Tuto 1.4 Generating waveforms

November 18, 2020

1 Gravitational Wave Open Data Workshop #3

Tutorial 1.4: Generating Waveforms We will be using the PyCBC library, which is used to study gravitational-wave data, find astrophysical sources due to compact binary mergers, and study their parameters. These are some of the same tools that the LIGO and Virgo collaborations use to find gravitational waves in LIGO/Virgo data

In this tutorial we will walk through how find a specific signal in LIGO data. We present how to generate the waveform of a gravitational-wave merger and matched filtering, which is optimal in the case of Gaussian noise and a known signal model. In reality our noise is not entirely Guassian, and in practice we use a variety of techniques to separate signals from noise in addition to the use of the matched filter.

Additional examples and module level documentation are here

Click this link to view this tutorial in Google Colaboratory

1.1 Installation (execute only if running on a cloud platform or if you haven't done the installation already!)

PyCBC is installable through pip. It relies on portions of the LALSuite c-library. A bundled version of this suitable for use with PyCBC is also available on Mac / Linux through pip. **It is recommended** to use conda on your own machine, as explained in the installation instructions. This usage might look a little different than normal, simply because we want to do this directly from the notebook.

[1]:
```
# -- Uncomment following line if running in Google Colab
#! pip install -q 'PyCBC==1.15.3' 'lalsuite==6.66'
```

Important: With Google Colab, you may need to restart the runtime after running the cell above.

1.2 Initialization

[10]:
```
%matplotlib inline

from pycbc.waveform import get_td_waveform
import pylab
```

1.2.1 Generate your first waveform !

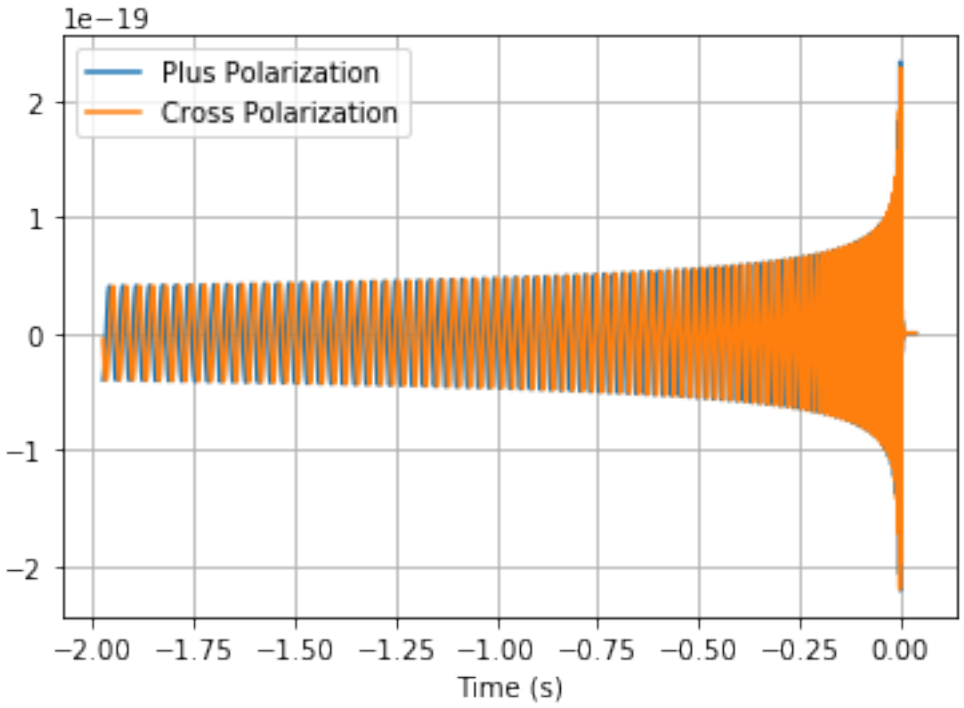
Here we'll generate the gravitational waveform using one of the available waveform approximants. These can be generated as a time series using `get_td_waveform`. There are some additional examples using this interface here. The key parameters are the masses of the binary (given in solar masses), the time between samples (in seconds), the starting gravitational-wave frequency (Hz) and the name of the approximant we'd like to generate. A variety of approximants are available that include different physical effects. A full review of the different models is outside of the scope of this tutorial.

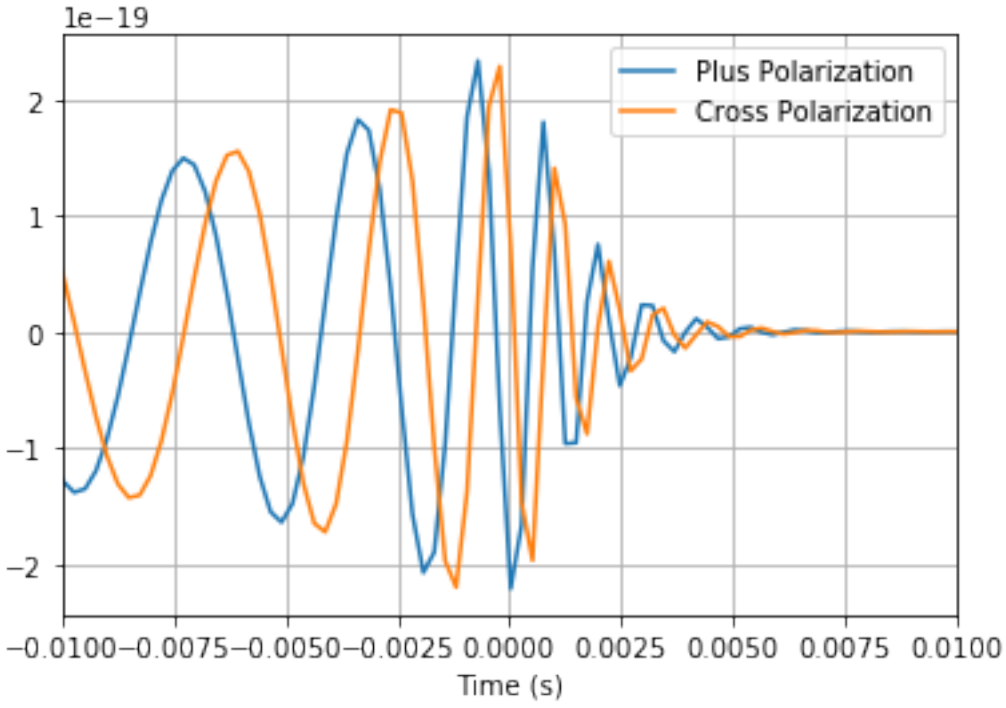
In this example, we've chosen to use the 'SEOBNRv4_opt' model. There are many others available as well with different methodologies and which include different physical effects. This is an implementation of the model introduced in this paper. It models the gravitational waveform of inspiralling and merging black holes, and includes the ability for each black hole to spin in the same direction as the orbit (aligned spin).

[11]:
```
# The output of this function are the "plus" and "cross" polarizations of the␣
 ↪gravitational-wave signal
# as viewed from the line of sight at a given source inclination (assumed␣
 ↪face-on if not provided)
hp, hc = get_td_waveform(approximant="SEOBNRv4_opt",
                         mass1=10,
                         mass2=10,
                         delta_t=1.0/4096,
                         f_lower=30)

pylab.plot(hp.sample_times, hp, label='Plus Polarization')
pylab.plot(hp.sample_times, hc, label='Cross Polarization')
pylab.xlabel('Time (s)')
pylab.legend()
pylab.grid()
pylab.show()

# Zoom in near the merger time#
pylab.plot(hp.sample_times, hp, label='Plus Polarization')
pylab.plot(hp.sample_times, hc, label='Cross Polarization')
pylab.xlabel('Time (s)')
pylab.xlim(-.01, .01)
pylab.legend()
pylab.grid()
pylab.show()
```





We can see that in the this case, the two polarizations differ only by the phase of the signal. This

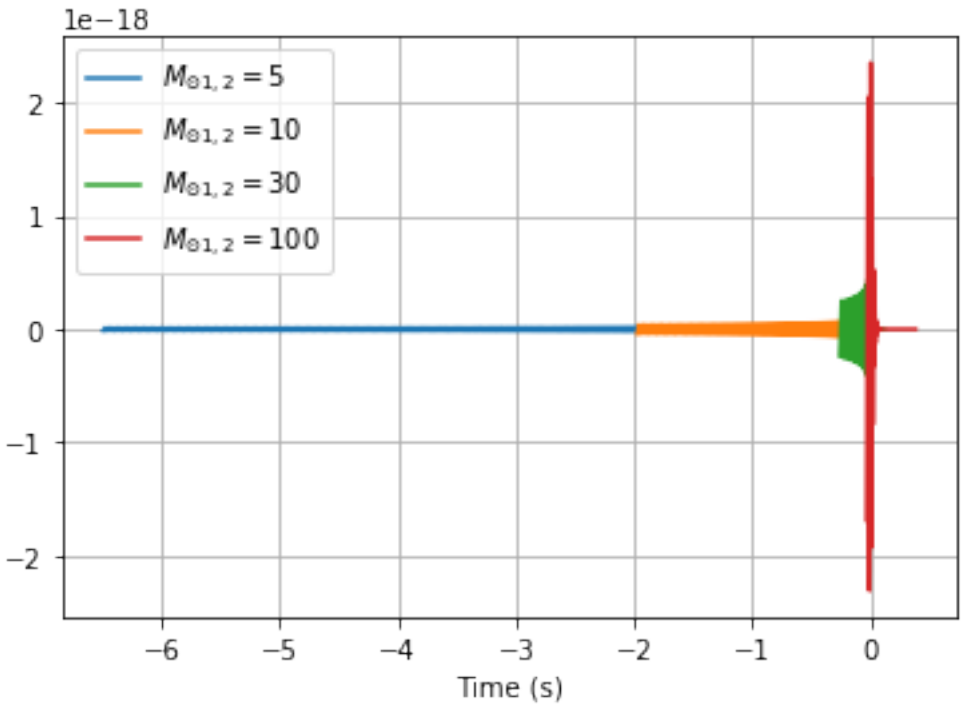
holds for systems where the orbital plane of the binary doesn't precess. In the zoom-in plot, we can see the merger itself and the ringdown that follows.

1.2.2 How does the waveform change with the mass of the binary?

Below you can see how the length of the waveform increases for lower mass binary mergers.

[12]:
```
# Component mass of each binary component. We'll simplify here and assume that␣
 ↪each
# component of the binary has the same mass. Again, units are in solar masses.
for m in [5, 10, 30, 100]:
    hp, hc = get_td_waveform(approximant="SEOBNRv4_opt",
                             mass1=m,
                             mass2=m,
                             delta_t=1.0/4096,
                             f_lower=30)

    pylab.plot(hp.sample_times, hp, label='$M_{\odot 1,2}=%s$' % m)
pylab.legend()
pylab.grid()
pylab.xlabel('Time (s)')
pylab.show()
```

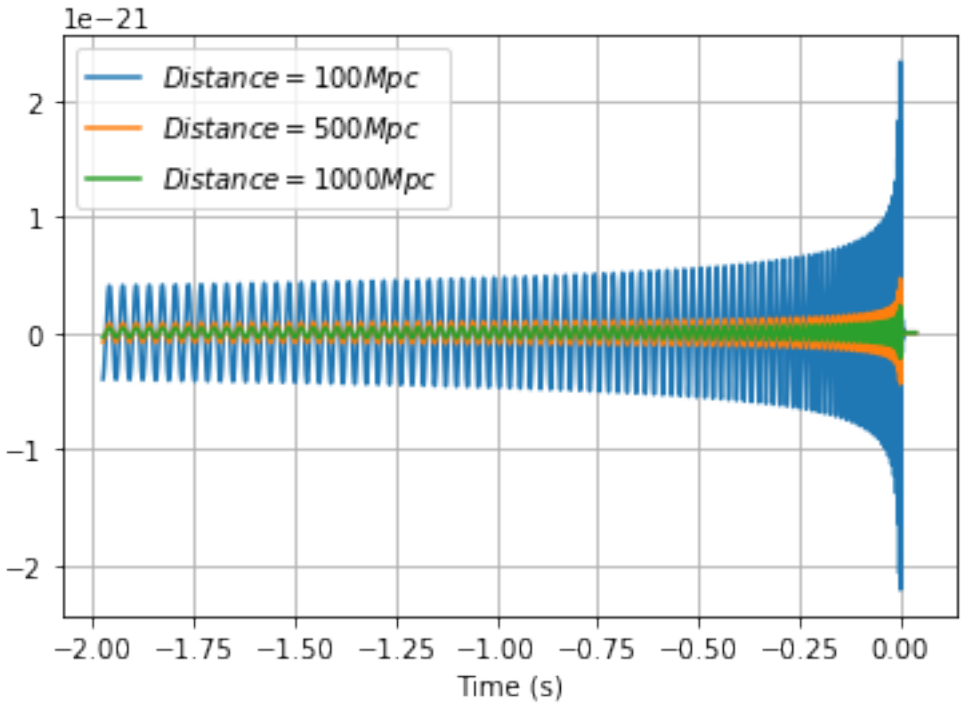


1.2.3 Changing the distance of the waveform

The distance of the waveform is also selectable when you generate a waveform. The units used are Megaparsecs. Keep in mind that no redshift effects are taken into account here, so there is a simple linear relationship between distance and amplitude

[13]:
```
for d in [100, 500, 1000]:
    hp, hc = get_td_waveform(approximant="SEOBNRv4_opt",
                             mass1=10,
                             mass2=10,
                             delta_t=1.0/4096,
                             f_lower=30,
                             distance=d)

    pylab.plot(hp.sample_times, hp, label='$Distance=%sMpc$' % d)
pylab.legend()
pylab.grid()
pylab.xlabel('Time (s)')
pylab.show()
```



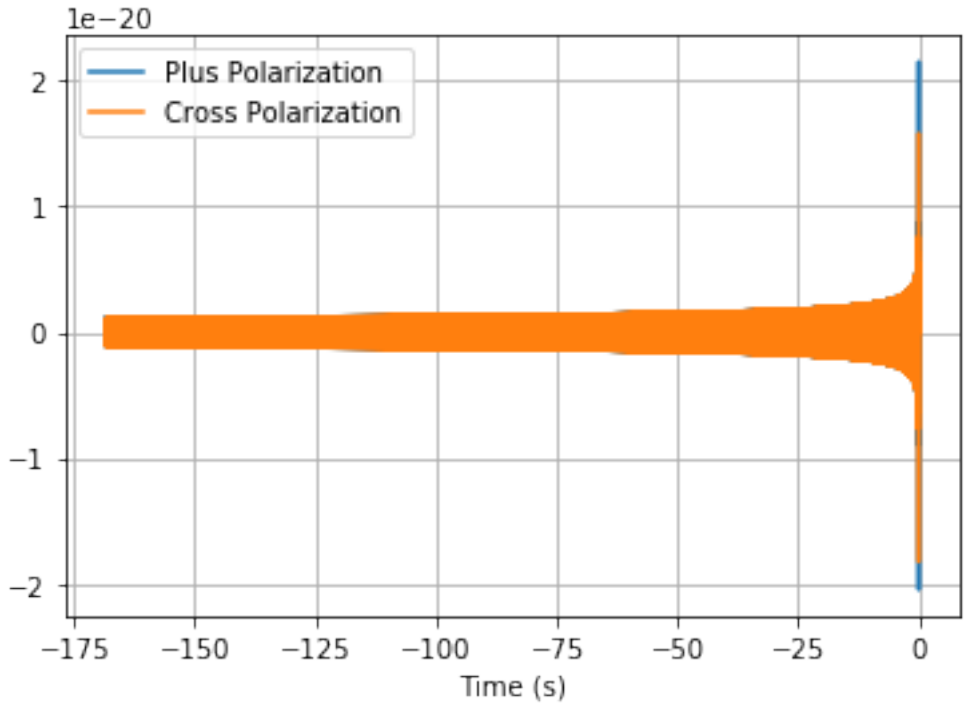
1.2.4 Exercise

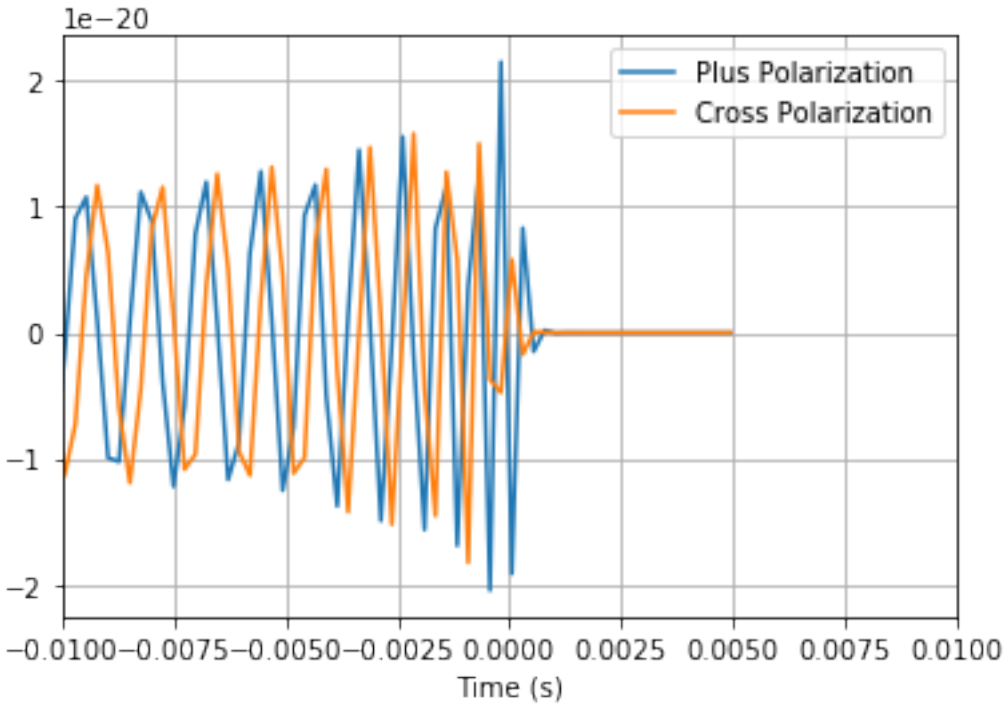
Generate and plot the waveform associated to the binary neutron star merger GW170817. Look up the GWTC#1 catalog page to obtain the estimated parameter for this source.

[18]:
```
hp_GW170817, hc_GW170817 = get_td_waveform(approximant="SEOBNRv4T",
                           mass1=1.46,
                           mass2=1.27,
                           delta_t=1.0/4096,
                           f_lower=20,
                           d=40)

pylab.plot(hp_GW170817.sample_times, hp_GW170817, label='Plus Polarization')
pylab.plot(hp_GW170817.sample_times, hc_GW170817, label='Cross Polarization')
pylab.xlabel('Time (s)')
pylab.legend()
pylab.grid()
pylab.show()

# Zoom in near the merger time#
pylab.plot(hp_GW170817.sample_times, hp_GW170817, label='Plus Polarization')
pylab.plot(hp_GW170817.sample_times, hc_GW170817, label='Cross Polarization')
pylab.xlabel('Time (s)')
pylab.xlim(-.01, .01)
pylab.legend()
pylab.grid()
pylab.show()
```




1e−20
2
1
0
−1
−2
−0.0100
−0.0075
−0.0050
−0.0025
0.0000
0.0025
0.0050
0.0075
0.0100
Time (s)
Plus Polarization
Cross Polarization

[ ]: