ASTR4004/ASTR8004
Astronomical Computing
Assignment 4 – Practice Draft

Christoph Federrath

due Tuesday, October 18, 2016, 09:15am

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 near the Galactic Centre, 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 astro-py 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 of how this should look like.

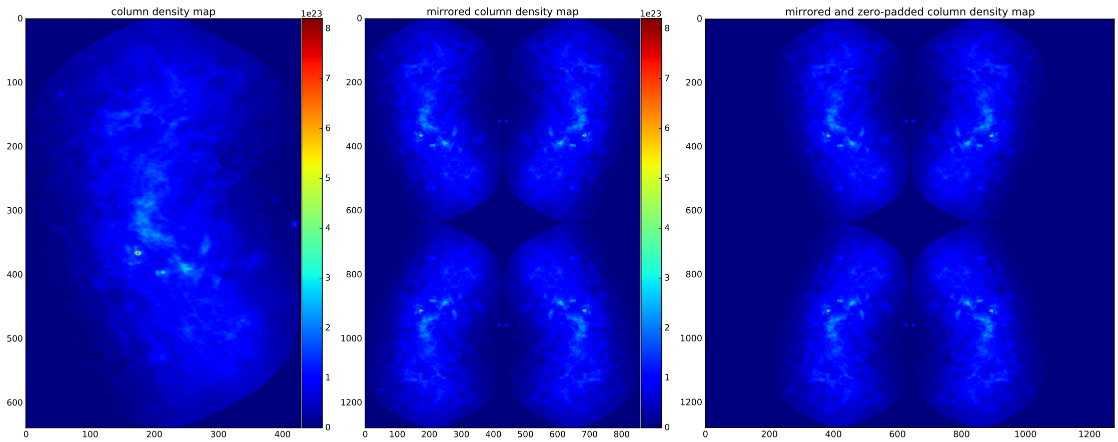

Figure 1: Left to right: original column density map, mirrored, and zero-padded.

3. Use the numpy functions `np.fliplr` and `npflipud` to produce a mirrored array and image. Write the image to a pdf file called 'brick_mirrored.pdf' (see the middle panel of Figure 1 for a thumbnail).

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

5. Install `pyfftw`, e.g., via `sudo port install py-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'.

6. Bin the Fourier-transformed mirrored-and-zero-padded column density array $\hat{a}$ in wavenumber $k = \sqrt{k_1^2 + k_2^2}$, to obtain the power spectrum,

$$P(k) = 2\pi \int \hat{a}(\mathbf{k})\, \hat{a}^{\star}(\mathbf{k})\, k\, dk, \tag{1}$$

   where $\hat{a}^{\star}$ is the complex conjugate of $\hat{a}$. Do this by defining concentric shells in wavenumber space around the centre of the Fourier-transformed image with dk equal to one cell size in the Fourier image.

7. Make a log-log plot of the power spectrum, $P(k)$, and write this out as an image called 'brick_power_spectrum.pdf'.

8. 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 an executable python script that runs the entire analysis with the input file (the column density map fits file) sitting in the same folder. The script should automatically produce the images (original column density image, mirrored, zero-padded, and Fourier image), as well as the final plot of the column density power spectrum.

2 Python project 2 – Markov Chain Monte Carlo

In this assignment you will use `emcee` in python. You will simulate a periodic data set and fit a function to it. Although this is written as if you are fitting to velocities (e.g. searching for exoplanets), the same approach works in fitting to many other data.

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

$$v = a_0 + a_1 t + a_2 \sin\left(2\pi\left(a_3 t + a_4\right)\right) \tag{2}$$

   You should simulate data at a number of random times over an interval, and include Gaussian errors for the data.

2. Setting $a_0 = 0$, $a_1 = 1$, $a_2 = 1$, $a_3 = 1$ and $a_4 = 0$, simulate a data set from times $t = 0$ to $t = 1.5$, containing 100 points with Gaussian errors with uncertainty 1.0.

3. Use `emcee` to fit to this dataset. [report correlation coefficient, etc...]

4. Change the time interval from $t = 0$ to $t = 0.75$. How does the result change?

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 $m = 1\,\mathrm{kg}$.

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 $k_2 = 10^3\,\mathrm{N/m^3}$.

Please send your scripts/responses/produced figures for these projects to mailto:christoph.federrath@anu.edu.au *by the assignment deadline.*