aLIGO-wxPython

Graphical User Interface for Gravitational Wave Data Analysis

Manuel Pichardo Marcano

SUMMER RESEARCH PROJECT, UNIVERSITY OF GLASGOW

HTTPS://GITHUB.COM/MANUELMARCANO22/ALIGO-WXPYTHON

This research was done under the supervision of Dr. Martin Hendry from June 18th to August 25th of 2015.

First release, August 2015

Contents

1. The Essentials

1.1 Python

1.1.1 Why Python?

Python is a widely use programming language. The community is a very active and growing one. It is free and open-source software with many open, popular libraries of scientific tools. Some of the most used ones being SciPy [1], NumPy [2], Matplotlib [3] and Astropy [4] . Besides these general reasons, there are other good Python resources more specific to Gravitational Waves data analysis. Three of them are described below:

LIGO Open Science Center: It provides an introduction to LIGO data files. The webpage provides a number of Python scripts to learn to download and plot actual LIGO data. The tutorials, including the first on how to install the necesarry programs, scripts and plot gallery can be accessed from the page `https://losc.ligo.org/tutorials/`.

GWpy: GWpy is a Python package with "tools and methods for studying data from gravitational waves detectors. The page `https://gwpy.github.io/` provides instructions for installation and short tutorial on its use.

LALSuite: LALSuite is maintain by the LIGO Scientific Collaboration Research Group (LSC). It is "compromised of various gravitational wave data analysis routines written in C." I can be installed as a RPM, Dev or from the Git repository. Instruction on how to install: `https://www.lsc-group.phys.uwm.edu/daswg/docs/howto/lal-install.html`. Installing from the Git repository didnt prove to be too difficult, but knowing these web resources were useful in the process:

Softaware prerequistes: This pages provides a step-by-step instruction to install the LSC software required for the LALSuite libraries and tools. (`https://www.lsc-group.`

[1]http://www.scipy.org/
[2]http://www.numpy.org/
[3]http://matplotlib.org/
[4]http://www.astropy.org/

phys.uwm.edu/daswg/docs/howto/lscsoft-install.html). Just make sure to install the right version. The page apparently is a bit old and I had a trouble when building LALSuite and had to installed a newer version. Also had a problem when running the ./configure for libmetaio. One solution was to comment out a line as was explain in this installation notes (http://www.ph.unimelb.edu.au/~cchung/installnotes.pdf). Also had to put on my .bashrc the line

```
export PKG_CONFIG=/usr/bin/pkg-config
```

I was having the same problem mentioned in this forum (http://ubuntuforums.org/archive/index.php/t-2217198.html)

SWIG: is an open source software tool used to connect computer programs or libraries written in C. To be able to use the LALSuite routines in Python when configuring LALSuite to install it need to include the prefix for swig-python:

```
$ ./configure --prefix=${LALSUITE_PREFIX} --enable-swig-python
```

R Besides the two LSC step-by-step instructios to install LALSuite and the LSC softwares, I list other pages that at some point on the installatin process of LALSuite routines were useful:

- Problem with *./configure* for lal : http://peekaboo-vision.blogspot.ca/2012/09/segmentation-algorithms-in-scikits-image.html
- Building LIGO's Global Diagnostic System (GDS): Useful installing Frame Library and other dependencies. http://www.spy-hill.net/~myers/help/ligo/GDS-build.html#framecpp
- Christine Chung LSC installation notes: http://www.ph.unimelb.edu.au/~cchung/installnotes.pdf
- http://smirshekari.wikidot.com/lal

1.1.2 The Python Must have

As stated above Python have many packages and libraries for many different purposes. The ones that I found essentials while working on the GUI were the following:

Packages:

These ones can be easily installed from the command line using the Python Package Index (PyPI). PyPI is a great tool to install the most up-to-date version of many python packages. It can be install from the command line on a Debian and Ubuntu machine running:

```
$ sudo apt-get install python-pip
```

Astropy contains tools to handle coordinate systems, units, and many more things useful for research in astronomy and astrophysics. I used it to change from gps time to local sidereal time on the observatories. http://www.astropy.org/

Numpy is a wonderful package to work with arrays. This tutorial is a great way to start learning about the powerful *Numpy arrays*. http://wiki.scipy.org/Tentative_NumPy_Tutorial.

SciPy have user-friendly and efficient numerical methods routines. http://www.scipy.org/.

Matplotlib. It is a very powerful tool to visualize data. A gallery with examples and the source code can be found here `http://matplotlib.org/`. Also the power of matplotlib can be extended using add-on toolkits. A toolkit for plotting 2D data on maps is called **Basemap** (http://matplotlib.org/basemap/).

1.2 wxPython: The GUI toolkit

wxPython is wrapper of a cross platform GUI library written in C++, wxWidgets. wxPython like wxWidgets is open source and cross-paltform. Its initial release was in 1988. So wxPython has been for a long time. This and the fact that there are many resources online makes wxPython a good choice to develope a GUI using Python.

R A great resource to start working in wxPython was the book written by one of its developers Robin Dunn. The book publish by *Manning Publications* in 2006 is called **wxPython in action**.

2. The Theory

The quest for gravitational waves started a while ago with the efforts of Joseph Weber. The first gravitational wave detectors where so called 'Weber bars' or resonant bar detectors. The idea was simple, to try to detect the waves by trying to measure the vibrations of the metal bar due to the in passing gravitational waves. There are still some functioning bar detectors around the world with great improve in sensitivities since the mid-60s. Recently the focus have to try to interferometers observatories, ground and space based interferometers. By examining closely the interference pattern of these couples of kilometers long interferometers higher sensitivities can be achieved and these observatories are the ones that will probable detect the first gravitational waves. In this report we will only focus and discuss the peculiarities of these type of detectors.

The challenge is hard, but the final goal remains: to be able to directly detected them and see the universe through a complete new window. To quantify detection and detectability a useful concept is the Signal-to-Noise Ratio (SNR). To calculated it we need to know about the source waveform, detectors antenna pattern and understand the different noise sources.

The subsequent sections will describe what we mean by SNR in the GW astronomy context, and explore in some detail what we need to know to calculated it. This approached was chosen since it will require to learn about waveform from compact binaries and its evolution, about different noise sources for different generations of interferometers, antenna patterns, power spectral density, and network of detectors.

2.1 Signal-to-Noise Ratio (SNR)

We first need to define what we mean by Signal-to-noise ratio. The ideal signal-to-noise ratio, ρ, characterizes the detectability of a signal in a detector with a given noise power spectrum. It is defined as :

$$\rho = \sqrt{4\int_{fmin}^{fmax} \frac{\tilde{\mathrm{h}}^*(f)\tilde{\mathrm{h}}(f)}{S(f)}df} \tag{2.1}$$

where $\tilde{\mathrm{h}}(f)$ is the Fourier transform of the quantity h called the *Characteristic strain*. The strain amplitude is a dimensionless quantity representing the amplitude of a gravitational wave. It gives the fractional change in displacement between two nearby masses due to the gravitational wave. The * denotes the complex conjugate. Gravitational radiation has two independent polarization states, $+$ and $\times$, and h is actually a linear combination of the two states, $h^2 = h_+^2 + h_\times^2$. How we can calculate the gravitational wave strain h from a source like a compact binary coalescence is discussed in the next subsection 2.2.

The other variable introduced was $S(f)$. $S(f)$ is the detector power spectral density. The square-root of the power spectral density, $S^{1/2}(f)$, is known as the *amplitude strain densitivity* or *amplitude spectral density (ASD)*. It has unit of $[\mathrm{Hz}^{-1/2}]$. This would be important when we discuss noise sources and how to get the ASD from all the noise sources in sec 2.3.3.

2.2 The Waveform

As we saw in the last section we can characterize the strength of a know signal in a detector by the quantity ρ using equation 2.1. The next question is how we can calculate h?

A lot of research have been made in this area to try to simulate and find the analytical expression from different gravitational waves sources (continuous, burst and coalescence). The strain, h, is time dependent but for simplicity we can work entirely on the frequency space. Different sources have different expressions for the characteristic strain, here using the quadrapole and stationary phase approximation we can calculate the real part of the strain in the frequency domain for a binary system in optimal location and orientation at a distance as:

$$\tilde{\mathrm{h}}(f) = \left(\frac{5\pi}{24}\right)^{1/2} \frac{G^2\mathscr{M}^2}{c^5 D} \left(\pi G \mathscr{M} f/c^3\right)^{-7/6} \tag{2.2}$$

where $\mathscr{M} = \mu^{3/5} M_{\text{total}}^{2/5}$ ($\mu = M_1 * M_2 / M_{\text{total}}$ being the reduced mass) is the *chirp mass*, D is the *luminosity distance*.

This is the expression we will use to calculate the SNR of a source with a given chirp mass and at a given distance.

With LALSuite routines installed we can generate different kind of waveforms in the frequency domain. An example using lalsimulation in python to generate the plus and cross polarization components for non-spinning binaries is:

```python
def fdwaveform(phiref, deltaF, m1, m2, fmin, fmax, dist, incl):
ampO = 0   # 0 pN order in amplitude
phaseO = 7 # 3.5 pN order in phase
```

```
approx = lalsimulation .TaylorF2 # Taylor F2 approximant
 fref = fmin

 hptilde , hctilde = lalsimulation .SimInspiralChooseFDWaveform(phiref, deltaF,
m1*lal.MSUN_SI, m2*lal.MSUN_SI, 0., 0., 0., 0., 0., 0., fmin,
fmax, fref , dist *1e6*lal .PC_SI, incl , 0., 0., None, None, ampO, phaseO, approx)

# return the frequency domain plus and cross waveforms
 return hptilde .data.data, hctilde .data.data
```

Listing 2.1: Frequency domain non-spinning inspiral waveform

2.3 Detectors

Now that we now how the signal looks like in the frequency domain we need to look at the Detectors response to the gravitational wave, and what limit its response. The specific examples on how to calculate the ASD and main source noises of a detector would be discussed only for a second generation one, but it applies also for the other generations.

2.3.1 Evolution

First Generation

This consist of 5 different observatories in three different continents. iLIGO (initial LIGO) and then eLIGO (enhanced LIGO) in Hanford and Livingston in the USA. Virgo in Italy, GEO 600 in Germany and TAMA 300 in Japan. For the analytical formulas of the noises for eLIGO see Appendix A.2 of *Creigton and Anderson* **Gravitational-Wave Physics and Astronomy** by WILEY publisher. It approximate values for parameters of the 4 km ideal initial LIGO and discusses the main source noises. There are also C routines part of lalsimulation to calculate the noises and ASD for eLIGO and other first generation detectors. Click here to see the list of the functions. See listing 2.2 in the next subsection for an example on using one of those functions part of lalsimulation in Python to calculate the ASD for two second generation detector, aLIGO and AdvVirgo.

Second Generation

Second generations include the two Advanced LIGO (aLIGO), Advanced Virgo and the Kamiola Gravitational Wave Detector (KAGRA). This last one was formerly the Large Cryogenic Gravitational Wave Telescope (LCGT).

As for the first generation we can use lalsimulation routines if we wish to generate the sensitivity curves and different noises for aLIGO and other second generation detectors. The list of the C routines to generate the different noise curves can be found in this webpage: `https://ligo-vcs.phys.uwm.edu/cgit/lalsuite/tree/lalsimulation/src/LALSimNoisePSD.c`. Two examples using those routines in Python would be:

```
import lalsimulation
import lal
import numpy as np
```

```

 [...]

#First way
#Create a FrequencySeries type:
psd = lal.CreateREAL8FrequencySeries('name', t0, 0., deltaF, lal.Unit(), len(hp)) # to hold psd
ret = lalsimulation.SimNoisePSDaLIGODesignSensitivityP1200087(psd, fmin)
psd.data.data[psd.data.data == 0.] = np.inf
asd = np.sqrt(psd.data.data) # convert PSD to ASD

#A more straight forward way for AdvVirgo:
freqrange = np.linspace(fmin, fmax, len(datapoints))
asd = np.zeros(len(freqrange))
for i in np.arange(len(freqrange)):
  asd[i] = lalsimulation.SimNoisePSDAdvVirgo(freqrange[i])
asd = np.sqrt(asd)

 [...]
```

Listing 2.2: Noise power spectrum for aLIGO and AdvVirgo

Where *CreateREAL8FrequencySeries* creates a frequency spectrum of data sampled over uniform frequency intervals. As it says in the documentation it includes extra fields "defining the sample frequencies, the timestamp of the spectrum, and the type of data being sampled." The arguments are the name of the data series, the start time of the time series in LIGOTimeGPS, the lowest frequency being sampled, in Hertz, the frequency sampling interval, in Hertz, and the physical units of the quantity being sampled. Click here for the documentation about the function. In the second way of doing it we simply evaluate the function *SimNoisePSDAdvVirgo* for every desired frequency.

Third Generation and Space-borne Observatories

- Einstein Telescope
- LISA Path finder and eLISA

2.3.2 Antenna Pattern

We can do it in two ways of doing it. An example code to do this in Python is in the git repository of the GUI here. We can also use a routine from lal to get the antenna pattern. Such a function on Python would look like:

```

def antenna_response( gpsTime, ra, dec, psi, det ):
gps = lal.LIGOTimeGPS( gpsTime )
gmst_rad = lal.GreenwichMeanSiderealTime(gps)

# create detector-name map
detMap = {'H1': lal.LALDetectorIndexLHODIFF, \
  'H2': lal.LALDetectorIndexLHODIFF, \
  'L1': lal.LALDetectorIndexLLODIFF, \
  'G1': lal.LALDetectorIndexGEO600DIFF, \
  'V1': lal.LALDetectorIndexVIRGODIFF, \
  'T1': lal.LALDetectorIndexTAMA300DIFF}

try:
    detector =detMap[det]
except KeyError:
    raise ValueError, "ERROR. Key %s is not a valid detector name." % (det)

# get detector
detval = lal.CachedDetectors[detector]

response = detval.response

# actual computation of antenna factors
fp, fc = lal.ComputeDetAMResponse(response, ra, dec, psi, gmst_rad)

return fp, fc
```

Listing 2.3: Antenna Response of a detector

The function defined above returns the beam pattern responses to the plus and cross polarization status of the gravitational waves. We see from the listing above that for each detector it depends on the time, position of the source in the sky and polarization angle. In theory it is frequency dependent too but in the long-wavelength limit the bean pattern functions become frequency-independent.

R Antenna pattern for different detectors and networks see Figure 6 of A Comprehensive Bayesian Approach to Gravitational Wave Astronomy

Also Chapter 6 of *Creighton and Anderson* **Gravitational-Wave Physics and Astronomy** by WILEY.

2.3.3 Main Noise Sources

Seismic

Thermal Noise

- Suspension Thermal Noise
- Coating Brownian Noise

Quantum Noise

2.4 Sky Source

Earth-fixed coordinate system

We can convert Right Ascension (ra) and Declination (dec) in the sky of the source at a given epoch to an Earth-fixed coordinate system.

2.5 Network

To get the average network SNR we add SNR of the individual detectors in quadrature. For example, for a network of two detectors the average SNR of the network, ρ_{net}, can be calculated doing:

$$< \rho_{\text{net}} >= \sqrt{\rho_1^2 + \rho_2^2} \tag{2.3}$$

The *network antenna pattern* P_N is simply:

$$P_N = \sum_I R_I P_I \tag{2.4}$$

where R_I is the *sensitivity ratio* for the I detector. If we choose a arbitrary a reference detector ref we compute R_I as:

$$R_I = \frac{\int_{-\infty}^{+\infty} df \frac{|h(f)|^2}{S_I(f)}}{\int_{-\infty}^{+\infty} df \frac{|h(f)|^2}{S_{ref}(f)}} \tag{2.5}$$

The integral are simply the SNR for each detector.

R

- A great website with a plot of some sensitivities plot and a paper in arXiv explain in the graph and all about the sensitivity curves is Gravitational Wave Sensitivity Curve Plotter
- Another great is the lectures and exercises of the Caltech Gravitational Wave Summer School. Specially the ipython notebook from the Dr. Weinstein.
- Book on the subject: *Creigton and Anderson* **Gravitational-Wave Physics and Astronomy** by WILEY publisher.
- For Network of detectors: *L. Wen and Schutz* discussion on chapter 5 of **Advanced Gravitational Wave Detectors** edited by David G. Blair.

TO DO

- Extend Third Generation discussion and Noises
- More benefit Network like sky position

3. Real Noise

3.1 Quadruple Suspension Design for Advanced LIGO

The routines part of lalsimulation are only approximation and theoretical models. To have a more realistic SNR calculations for aLIGO we need more accurate representation of the noises.

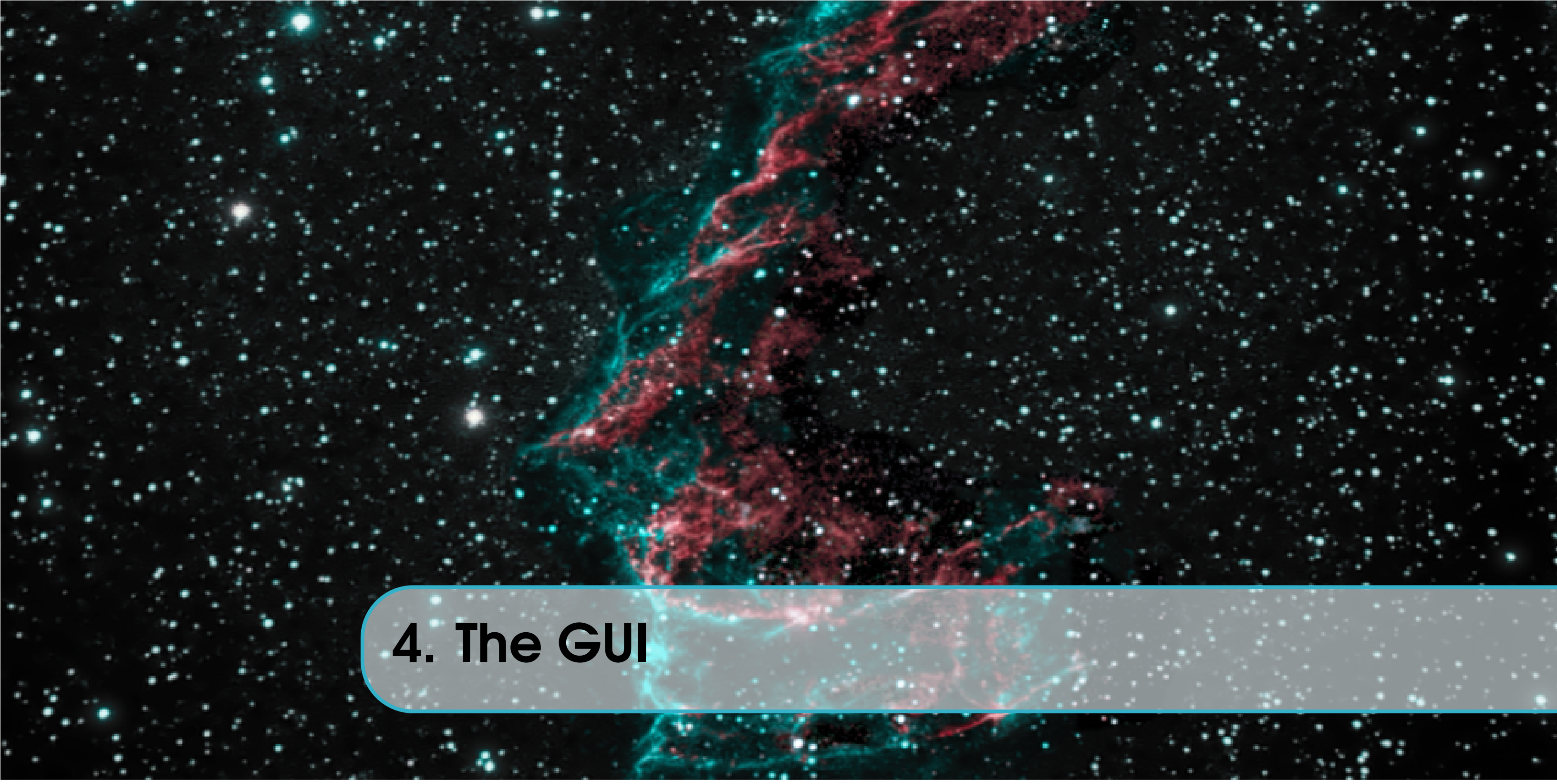
For the fist prototype of the GUI we will only consider the effect of changing the suspension design parameters on the SNR of a particular source. Dr. Giles Hammond and Ms Kyung Ha lee, both at the University of Glasgow, generated data for four different noises. Each data set is a 4500×10 .txt file that is read into the program to calculate the SNR for a given suspension configuration. aLIGO has a quadruple suspension and we can change three important things on the Suspension of the interferometer.The possible configurations for the three design parameters that we can change are:

- Total mass of the suspension. 9 possible masses from 40 kg to 120 kg.
- Total length. 3 possible configurations (1.6, 1.87 and 2.14 meters).
- Sections length (ribbons). Also 3 possibilities (0.6, 0.85 and 1.1 meters).

TO DO
- Ask Dr. Hammond and Kyung-Ha Lee on detail on computed data
- photo or diagram of the suspension

R

- `http://labcit.ligo.caltech.edu/docs/public_html/public/P/P020001-A/P020001-A.pdf`



4. The GUI

4.1 Putting it all together

For an specific compact binary system of the same mass and at a given distance it calculates the Signal-to-Noise Ratio, Horizon Distance. Average Range, Time to innermost stable circular orbits (ISCO), and Time in view for three different detectors configuration (LIGO Hanford, LIGO Livingston and a Network of the two detectors). Three design parameters can be change: 1-Total Suspension Length, 2- Section Suspension Length and 3-Total mass of the suspension. Can select from 9 different masses, and 3 lengths for both the total and section length. Four source parameters can be changed: 1-Mass of the two binaries, 2-Distance of the binaries, 3-Position in the Sky (in Earth-fixed coordinates), 4-Maximun gravitational wave (GW) frequency (GW freq. is twice the orbital frequency). It plots and calculate the amplitude spectral density (ASD) from the Seismic, Suspension Thermal, Coating Brownian and Quantum Noise.

The code assumes polarization angle of 0.343, observation time at $t_0 = 9 \times 10^8$ (gps sec), and inclination of 0 (best scenario). It is a stationary phase approximation.

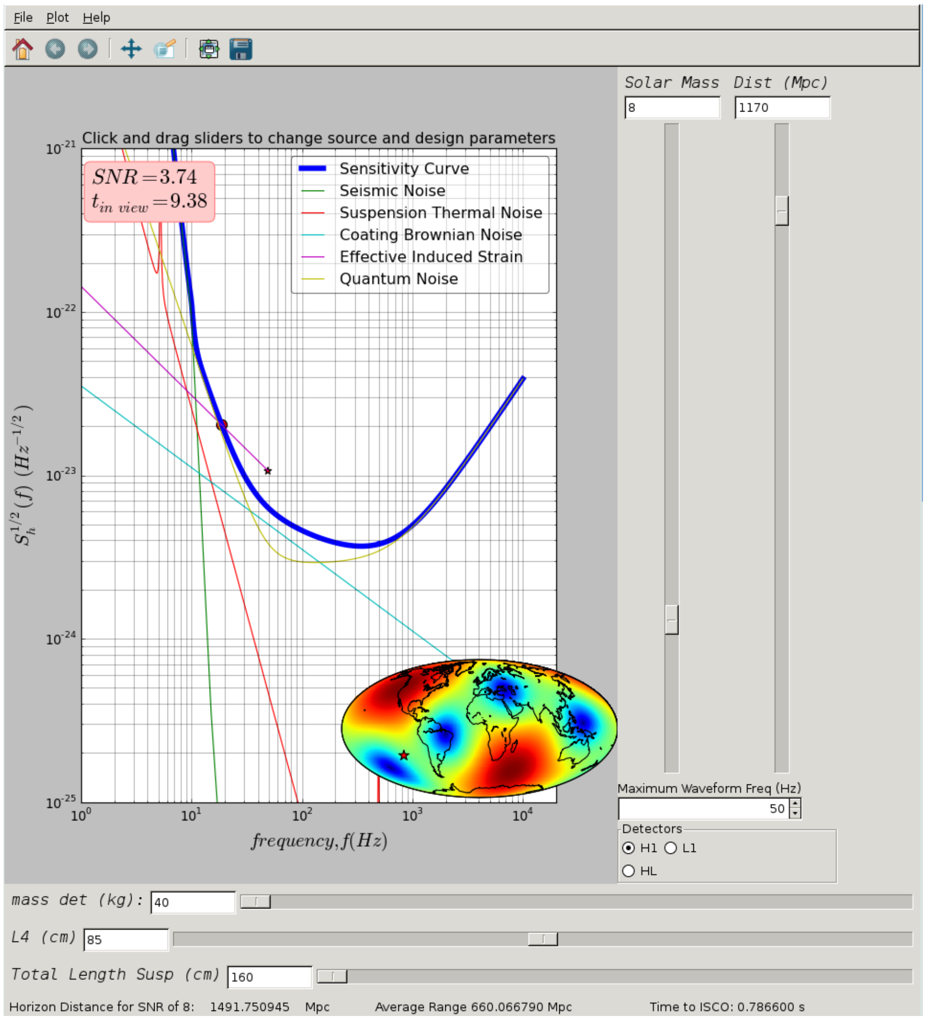
4.2 Assumptions

Stationary phase approximation. The code assumes polarization angle of 0.343, observation time at t0=9e8 (gps sec), and inclination of 0 (best scenario). Assumes no spinning binaries in a circular orbit.

4.3 Future Work

- Include gravity gradient noise
- Include time as parameter
- More data points for the sliders
- Include contribution to SNR beyond the ISCO
- Inclination as parameter or randomize

- Include eccentric and spinning binaries



Solar Mass: 8

Dist (Mpc): 1170

Maximum Waveform Freq (Hz): 50

Detectors: [✓ H1] L1 HL

mass det (kg): 40

L4 (cm): 85

Total Length Susp (cm): 160

Horizon Distance for SNR of 8: 1491.750945 Mpc Average Range 660.066790 Mpc Time to ISCO: 0.786600 s

Figure 4.1: GNU screenshot