DRAFT VERSION JANUARY 25, 2017

PROBLEM SET 1 (DUE WED FEB 1).

Detrending Kepler data (20 pts):

The *Kepler* spacecraft monitored stars for several years, providing amazing photometric lightcurves with precisions in the parts-per-million range. This exquisitely low error is requisite for *Kepler's* main science objective: detecting Earth-like planets around other stars. The raw data that comes off the spacecraft however has large, long-timescale variations due to a variety of factors including thermal expansion/contraction of the telescope's optical path and stars drifting out of their designated "pixel aperture". These long-timescale variations can be removed by detrending the data in various ways. Here, we're going to unleash the power of linear regression to remove a low-order polynomial from real *Kepler* data. Our target is **KIC-7200111**, a Sun-like star, and we'll look at it's 3rd quarter data[1]. Kepler data, as with many other forms of astronomical data, is in FITS format. To get started, install `pyfits` by doing `$pip3 install pyfits` at your command line. This problem will also provide your introduction to python3[2], to bitbucket, to plotting, and to Latex[3].

NOTE: the code snippet produces two lightcurves: an uncorrected one and a PDC lightcurve, which is the Kepler "first pass" at error correction. Throughout this homework, do the fitting, etc. to the uncorrected lightcurve. We will compare our results to PDC at the end.

0. Go to the MAST website, enter **7200111** in the field "Kepler ID" and click on the quarter 3 dataset. At the top of the page, download the light curve (.fits) file.
1. Using the code provided, read in the Kepler FITS file data. Plot the lightcurve and make sure it agrees broadly with the image you saw on MAST.
2. Using linear regression and least squares, produce polynomial fits for the uncorrected lightcurve. Sample a variety of polynomial degrees (at least from 1 to 4). Plot the resulting polynomial fits over the data. Comment.
3. Detrend the data by removing a $N = 3$ polynomial. Plot this detrended data, and on the same plot, show the PDC lightcurve from the *Kepler* pipeline[4]. Comment on the agreement between our crude polynomial fitting technique and PDC.
4. What could be the source of the remaining wiggles in the stellar lightcurve? Justify.

Analyzing Kepler data (20 pts):

Here we continue on with our *Kepler* explorations. Our target remains **KIC-7200111**. Now it's time to try Fourier transforms of the data to obtain periods.

0. Start with your detrended solution from Part 1: subpart
1. Library FFTs (Fast Fourier Transforms) require data on uniformly sampled locations. Use `np.interp()` to interpolate your data onto a uniform grid with 8192 equally spaced time points. The function `np.linspace()` may be helpful. Confirm that your interpolated data agrees reasonably with your original detrended data (plotting and chi-by-eye is sufficient here).
2. Perform a FFT on your data, using `np.fft.fft()`. How do the returned coefficients in the FFT array map onto temporal frequencies? (e.g., what frequency is stored in FFT[0]? What frequencies are stored in increasing locations within the array? What happens after we pass the mid-point in the array?) *hint: read the numpy docs*.
3. Form the power spectrum by multiplying the FFT by it's complex conjugate. Plot the power spectrum versus frequency for the positive frequencies. Use a loglog scale.
4. Identify the two highest peaks of power. What frequencies do these correspond to? What periods do these correspond to ($P = 1/f$)? What might these periods represent?
5. The FFT assumes that your data is periodic. Is this data periodic? What problems could arise from non-periodicity?

Asteroseismology (10 pts):

Let's try playing with Short Cadence *Kepler* data. Go to the MAST webpage and search for *16 Cyg a*, one of the brightest stars observed with *Kepler*. Find the Q7 Short Cadence (target type: SC) data and download the FITS file. The target is **KIC-100002741**, and the file is `kplr100002741-2010296114515_slc.fits`. Now it's time to try Fourier transforms of the data to obtain oscillations.

1. Use least squares to fit and remove a $N = 3$ polynomial from the data. Interpolate the data onto a uniform grid with $2^{16} = 65536$ points in time. Take the FFT and form the powerspectrum. Plot the powerspectrum on a log-log scale. Comment.
2. Plot the powerspectrum a second time on a linear-linear scale, with units of seconds^{-1} on the frequency axis (previously you used day^{-1}). Zoom in on the frequency range from 1.5×10^{-3}–3.0×10^{-3} sec^{-1}, and choose a y-scale that emphasizes the peaks. How does the peak oscillation frequency compare to the peak of the solar helioseismic oscillations?
3. *Extra credit:* what could you infer about the qualitative stellar properties of *Cyg 16* a compared to the Sun based on the oscillation frequencies?

To hand in: answers to the above problems, plus labelled, clearly designed figures for all requested plots. We would also like your final code producing the figures. Hand code in via https://bitbucket.org/[5]; we might consider other options for handing in code, depending on preferences from our grader.

```
#
# hw1.py
# Benjamin Brown
# ASTR 5550
# January 24, 2017
#
import numpy as np
```

```python
    data = file['LIGHTCURVE'].data
    raw_times = data['TIME']
    raw_lightcurve = data['SAP_FLUX']
    raw_PDC_lightcurve = data['PDCSAP_FLUX']

    # clean out NANs and infs in raw data stream
    # slightly more points are bad in "PDC" version.
    good_data = np.isfinite(raw_PDC_lightcurve)
    lightcurve = raw_lightcurve[good_data]
    PDC_lightcurve = raw_PDC_lightcurve[good_data]
    times = raw_times[good_data]

    N_good_points = len(lightcurve)
    N_bad_points = len(raw_lightcurve)-N_good_points
    print("{:d} good points and "
          "{:d} bad points in lightcurve".format(N_good_points, N_bad_points))

    # note: the PDC_lightcurve is a corrected lightcurve
    # from the Kepler data pipleine, that fixes some errors.
    # PDC means "Pre-Search Data Conditioning"
    return times, lightcurve, PDC_lightcurve

if __name__ == "__main__":

    filename = 'kplr007200111-2009350155506_llc.fits'
    times, lightcurve, PDC_lightcurve = read_data(filename)
```