ASTR4004/ASTR8004
Astronomical Computing
Assignment 4 (exam assignment)

Christoph Federrath, Michael Ireland, Mark Krumholz

due Friday, November 01, 2019

1 Python project 1 – Fourier transforms and parallel computing (multi-threaded FFT)

Here you will make a python program that reads a column density map of a molecular cloud called 'The Brick' near the Galactic Centre (you can read more about this cloud in Federrath et al., 2016), apply mirroring and zero-padding to the image, compute the Fast Fourier Transform (FFT) with the `pyFFTW` library (https://pypi.python.org/pypi/pyFFTW), make a Fourier image and compute the power spectrum of the column density map.

1. Download the observational column density map from http://www.mso.anu.edu.au/~chfeder/teaching/astr_4004_8004/material/brick.fits. Use the astropy lib to read the data map in the fits file (http://docs.astropy.org/en/stable/io/fits/) into a numpy array.

2. Make a python function to produce an image of the map with a colour bar and write the image to a pdf file named 'brick.pdf'. See the left-hand panel of Figure 1 for an example thumbnail image of how this should look like.

3. Use the numpy functions `np.fliplr` and `np.flipud` to produce a mirrored array and image. Write the image to a pdf file called 'brick_mirrored.pdf' (see

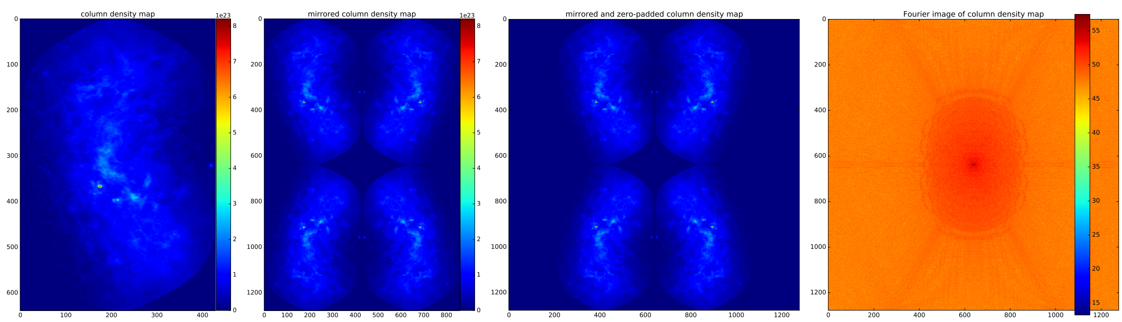

Figure 1: Left to right: original column density map, mirrored, zero-padded, and $\log_{10}$ Fourier image. Make sure to reproduce these not so small as in this assignment, but with readable font sizes; these are just meant as thumbnails to give you some idea of what the output of your script should look like.

the middle panel of Figure 1 for a thumbnail).

4. Now use the numpy function `np.pad` to pad zeros symmetrically to the left and right of the image, such that the total dimensions become (1278, 1278). Make an image of this called 'brick_mirrored_zped.pdf' (see Fig. 1 for how this should look.)

5. Install `pyfftw`. Make a 2D threaded FFTW (use 1, 2 or 4 threads) of the mirrored-and-zero-padded column density map. Shift the $\mathbf{k} = (0, 0)$ position to the centre of the Fourier image and write out an image called 'brick_fourier_image.pdf'. The result of this should look like the last panel of Figure 1.

6. Compute the 1D power spectrum $P(k)$ of the mirrored and zero-padded column density, where $k = \sqrt{k_x^2 + k_y^2}$. Make a log-log plot of the power spectrum, $P(k)$, and write this out as an image called 'brick_power_spectrum.pdf'.

7. (Optional) scaling test: replicate the mirrored and zero-padded image $N\times$ in the x and y direction. Try $N = 10$ to produce a very big array with (12780, 12780) points. Test the multi-threaded FFTW with 1, 2, 4, and 8 threads for the parallelised FFT and produce a plot of speedup versus number of threads for the FFTW part of your script.

Put everything into a single Bash-shell-executable python script that runs the entire analysis with the input file (the column-density fits file) sitting in the same folder. The script should automatically produce the images with the requested filenames above (original column density image, mirrored, zero-padded, and Fourier image), as well as the final plot of the column-density power spectrum.

(10 points)

2 Python project 2 – Markov Chain Monte Carlo

In this assignment you will use `emcee` in python (http://dfm.io/emcee/current/) or on github. You will simulate a periodic data set and fit a function to it. This could be, e.g., a photometric dataset from Kepler, or a series of radial velocity points. Some skeleton code (with many gaps!) is available here: http://www.mso.anu.edu.au/~chfeder/teaching/astr_4004_8004/material/mcmc_assignment_hints.py.

1. Create a function using python and `numpy` that simulates data that take a periodic function with a form:

$$v = a_0 + a_1 t + a_2 \sin(a_3 t + a_4) \tag{1}$$

You should simulate data at a number of random times over an interval, and include Gaussian errors for the data. The inputs a_i should take the form of a 1-dimensional python array.

2. Setting $a_0 = 0$, $a_1 = 1$, $a_2 = 1$, $a_3 = 1$ and $a_4 = 0$, simulate a data set from times $t = 10$ to $t = 30$, containing 100 points with Gaussian errors with uncertainty 0.3.

3. Use `emcee` to fit to this dataset. Plot histograms of the fitted parameters – do the results make sense? Are any of the parameter fits correlated? Try this again for $a_4 = 3$.

4. Show that the following is a re-parameterisation[1] of Equation (2):

$$v = b_0 + b_1(t - 20) + b_2 \sin(b_3 t) + b_4 \cos(b_3 t) \tag{2}$$

   Which is better – Equation (2) or Equation (1) for a reliable run of `emcee`, and why? The second set of parameters above ($a_0 = 0$, $a_1 = 1$, $a_2 = 1$, $a_3 = 1$ and $a_4 = 3$) illustrates the difference well.

5. If Equation (1) is your model with uniform priors in all parameters but Equation (2) is used in `emcee` instead with uniform priors, this produces an implicit prior on a_2. What is it?

Include all python code in your assignment, as well as a write-up.

(10 points)

3 Python project 3 – numerical solution of differential equations

Consider a simple harmonic oscillator: a spring with spring constant k is attached to a mass m, and the displacement of the mass from its rest position is x. The mass experiences a restoring force,

$$F = -kx. \tag{3}$$

At time $t = 0$, the mass is released at rest at the initial position $x(0)$, and is allowed to oscillate.

3.1 Part 1

Write a python function that takes as inputs the value of the spring constant k, the mass m, the initial displacement $x(0)$, the amount of time for which to integrate T, and the number of times N at which the position should be recorded, and returns the position and velocity of the mass at times 0, $T/(N-1)$, $2T/(N-1)$, $\ldots$, T. Verify that your code matches the analytic solution for $x(0) = 0.1\,\mathrm{m}$, $k = 50\,\mathrm{N/m}$, and $m = 1\,\mathrm{kg}$.

[1]Re-parameterisation means that the b_0 through b_4 can be written in terms of a_0 through a_4.

3.2 Part 2

Modify your routine to that it works for a nonlinear spring; one with a restoring force

$$F = -k_1 x - k_2 x^3. \tag{4}$$

Make a plot comparing the solution for a simple harmonic oscillator with the parameters given in Section 3.1 and the solution for a non-linear oscillator with the same values of $x(0)$, k_1, and m, and a nonlinear coefficient $k_2 = 10^3\,\mathrm{N/m^3}$.

(10 points)

(Total 30 points)

References

Federrath, C., Rathborne, J. M., Longmore, S. N., et al. 2016, *Astrophys. J.*, 832, 143