ASTR 310 Spring 2024 — Final Project

116 points total + 10 possible extra credit

Instructions

- The final project may not be submitted late, but you may upload draft files and replace them with a final version at any time up until the deadline. It's not as long as it looks, but don't wait until the last minute to start!
- As this project stands in lieu of a final exam, it must be your own work and must not be discussed with your peers prior to submission.
- We will answer clarifying questions, but we will not offer as much assistance as we do with homework assignments. This is a summative assignment meant to show what you can do on your own after taking the course.
- You are encouraged to consult the lecture and reading slides as well as Python, NumPy/SciPy, and Astropy documentation that you can find online. You can also ask us for clarification on the materials in the lecture and reading slides. Do not ask any other entities (peers, online forums, other faculty, or LLMs) to help or do the work for you; violations will be treated as academic integrity infractions.

---

1. Fourier Analysis

   Here we're going to construct the Fourier series for a discretely sampled function and show how the Fourier series allows us to approximate the original function.

   (a) Download the file `plot-data.csv` and read it into python. The data points $y(x)$ form the function we'll be analyzing. [3 pts]

   (b) Notice that the data points are not regularly sampled. Therefore, interpolate the data onto a regular grid of $x-$values. If you use somewhere between 200 and 1000 grid points you won't lose much information. For future reference, we will call the number of grid points N and the interpolated data points $f(x_\ell)$ for $\ell = 0, 1, ...N-1$. [3 pts]

   (c) Plot the original function and the interpolated function on the same plot, to verify that the interpolation hasn't done anything surprising. [3 pts]

   (d) Here we use the following set of definitions. The complex Fourier amplitudes are given by

   $$A_n = a_n + ib_n = \sum_{\ell=0}^{N-1} f(x_\ell) \exp(-2\pi i k_n x_\ell)$$

   and the original function can be reconstructed with those amplitudes by

   $$f(x_\ell) = \frac{1}{N} \sum_{n=0}^{N-1} \mathrm{Re}(A_n \exp(2\pi i k_n x_\ell)) = \frac{1}{N} \sum_{n=0}^{N-1} \Big[a_n \cos(2\pi k_n x_\ell) - b_n \sin(2\pi k_n x_\ell) \Big].$$

   Form the Fourier series of your interpolated data points. [4 pts]

(e) Reconstruct successive approximations to $f(x)$ by summing just the first few terms in the series. In other words, the $m-$th approximation $\tilde{f}_m(x)$ keeps terms up to m and can be written

$$\tilde{f}_m(x_\ell) = \frac{1}{N}\sum_{n=0}^{m} \mathrm{Re}(A_n \exp(2\pi i k_n x_\ell)).$$

Notice that the sum only goes up to m now instead of all the way to $N-1$. Two points of note: (i) The helper function `rfftfreq` gives you the spatial frequency $k_n = n/L$, where L is the period of the function in question (the total length of the dataset, here), and (ii) `rfft` throws away the information associated with negative frequencies so all the amplitudes aside from A_0 need to be multiplied by 2 to recover the original function correctly.

Plot the interpolated data with overlays of the successive approximations $\tilde{f}_m(x)$ for $m = 0, 2, 5, 15$, and 31. Include a legend so that we can distinguish which line is which. [10 pts]

(f) As a sanity check to verify that you understand the Fourier amplitudes, do the following computation. Choose $n = 1$; form the products $f(x_\ell)\cos(2\pi k_n x_\ell)$ and $f(x_\ell)\sin(2\pi k_n x_\ell)$, and make a new figure showing those products overlaid on the data points. Compute A_n from the explicit sum in the top equation and compare it to the corresponding value of A_n returned by `rfft`. [5 pts]

2. ODEs: Orbits

Finding the orbits of objects moving in a gravitational field basically amounts to solving the second-order ODE $d^2\mathbf{r}/dt^2 = -\vec{\nabla}\Phi$, where Φ is the gravitational potential. As you have seen, we tackle this second-order ODE by recasting it in coupled first-order ODEs:

$$\frac{dx}{dt} = v_x; \quad \frac{dy}{dt} = v_y; \quad \frac{dv_x}{dt} = -\frac{\partial\Phi}{\partial x}; \quad \frac{dv_y}{dt} = -\frac{\partial\Phi}{\partial y}.$$

(We're going to stick with 2D orbits.) Here we'll try a few different types of gravitational potentials and explore the shapes of the orbits produced in those potentials.

(a) Our old friend the Keplerian potential for a mass M at the origin of our coordinate system is $\Phi = -GM/r = -GM(x^2+y^2)^{-1/2}$. For simplicity we will work in a dimensionless coordinate system in which $GM = 1$. Thus $\frac{\partial\Phi}{\partial x} = x(x^2+y^2)^{-3/2}$ and $\frac{\partial\Phi}{\partial y} = y(x^2+y^2)^{-3/2}$. Define a function that will calculate the acceleration of a particle moving in the Keplerian potential. [4 pts]

(b) Now integrate to find the orbit of such a particle. I suggest you integrate from $t = 0$ to $t = 30$, starting with the initial conditions $x = 1$, $y = 0$, $v_x = 0$, and $v_y = 0.6$. You can use the leapfrog scheme outlined in our Reading/Lecture notes, or you can use the canned routines in SciPy. If you use the canned routines, note that the default Runge-Kutta 'RK45' method is not good at conserving energy. You'll get better results out of the 'DOP853' method. [5 pts]

(c) Plot your orbit shape (x vs y). You should find a closed elliptical orbit, just like Mr. Newton predicted. [3 pts]

(d) How well did you do with energy conservation? Compute the specific total energy $E = \vec{v}\cdot\vec{v}/2 + \Phi$ and plot it as a function of time. If you used the canned SciPy routines, try both 'RK45' and 'DOP853' methods, and plot the total energies of both solutions together on the same plot with appropriate labels. [4 pts]

(e) Now try a slightly different form of the gravitational potential. The Plummer potential is

$$\Phi = \frac{-GM}{\sqrt{r^2 + a_p^2}} = -GM(x^2 + y^2 + a_p^2)^{-1/2},$$

where a_p is a constant. This function has a smoother behavior at $r = 0$ and is a better match to the potential within a non-point mass distribution like the bulge of a galaxy. Write a new function to calculate the acceleration of a particle moving in the Plummer potential, and integrate it to find the orbit of a particle with the same ICs as you used for the Kepler potential. Try $a_p = 0.2$. Plot the orbit shape. [7 pts]

(f) Check energy conservation again for this orbit in the Plummer potential. [3 pts]

(g) Things get really wild when the potential is not axisymmetric. This can happen in a barred spiral galaxy, for example. The force on the particle will not be radial, and therefore the orbit's angular momentum is not conserved. Experiment with a bar potential:

$$\Phi = \frac{1}{2}\ln(R_c^2 + x^2 + y^2/q^2),$$

where R_c is another constant and q describes the flattening of the equipotential surfaces ($0 \leq q \leq 1$). This function gives $\partial\Phi/\partial x = x(R_c^2 + x^2 + y^2/q^2)^{-1}$. Write another function to integrate orbits in this potential; try $R_c = 0.14$ and $q = 0.8$, and integrate again for the same ICs you've been using. Plot the orbit shape.
This set of ICs describes an orbit with a relatively high angular momentum and while the angular momentum isn't constant, the orbit does at least have a constant sense of circulation around the origin. [7 pts]

(h) Try a couple of other sets of initial conditions. This potential produces lots of orbits that look a bit like Lissajous curves and have names like fish, banana, and pretzel orbits. In these orbits, the angular momentum can pass through zero, so you can find many of them by releasing your particle from rest at a variety of different positions that are not on the $x-$ or $y-$ axes. Plot at least two interesting orbit shapes. [6 pts]

(i) Verify that the orbits in the Plummer potential conserve angular momentum whereas the ones in the asymmetric potential don't. In other words, plot the (specific) angular momentum $\mathbf{L} = \mathbf{r} \times \mathbf{v}$ as a function of time for your orbit in the Plummer potential and for another in the asymmetric bar potential. [5 pts]

3. Work with a FITS data cube and velocity field

Here we'll carry out some relatively simple mathematical operations on a data cube to estimate the rotation velocity of a galaxy.

(a) Open and read the image cube FITS file `ngc4596.fits`. It gives the intensity of spectral line emission from the CO molecule, as a function of two spatial dimensions (positions on the sky) and one frequency or velocity dimension. (Frequency maps to velocity via the Doppler shift.) [2 pts]

(b) Inspect the FITS header and the world coordinate system (WCS) of the data cube. Notice that the FITS file's first axis is Right Ascension, the second axis is Declination, and the third axis is velocity. However, when the cubes are read into numpy arrays, the first axis becomes velocity, the second is Dec, and the third is RA. Compare NAXIS1, NAXIS2, and NAXIS3 to the shape of the numpy arrays so that you can satisfy yourself on this point. [2 pts]

(c) Integrated intensity image

Project the data cube into a 2D image by summing over the velocity axis. This is a simple way to create an integrated intensity image. [3 pts]

(d) Display the integrated intensity image with nicely formatted astronomical RA and Dec coordinates on the axes. Hint — if you read the file with `astropy.io.fits` you can create a wcs object from the header (see `astropy.wcs.WCS`). It will have 3 axes. To get correct axis labels on the 2D image, you will need to make a new wcs with only the 2 spatial axes; see `wcs.dropaxis()`. If you read the file with `CCDData.read`, it will ignore the velocity axis entirely, so you won't need to manually drop the 3rd axis.

Checkpoint: You should see a moderately bright peak in the center of the image and an extended, diffuse low surface brightness disk that is roughly elliptical in shape. Pixels around the outside edges and in the corners are filled with thermal noise. [4 pts]

(e) Velocity information

Use the header parameters of the original data cube to obtain the velocity information (the mapping between frequency channel number and Doppler velocity). CRVAL3 is the velocity value at channel or pixel number CRPIX3 and CDELT3 is the channel width (the step size between one channel and the next). Create 1D arrays storing the frequency channel information and the velocities of the corresponding channels. You'll need the velocity array for later analysis.

Checkpoint: You should find the spectrum covers velocities 1809 to 2249 km s^{-1} with a channel width of 0.63 km s^{-1}. (The FITS standard is 1-based, rather than 0-based like python.) [3 pts]

(f) Velocity field, version 1

We're going to estimate the velocity of the gas at each position in the galaxy. To explain how this is done, we need to talk through a little math first.

Consider an individual spectrum. It consists of intensity measurements (I_j) as a function of velocity (v_j), for a fixed spatial position. Here j is just a dummy index that runs over the velocity axis, which has N channels.

We calculated the integrated intensity by computing the sum $I_{tot} = \sum_{j=0}^{N-1} I_j$. If we wanted to calculate the average intensity $\langle I \rangle$ we would normalize that sum:

$$\langle I \rangle = \frac{\sum_{j=0}^{N-1} I_j}{\sum_{j=0}^{N-1} 1} = \frac{1}{N} \sum_{j=0}^{N-1} I_j.$$

We're writing the average in that slightly pendantic way in order to motivate the idea of a weighted average, which we'll do next.

Estimate the velocity at each position in the galaxy by calculating the intensity-weighted mean velocity at each spatial position. In other words:

$$\langle v \rangle = \frac{\sum_{j=0}^{N-1} v_j I_j}{\sum_{j=0}^{N-1} I_j}.$$

Compute the intensity-weighted mean velocity at each spatial position in the galaxy, as described above. For full credit, you should carry out this task without using explicit loops (e.g. `for` or `while`). You should end up with a 2D image where the values in the image are those mean velocities. [7 pts]

(g) Display your image of the velocity field, with proper astronomical coordinates on the axes. Include a colorbar (with a label) to help readers interpret the color information in terms of velocities.

Checkpoint: We should be able to see high velocities on one side of the galaxy and low velocities on the other side, because the galaxy is rotating. There will be lots of noisy junk around the edges of the image where there is no real signal. [4 pts]

(h) For the next iteration, we need an estimate of the noise value σ in the cube. You can obtain this estimate from the standard deviation of all the values in the cube. (Strictly speaking we should use some kind of clipping or outlier rejection to eliminate the signal before computing the standard deviation, but it's not a big deal in this case since there aren't very many pixels with signal.) [2 pts]

(i) As a sanity check on your estimate of the standard deviation, plot a histogram of the values in the datacube. Adjust the range of the histogram and/or the bin size so that you see a well-sampled, approximately Gaussian peak in the center of the histogram. This peak will come from the large number of pixels that contain no signal and only thermal noise, and its full width at the half-maximum points should be very roughly twice the rms value you calculated above. Label the histogram appropriately – the brightness unit in the datacube is given by the `BUNIT` parameter of the header. [3 pts]

(j) Velocity field, version 2

Now make an improved estimate of the velocity field. The idea here is to compute the mean velocity as in part (f) above, but rather than using all of the datapoints, to use *only* the datapoints whose intensities $I_j > 3.0\sigma$. Those intensities are unlikely to represent thermal noise and are more likely to represent real signal. Implement this selection criterion, recalculate the estimated velocity field, and display the new version. Include all the axis labels and the colorbar as before. You should have a much cleaner image, with only a few scattered noisy points outside the galaxy's disk. [7 pts]

(k) Velocity field, version 3

Make a third estimate of the velocity field. Use the selection criteria again, but this time instead of weighting by the intensity (I_j), weight by the intensity squared. Subtract your last two velocity field estimates, producing a 2D image of the difference. What is the maximum value of the difference in the center part of the galaxy? [7 pts]

(l) *Extra Credit*

Download the HST image `hst_15133_49_wfc3_uvis_f475w_idkv49_drc.fits`. It shows a similar region of the sky to the radio data cube. Make a cutout of the HST image with the appropriate size and shape such that the cutout shows the exact same region on the sky as the radio integrated intensity image (part c-d). Display the radio image next to the HST cutout. You can figure out how to specify the size and shape of the cutout by trial and error, or you can work with the two wcs's. Choosing the color scale on the HST image is nontrivial, but if you get it right, you should be able to see some darker dust clouds coincident with the radio disk. [10 pts]