CA2_DataScience_Report_2025

September 11, 2025

Analysing gravitational wave signals

Data science - Coursework 2 (55%)

Deadline Thursday week 11, 2pm

This coursework assesses learning outcomes from **all chapters**, with particular focus on **Chapters 6, 7, 8, and 9**. You may complete this assessment **individually or in pairs**.

The assessment consists of:
* A written **Data Science Report** * A **short video** explaining your work

This Jupyter notebook contains all the questions you must complete for the Data Science Report. Work through the notebook step by step, answering each question in the space provided. Your final submission (converted to a PDF) should include both your code (with outputs) and your written explanations and interpretation.

Before starting:

- **Read the Instruction document in Learning Central > Assessment and Feedback > CA 2.**
- Read the Chapter 8 Jupyter Notebook
- Conduct any additional research needed, such as reviewing the LIGO consortium publication.

Gravitational waves are disturbances in the curvature of spacetime, generated by accelerated masses, that propagate as waves outward from their source at the speed of light. They are predicted in General Relativity and other theories of gravity and since 2017, they have now been observed!

In this exercise we will analyse some mock gravitational wave data from two unknown astrophysical objects merging together and coalescing. We will use a Monte Carlo Markov Chain (MCMC) to compare a scaled model that predicts how the wave changes depending on the total mass of the merging objects and their distance from us to the observed waveform. This will allow us to determine the nature of the orbiting objects that merged to form the gravitational wave using MCMC, whether for instance they could be originating from merging white dwarfs, neutron stars or black holes.

The mock or simulated waveforms measure the strain as two compact, dense astrophysical objects coalesce. The strain describes the amplitude of the wave. The system is parameterised by the masses of the merging objects, M_1 and M_2, and their distance from the observer D. Other useful parameters and equations relevant for this assessment are given in the Chapter 8 Jupyter Notebook.

Part A - Exploring the Observed Gravitational Waveform [15 marks]

We first need to access the observational data measured with the gravitational wave detectors (the waveform observed when two compact, dense astrophysical objects coalesce), and format it correctly so that later we can compare it with some model gravitational waves. We will use the `pandas` package to read in the data, with eg `dataIn=pd.read_csv('filename.csv')`.

Using `Observedwaveform.csv`, prepare the dataset for comparison with model waveforms.

1. Load and present the data.
   [5 marks]
2. Align the merger with $t = 0$s.
   [4 marks]
3. Quantify the noise in the waveform. Tip - estimate the average value of the noise and its standard deviation in the wave. This requires careful thought about where the noise can be seen in the waveform.
   [6 marks]

Part B Waveform Interpolation Exercise [10 marks]

In this exercise, you will learn how to match the time samples between a **reference waveform** and a **mock data waveform** with the same mass and distance. This step is important because in later parts of the assignment we will compare our observed gravitational wave data to a rescaled reference waveform. Here we will practice this method.

The reference waveform is given in `reference_waveform_40Msun_1Mpc.csv`. A mock dataset with the same parameters is in `mockdata_waveform_40Msun_1Mpc.csv`. Both reference and mock data are aligned with $t = 0$ at the merger.

1. Find the starting time of the mock data waveform, t_{min}. Keep only values with $t > t_{\mathrm{min}}$ and $t < 0$ (before the merger, and only where data exists). Verify that this has worked.

   [5 marks]

2. Convert the reference waveform so that it has the same time sampling as the mock data by interpolating the reference strain onto the mock data's time samples.

   [5 marks]

Hint: You can use the following interpolation pattern:

```
from scipy.interpolate import interp1d

# Create interpolation function
interp_fn = interp1d(ref_x, ref_y, bounds_error=False)

# Interpolate onto mock time samples
interp_strain = interp_fn(mock_x)
```

Part C – Using model waveforms to estimate the total mass and distance (a by-eye estimate) [23 marks]

You will need to remind yourself of how the time t and strain h of a gravitational waveform with $q = 1$, total mass M and distance D depends on mass and distance (see theChapter 8 Jupyter Notebook).

Using the reference waveform `reference_waveform_40Msun_1Mpc.csv` ($M = 40M_{sun}$, $D = 1$Mpc, $q = 1$) and the observed waveform `Observedwaveform.csv`:

1. Which *one* of the following code snippets (A–D) correctly scales and regrids the reference waveform to the observed time samples for a new total mass ($M = 70M_{sun}$, $D = 5$Mpc, $q = 1$)?
   [**5 marks**]

```
# reference data with mass and distance
refData == reference_waveform_40Msun_1Mpc.csv
# observed waveform - we don't know mass and distance yet
obsData == Observedwaveform.csv
# mock data with different mass and distance
70_Data == waveform_70Msun_1Mpc.csv

interp_fn = interp1d(refTime, refStrain, bounds_error=False)

Mref = 40.
Dref = 1.
Mnew = 70.
Dnew = 1.
```

Code A

```
def scale(tOut, ref_interp, mref, dref, mtot, dist):
    tRef = tOut * (mtot / mref)
    hOut = (mtot / mref) * (dref / dist) * ref_interp(tRef)
    return tRef, hOut

t, h = scale(obsTime, interp_fn, Mref, Dref, Mnew, Dnew)
```

Code B

```
def scale(tOut, ref_interp, mref, dref, mtot, dist):
    # determine the time in the reference_waveform that we need to interpolate from
    tRef = tOut * (mref / mtot)  # students need to rearrange this from the eq in notes
    hOut = (mtot / mref) * (dref / dist) * ref_interp(tRef)
    return tOut, hOut

t, h = scale(obsTime, interp_fn, Mref, Dref, Mnew, Dnew)
```

Code C

```
def scale(tOut, ref_interp, mref, dref, mtot, dist):
    tRef = tOut * (mref / mtot)
    hOut = (mref / mtot) * (dref / dist) * ref_interp(tRef)
    return tRef, hOut

t, h = scale(obsTime, interp_fn, mref, dref, Mnew, Dnew)
```

Code D

```
def scale_nointerp(tOut, ref_interp, mref, dref, mtot, dist):
    # Time scaling is correct
    tRef = tOut * (mref / mtot)
    # Strain scaling has the mass factor inverted (wrong!)
    hOut = (mref / mtot) * (dref / dist) * refStrain  # WRONG! should be mtot/mref
    return tRef, hOut
```

2. Justify your choice.
   [8 marks]
3. Use the selected scaling function from Part C1 to produce a scaled waveform and, by eye, estimate the total mass and distance that best fit the observed data (report values to within (e.g. to within +/- 5 Msun, +/- 100 Mpc). State the values you chose and verify they are correct.
   [10 marks]

Part D – Estimating the Total Mass and Distance Using MCMC [75 marks]

Using your scaled reference waveform function and matched time sampling from Part C3:

1. Choose **one** of the following options and implement an MCMC to sample the parameter(s):

   Option 1: Sample the total mass to find its best value.

   [20 marks]

   Option 2: Sample both total mass and distance to find their best values.

   [40 marks]

2. Display your results clearly and comment on them, including justifying whether the MCMC has converged.

   [20 marks]

3. Report the **median** and **90% credible limits** for the parameter(s). Compare the waveform generated from your MCMC result with the observed waveform, and comment on the agreement.

   [15 marks]

*Hints: * Think carefully about what the likelihood function will be in this case (see Chapters 6-9) since we are trying to find out how good our model is matching the data.*
** You should work with "log(Likelihood)" to avoid numerical errors - note this will affect both your posterior and the step in the MCMC chain where we usually write* $p_{\text{proposed}}/p_{\text{current}}$.

- *The step size between samples of the MCMC is quite important. A suggested value is* $0.1\,M_{sun}$.

- *The initial guess of your mass is also very important. You may find yourself getting into a local minimum rather than your code finding the true minimum. Try starting close to the estimate from Part C3.*
- *Test your MCMC on a small number of samples (e.g. 10-100) before trying it with a larger number.*
- *At the end, ask yourself if you need to include every sample?*
- *Depending on your step size, this part can take a long time to run. Suggest that you move all your plotting routines to a different code cell to save you re-running everything 10000s of times when you just want to change a plot command.*
- *To find out how long it will take for a Jupyter notebook to compile the MCMC code cell, add the following snippet to your code before you go into your MCMC loop (where Nsteps is the number of steps your MCMC is using):*

```
def time_spent_waiting(n):
from datetime import datetime,timedelta
preddur=[n*0.01,n*0.02]
print('predicted duration: {:.2f}-{:.2f} mins'.format(preddur[0]/60.,preddur[1]/60.))
return
```

Part E – Putting It All Together [32 marks]

1. Using your mass and distance estimates from previous parts (by-eye values or MCMC), determine the chirp mass and the individual masses of the merging bodies. Describe your reasoning and comment on the individual masses.

   [6 marks]

2. Estimate the period of the gravitational wave around its peak amplitude from your observed waveform outlining any assumptions.

   [10 marks]

3. Assuming non-spinning, circular orbits, use your measured period to estimate the orbital separation (in km) of the two bodies around the peak amplitude. Include all steps.

   [11 marks]

4. Comment on the astrophysical nature of the merging objects based on your analysis.

   [5 marks]

Understanding, Analysis, Interpretation and Presentation [25 marks]

Marks will be awarded for evidence of understanding, explanations, clarity of plots, interpretation of outputs, formatting, and overall presentation of results throughout the entire report. Bonus marks may be awarded for additional investigations or high-quality analysis.