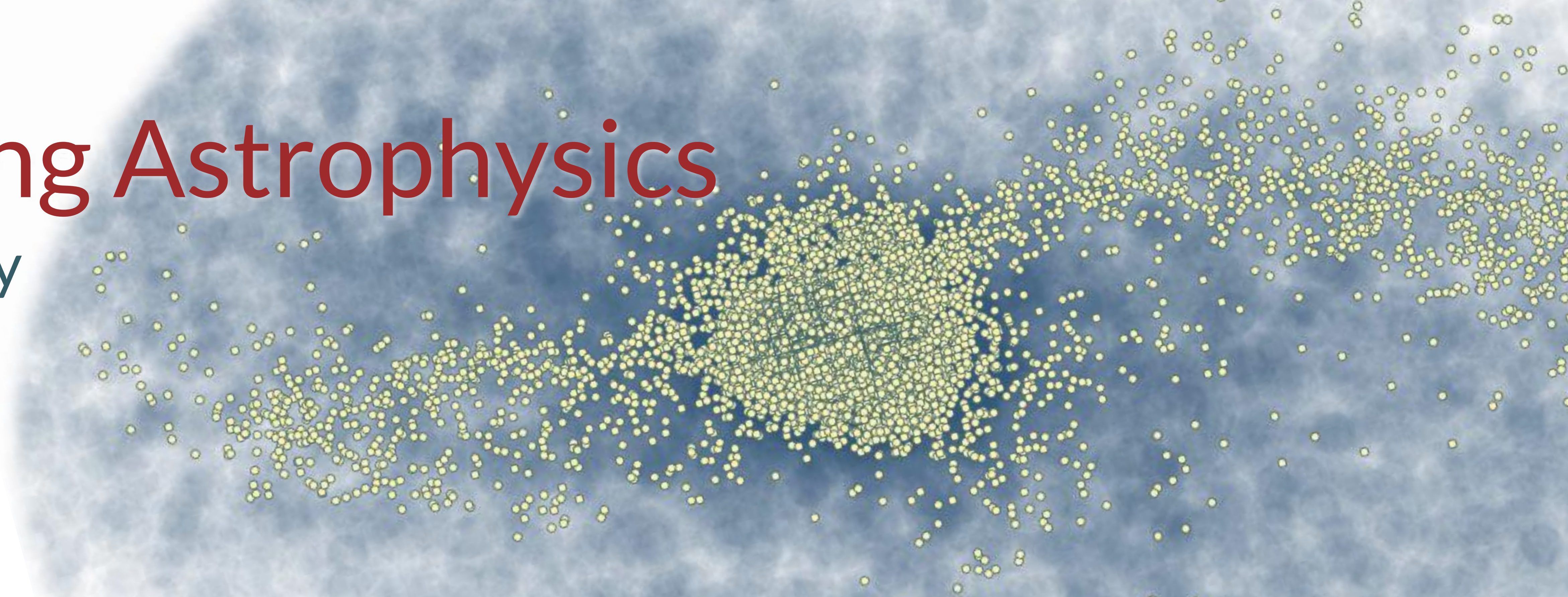
Teaching Advanced Python Techniques using Astrophysics

Joseph D. MacMillan, Ontario Tech University

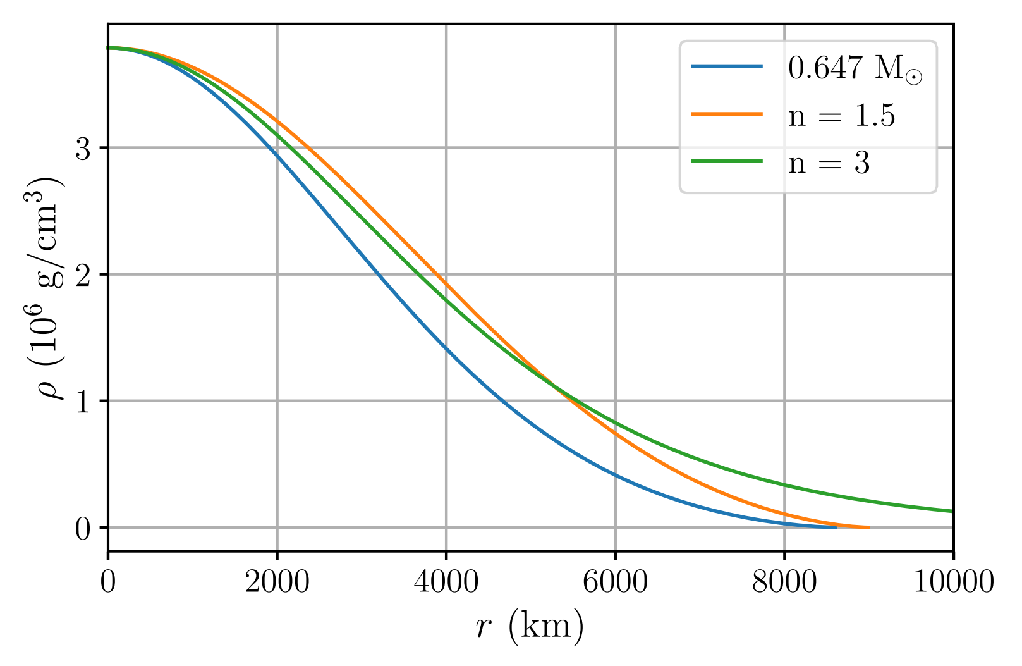


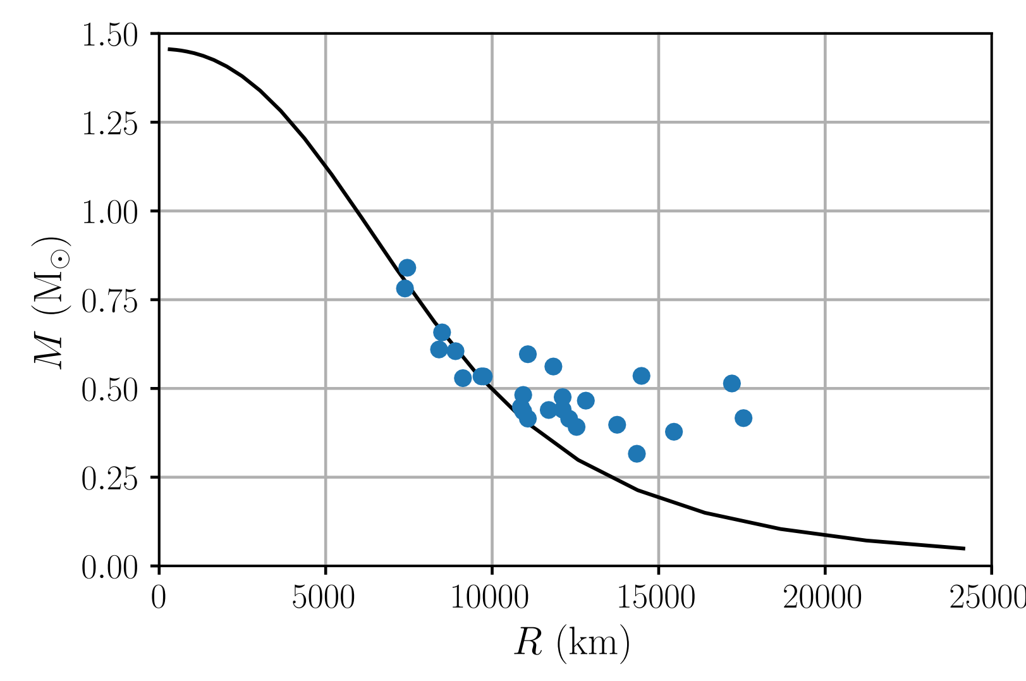
At Ontario Tech University we teach a course called PHY 4910U Techniques of Modern Astrophysics, which is required for our students who specialize in Astrophysics and a popular elective for other physics and computer science students. This course is a survey of a variety of computational techniques used in modelling stars, analyzing observational data, simulating astrophysical systems, and analyzing data from non-optical astronomical sources. Along the way, we train our students to use advanced features of the Python programming language and various libraries such as NumPy and AstroPy. Students are encouraged to use the Linux operating system and post code on GitHub. This poster showcases four specific examples of work from the course that illustrate the wide range of tools and skills students learn. All example code shown here is available on the course GitHub page: https://github.com/phy4910.

Example 1: Modelling White Dwarf Stars

We explore simple stellar models starting from the equations of hydrostatic equilibrium and Newtonian gravity. Polytropic models, with an equation of state with the pressure proportional to the density to some power ($P \propto \rho^\gamma$), are used as simple models of *relativistic* (γ = 4/3) and *nonrelativistic* (γ = 5/3) white dwarf stars. We then use a slightly more sophisticated model that combines both regimes and compare with observations of white dwarfs. Solving the differential equation for density using the 4th order Runge-Kutta method gives density profiles of different white dwarf stars; doing it 20 times allows students to plot the famous mass-versus-radius relationship and compare with observations (blue dots).

```python
def ode_rk4(x_start, x_end, h, y0, z0, f, g, stop = lambda : False, data = None):

    x = np.arange(x_start, x_end, h)
    N = len(x)
    y= np.zeros(N)
    y[0] = y0
    z = np.zeros(N)
    z[0] = z0

    for i in range(0, N-1):
        k1 = h * f(x[i], y[i], z[i])
        l1 = h * g(x[i], y[i], z[i])
        k2 = h * f(x[i] + 0.5 * h, y[i] + 0.5 * k1, z[i] + 0.5 * l1)
        l2 = h * g(x[i] + 0.5 * h, y[i] + 0.5 * k1, z[i] + 0.5 * l1)
        k3 = h * f(x[i] + 0.5 * h, y[i] + 0.5 * k2, z[i] + 0.5 * l2)
        l3 = h * g(x[i] + 0.5 * h, y[i] + 0.5 * k2, z[i] + 0.5 * l2)
        k4 = h * f(x[i] + h, y[i] + k3, z[i] + l3)
        l4 = h * g(x[i] + h, y[i] + k3, z[i] + l3)

        y[i+1] = y[i] + (k1 + 2.0 * k2 + 2.0 * k3 + k4) / 6.0
        z[i+1] = z[i] + (l1 + 2.0 * l2 + 2.0 * l3 + l4) / 6.0

        if stop(x[i+1], y[i+1], z[i+1], data):
            return x[0:i], y[0:i], z[0:i]

    return x, y, z
```





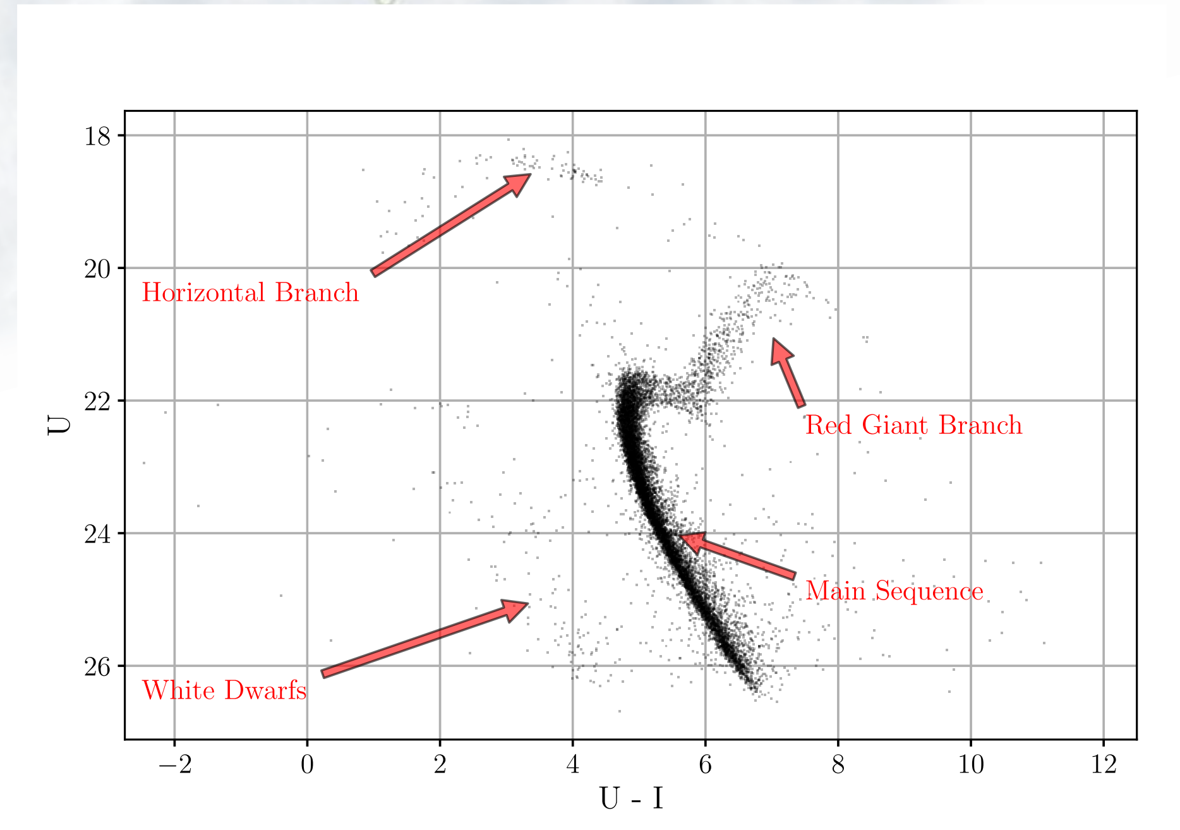
Astrophysics concepts: stellar modelling, fluid dynamics, Newtonian gravity, hydrostatic equilibrium, the Lane-Emden equation, white dwarf stars, equations of state, literature searching, data extraction.

Computational concepts: loops, arrays, saving data, reading data, plotting, functions, NumPy arrays and functions, the MatPlotLib library; students build a fourth order Runge-Kutta differential equation solver by hand.

Example 2: A Colour-Magnitude Diagram for a Globular Cluster

M22 is one of the brightest globular clusters in the sky. In this assignment, we find and download multiple science images from the Hubble Legacy Archive as FITS files, and use the popular AstroPy library to read and analyze those images. First, from three different filters, students combine the images to produce a false-colour photograph. Then all stars in the field of view are detected using the IRAF star finder method, aperture photometry is done, and a colour-magnitude diagram is produced.

```python
import matplotlib.pyplot as plt
import numpy as np
from astropy.io import fits
from astropy.visualization import LogStretch, PercentileInterval
from astropy.stats import sigma_clipped_stats
from astropy.wcs import WCS
from astropy.nddata import Cutout2D
from photutils import IRAFStarFinder, CircularAperture

# first read in the entire image and the wcs from the header
fi = fits.open("hst_12311_07_wfc3_uvis_total_sci.fits")['SCI']
header = fi.header
image_detection = fi.data
wcs = WCS(header)

# now cut out the central part ...
cutout = Cutout2D(image_detection, position=(3000, 3000), size=2800, wcs=wcs)
image_detection = cutout.data
wcs = cutout.wcs

# get image statistics
mean, median, std = sigma_clipped_stats(image_detection, sigma=3.0)

# search for the stars ...
iraffind = IRAFStarFinder(fwhm=3.0, threshold=10*std)
sources = iraffind(image_detection - median)

sources.write("stars.txt", format='ascii.fixed_width', overwrite=True)
```



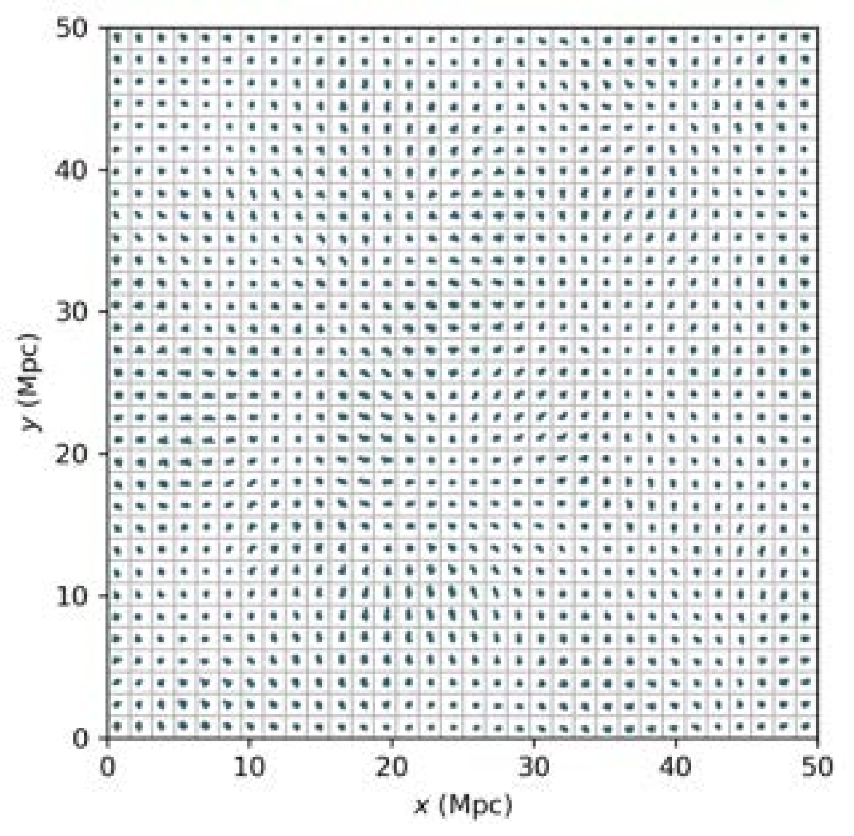
Astrophysics concepts: optical astronomy, right ascension and declination, filter systems, apparent and absolute magnitude, colour-magnitude diagrams, stellar evolution, the Hubble Legacy Archive, FITS file format.

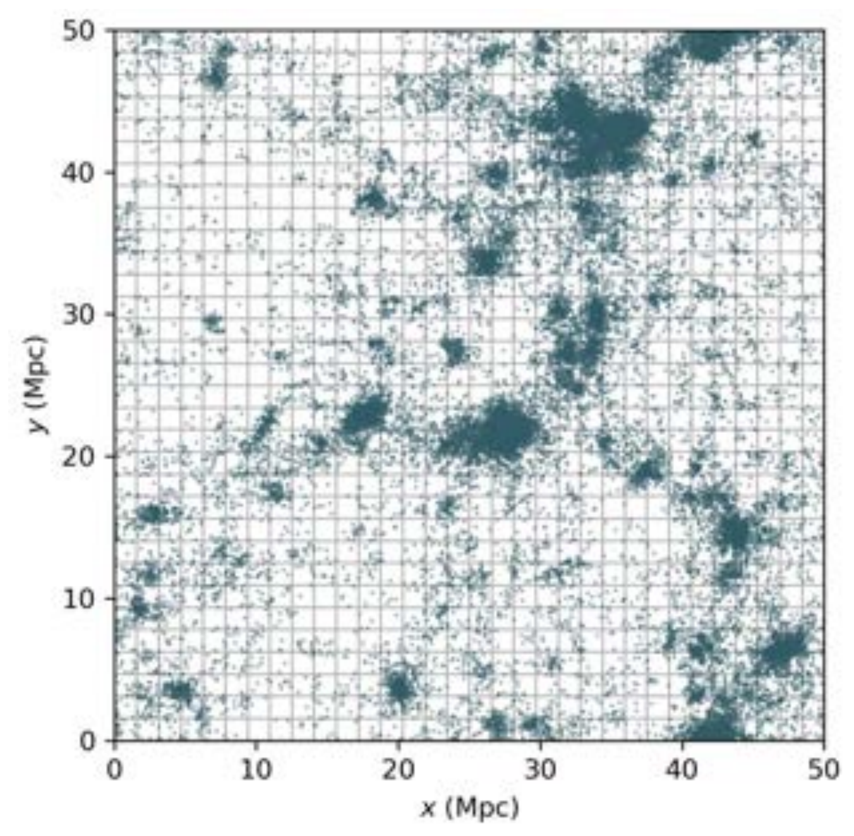
Computational concepts: the AstroPy and PhotUtils libraries, reading and manipulating FITS files and image data, transformations, image analysis, advanced plotting techniques.

Example 3: Simulating the Universe Using a Mesh Code

After exploring general *N*-body methods using our solar system and galaxy collisions as examples, we simulate a large cubical region of the universe. This is done by calculating the density on a mesh and using a Fourier transform to find the gravitational potential, using $N_p = 32^3$ particles and $N_G = 64^3$ mesh cells. The particles at the initial time (corresponding to a redshift of z = 30) are perturbed using standard techniques and provided to students. Realistic large-scale structure forms by z = 0 and students can do basic analysis (e.g., mass of largest structure formed) on the data.

```python
class Particle:

        def __init__(self, mass, pos, vel, ID = 0):
                self.mass = mass
                self.position = pos
                self.velocity = vel
                self.ID = ID
                self.accel = [0.0, 0.0, 0.0]

        def radius(self):
                return sqrt(self.position[0]**2 + self.position[1]**2 + self.position[2]**2)

        def distance_from(self, p):
                return sqrt( (self.position[0] - p.position[0])**2 + (self.position[1] -
p.position[1])**2 + (self.position[2] - p.position[2])**2 )

        def velocity2(self):
                return (self.velocity[0]**2 + self.velocity[1]**2 + self.velocity[2]**2)

class System:

        def __init__(self, parts, t = 0.0, G = 1.0):
                self.N = len(parts)
                self.time = t
                self.particles = parts

                self.T = 0.0
                self.U = 0.0

                self.G = G
```





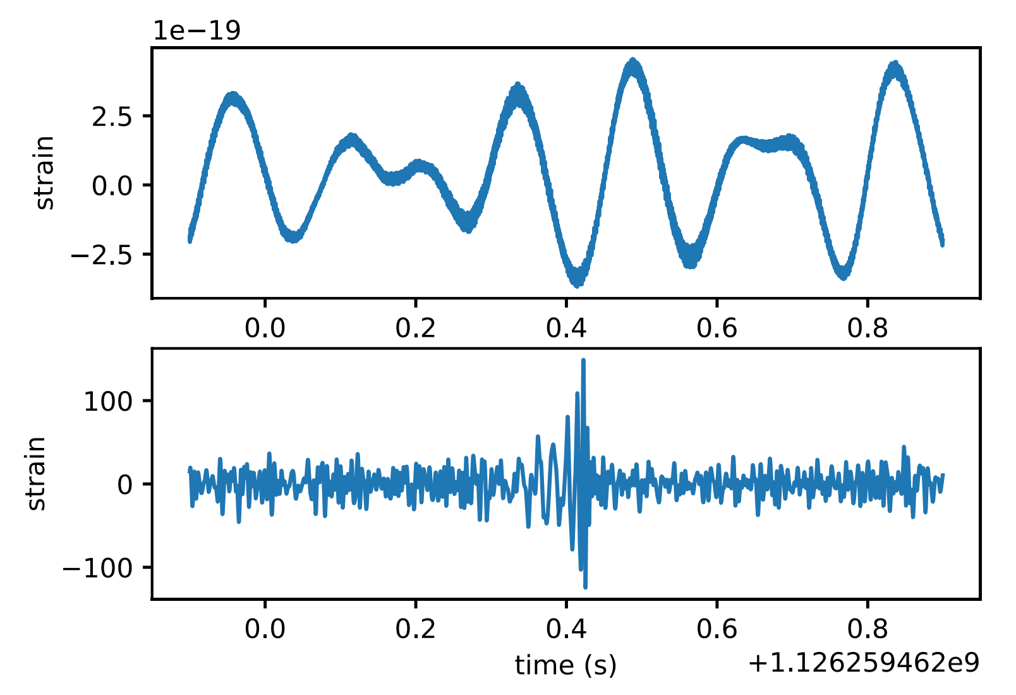
Astrophysics concepts: Gravity, N-body methods, cosmology (including the Robertson-Walker metric and the Friedmann equation, the Hubble parameter, and comoving coordinates), large-scale structure formation.

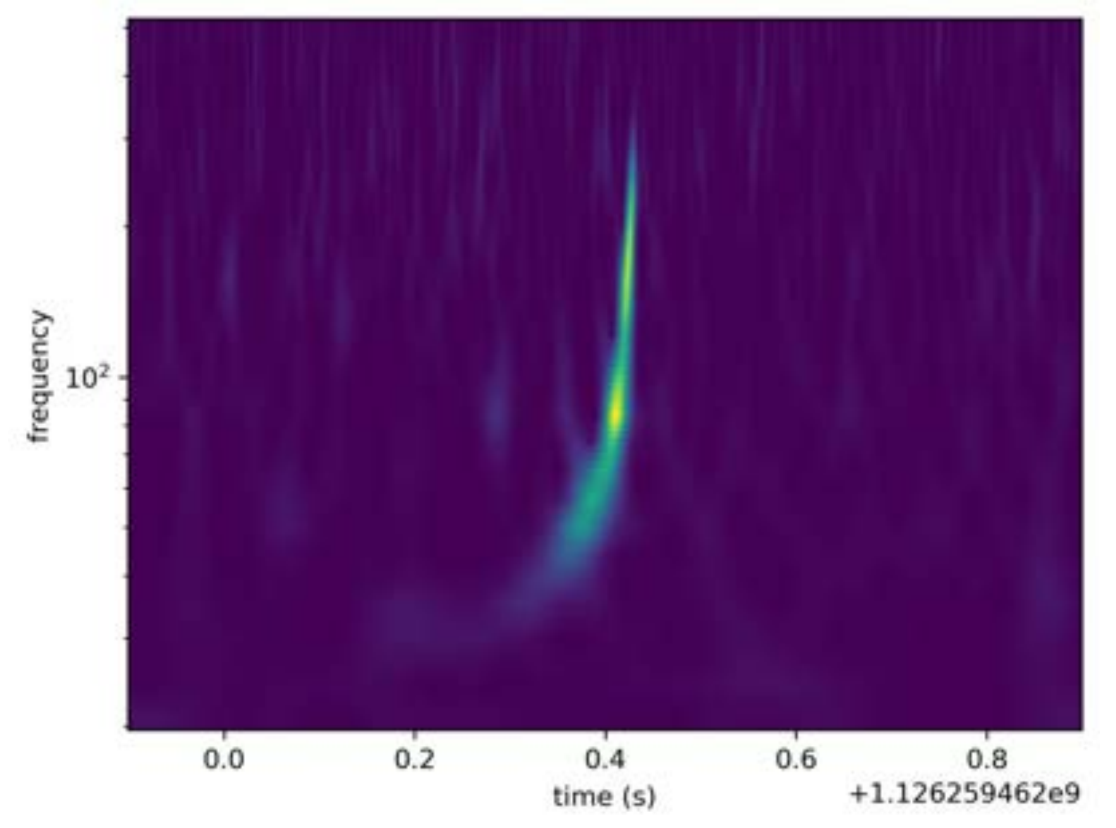
Computational concepts: Python classes, class and string methods, reading and writing binary data, main method, animations with MatPlotLib, fast Fourier transforms using NumPy, numerical integration using SciPy, periodic boundary conditions; students build and use their own Python library.

Example 4: Exploring Gravitational Waves from Black Hole Mergers

The successful operation of LIGO and other gravitational wave interferometers across the world have opened an exciting new avenue in astronomy. In this example, we take a close look at the signal from the Hanford site of LIGO of black hole merger GW150914, one of the earliest and strongest signals detected. The data is the strain $h = \Delta L/L$ – how much the space between interferometer mirrors is stretched and squeezed by the passing gravitational wave. After whitening and filtering the strain, a characteristic "chirp" signal is clearly visible in the data, and is even more clear in the time-frequency map ("Q-transform").

```python
from pycbc import catalog
import matplotlib.pyplot as plt
import numpy as np

# Get data from the catalog about this event
m = catalog.Merger('GW150914')

# look at the strain data from the Hanford site
h = m.strain('H1')
plt.plot(h.sample_times, h)
plt.title("Raw strain data for GW150914 H1")
plt.show()

# in terms of signal analysis, the first thing we want to do is whiten the data:
hw = h.whiten(4, 4) # 4 seconds of each sample used in PSD estimate, 4 second duration of
filter

# still lots of high-frequency and low-frequency noise, so let's run
hf = hw.highpass_fir(30, 512)
hf = hf.lowpass_fir(250, 512)

plt.plot(hf.sample_times, hf)
plt.title("Whitened and filtered strain data for GW150914 H1")
plt.show()

# what's that around the middle?  Zoom in (we know the actual merger time, so use that)
hz = hf.time_slice(m.time - 0.5, m.time + 0.5)
plt.plot(hz.sample_times, hz)
plt.title("Zoomed in on the signal")
plt.show()
```





Astrophysics concepts: Black holes and mergers, the basics of gravitational waves, interferometers, strain.

Computational concepts: Catalogue searching and signal analysis with the PyCBC library, high- and low-pass filtering, plotting images with MatPlotLib.

OntarioTech UNIVERSITY | PHYSICS