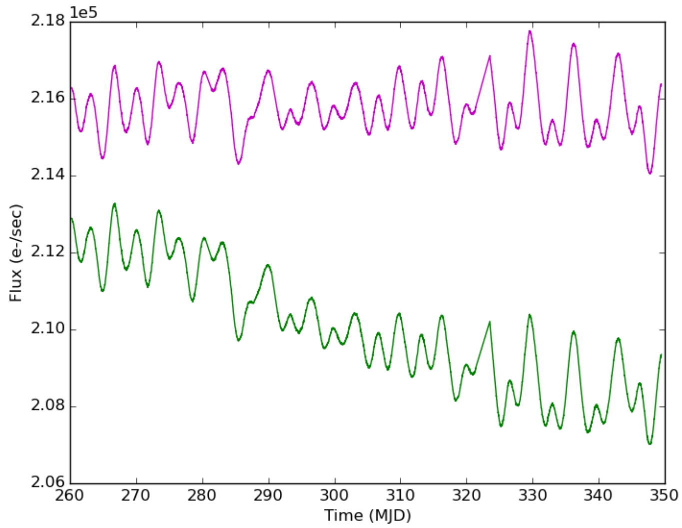
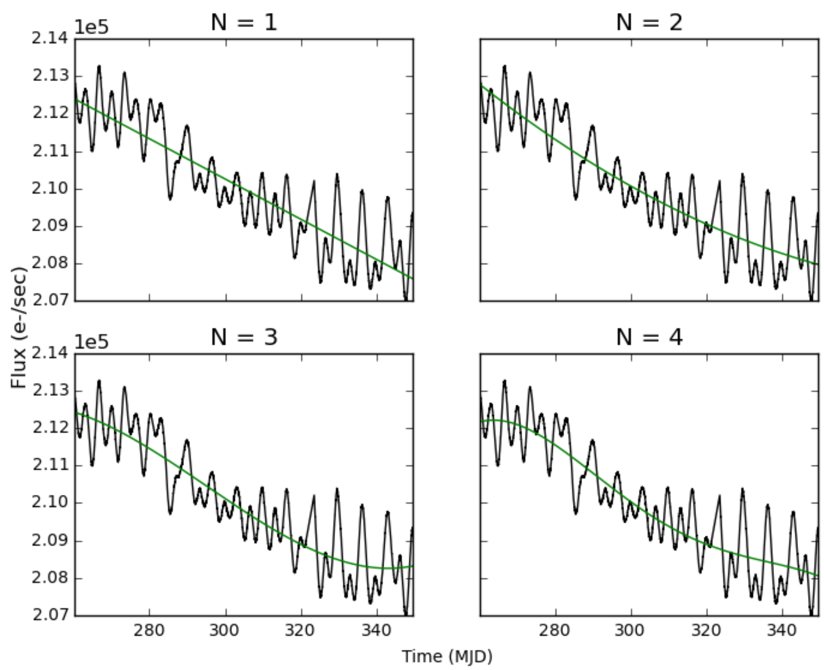
Problem Set 4: Detrending Kepler Data

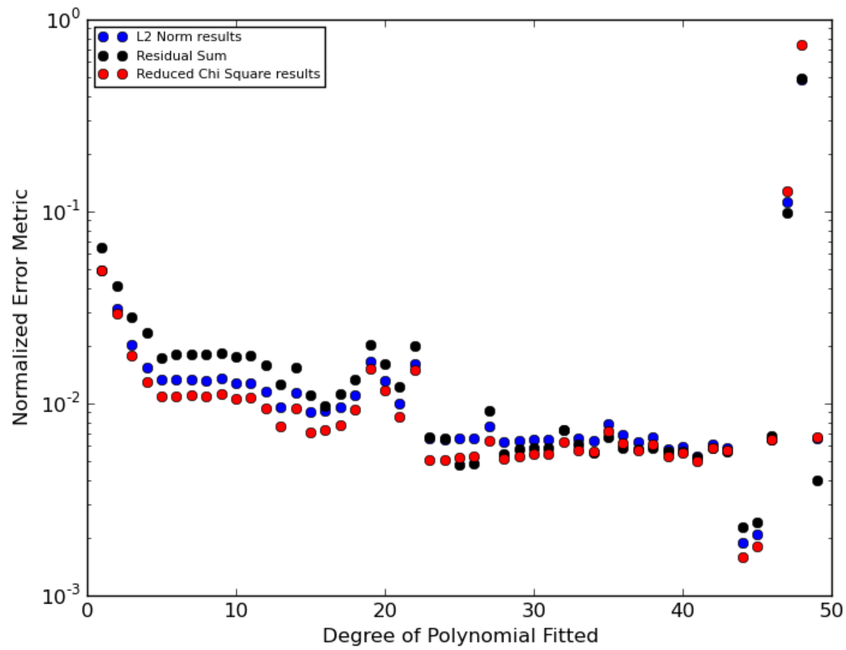
1. It agrees!



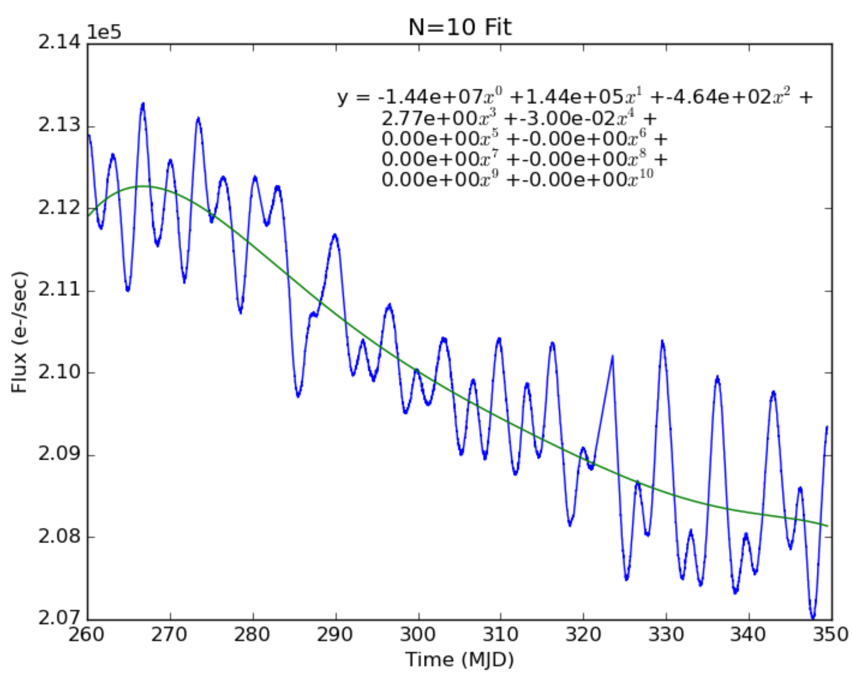
2. The fit becomes progressively better as N increases, but even the linear solution is not terrible.



3. As N increases, the various metrics of error all decrease, until about $N = 5$, when the fit no longer significantly improves with N, and actually (around N = 40) can go haywire and become terrible. This effect is due to precision errors and the conditioning number reaching the limits of double precision numbers' range.

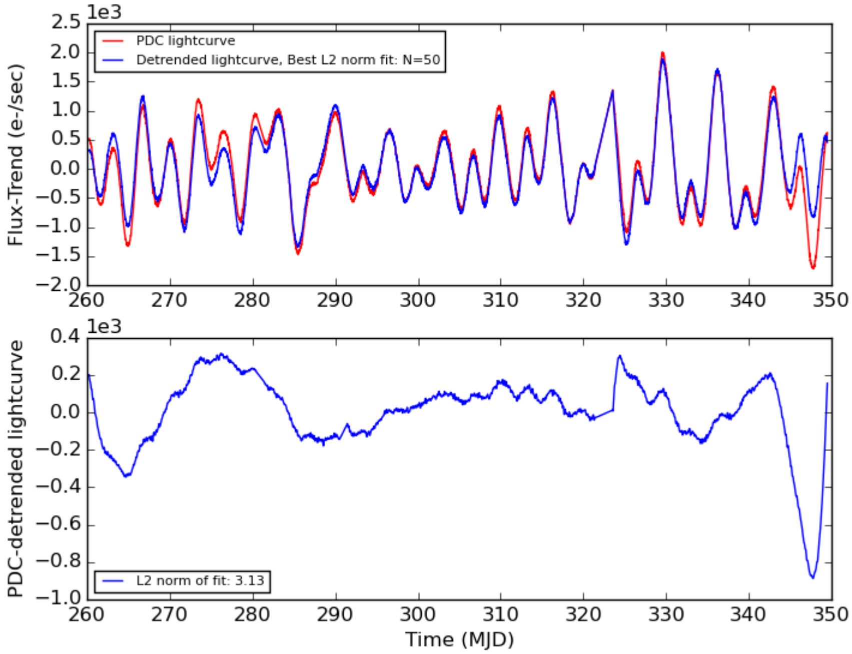


4. Only the polynomials up to $N = 4$ in this solution make any significant difference to the final solution, therefore $N = 4$ is a reasonable choice for detrending the data.

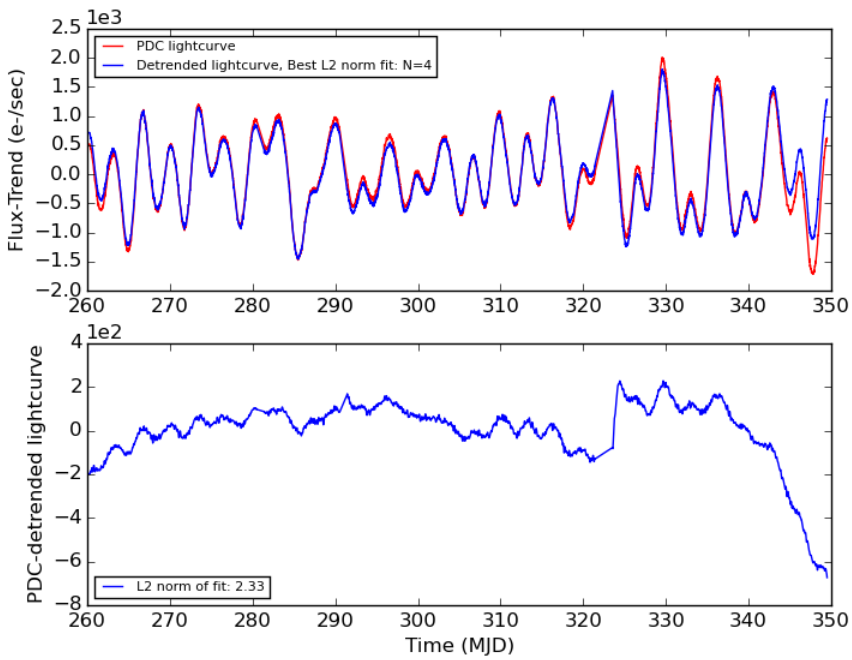


5. Interestingly enough, if you choose a polynomial based purely on minimizing the L2 Norm, the sum of the residuals, or the reduced chi square, the “best” fit to detrend with would be $N = 50$, which is plotted below and

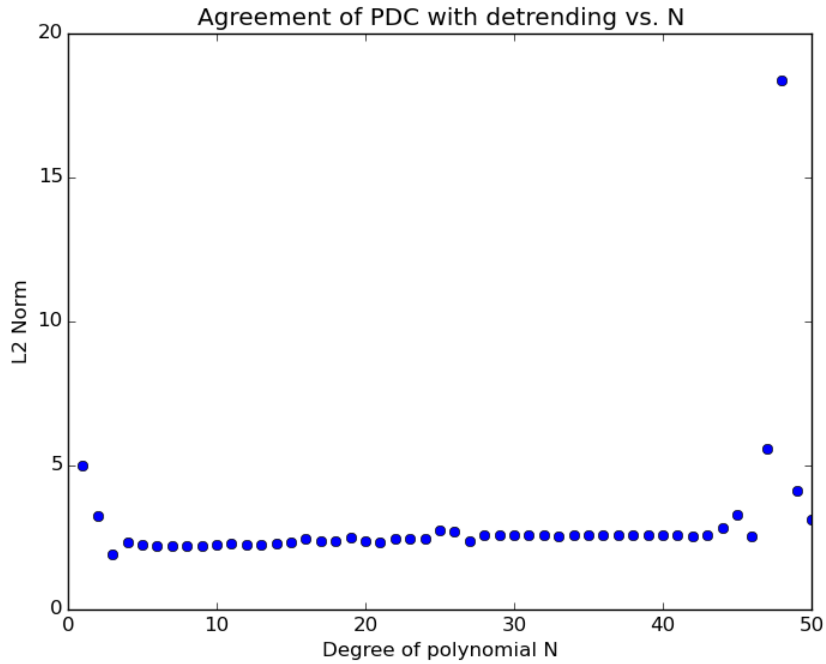
the L2 norm calculated to quantify the agreement with the PDC lightcurve.



If instead, following question 4, I were to choose $N = 4$, I would find the following,



And if I were to look at the smallest value of N with good agreement (below), I would find that the smallest reasonable polynomial to use would be $N = 3$, which is actually the best of them all.



6. The remaining wiggles in the stellar lightcurve are from a number of things. As this lightcurve was produced for a star with a possibly transiting exoplanet, some of the wiggles are from exoplanet transits, others may be from stellar activity such as pulsations and flares, and others from quasi-periodic variation due to stellar rotation, as well as good old fashioned noise (much smaller variations). The overall trend of our residual curve displays a mostly flat morphology with curvature towards the ends, which I would blame on fitting issues in dealing with the ends of a data set as well.

   Given that this star does not display transits (at least according to the Kepler results so far), and that the wiggles are fairly stable and consistent, I would describe the large wiggling pattern as likely the result of observing a semi-regular variable star or RV Tauri.

Code involved in this assignment:

```
#!/usr/bin/python3.4

import numpy as np
from numpy import linalg as la
from matplotlib import pyplot as plt
from copy import deepcopy
import time
import pyfits
from os import sys
import pdb
from decimal import Decimal


def read_data(file):
    file = pyfits.open(file)
    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_pts = len(lightcurve)
    N_bad_pts = len(raw_lightcurve)-N_good_pts
    print("{:d} good points and"
          " {:d} bad points in the lightcurve".format(N_good_pts, N_bad_pts))

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


def LS_fitter(x_data, y_data, N):
    A = np.array([x_data**i for i in range(N+1)])#.transpose()
    y_data = np.array(y_data).reshape(1,y_data.shape[0])

    x = np.linalg.solve(np.dot(A,A.transpose()),
                        np.dot(y_data,A.transpose()).transpose())

    fit = np.dot(x.transpose(),A)  ;  res = (y_data-fit)**2
    abs_err = [np.sqrt(res.sum())/len(fit[0]),np.abs(np.sqrt(res)).sum(),(res/y_data).sum()/(len(fit[0])-N)]
    # L2 Norm, reduced X^2

    return fit, x, abs_err





if __name__=="__main__":
    times, lightcurve, PDC_lightcurve = read_data(sys.argv[1])

    #PLOTTING##THE##FIRST##FIGURE##########################
    fig = plt.figure()
    ax = fig.add_subplot(1,1,1)
    ax.plot(times, lightcurve, color='g')
    ax.plot(times, PDC_lightcurve, color='m')
    ax.set_xlabel("Time (MJD)")
    ax.set_ylabel("Flux (e-/sec)")
    ax.ticklabel_format(style='sci', axis='y', scilimits=(0,0))
    fig.savefig('hw4_fig1.png')
    fig.clf()


    #PERFORMING##THE##FITS################################
    N_fits = int(sys.argv[2])
    fit = np.zeros((N_fits, len(times)))
    err = np.zeros((N_fits,3))
    res = []

    for i in range(1,N_fits+1,1):
        fit[i-1], bit, err[i-1] = LS_fitter(times, lightcurve, i)
        res.append(deepcopy(bit))

```

```
    #PLOTTING##THE##N=1-4##CASES###############################
    fig = plt.figure()
    ax = fig.add_subplot(111)
    ax.spines['top'].set_color('none')
    ax.spines['bottom'].set_color('none')
    ax.spines['left'].set_color('none')
    ax.spines['right'].set_color('none')
    ax.tick_params(labelcolor='w',top='off',bottom='off',left='off',right='off')

    for i in range(1,5,1):
        ax2 = fig.add_subplot(2,2,i)
        #indx = i % 2
        #indy = np.floor(i/2)
        ax2.plot(times, lightcurve, color='k', label='Raw lightcurve')
        ax2.plot(times, fit[i-1], color='g', label='Fitted polynomial')
        ax2.set_title('N = '+str(i))
        ax2.locator_params(nbins = 5, axis='x')
        ax2.set_xlim(min(times), max(times))
        ax2.tick_params(axis='both',labelsize=10)
        if (i ==1) or (i == 2):
            ax2.set_xticklabels([])
        if (i ==4) or (i == 2):
            ax2.set_yticklabels([])
        else:
            ax2.ticklabel_format(style='sci',axis='y', scilimits=(0,0))


    ax.set_xlabel("Time (MJD)", fontsize = 10)
    ax.set_ylabel("Flux (e-/sec)")
    ax.ticklabel_format(style='sci', axis='y', fontsize=10)
    fig.savefig('hw4_fig2.png')
    fig.clf()


    #PLOTTING##THE##N=1-50##CASES##BY##RESIDUAL################
    fig = plt.figure()
    ax = fig.add_subplot(111)
    ax.plot(np.arange(1,N_fits+1,1), (err[:,0]-err[:,0].min())/err[:,0].max(),
            'bo', label='L2 Norm results')
    ax.plot(np.arange(1,N_fits+1,1), (err[:,1]-err[:,1].min())/err[:,1].max(),
            'ko', label='Residual Sum')
    ax.plot(np.arange(1,N_fits+1,1), (err[:,2]-err[:,2].min())/err[:,2].max(),
            'ro', label='Reduced Chi Square results')
    ax.set_xlabel("Degree of Polynomial Fitted")
    ax.set_ylabel("Normalized Error Metric")
    ax.set_yscale('log')
    ax.legend(loc='upper left', fontsize = 8)
    fig.savefig('hw4_fig3.png')
    fig.clf()


    #PLOTTING##THE##N=10##CASES##BY##RESIDUAL##################
    fig = plt.figure()
    ax = fig.add_subplot(111)
    ax.plot(times,lightcurve)
    ax.plot(times,fit[9])
    soln = 'y = '
    for i in range(11):
        soln += '{:.2e}'.format(np.round((res[10])[i][0],2))+r"$x^{}$+".format(i)
        if (i % 2 == 0) & (i != 0):
            if i == 10:
                soln = soln[:-7]+"$x^{10}$"
```

```
            ax.annotate(soln, xy=(290, 2.135e5-i/2*.0025e5))
            soln = '        '
    ax.set_xlabel("Time (MJD)")
    ax.set_ylabel("Flux (e-/sec)")
    ax.set_title("N=10 Fit")
    ax.ticklabel_format(style='sci', axis='y',scilimits=(0,0))
    fig.savefig('hw4_fig4.png')
    fig.clf()


    #DETRENDING##THE##DATA######################################
    fig = plt.figure()
    ax = fig.add_subplot(111)
    ax.spines['top'].set_color('none')
    ax.spines['bottom'].set_color('none')
    ax.spines['left'].set_color('none')
    ax.spines['right'].set_color('none')
    ax.tick_params(labelcolor='w',top='off',bottom='off',left='off',right='off')

    ax2 = fig.add_subplot(211)
    ax3 = fig.add_subplot(212)
    det_PDC = PDC_lightcurve-np.mean(PDC_lightcurve)
    ax2.plot(times, det_PDC, color='r', label='PDC lightcurve')
    lbl = ['Best L2 norm fit: N=','Best residual sum fit: N=','Best chi square fit: N=' ]
    clr = ['b','y','m']
    for i in range(1):#3):
        best = np.where(err[:,i] == min(err[:,i]))
        print(best)
        det_raw = lightcurve-fit[best].reshape(len(times))
        ax2.plot(times, det_raw, color=clr[i], label='Detrended lightcurve, '
                +lbl[i]+str(best[0][0]+1))
        diff = det_PDC-det_raw
        quant = diff**2
        quant = np.round(np.sqrt(quant.sum())/float(len(times)),2)
        ax3.plot(times, diff, color=clr[i],label='L2 norm of fit: '+str(quant))


    ax2.set_ylabel('Flux-Trend (e-/sec)')
    ax2.ticklabel_format(style='sci', axis='y',scilimits=(0,0))
    ax2.legend(loc='upper left', fontsize=8)


    ax3.set_ylabel('PDC-detrended lightcurve')
    ax3.ticklabel_format(style='sci', axis='y',scilimits=(0,0))
    ax3.legend(loc='lower left', fontsize=8)

    ax.set_xlabel('Time (MJD)')
    fig.savefig('hw4_fig5.png')
    fig.clf()


    #DETRENDING##THE##DATA##FOR##N=4############################
    fig = plt.figure()
    ax = fig.add_subplot(111)
    ax.spines['top'].set_color('none')
    ax.spines['bottom'].set_color('none')
    ax.spines['left'].set_color('none')
    ax.spines['right'].set_color('none')
    ax.tick_params(labelcolor='w',top='off',bottom='off',left='off',right='off')

    ax2 = fig.add_subplot(211)
    ax3 = fig.add_subplot(212)
```

```
    det_PDC = PDC_lightcurve-np.mean(PDC_lightcurve)
    ax2.plot(times, det_PDC, color='r', label='PDC lightcurve')
    lbl = ['Best L2 norm fit: N=','Best residual sum fit: N=','Best chi square fit: N=' ]
    clr = ['b','y','m']
    for i in range(1):#3):
        best = 3
        det_raw = lightcurve-fit[best].reshape(len(times))
        ax2.plot(times, det_raw, color=clr[i], label='Detrended lightcurve, '+lbl[i]+str(best+1))
        diff = det_PDC-det_raw
        quant = diff**2
        quant = np.round(np.sqrt(quant.sum())/float(len(times)),2)
        ax3.plot(times, diff, color=clr[i],label='L2 norm of fit: '+str(quant))


    ax2.set_ylabel('Flux-Trend (e-/sec)')
    ax2.ticklabel_format(style='sci', axis='y',scilimits=(0,0))
    ax2.legend(loc='upper left', fontsize=8)


    ax3.set_ylabel('PDC-detrended lightcurve')
    ax3.ticklabel_format(style='sci', axis='y',scilimits=(0,0))
    ax3.legend(loc='lower left', fontsize=8)

    ax.set_xlabel('Time (MJD)')
    fig.savefig('hw4_fig6.png')
    fig.clf()

    #LOWEST##N##WITH##GOOD##PDC##AGREEMENT#####################
    fig = plt.figure()
    ax3 = fig.add_subplot(111)
    quant = np.array([np.sqrt(((det_PDC-(lightcurve-i))**2).sum())/float(len(times)) for i in fit
    ])
    ax3.plot(np.arange(1,N_fits+1,1), quant,'bo')
    ax3.set_xlabel('Degree of polynomial N')
    ax3.set_ylabel('L2 Norm')
    ax3.set_title('Agreement of PDC with detrending vs. N')
    fig.savefig('hw4_fig7.png')
    fig.clf()
```