Hand-in Exercise 3

Deadline: Nov 30, 23:59

Read these instructions carefully before you start coding.

This third hand-in exercise covers material from **lectures 7 through 9** (up to and including the FFT and applications). Unless noted otherwise, you are expected to code up your own routines using the algorithms discussed in class and **cannot use special library functions** (except exp()). Extremely simple functions like `arange`, `linspace` or `hist` (without using advanced features) are OK. When in doubt, write your own, or ask us. You can use the routines you wrote for the tutorials or previous assignments, however, your routines must be written **by yourself**. Codes will be checked for blatant copying (with other students as well as other sources), routines which are too similar get **zero points**.

For every main question you should write a **separate program**. We **must** be able to run everything with **a single call to a script `run.sh`**[1], which downloads any data (data needed for an exercise **may not** be included in your code package but needs to be retrieved at runtime), runs your scripts and generates a PDF containing all your source code and outputs **in the following format**:

- Per main question, the code of any shared modules.
- Per sub-question, an explanation of what you did.
- Per sub-question, the code specific to it.
- Per sub-question, the output(s) along with discussion/captions.

Your code may be handed however you'd like, for example by emailing a zip file or by sharing a github repository. If you organize your code in multiple folders, `run.sh` should be **in the top folder**. Have a fellow student test that your code works by running ./run.sh to make sure there are no permission errors! Exercises that are not run with this single command or do not have their code and output in the PDF get **zero points** (this includes solutions in Jupyter notebooks).

Ensure that your code runs to completion on the `pczaal` computers using `python3`(!). Codes that do not get **zero points**. Your code should have a total run time of **at most 10 minutes**, solutions generated will not be checked beyond this limit. If, **during testing**, a part of your code takes long to run, remember that you can run it once and read in its output (saved to file) in the rest of your code – however, the rules as stated above still apply to whatever you hand in at the end (everything run with a single command, all outputs produced on the fly, total runtime limit, etc). It is possible to test your own code on the pczaal remotely by using ssh, you can directly log into a pczaalXX computer (XX is between 00 and 21) using `ssh -XY [username]@pczaalXX.strw.leidenuniv.nl` to test your code.

For all routines you write, **explain** how they work in the comments of your code and **argue** your choices! Similarly, whenever your code outputs something **clearly indicate** next to the output/in the PDF what is being printed. This includes discussing your plots in their captions.

If a part of your code **does not run**, explain what you did so far and what the problem was and still include that part of the code in the PDF for possible partial credit. If you are unable to get a routine to work but you need it for a follow-up question, use a library routine in the follow-up (and clearly indicate that and why you do so).

Each sub-question starts with a reference to a relevant tutorial (*T*) and/or a reference to the relevant lecture (*L*). For example, *[L2,T2.3]* means the question is related to lecture 2 and question 3 in tutorial 2. Your previously written code for these will be a great starting point!

[1]See the example here: `https://github.com/Fonotec/NURSolutionTemplate`.

1. **Satellite galaxies around a massive central – part 3**
Recall the number density satellite profile from the first two assignments:

$$n(x) = A\,\langle N_{\rm sat}\rangle \left(\frac{x}{b}\right)^{a-3} \exp\left[-\left(\frac{x}{b}\right)^{c}\right]. \tag{1}$$

Here x is the radius relative to the virial radius, $x \equiv r/r_{\rm vir}$, with $x < x_{\rm max} = 5$, and a, b and c are free parameters. $A = A(a, b, c)$ normalizes the profile and $\langle N_{\rm sat}\rangle$ is the mean number of satellites in each halo.

In previous assignments we explored the properties of this model and sampled from it. This time, we're going to fit it to some data. Have your `run.sh` download these files (do NOT directly include them in the solutions you hand in, they are too large):

```
https://home.strw.leidenuniv.nl/~daalen/files/satgals_m11.txt
https://home.strw.leidenuniv.nl/~daalen/files/satgals_m12.txt
https://home.strw.leidenuniv.nl/~daalen/files/satgals_m13.txt
https://home.strw.leidenuniv.nl/~daalen/files/satgals_m14.txt
https://home.strw.leidenuniv.nl/~daalen/files/satgals_m15.txt
```

Each file contains haloes in a certain halo mass bin with variable numbers of satellites. For example, the first file contains over 20 000 000 low-mass haloes, most of which contain zero satellites, and the last contains only two (cluster-sized) haloes, each containing several hundred satellites.

Since satellite galaxies are discrete objects, we are dealing with a Poisson realization here, and we should therefore write down a Poisson likelihood. In this exercise, we will compare the results of an unbiased fit with that of an "easy" χ^2 fit with the correct Poisson variance $\sigma^2 = \mu$. Regardless of how we fit the profile, keep in mind that A depends on the parameters a, b and c, so recalculate A with your own integration routine (assignment 1) at every step!

(a) (5 points *[+2]*) *[L7]* First we'll take the (incorrect but easy) χ^2 approach. Choose any number of radial bins, space (real or log), and range (e.g. $x \in [0, x_{\rm max}]$). Argue your choices. Then, separately for each file, calculate $\langle N_{\rm sat}\rangle$ and bin the satellite galaxy radii to get a mean number of satellites *per halo* in each radial bin, N_i. Minimize a χ^2 for each, using that both the model mean and variance for bin $[x_i, x_{i+1}]$ are given by $\tilde{N}_i = 4\pi \int_{x_i}^{x_{i+1}} n(x) x^2\,{\rm d}x$. You can reuse one of the minimization routines you wrote for assignment 2, or, for bonus points, write and use a Levenberg-Marquardt routine. For each of the five data sets, report $\langle N_{\rm sat}\rangle$, your best-fit a, b and c, and the minimum value for χ^2. Also plot your binned data together with the best-fit profiles in a log-log plot ($\tilde{N}_i$ vs x).

(b) (8 points) *[L7,T7.2]* Now we take the correct, unbiased approach. Write down a Poisson log-likelihood for model counts $\mu = \tilde{N}_i$ and N_i mean observed counts in each bin. Explain the steps you take. Either use the same bins as in (a) or no bins at all. For each of the five data sets, minimize $-\ln \mathcal{L}(a, b, c)$ using one of the minimization routines you wrote for assignment 2 (or write a new one), and again report your best-fit a, b and c, and the minimum value for $-\ln \mathcal{L}(a, b, c)$. Also plot your binned data together with the best-fit profiles in a log-log plot.

(c) (5 points *[+2]*) *[L8]* Now we want to see which of the two methods yielded the models most consistent with the data. Since the data is binned, we'll do a G-test (if you did not use bins for (b), bin the best-fit model by integrating over the bins from (a), or, for bonus points, use a K-S test for (b) instead). Calculate the statistic G for each method and each data file (so 10 total). Argue what the number of degrees of freedom is for each data set and calculate the significance Q of G using the χ^2 distribution.[2] Comparing the Q-values for each data set between the two methods, what can you conclude?

(d) **BONUS QUESTION** *(4 points) [L8]* For bonus points, do Monte Carlo simulation, as follows. Choose one particular data file and take your corresponding best-fit parameters for both (a) and (b). For both sets of parameters, generate another set of haloes, using the same number of haloes and $\langle N_{\rm sat}\rangle$ as in the original data files, by sampling from these distributions as in assignment 2. Find the best-fit parameters for each mock data set using the corresponding

[2]You can use a library function for the (incomplete) gamma functions.

method. Repeat this process as many times as you can (using your original best-fit parameters to sample each time) and plot the best-fit profile for each sample in one plot for each method (so one plot for χ^2, one plot for Poisson likelihood). Overplot the original best fits in a different colour, and also plot the mean profile of all resamplings. Does the mean reasonably match the original profile in both cases? What can you conclude from this experiment?

2. **Calculating forces with the FFT**

When performing a numerical simulation with a lot of particles, one classical challenge is that calculating the gravitational forces between each pair of particles is highly time-consuming. If every time step each of the N particles needs to know the gravitational pull of each of the other $N-1$ particles, we're naively dealing with an $\mathcal{O}(N^2)$ calculation. Luckily, there are several (very different!) ways of making this an $\mathcal{O}(N \log N)$ problem, one of which is using the FFT.

Our "simulation" will be a 3D volume of $16 \times 16 \times 16$ units, with 1024 particles that each have a mass of 1 unit. The volume has periodic boundary conditions: that is, the edges connect such that $x = 16 \equiv 0$ (and the same for y and z). Each particle has a coordinate $(x_\mathrm{p}, y_\mathrm{p}, z_\mathrm{p})$. Generate these particles **exactly** as follows:

```
#!/usr/bin/env python3
import numpy as np
np.random.seed(121)
positions = np.random.uniform(low=0,high=16,size=(3,1024))
```

(a) (3 points) *[L1, H1.1]* We first need to interpolate the particles to a grid so we can use the FFT. Take a cubic grid with 16^3 grid points. The grid points are each in the center of a cell with boundaries $[i, i+1]$, $[j, j+1]$, $[k, k+1]$ in x, y, z, so grid point g_{ijk} has coordinates $(x_\mathrm{g}, y_\mathrm{g}, z_\mathrm{g}) = (i + \frac{1}{2}, j + \frac{1}{2}, k + \frac{1}{2})$. We'll assign a mass to each grid point using the Cloud-In-Cell method: for each particle, determine the 8 grid points it is closest to (i.e. those that would fall in a box around the particle, don't use the Euclidean norm) and add a fraction of its mass to these 8 points, given by the weights $(1 - |x_\mathrm{p} - x_\mathrm{g}|)(1 - |y_\mathrm{p} - y_\mathrm{g}|)(1 - |z_\mathrm{p} - z_\mathrm{g}|)$.[3] Finally, convert the mass assigned to each grid point to a density contrast $\delta = (\rho - \bar{\rho})/\bar{\rho}$ by using that for the mean density $\bar{\rho} = 1024/16^3$.

Plot 2D slices of your grid at $z = 4.5$, 9.5, 11.5 and 14.5, showing the δ assigned to each grid point (e.g. as a colormap).

Hint: don't forget that there are periodic boundary conditions, so one of the closest grid points for a particle near $x = 0$ for example may be at $x_\mathrm{g} = 15.5$.

(b) (7 points) *[L9, T9.1]* To calculate the gravitational forces at each grid Poisson, we need to know the potential, which is given by the Poisson equation: $\nabla^2\phi = \nabla \cdot \nabla\phi = 4\pi G\rho = 4\pi G\bar{\rho}(1 + \delta)$. We only need to solve the part of the equation with a spatial dependence, $\nabla^2\Phi \propto \delta$. This we can easily do in Fourier space, by taking the FFT, calculating the transformed potential and then taking the inverse FFT:

$$\nabla^2\Phi \propto \delta \xrightarrow{\text{FFT}} k^2\tilde{\Phi} \propto \tilde{\delta} \xrightarrow{\text{rewrite}} \tilde{\Phi} \propto \frac{\tilde{\delta}}{k^2} \xrightarrow{\text{iFFT}} \Phi. \qquad (2)$$

Calculate the potential Φ for the grid you calculated at (a) and plot the results for the same slices as before. Also plot the log of the absolute value of the Fourier-transformed potential, $\log_{10}(|\tilde{\Phi}|)$, for the same slices.

Most recent changes:

Nov 21: The number of particles in the text of question 2 was 1024^3 but should have been 1024.

[3]Note that these weights sum to 1 over the 8 grid points, so mass is conserved.