

# Io 30s

```
In [1]: import numpy as np
import os
from astropy.io import fits
from astropy import units as u
from astropy.modeling.polynomial import Polynomial1D
from astropy.modeling.models import Gaussian1D, Linear1D
from astropy.modeling.fitting import LinearLSQFitter
from IPython.display import Image
# astroquery provides an interface to the NIST atomic line database
from astroquery.nist import Nist
from IPython.display import Image
import glob

from PIL import Image
import numpy as np
import pylab as pl
pl.style.use('dark_background')

from astropy.modeling.polynomial import Polynomial1D
from astropy.modeling.fitting import LinearLSQFitter

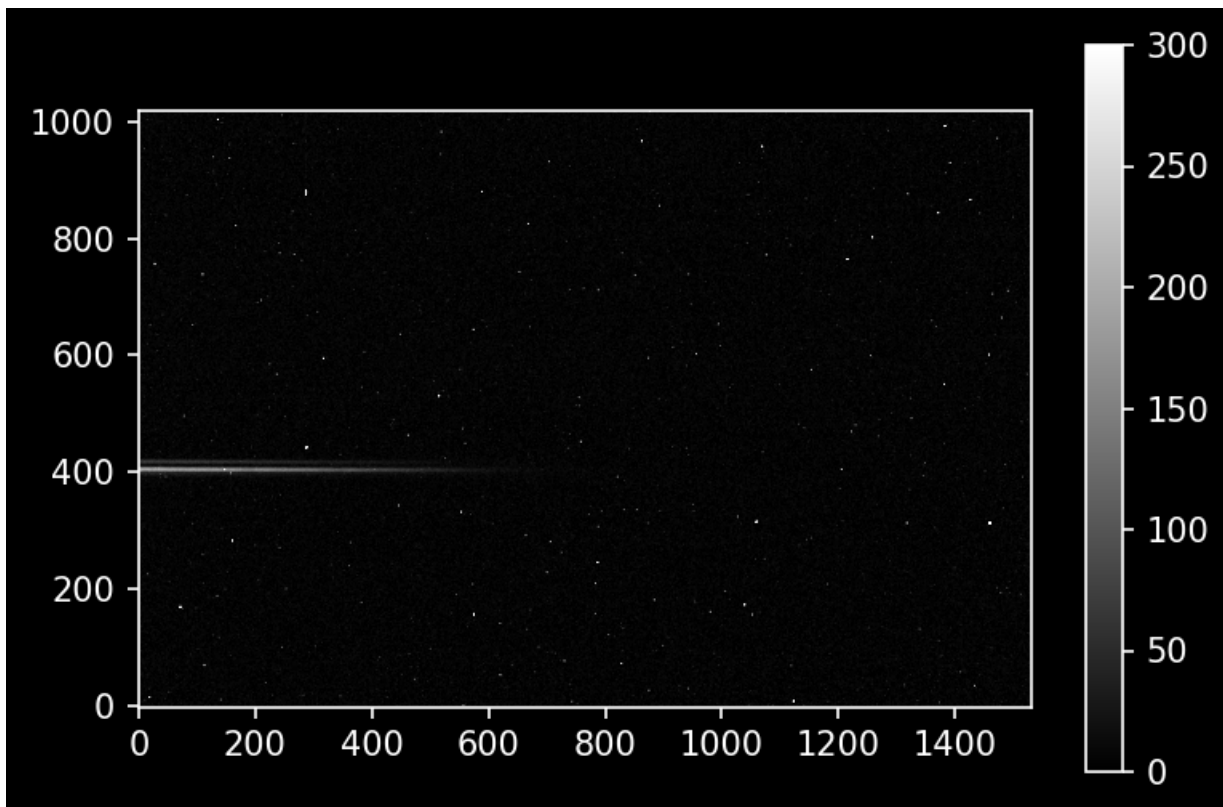
from astropy.modeling.models import Gaussian1D
from astropy.modeling.fitting import LevMarLSQFitter
```

```
In [2]: io_30s_image_data = (np.mean([fits.getdata(x) for x in glob.glob("\\Users\\Sydney\\OneDrive\\Documents\\io_30s\\io_30s_*.fits")],
axis=0)
- np.mean([fits.getdata(x)
for x in glob.glob("\\Users\\Sydney O'Donnell\\OneDrive\\Documents\\io_30s\\io_30s_*.fits")],
axis=0)
)
```

```
In [3]: io_array = np.array(io_30s_image_data)
io_array = io_array - np.median(io_30s_image_data)
```

```
In [4]: %matplotlib inline
import pylab as pl
pl.rcParams['image.origin'] = 'lower'
pl.rcParams['figure.dpi'] = 150
pl.matplotlib.style.use('dark_background') # Optional!
pl.imshow(io_30s_image_data, cmap='gray', vmax=0, vmin=300)
pl.colorbar()
```

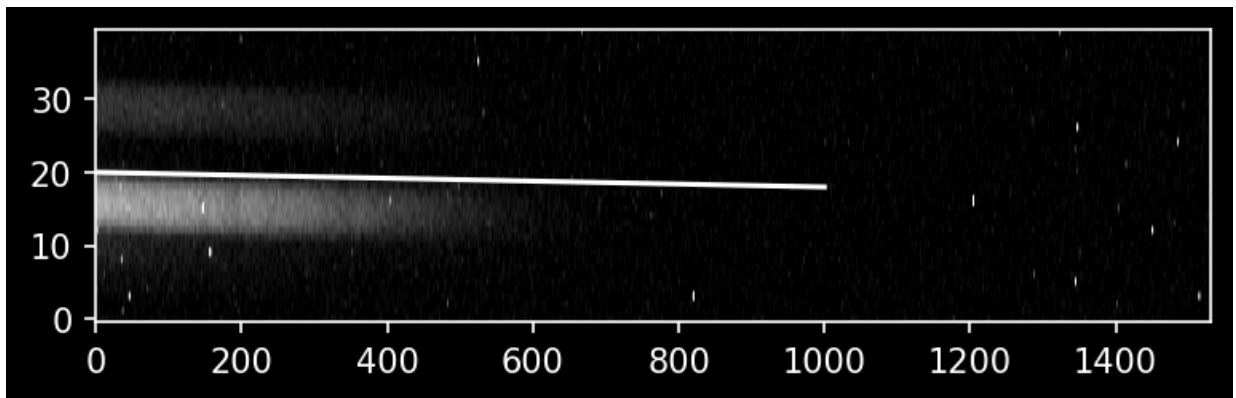
Out[4]: <matplotlib.colorbar.Colorbar at 0x1e8c3e38250>



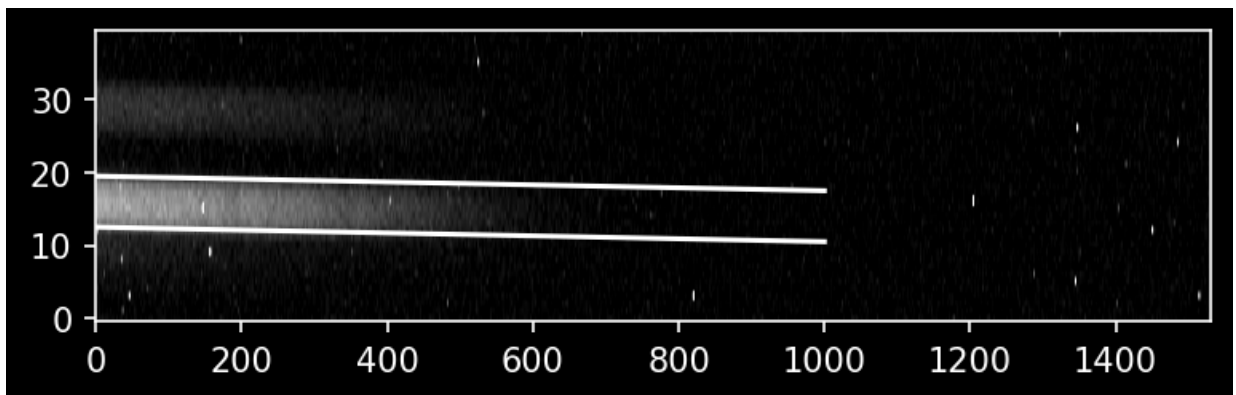
```
In [5]: dy = -2
dx = 1000
slope = dy/dx

ystart = 390
yend = 430

pl.imshow(io_array[ystart:yend,:], cmap='gray', vmax=300, vmin=0)
pl.plot([0,1000], 20 + np.array([0,1000]) * slope, color='w')
pl.gca().set_aspect(10)
```



```
In [6]: intertrace_cuts = np.array([12.5,19.5])
pl.imshow(io_array[ystart:yend,:], cmap='gray', vmax=300, vmin=0)
pl.plot([0,1000], intertrace_cuts + np.array([0,1000])[0,:None] * slope, color='w')
pl.gca().set_aspect(10)
```



```
In [7]: npixels_to_cut = 4 # very conservative - we'll see why below
xvals = np.arange(io_array.shape[1])
trace_center = ystart+(intertrace_cuts[0] + intertrace_cuts[1])/2 + xvals * slope
cutout_trace = np.array([io_array[int(yval)-npixels_to_cut:int(yval)+npixels_to_cut]
                        for yval, ii in zip(trace_center, xvals)]).T
```

```

In [8]: # to get the y-axis values corresponding to each part of our cutout trace, we do
yaxis_full = np.arange(io_array.shape[0])
yaxis = np.array([yaxis_full[int(yval)-npixels_to_cut:int(yval)+npixels_to_cut]
                  for yval, ii in zip(trace_center, xvals)]).T
xend = 800
weighted_yaxis_values = np.average(yaxis[:, :xend], axis=0,
                                   weights=cutout_trace[:, :xend])

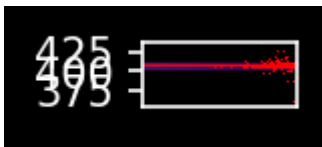
pl.figure(figsize=(8,3))
traces = {}
for trace_index in range(len(intertrace_cuts)-1):
    yoffset = ystart + (intertrace_cuts[trace_index] + intertrace_cuts[trace_index+1])
    trace_center = yoffset + slope * xvals

    cutout_trace = np.array([io_array[int(yval)-npixels_to_cut:int(yval)+npixels_to_cut]
                             for yval, ii in zip(trace_center, xvals)]).T
    yaxis = np.array([yaxis_full[int(yval)-npixels_to_cut:int(yval)+npixels_to_cut]
                     for yval, ii in zip(trace_center, xvals)]).T
    weighted_yaxis_values = np.average(yaxis[:, :xend], axis=0,
                                       weights=cutout_trace[:, :xend])

    # it takes a little mental gymnastics to get to this, but: to show the trace
    # we need to calculate the local version
    local_weighted_yaxis_values = np.average(np.arange(npixels_to_cut*2)[ :, None],
                                             axis=0, weights=cutout_trace[:, :xend])

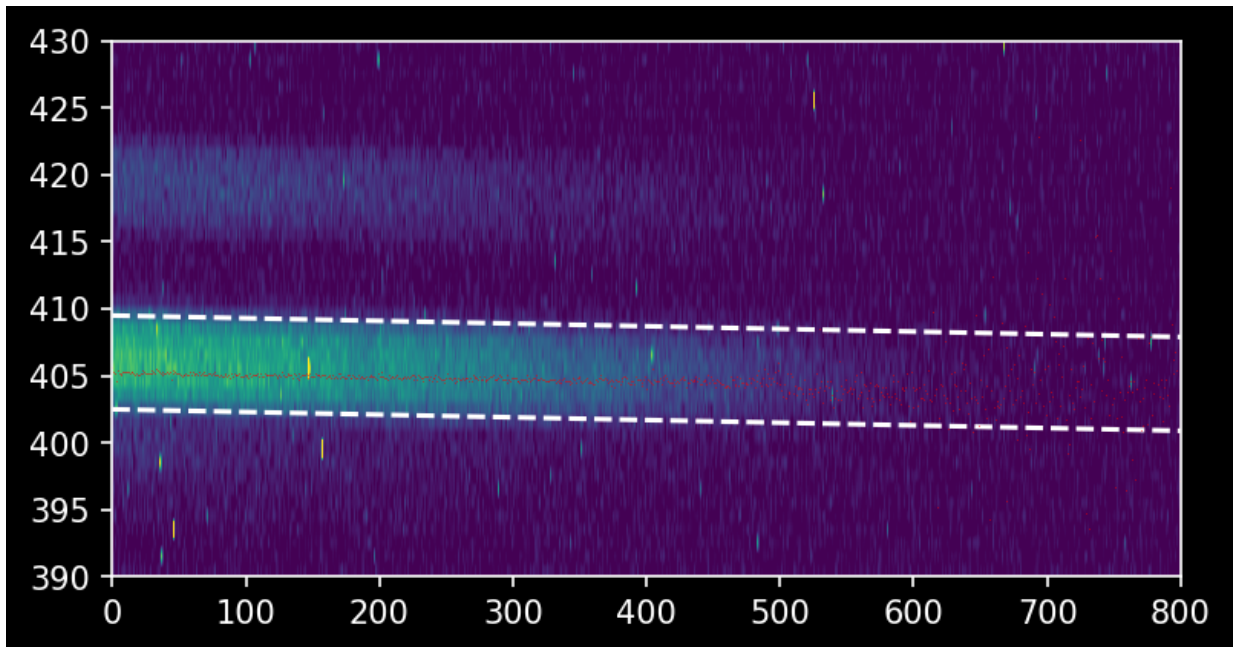
    traces[trace_index] = weighted_yaxis_values
    ax = pl.subplot(7, 1, trace_index+1)
    ax.imshow(cutout_trace[:, :xend], extent=[0, xend, yoffset-npixels_to_cut, yoffset+npixels_to_cut])
    ax.plot(xvals[:xend], yoffset - npixels_to_cut + local_weighted_yaxis_values)
    ax.set_aspect(4)
    ax.set_xticks([])
pl.tight_layout()

```



```
In [9]: # then we can plot the "global" version here
pl.imshow(io_array[ystart:yend, :xend],
          extent=[0,xend,ystart,yend], vmax=300, vmin=0)
pl.plot([0,xend], ystart + intertrace_cuts + np.array([0,xend])[:,None] * slope,
pl.gca().set_aspect(10)
for trace in traces.values():
    pl.plot(xvals[:xend], trace[:xend], 'r', alpha=0.5)
pl.axis((0,xend,ystart,yend))
```

Out[9]: (0.0, 800.0, 390.0, 430.0)



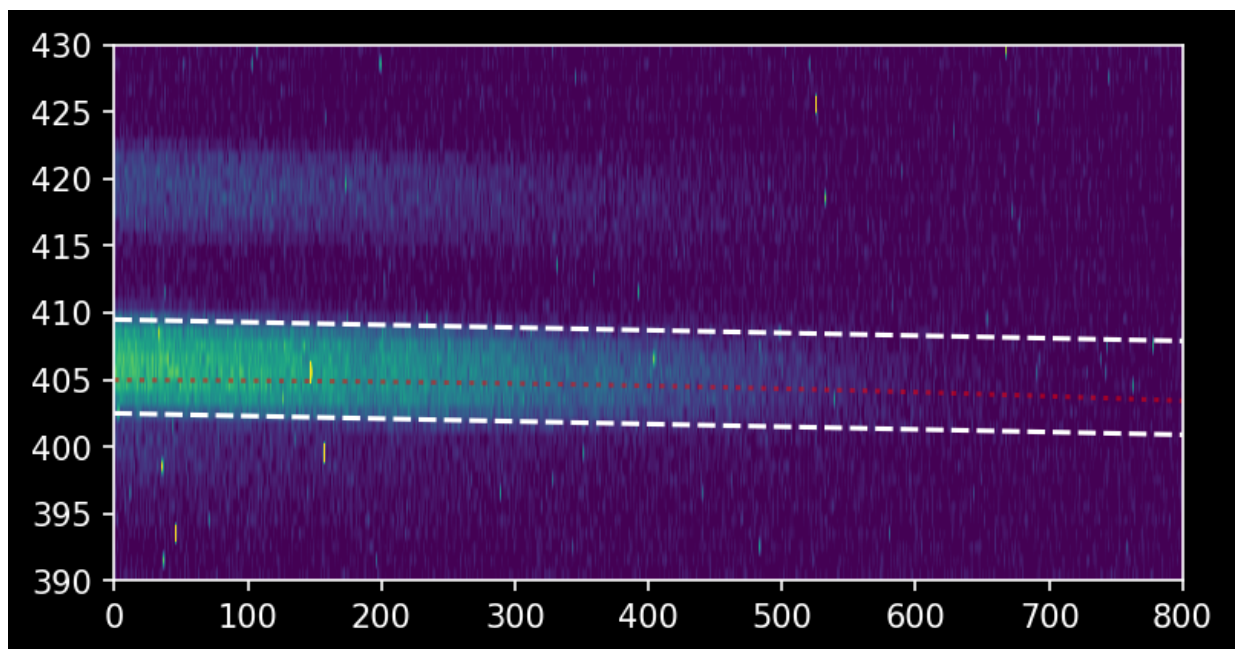
```

In [10]: # We fit a 2nd-order polynomial
polymodel = Polynomial1D(degree=2)
linfitter = LinearLSQFitter()
fitted_polymodels = {index: linfitter(polymodel, xvals[:xend], weighted_yaxis_val
                                for index, weighted_yaxis_values in traces.items())}

pl.imshow(io_array[ystart:yend, :xend],
          extent=[0,xend,ystart,yend],
          vmin=0, vmax=300,
          )
pl.plot([0,xend], ystart + intertrace_cuts + np.array([0,xend])[:,None] * slope,
pl.gca().set_aspect(10)
for tracefit in fitted_polymodels.values():
    pl.plot(xvals[:xend], tracefit(xvals[:xend]), 'r:', alpha=0.5)
pl.axis((0,xend,ystart,yend))

```

Out[10]: (0.0, 800.0, 390.0, 430.0)



```

In [11]: lmfitter = LevMarLSQFitter()
guess = Gaussian1D(amplitude=160, mean=0, stddev=5)

npixels_to_cut_trace = 4
spectra = {}
for trace_index, polymodel_trace in fitted_polymodels.items():
    trace_center = polymodel_trace(xvals)

    cutout_trace = np.array([io_array[int(yval)-npixels_to_cut_trace:int(yval)+npixels_to_cut_trace, ii]
                             for yval, ii in zip(trace_center, xvals)]).T

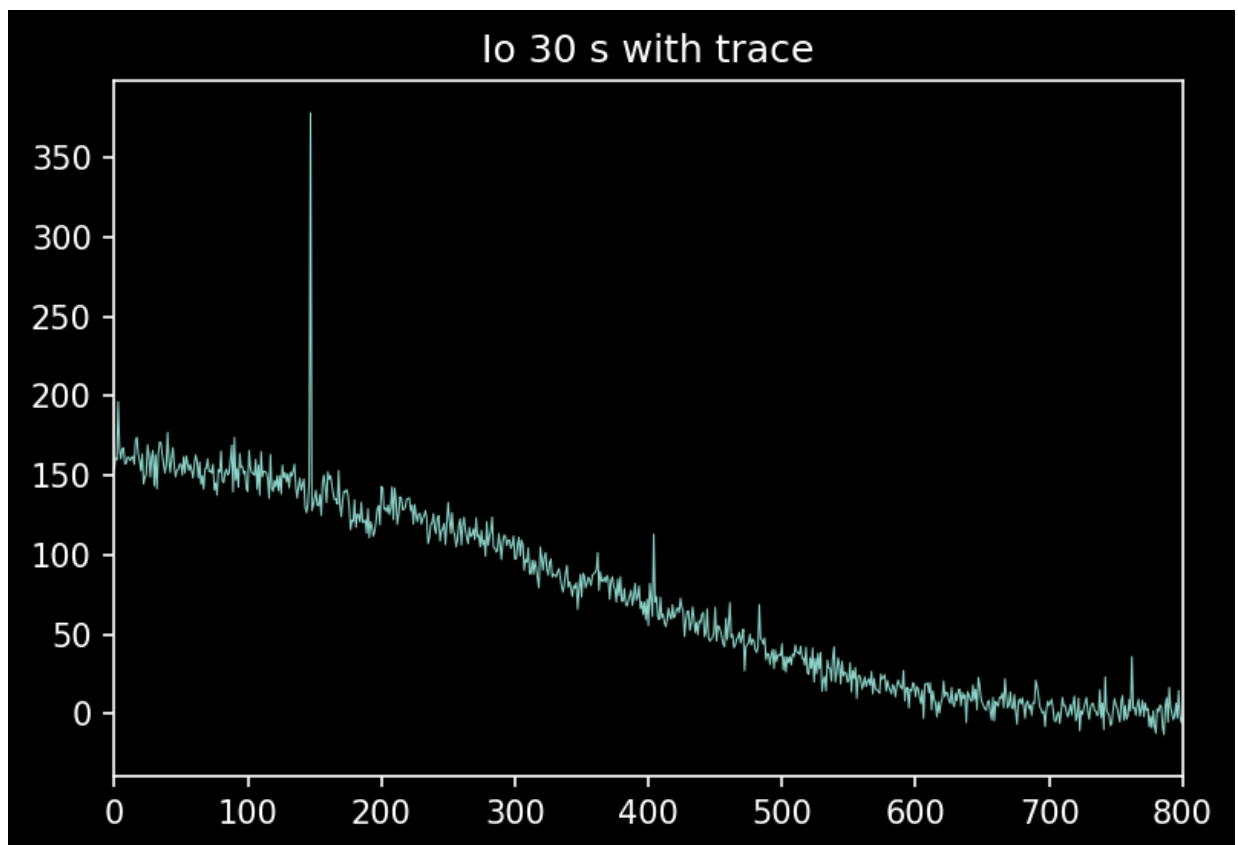
    trace_profile = cutout_trace.mean(axis=1)
    trace_profile_xaxis = np.arange(len(trace_profile))
    fitted_trace_profile = lmfitter(model=guess, x=trace_profile_xaxis, y=trace_profile)
    model_trace_profile = fitted_trace_profile(trace_profile_xaxis)

    trace_avg_spectrum = np.array([np.average(
        io_array[int(yval)-npixels_to_cut_trace:int(yval)+npixels_to_cut_trace, ii],
        weights=trace_profile)
        for yval, ii in zip(trace_center, xvals)])
    spectra[trace_index] = trace_avg_spectrum

for index in spectra:
    pl.plot(spectra[index], linewidth=0.5)
    pl.title("Io 30 s with trace")
pl.xlim(0,800)

```

Out[11]: (0.0, 800.0)

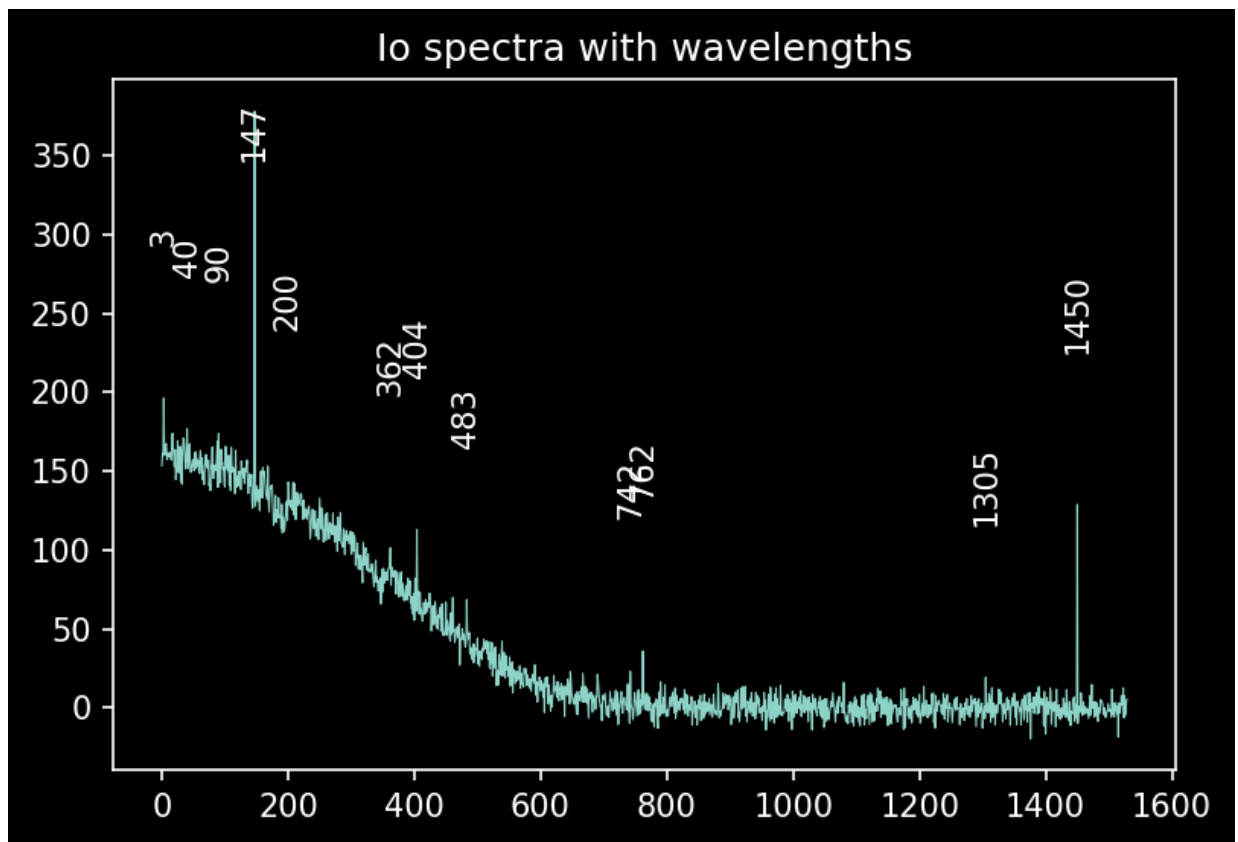


```
In [15]: import scipy.signal

mean_he = np.nanmean([spectra[ind] for ind in spectra], axis = 0)
pl.plot(mean_he, linewidth = 0.5)
pl.title("Io spectra with wavelengths")

peaks,_ = scipy.signal.find_peaks(mean_he, prominence = 30)

for peak in peaks:
    pl.text(peak,
           min([mean_he[peak] + 100, 350]), peak,
           rotation = 90, horizontalalignment = 'center',)
```



```
In [16]: peaks
```

```
Out[16]: array([ 3, 40, 90, 147, 200, 362, 404, 483, 742, 762, 1305,
                1450], dtype=int64)
```

## Ring

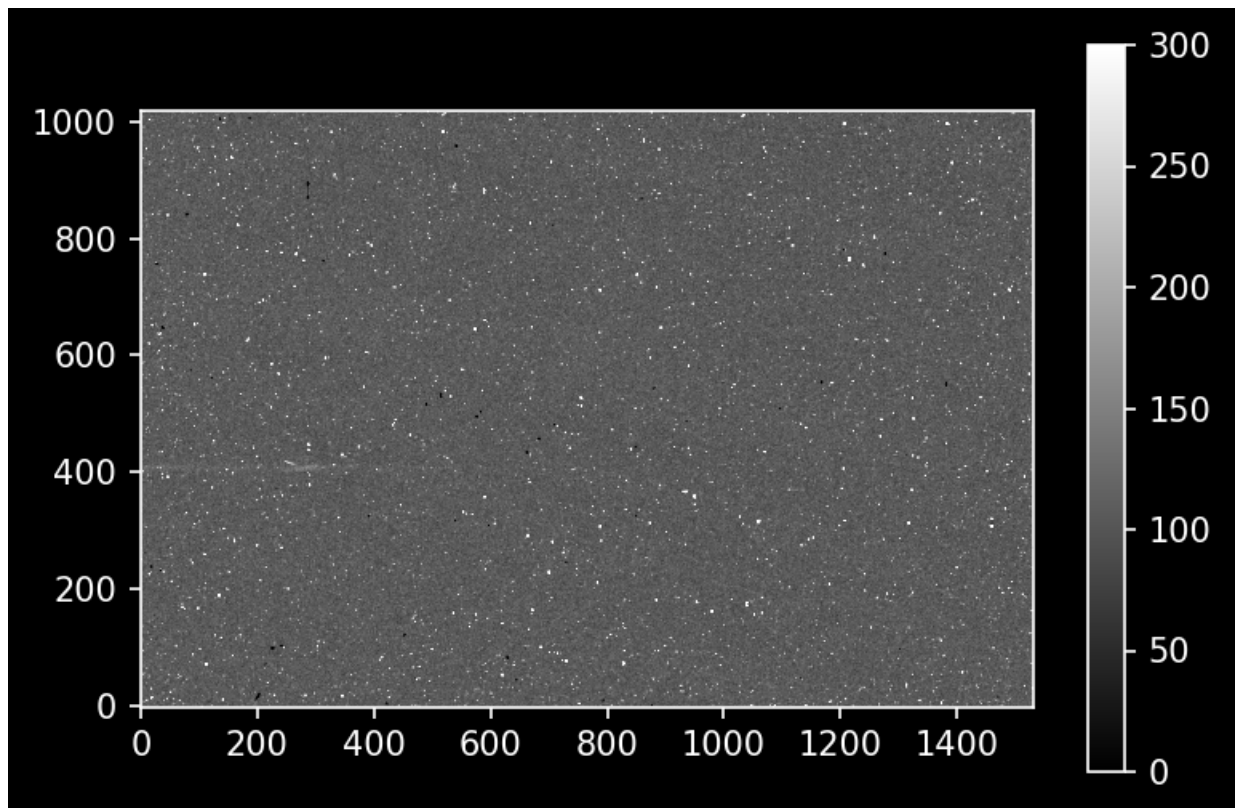


```
In [12]: ring_5m_image_data = (np.mean([fits.getdata(x) for x in glob.glob("\\Users\\Sydr
axis=0)
        - np.mean([fits.getdata(x)
                    for x in glob.glob("\\Users\\Sydney O'Donnell\\OneDrive\\
axis=0)
        )
```

```
In [13]: ring_array = np.array(ring_5m_image_data)
ring_array = ring_array - np.median(ring_5m_image_data)
```

```
In [14]: %matplotlib inline
import pylab as pl
pl.rcParams['image.origin'] = 'lower'
pl.rcParams['figure.dpi'] = 150
pl.matplotlib.style.use('dark_background') # Optional!
pl.imshow(ring_5m_image_data, cmap='gray', vmax=0, vmin=300)
pl.colorbar()
```

Out[14]: <matplotlib.colorbar.Colorbar at 0x245c6111370>



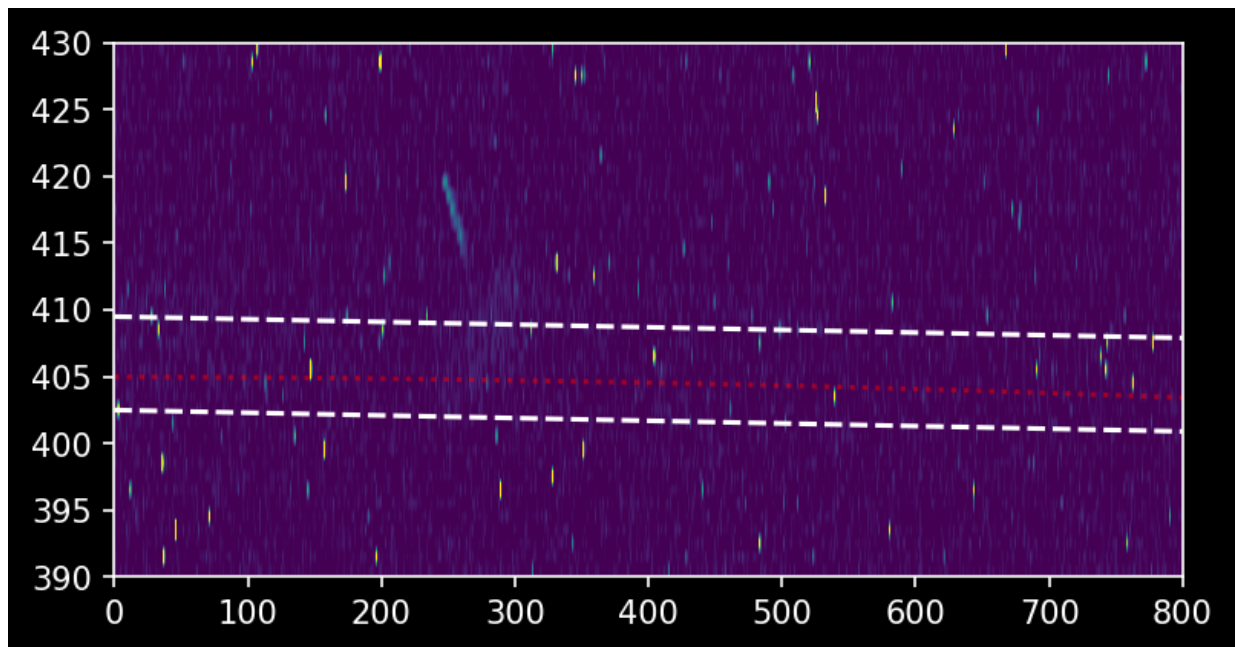
```

In [15]: # We fit a 2nd-order polynomial
polymodel = Polynomial1D(degree=2)
linfitter = LinearLSQFitter()
fitted_polymodels = {index: linfitter(polymodel, xvals[:xend], weighted_yaxis_val
                                for index, weighted_yaxis_values in traces.items())

pl.imshow(ring_array[ystart:yend, :xend],
          extent=[0,xend,ystart,yend],
          vmin=0, vmax=700,
          )
pl.plot([0,xend], ystart + intertrace_cuts + np.array([0,xend])[:,None] * slope,
pl.gca().set_aspect(10)
for tracefit in fitted_polymodels.values():
    pl.plot(xvals[:xend], tracefit(xvals[:xend]), 'r:', alpha=0.5)
pl.axis((0,xend,ystart,yend))

```

Out[15]: (0.0, 800.0, 390.0, 430.0)



```

In [16]: lmfitter = LevMarLSQFitter()
guess = Gaussian1D(amplitude=160, mean=0, stddev=5)

npixels_to_cut_trace = 4
spectra = {}
for trace_index, polymodel_trace in fitted_polymodels.items():
    trace_center = polymodel_trace(xvals)

    cutout_trace = np.array([ring_array[int(yval)-npixels_to_cut_trace:int(yval)+npixels_to_cut_trace, ii]
                             for yval, ii in zip(trace_center, xvals)]).T

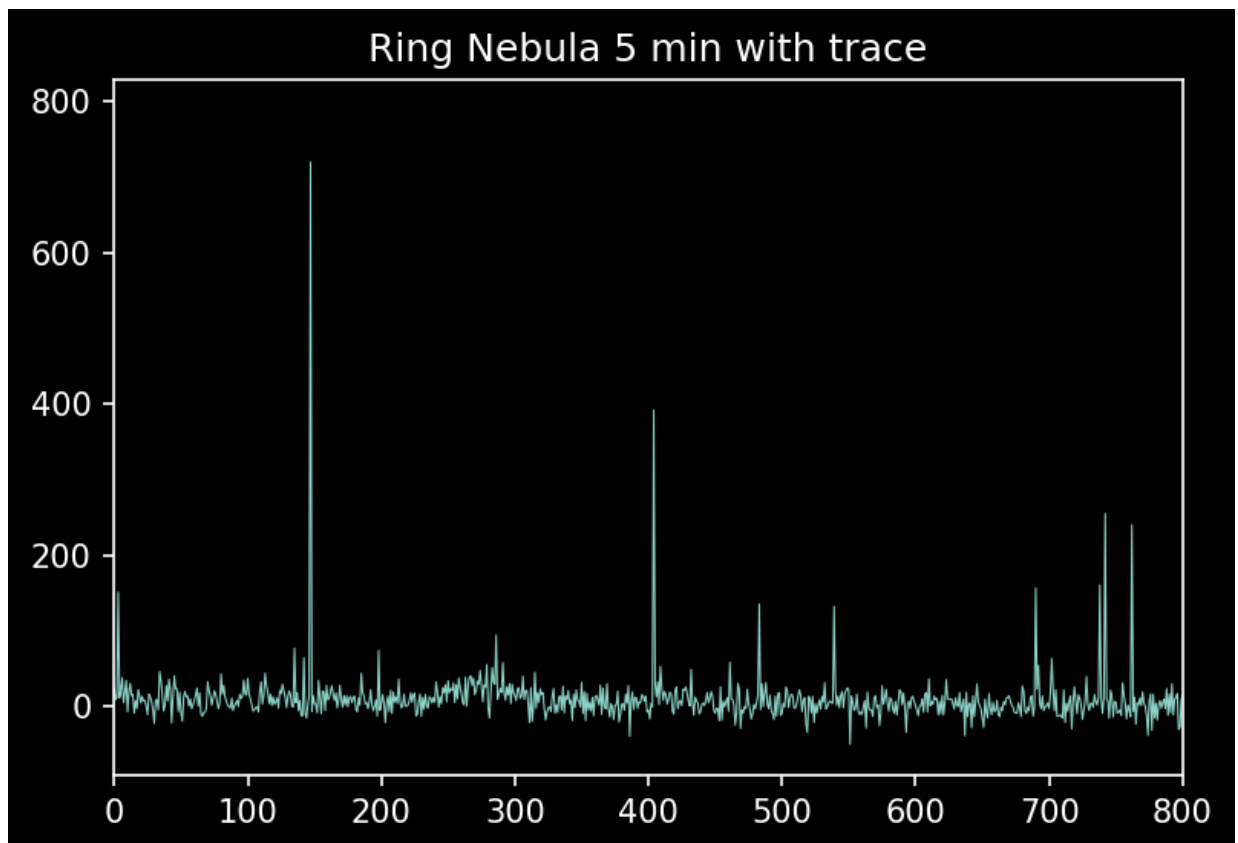
    trace_profile = cutout_trace.mean(axis=1)
    trace_profile_xaxis = np.arange(len(trace_profile))
    fitted_trace_profile = lmfitter(model=guess, x=trace_profile_xaxis, y=trace_profile)
    model_trace_profile = fitted_trace_profile(trace_profile_xaxis)

    trace_avg_spectrum = np.array([np.average(
        ring_array[int(yval)-npixels_to_cut_trace:int(yval)+npixels_to_cut_trace, ii],
        weights=trace_profile)
        for yval, ii in zip(trace_center, xvals)])
    spectra[trace_index] = trace_avg_spectrum

for index in spectra:
    pl.plot(spectra[index], linewidth=0.5)
    pl.title("Ring Nebula 5 min with trace")
pl.xlim(0,800)

```

Out[16]: (0.0, 800.0)

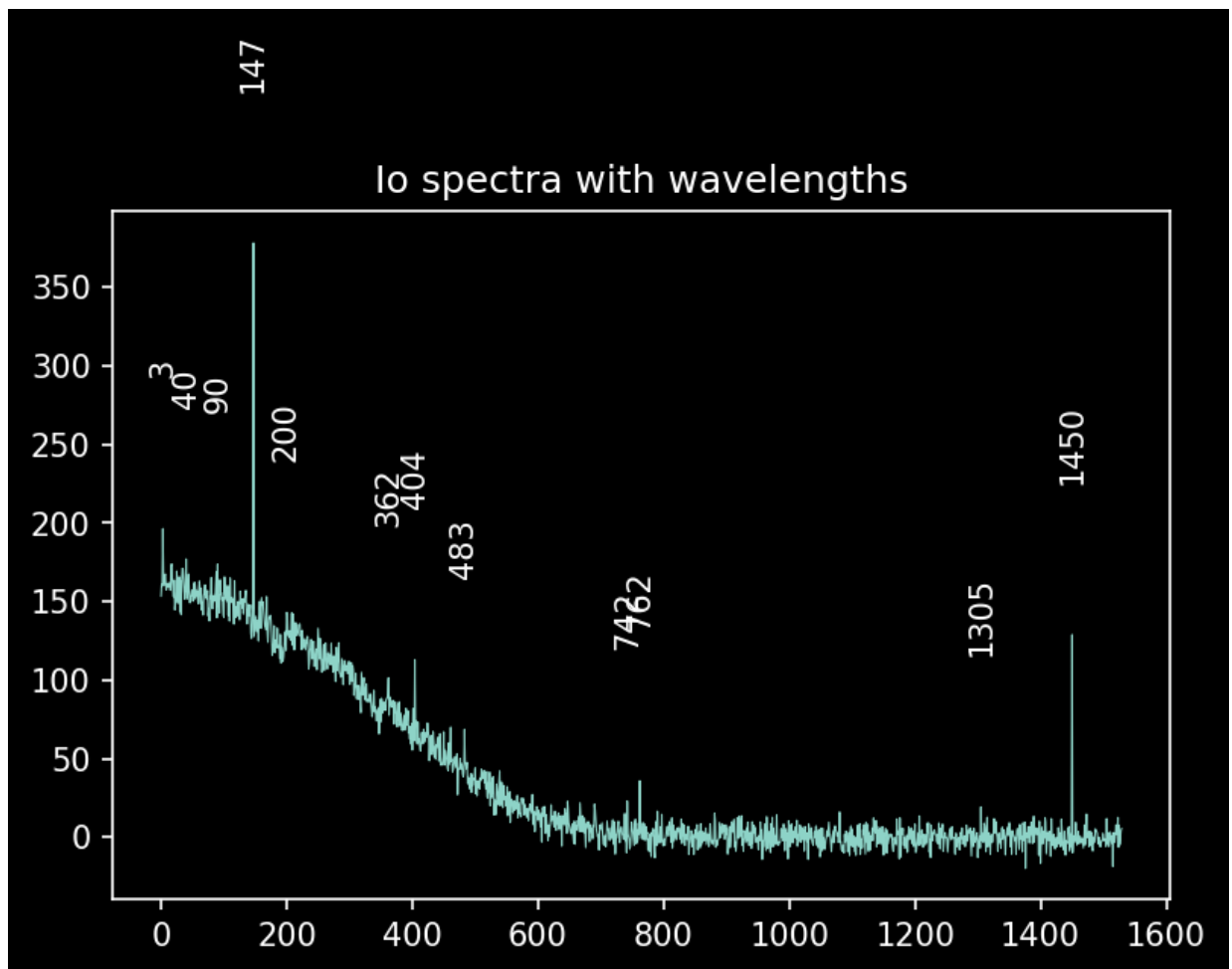


```
In [17]: import scipy.signal

mean_he = np.nanmean([spectra[ind] for ind in spectra], axis = 0)
pl.plot(mean_he, linewidth = 0.5)
pl.title("Io spectra with wavelengths")

peaks,_ = scipy.signal.find_peaks(mean_he, prominence = 30)

for peak in peaks:
    pl.text(peak,
            min([mean_he[peak] + 100, 800]), peak,
            rotation = 90, horizontalalignment = 'center',)
```



In [ ]:

