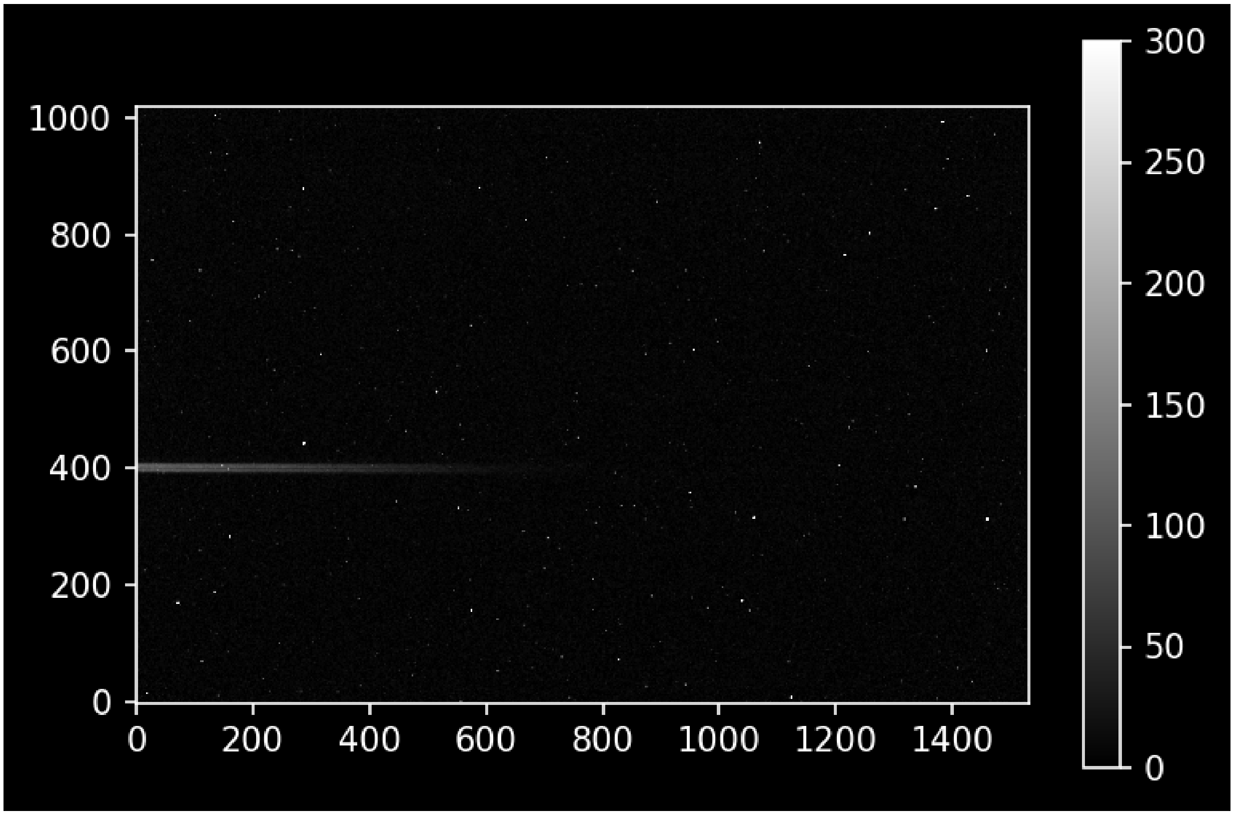
Europa 30s

In [1]:

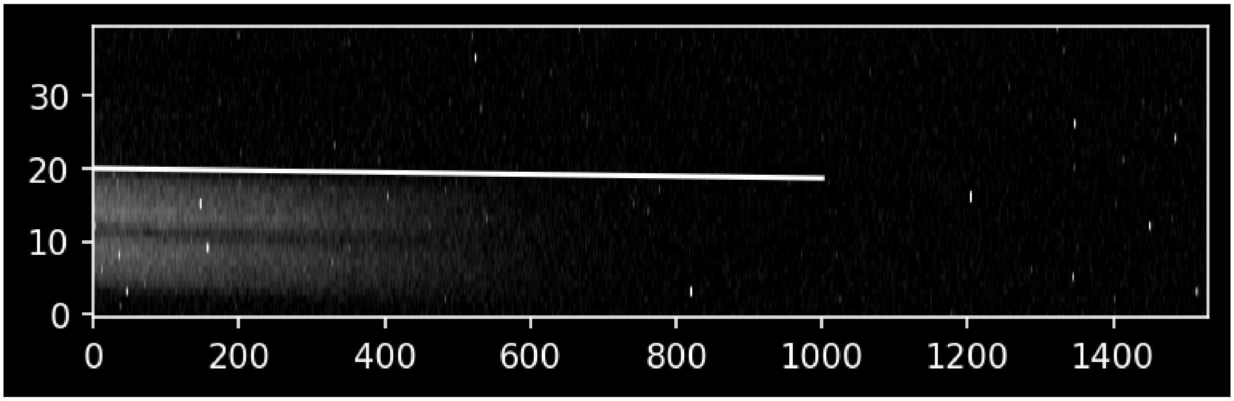
```
import numpy as np
import os
from astropy.io import fits
from astropy import units as u
from astropy.modeling.polynomial import Polynomial1D
from astropy.modeling.models import Gaussian1D, Linear1D
from astropy.modeling.fitting import LinearLSQFitter
from IPython.display import Image
# astroquery 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]:
```
europa_30s_image_data =  (np.mean([fits.getdata(x) for x in glob.glob("\\Users\\S
                      axis=0)
              - np.mean([fits.getdata(x)
                        for x in glob.glob("\\Users\\Sydnee O'Donnell\\OneDrive\
                        axis=0)
             )

eur_array = np.array(europa_30s_image_data)
eur_array = eur_array - np.median(europa_30s_image_data)

%matplotlib inline
pl.rcParams['image.origin'] = 'lower'
pl.rcParams['figure.dpi'] = 150
pl.matplotlib.style.use('dark_background') # Optional!
pl.imshow(europa_30s_image_data, cmap='gray', vmax=0, vmin=300)
pl.colorbar()
```

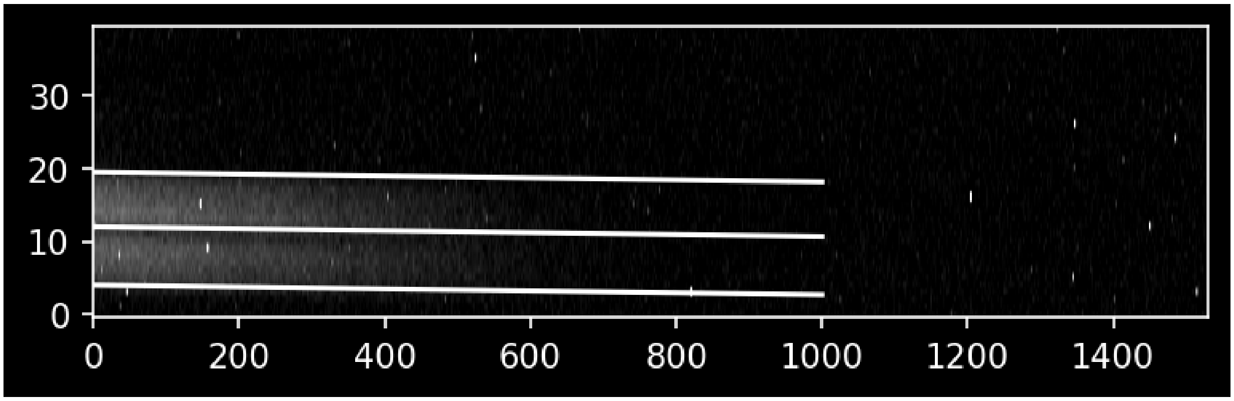
Out[2]:
```
<matplotlib.colorbar.Colorbar at 0x26571949760>
```



In [3]:
```
dy = -2
dx = 1500
slope = dy/dx

ystart = 390
yend = 430

pl.imshow(eur_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)
```



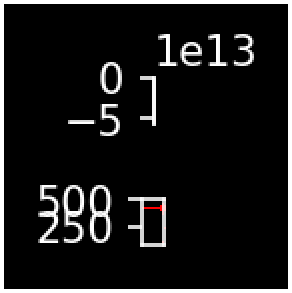
In [4]:
```
intertrace_cuts = np.array([4,12,19.5])
pl.imshow(eur_array[ystart:yend,:], cmap='gray', vmax=300, vmin=0)
pl.plot([0,1000], intertrace_cuts + np.array([0,1000])[:,None] * slope, color='w'
pl.gca().set_aspect(10)
```



In [5]:
```
npixels_to_cut = 4 # very conservative - we'll see why below
xvals = np.arange(eur_array.shape[1])
trace_center = ystart+(intertrace_cuts[0] + intertrace_cuts[1])/2 + xvals * slope
cutout_trace = np.array([eur_array[int(yval)-npixels_to_cut:int(yval)+npixels_to_
                    for yval, ii in zip(trace_center, xvals)]).T
```

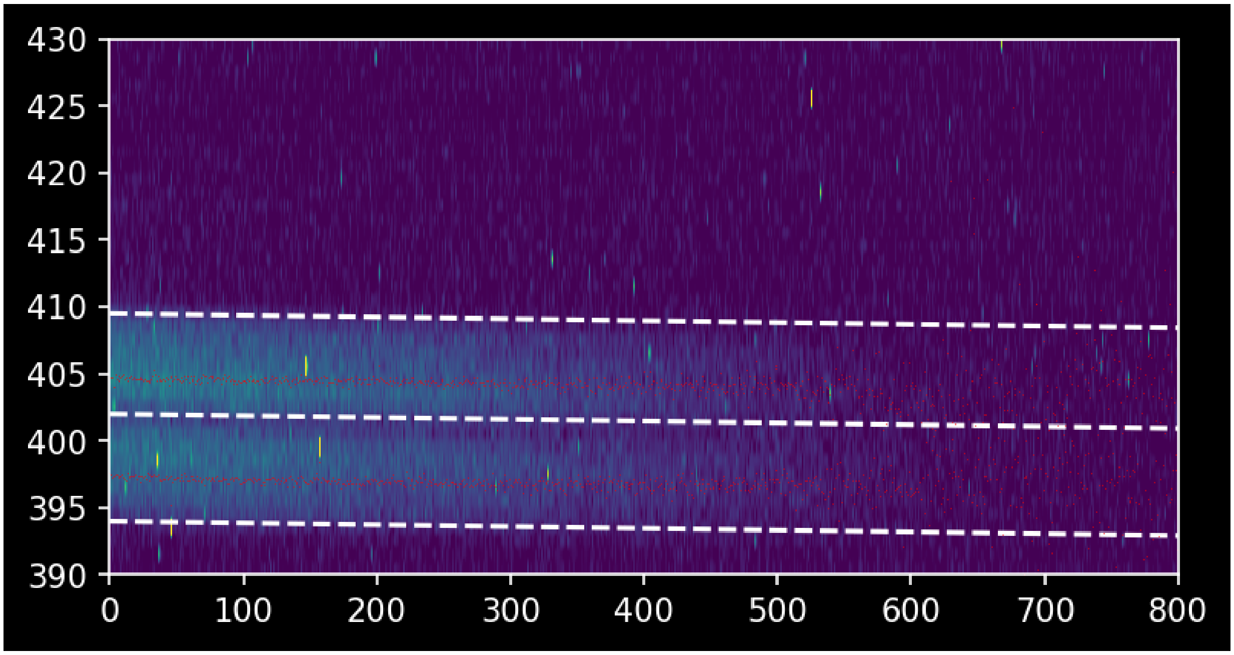
In [6]:

```
# to get the y-axis values corresponding to each part of our cutout trace, we do
yaxis_full = np.arange(eur_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_inde
    trace_center = yoffset + slope * xvals

    cutout_trace = np.array([eur_array[int(yval)-npixels_to_cut:int(yval)+npixels
                        for yval, ii in zip(trace_center, xvals)]).T
    yaxis = np.array([yaxis_full[int(yval)-npixels_to_cut:int(yval)+npixels_to_cu
                     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, yo
    ax.plot(xvals[:xend], yoffset - npixels_to_cut + local_weighted_yaxis_values
    ax.set_aspect(4)
    ax.set_xticks([])
pl.tight_layout()
```



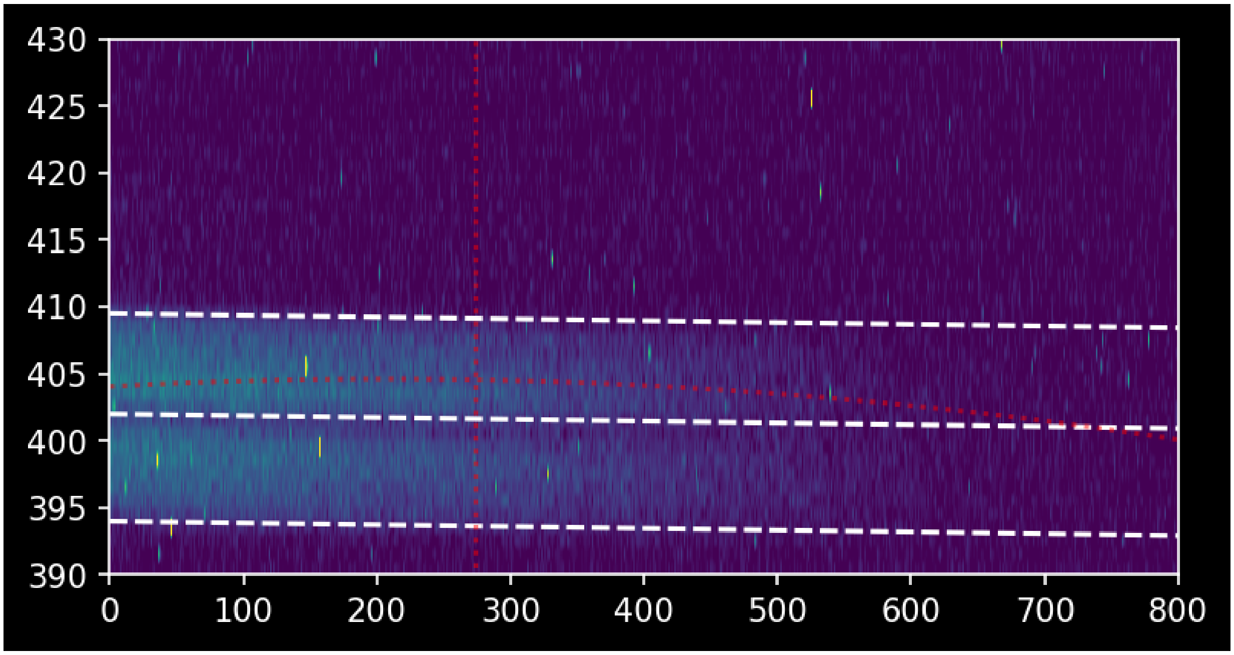
In [7]:
```
# then we can plot the "global" version here
pl.imshow(eur_array[ystart:yend, :xend],
          extent=[0,xend,ystart,yend], vmax=300, vmin=0)
pl.plot([0,xend], ystart + intertrace_cuts + np.array([0,xend])[:,None] * slope,
pl.gca().set_aspect(10)
for trace in traces.values():
    pl.plot(xvals[:xend], trace[:xend], 'r,', alpha=0.5)
pl.axis((0,xend,ystart,yend))
```

Out[7]:
```
(0.0, 800.0, 390.0, 430.0)
```



In [13]:
```
# We fit a 2rd-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(eur_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[13]:
```
(0.0, 800.0, 390.0, 430.0)
```



In [14]:

```
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([eur_array[int(yval)-npixels_to_cut_trace:int(yval)+n
                        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_p
    model_trace_profile = fitted_trace_profile(trace_profile_xaxis)

    trace_avg_spectrum = np.array([np.average(
            eur_array[int(yval)-npixels_to_cut:int(yval)+npixels_to_cut, 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("Europa 30 s with trace")
pl.xlim(0,800)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-14-5ce2621658c1> in <module>
     12     trace_profile = cutout_trace.mean(axis=1)
     13     trace_profile_xaxis = np.arange(len(trace_profile))
---> 14     fitted_trace_profile = lmfitter(model=guess, x=trace_profile_xaxis,
y=trace_profile)
     15     model_trace_profile = fitted_trace_profile(trace_profile_xaxis)
     16

~\AppData\Roaming\Python\Python38\site-packages\astropy\modeling\fitting.py in
wrapper(self, model, x, y, z, **kwargs)
    259         else:
    260
--> 261             return func(self, model, x, y, z=z, **kwargs)
    262
    263     return wrapper

~\AppData\Roaming\Python\Python38\site-packages\astropy\modeling\fitting.py in
__call__(self, model, x, y, z, weights, maxiter, acc, epsilon, estimate_jacobia
n)
   1154             dfunc = self._wrap_deriv
   1155         init_values, _ = _model_to_fit_params(model_copy)
-> 1156         fitparams, cov_x, dinfo, mess, ierr = optimize.leastsq(
   1157             self.objective_function, init_values, args=farg, Dfun=dfunc
,
   1158             col_deriv=model_copy.col_fit_deriv, maxfev=maxiter, epsfcn=
epsilon,
C:\ProgramData\Anaconda3\lib\site-packages\scipy\optimize\minpack.py in leastsq(func, x0, args, Dfun, full_output, col_deriv, ftol, xtol, gtol, maxfev, epsfcn, factor, diag)
    412 
    413     if n > m:
--> 414         raise TypeError('Improper input: N=%s must not exceed M=%s' % (n, m))
    415 
    416     if epsfcn is None:

TypeError: Improper input: N=3 must not exceed M=0
```

```
In [ ]:
```