ASTR 660 Homework 2

Deadline: start of class on 10/24

There is one question, worth 50 marks.

Please combine answers into a single file and upload to your GitHub repository as `homework2.pdf`. All code should also be committed to your GitHub repository and referenced in your answers.

Refer to presentation guidelines in homework 1. In addition, ANY USE OF CHATGPT OR SIMILAR AI TOOLS MUST BE CLEARLY DECLARED.

Background information

The Sersic profile is a widely-used model for the projected (i.e. 2-dimensional) surface density distribution of stars in elliptical galaxies:

$$\Sigma(r) = \Sigma_e \exp\left\{ -b_n \left[\left(\frac{r}{r_e} \right)^{\frac{1}{n}} - 1 \right] \right\} \tag{1}$$

In this equation, r_e is the radius that contains half the mass of the galaxy (in projection), $\Sigma_e \equiv \Sigma(r_e)$ is the surface density at that radius and n is a dimensionless parameter that controls the shape of the profile (called the Sersic index).

$\Sigma(r)$ is the stellar mass per unit area at a point r measured outwards from $r = 0$ at the centre of the galaxy[1]. If r is measured in kiloparsecs and masses are measured in $M_\odot$, then the units of $\Sigma(r)$ will be $M_\odot$, kpc^{-2}. In this homework we will assume circular symmetry.

The coefficient b_n is a function of n. For this exercise, we can assume that it is given by the following polynomial relation (which is actually a fit to a more complicated function):

```
def bn_approx(n):
    """
    A 4th-order accurate approximation for bn in the Sersic profile.
    Appropriate for n > 0.36.

    Ciotti & Bertin 1999
    A&A, v.352, p.447-451
    """
    A = 4.0/405
    B = 46.0/25515
    C = 131.0/1_148_175
    D = 2194697.0/30_690_717_750
    return 2*m - (1/3) + (A/m) + (B/m**2) + (C/m**3) + (D/m**4)
```

Note the use of underscores "_" to write long numbers in Python.

For more details about the Sersic profile see: `https://ned.ipac.caltech.edu/level5/March05/Graham/Graham_contents.html`.

---

[1] Don't get $\Sigma(r)$ mixed up with the *average surface density within r*, $\Sigma(<r) \equiv M(<r)/4\pi r^2$).

Question 1

1. Use `scipy.integrate.quad` to find the total ***mass*** projected between $r = 1\,\mathrm{kpc}$ and $r = 100\,\mathrm{kpc}$ in a galaxy described by the Sersic parameters $\Sigma_e = 1 \times 10^6\,\mathrm{M}_\odot$, $r_e = 2\,\mathrm{kpc}$ and $n = 4$. Give your answer to 5.s.f.

   Be careful: the profile is one-dimensional (a function of radius only), but the galaxy (in projection) is two-dimensional. In this case the double integral (most naturally written over angle and radius in polar coordinates) reduces easily to a single integral:

   $$.\int\int \Sigma(r)\, d\phi\, dr = 2\pi \int r\Sigma(r)\, dr$$

2. Write a **python** program to solve the same integral as part (1) using the trapezoid method using only loops (no arrays). Provide your code and its output. Quote the mass you obtain for 10, 100, 1000 and 10000 trapezoid steps to 5.s.f.

   Here is a template code for this part of the problem. You can make any changes you want to this template:

```
def trapz_integrate(f,a,b,nsteps,args=None):
    """
    Trapezoid integration of f from a to b, using nsteps steps.

    Pass function arguments as a tuple in args.
    """
    area = 0
    x0   = a
    w    = (b-a)/nsteps
    for i in range(0,nsteps):
        ... Apply the trapezoid method to accumulate area ...
    return area
```

3. Re-write the code in part (2) to use `numpy` arrays instead of a loop. How much faster is this version? Why is it faster?

4. Write a Fortran program to solve the same integral as part (1), again using the trapezoid method. See the template here: `https://github.com/nthu-astr660/examples/homework2/integrate_template.f90`.

   You do not need to use Fortran arrays, modules, interfaces or subroutines to solve this part, only loops and functions.

5. Download the code `https://github.com/nthu-astr660/examples/homework2/sphere.f90`.

   Compile and run this code without changing anything and provide the output in your answers. Then, identify the bug in the code and carefully explain why that bug led to the expected output.