnitude squared of the length NFFT DFTs of the sections are averaged to form Pxx. Pxx is length NFFT/2+1 for NFFT even, (NFFT+1)/2 for NFFT odd, or NFFT if the signal X is complex. If you specify a scalar for WINDOW, a Hanning window of that length is used. Fs is the sampling frequency which doesn't affect the spectrum estimate but is used for scaling of plots.

**SAVEMATFILE** - Sets the 'Tag' of the current figure and
the name of the file that is automatically
created when INTENSITY/INTENSITYLOG functions
are executed.
This function needs to be executed before saving
INTENSITY/INTENSITYLOG plots.
gcf - get current figure - Matlab variable
name - name of the file that corresponds to the figure which is current.
Name is without any extension
example: If you plot bicoherence for LIGO 40m data and you
want to save that figure as LIGO40mbic.m then it is recommended
(for easier tracking) that you execute.

isomatfile(gcf,'figureLIGO40mbic');

Before executing this command make sure the figure you want
to save is current figure. The easiest way to do this is to
press 'zoom on' button on the figure twice. NOTE: Figure
% shouldn't be saved when zooming is enabled.

# Appendix

## Installing GRASP

GRASP (Gravitational Radiation Analysis and Simulation Package) is a public domain software toolkit designed for analysis and simulation of data from gravitational wave detectors. The LIGO project has adopted a data format standard called the FRAME format for time domain data. The 40-meter laboratory implemented this data format.
The GRASP package includes routines for reading and using data in the FRAME format. The Gravitational Radiation Toolbox provides a Matlab interface to both GRASP and the Frame Library. The GR toolbox links these two packages with Matlab - simultaneously exposing data to a problem solving environment of Matlab.

GRASP requires access to Numerical Recipes in C and to MPI and MPE libraries and optionally to the CLAPACK libraries. These packages must be installed, and then within GRASP a path to these must be defined.
Installation procedure has two steps. Editing file called SiteSpecific file and then running a shell script which will be shown shortly.
Detailed procedure on installation may be found in GRASP User Manual which can be obtained at: http://www.lsc-group.phys.uwm.edu/_ballen/grasp-distribution.

Site specific we made for our platform is:

```
-------------------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------------------------
#
## This is the SITE_SPECIFIC file
##
## This is the only file that you should need to edit to set up
## your own working copy of GRASP. Directions for editing this
## file may be found in the GRASP USERS MANUAL. A postscript version
## of the manual may be found in the doc/ subdirectory of this file,
## in manual.ps
##
## First, edit this file.
## Second, run the shell script called InstallGRASP
##
##
## ------------------------------------------------------------##
## START EDITING HERE. PLEASE SAVE A COPY OF THIS ORIGINAL FILE
##
##
## edit to point to the home directory of the GRASP package. This
## is where GRASP is installed and where it must be built.
## If the directory structure is already where you want it, just set
```

```
## GRASP_HOME to the current directory!
GRASP_HOME=/home/dpetrovi/HOS/GRASP_1.9.2

## ----------------------------------------------------------##
## THIS SECTION DEFINES YOUR CHOICE OF COMPILER AND OPTIONS ##

# edit to the name of your C compiler:
 CC = cc
#CC = /apps/workshop/SUNWspro/bin/cc


# edit to include compilation flags that you desire:
# profiling
# CFLAGS=-fast -xO4 -xpg
# speed and optimization
# CFLAGS=-fast -xO4
# for debug execution on Sparc Ultra II chips
# CFLAGS=-g -dalign
# for fast execution on Sparc Ultra II chips
#CFLAGS= -fast -xO4 -dalign -xarch=v8plusa
# for fast execution on Sparc Ultra II chips with profiling
# CFLAGS= -fast -xO4 -dalign -xarch=v8plusa -xpg
# debugging and developement
 CFLAGS=-g

## ----------------------------------------------------------##
##  THIS SECTION REFERENCES THE NUMERICAL RECIPES LIBRARIES ##

# edit to point to directory containing the Numerical Recipes library
RECIPES_LIB=/home/ballen/numerical_recipes/lib

# edit to give the name of the numerical recipes library you wish to use:
# profiling
 LRECIPES=recipes_c
# speed and optimization
#LRECIPES=c_numrec
# debugging and developement
# LRECIPES=recipes_cg

## ----------------------------------------------------------##
## THIS IS THE SECTION THAT REFERENCES THE MPI/MPE LIBRARIES ##

# If you want to build the MPI code, set this variable to true, else to false
# If it is set to false, all the MPI defines below are ignored.
 BUILD_MPI = false
#BUILD_MPI = true
```

```
# edit to give the name of your local MPI C compiler (and any flags)
# MPICC = /usr/local/mpi/bin/mpicc -g
MPICC = $(CC)
# edit to give the names/paths of the MPI/MPE libraries you wish to use:
# if you do not want MPI/MPE, values are irrelevant (set to blank):
# You MAY need to include paths as well
MPI_LIBS = -L/usr/local/mpi/lib/solaris/ch_p4 -lmpe -lmpi -lsocket -lnsl
# MPI_LIBS = -L/usr/local/mpi/lib/solaris/ch_p4 -lmpi -lmpe
# MPI_LIBS = -lmpe
# edit to point to the MPI/MPE include directory
MPI_INCLUDES=-I/usr/local/mpi/include

## ------------------------------------------------------------##
## THIS IS THE SECTION THAT REFERENCES THE FRAME LIBRARIES   ##

# If you want to build code that depends upon the FRAME libraries,
# set this variable to true, else to false
# If it is set to false, the FRAME define is ignored
# BUILD_FRAME = false
BUILD_FRAME = true

# edit to point to the directory containing the FRAME directories
# include and lib. The required files are include/FrameL.h and
# lib/libFrame.a
FRAME_DIR=/home/dpetrovi/HOS/framelib/v3.74

## ------------------------------------------------------------##
## THIS IS THE SECTION THAT REFERENCES THE EPICS LIBRARIES   ##

# If you want to build code that will run on a real-time system (in the 40-meter lab)
# and gets the file names from there, you will need to have a set
# of epics libraries and include files available.
# unless you want to do this, please leave the following lines commented out!
# directories containing epics includes/libraries
#EPICS_INCLUDES=-I/home/ballen/epics2 -I/home/ballen/epics1
#EPICS_LIBS=-L/home/ballen/epics2                -L/home/ballen/epics1                -
D_POSIX_PTHREAD_SEMANTICS -lezca  -lca -lCom -lnsl -lsocket -ldl -lintl -lposix4
#BUILD_REALTIME = -DREALTIME

## ------------------------------------------------------------##
##  THIS IS THE SECTION THAT DEALS WITH CLAPACK
## REQUIRED IF YOU WISH TO 'CLEAN' DATA USING ENVIRONMENTAL
## INFORMATION

WITH_CLAPACK = false
```

```
#WITH_CLAPACK = false

#claplack library directory
  CLAPACK_LIB=/usr/local/CLAPACK/lib/
#
# Name of clapack and blas library
# Alpha
 LCLAPACK=lapack_alpha
 LBLAS=blas_alpha
# Sun
# LCLAPACK=libblas_solaris.a
# LBLAS=liblapack_solaris.a

# f2c library directory
  F2C_LIB=/usr/local/F2C/lib
  LF2C=F77
  LF2CI=I77
 F2C_INC=/usr/local/F2C/include

 CLAPACK_LIBS=-L$(CLAPACK_LIB) -l$(LCLAPACK) -l$(LBLAS) -L$(F2C_LIB) -
l$(LF2C)  -l$(LF2CI)

## ------------------------------------------------------------##
## THIS SECTION IS WHERE HARDWIRED OPTIMIZATIONS ARE SWITCHED

# The GRASP code includes some inline optimization tricks which speed
# up the code by having our own implementations of certain standard
# functions like pow(x,1.0/3.0) and sin and cos. This is at a price - these
# functions are less accurate than the routines that they replace.
# You should only uncomment the following lines if you are doing production
# work, and then only after verifying that the inline functions are
# sufficiently accurate for your purposes
# DEFINES = -DINLINE_CUBEROOT -DINLINE_TRIGS

## ------------------------------------------------------------##
## THIS SECTION IS WHERE ANY FUNCTIONS ARE OVERLOADED WITH.o
FILES
## FOR OPTIMIZATION PURPOSES, AND ANY EXTRA OPTIMIZATION LIBRAR-
IES ARE
## DEFINED. LEAVE THESE LINES COMMENTED OUT EXCEPT FOR BUILDING
## OPTIMIZED VERSIONS OF LIBRARY FUNCTIONS.

# optimization for the Intel Paragon Supercomputer (w/CLASSPACK library)
#                   OPTIMIZED_OBJECTS=$(GRASP_HOME)/src/optimization/paragon/
realft_paragon_risky.o
# OPTIMIZED_LIBS=-lkmath
```

```
# optimization for the Sparc (w/Sun Performance Library)
#    OPTIMIZED_OBJECTS=$(GRASP_HOME)/src/optimization/sparc/realft_sparc.o
#    OPTIMIZED_LIBS=-L/usr1/SUNWspro/SC4.2/lib -xlic_lib=sunperf
#    OPTIMIZED_LIBS=-L/apps/workshop/SUNWspro/lib -xlic_lib=sunperf

## ------------------------------------------------------------##
##  THIS IS THE SECTION THAT REFERENCES THE GRTOOLBOX WHICH
## PROVIDES A MATLAB INTERFACE TO GRASP AND THE FRAME
## LIBRARY.
##
## NOTE: IF YOU SET BUILD_GRTOOLBOX TO TRUE YOU MUST ALSO
##       HAVE SET BUILD_FRAME TO TRUE.
##


# If you want to build code that depends upon the GRtoolbox,
# set this variable to true, else to false
# If it is set to false, the varriables MEX, MEXFLAGS, and
# EXT varriables are ignored.
#
BUILD_GRTOOLBOX = true
#BUILD_GRTOOLBOX = false

# edit to the name of your mex compiler:
MEX = mex

# edit to include compilation flags that you desire:
# debugging and developement
MEXFLAGS=-g
# speed and optimization
#MEXFLAGS=-O

# Set the EXT variable to indicate your platform.
# Mex-files have different filenames for different
# platforms.
#
# Alpha
#EXT = axp
# HP9000 series 700
#EXT = mexhp7
# IBM RS6000
#EXT = mexrs6
# Linux
#EXT = lx
# SGI
```

```
#EXT = sg
# SGI 6.4
#EXT = sg64
# Solaris
EXT = sol
# SunOS4x
#EXT = 4


## ----------------------------------------------------------##
## NO EDITING SHOULD BE NEEDED BELOW THIS POINT

GRASP_I=$(GRASP_HOME)/include
```

---

---

When you have edited SiteSpecific file execute the shell script InstallGRASP in the top level GRASP directory, by typing the commands:

```
chmod +x InstallGRASP
./InstallGRASP name_of_the_SiteSpecific_file
```

From here on, the remainder of the installation should proceed automatically. Note that when installing GRASP you have the option of installing the Gravitational Radiation Toolbox. You should set this option to ‘true’ in the SiteSpecific file you edited if you want to use Gravitational Radiation toolbox.

Detailed description on how to use GRtool as part of the Gravitational Radiation Toolbox may be found in GRASP tutorial.