Trombini_Quentin_Week12

S12_01

S12_01_a

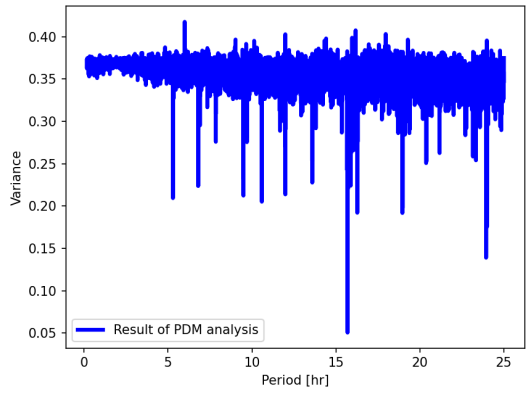
```
import matplotlib.pyplot as plt
import numpy as np
from matplotlib.figure import Figure
from matplotlib.backends.backend_agg import FigureCanvasAgg

ID = "1005497"

# ------------------------------ RETRIEVING DATA ---------------
with open(f"./S12/{ID}.dat", "r") as f:
    lines = f.readlines()

mjd_offset = 52000
data_mjd = []
data_mag = []
data_err = []

lines = lines[20:]
for line in lines:
    line = line.strip()
    _, mjd, mag, err = line.split()
    data_mag.append(float(mag))
    data_mjd.append(float(mjd) - mjd_offset)
    data_err.append(float(err))

# --------------------------------- CONSTANTS ------------------
period_min_min = 10.0
period_min_day = period_min_min / (60.0 * 24.0)
```

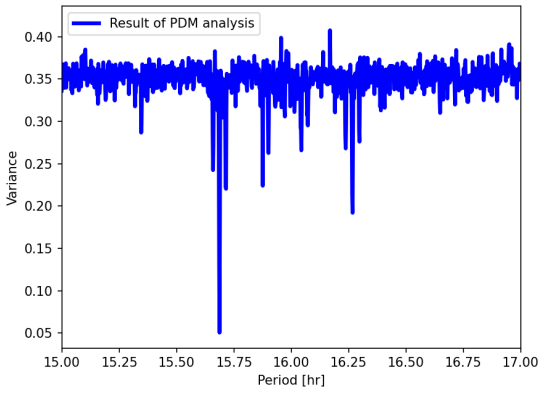
```
period_max_min = 1500.0
period_max_day = period_max_min / (60.0 * 24.0)
step_min = 0.1
step_day = step_min / (60.0 * 24.0)
n_bins = 10

# ------------------------------ PERIOD SEARCH ----------------
period_day = period_min_day
data_per = []
data_var = []

while period_day < period_max_day:
    data_phase = [(mjd / period_day - int(mjd / period_day)) for

    total_variance = 0.0

    for i in range(n_bins):
        bin_min = i / n_bins
        bin_max = (i + 1) / n_bins
        data_bin = [data_mag[j] for j in range(len(data_phase))

        if not data_bin:
            continue

        variance_in_bin = np.var(data_bin)
        total_variance += variance_in_bin

    data_per.append(period_day * 24.0)
    data_var.append(total_variance)
    period_day += step_day

# ------------------------------ PLOTTING ---------------------
fig = Figure()
canvas = FigureCanvasAgg(fig)
ax2 = fig.add_subplot(111)
```

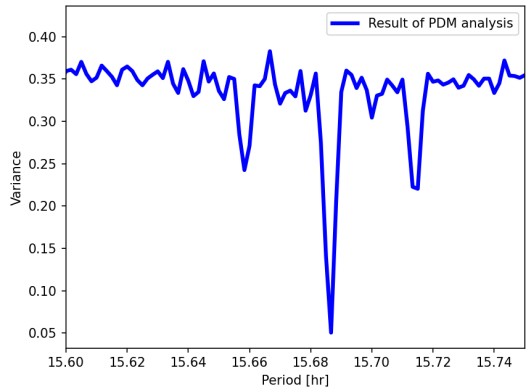
```
ax2.set_xlabel('Period [hr]')
ax2.set_ylabel('Variance')

ax2.set_xlim (15, 17)

ax2.plot(data_per, data_var, linestyle='-', color='blue', linew
ax2.legend()

fig.savefig(f"./S12/S12_01_zoom1.png", dpi=150)

fig.clf()

ax = fig.add_subplot(111)

ax.set_xlabel ('Phase')
ax.set_ylabel ('Apparent Magnitude [mag]')

ax.invert_yaxis ()
ax.errorbar (data_phase, data_mag, yerr=data_err, linestyle='No

fig.savefig(f"./S12/S12_01_error.png", dpi=150)
```

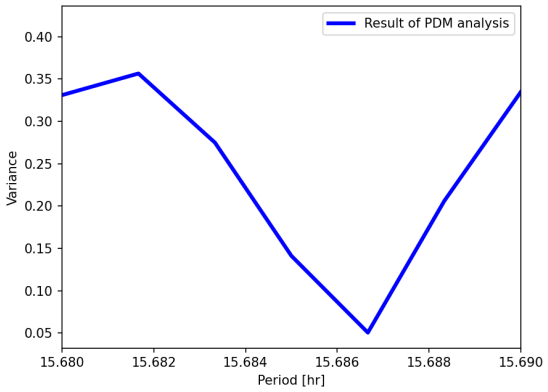
S12_01_b

I made different output with different zoom level you will find all of them below:

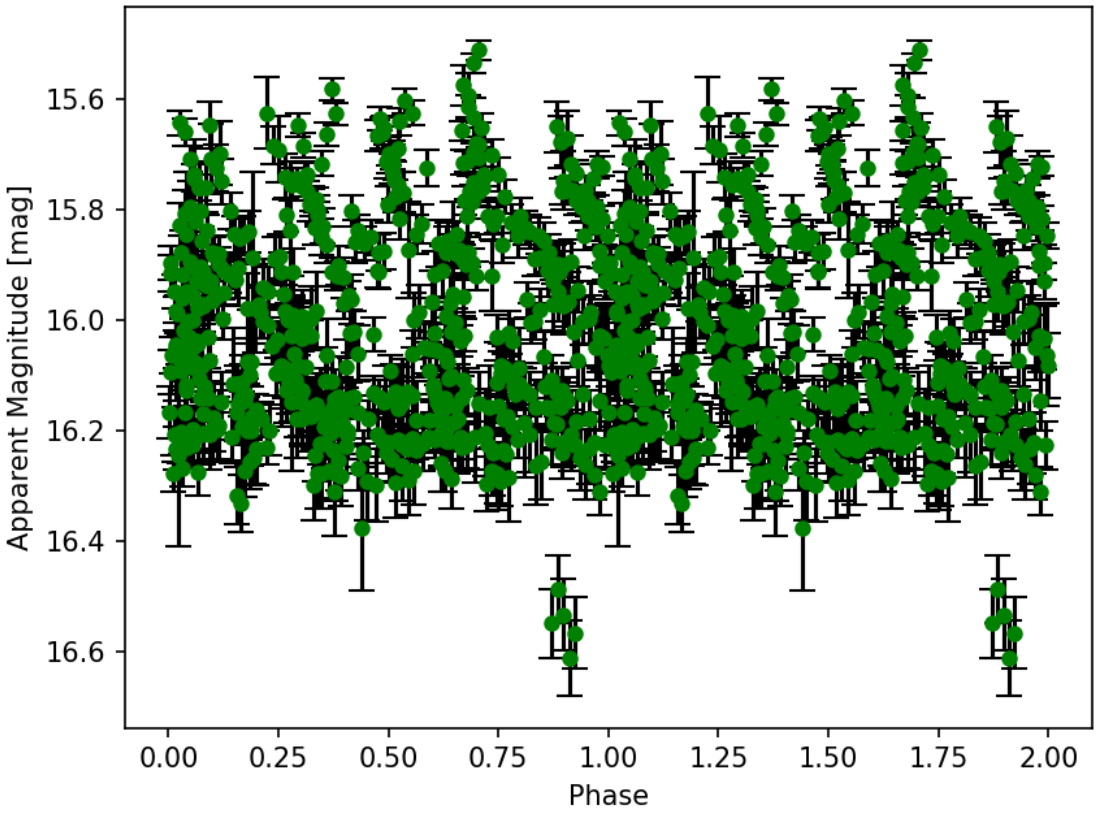








Below is the error curve this one is probably false:



S12_01_c

To retrieve the data I opened the file, suppressed every useless line and then extracted the data into different variable using .split().
Then I tried to get the period and the variance using the retrieved data.

Then using the collected data I will draw the PDM curve and make some zoom to find the lower spike.

Once this is done I try to create the light curve but I think I might have done some miscalculation somewhere because the result don’t look like a folded light curve even if we can find some pattern in the lower and upper bar

---

S12_02

S12_02/03_a

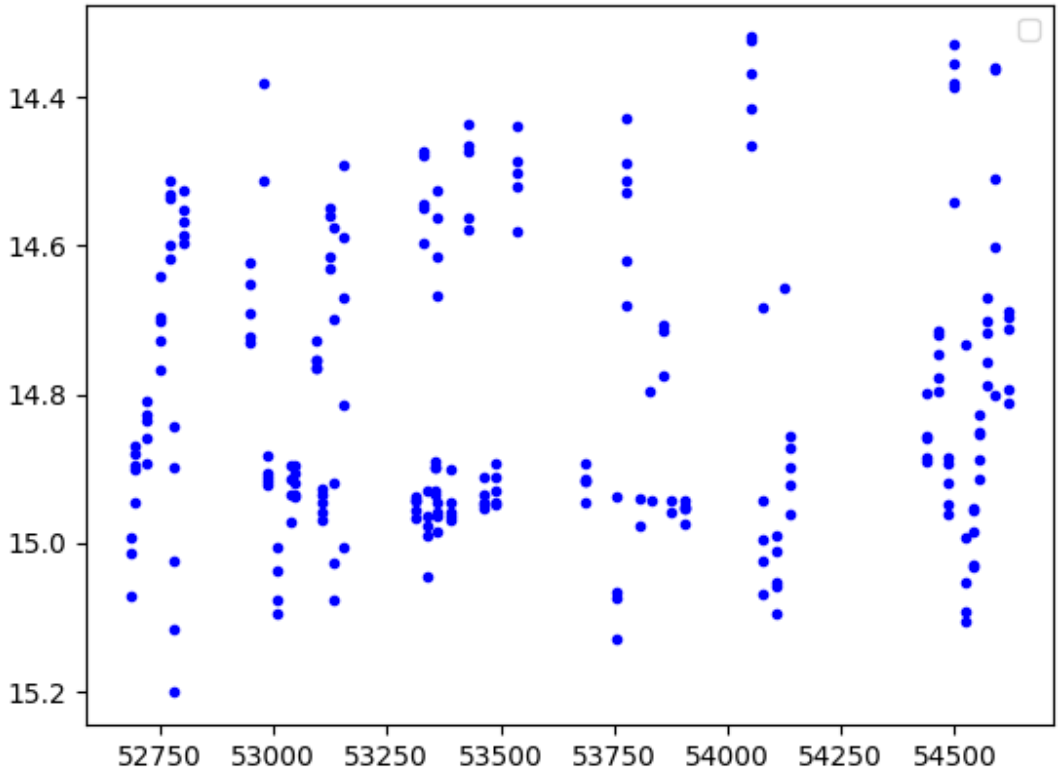
```
from operator import index
import numpy as np
import astropy.timeseries as timeseries
import matplotlib.pyplot as plt
from matplotlib.figure import Figure
from matplotlib.backends.backend_agg import FigureCanvasAgg

ID = "1010992"

# ------------------------------ RETRIEVING DATA ---------------
with open(f"./S12/{ID}.dat", "r") as f:
    lines = f.readlines()

trial = 169.37/24
data_mjd = []
data_mag = []
data_err = []
data_phase = np.array([])

lines = lines[20:]
for line in lines:
    line = line.strip()
    mjd, mag, err = line.split()
    data_mag.append(float(mag))
    data_mjd.append(float(mjd))
    data_err.append(float(err))

    phase = (float(mjd) - data_mjd[0])/trial
    phase -= int(phase)
    data_phase = np.append(data_phase, phase)

freqs, powers = timeseries.LombScargle(data_mjd, data_mag).autop
```

```
data_per_hr = []
data_power = []
for freq, power in zip(freqs,powers):
    data_per_hr.append(1/freq*24)
    data_power.append(power)

maximum = data_per_hr[data_power.index(max(data_power))]
per_min_hr = maximum - 3
per_max_hr = maximum + 3

# ------------------------------- PLOTTING DATA ----------------
fig = Figure()
canvas = FigureCanvasAgg(fig)
ax = fig.add_subplot(111)

ax.plot (data_per_hr, data_power, linestyle='-', linewidth=3, co

ax.legend()
fig.savefig("./S12/S12_02_lomb")

# ------------------------------- ZOOMING IN ------------------
fig.clf()
ax = fig.add_subplot(111)
ax.set_xlim(per_min_hr, per_max_hr)
ax.plot (data_per_hr, data_power, linestyle='-', linewidth=3, co
ax.legend()
fig.savefig("./S12/S12_02_lomb_zoom")



# ---------------------------------- FOLDING -------------------
fig.clf()
ax = fig.add_subplot (111)
ax.set_xlabel ('Phase')
ax.set_ylabel ('Magnitude [mag]')
ax.invert_yaxis ()
label = f"folded lightcurve constructed from $P = {trial * 24.0
```

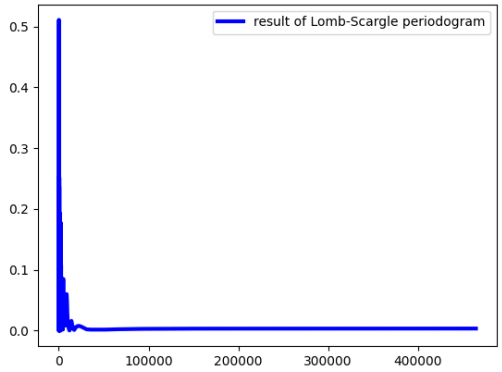
```
ax.errorbar (data_phase, data_mag, yerr=data_err, linestyle='No
ax.errorbar (data_phase + 1.0, data_mag, yerr=data_err, linesty
ax.legend (bbox_to_anchor=(1.0, 1.12), loc='upper right')
fig.savefig ("./S12/S12_02_folded", dpi=150)
```

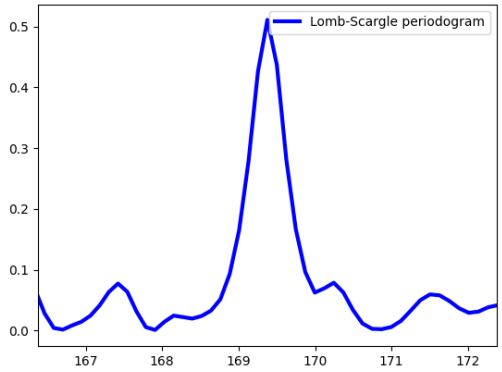
S12_02_b

Here are the different picture created by the program:

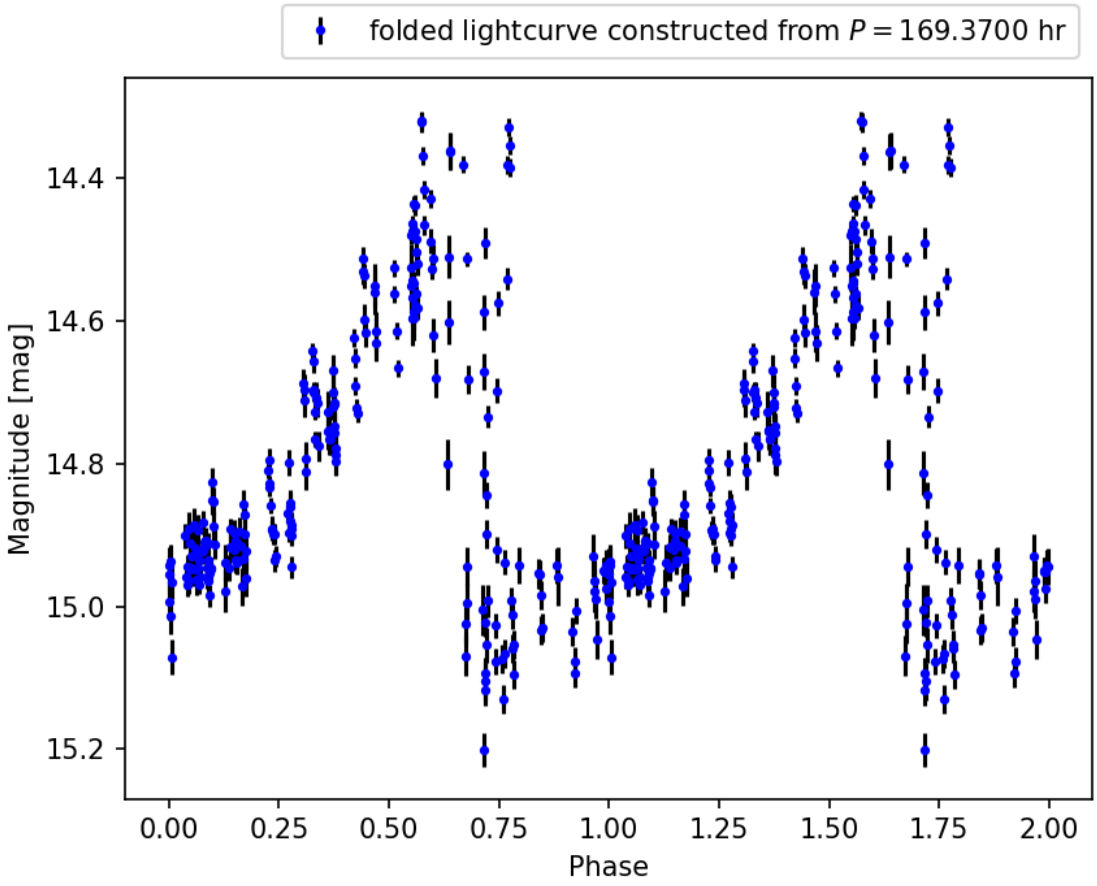


Below the different zoom in the Lomb-Scargle diagram:





Below the folded lightcurve:



The data retrieving is looking like the one of the previous exercise but this time we add the following lines:

S12_02_c

```
phase = (float(mjd) - data_mjd[0])/trial
phase -= int(phase)
data_phase = np.append(data_phase, phase)
```

to create a numpy array of phases

Once this is done the following line will take our date and magnitude and perform an analysis on it:

```
freqs, powers = timeseries.LombScargle(data_mjd, data_mag).autop
```

we will then used the frequency and the power outputted to draw a plot and find the maximum of this plot.

We use the following line to get time of the maximum of the plot by getting the index of the curves apex and then applying it to the frequency to get the right moment.

```
maximum = data_per_hr[data_power.index(max(data_power))]
```

Once we have the maximum we can divide it by 24 to get our unfolding period and draw the light curve using the following two lines:

```
ax.errorbar (data_phase, data_mag, yerr=data_err, linestyle='Non
ax.errorbar (data_phase + 1.0, data_mag, yerr=data_err, linestyl
```

S12_03_c

For this part I didn”t find the magnitude data in the Kepler fits files so I couldn’t make the program work. The code should be similar to the code in the S12_02_b since it’s following the same principle.