In [1]:
```
from astropy import units as u
import numpy as np
```

Pepito Characterization Exercise & Lab

The fiber output from Pepito has 7 spots that each have diameter 100μm. They are separated by less than that, maybe $\sim 10\mu$m.

The expected input is from an f/10 beam, so the first lens is an f/10 collimator.

Parallel rays come and hit the diffraction grating, which has 830 lines/mm. It is about 2cm long.

The diffracted light is then focused onto the CCD with an f/10 camera.

The SBIG has 9 micron pixels.

Fill in the blanks.

Questions are asked using

bigger

fonts

The angle of the bend in Pepito is *approximately* 20 degrees.

Using a protractor or other device, what angle do you measure for the whole system?

In [2]:
```
angle = 22
```

The angle of the grating is *supposedly* 0 degrees with respect to the collimated beam - we'll assume that at the start of this exercise. But, now's a good time to measure it.

Using the same tool, what is the angle of the grating with respect to the incoming light?

(you won't use this until the redesign section below

In [3]:
```
grating_angle = 0
```

The information above comes from lab notes and Amanda Townsend's thesis. We will aim to verify these measurements.

In [4]:
```
grooves_per_mm = 830 *u.mm**-1
pixel_size = 10*u.um
f_cam = 10 * u.cm
f_cam = f_cam.to(u.mm)
```

At what angle is the first order for $\lambda = 4000, 5000, 6000 \dot{A}$?

Recall the grating equation:

$$n\lambda = D\sin\theta$$

where

- n is the order number and must be an integer
- λ is the wavelength
- D is the distance between holes (gaps) in the grating
- θ is the angle defined such that $\theta = 0$ is perpendicular to the grating (or parallel to the direction of the light)

In [5]:
```
order_number = 1
wavelength = [4000,5000,6000]*u.AA

D = 1/ grooves_per_mm

theta = np.arcsin(wavelength / D)
theta = theta.to(u.deg)
theta
```

Out[5]: $[19.390212,\ 24.519316,\ 29.867769]\ ^\circ$

What wavelength is centered assuming the angle is 20 deg?

In [6]:
```
degrees20 = 20*u.deg
degrees20 = degrees20.to(u.rad)

wavelength_20 = (D) * np.sin(degrees20)
wavelength_20 = wavelength_20.decompose()
wavelength_20 = wavelength_20.to(u.AA)
wavelength_20
```

Out[6]: $4120.7246\ \mathring{A}$

What is the wavelength difference per pixel?

Recall that the spatial separation per unit wavelength is:

$$\frac{dx}{d\lambda} = \frac{\Delta x}{\Delta \lambda} = \frac{f_{cam} n}{D \cos \theta}$$

where n is the order number, D is the distance between gaps in the grating, $dx \sim \Delta x$ is the pixel spacing, and $d\lambda \sim \Delta\lambda$ is the wavelength spacing.

In [7]:
```
d_groove = grooves_per_mm**-1

dispersion = f_cam / d_groove /  np.cos(degrees20)
dispersion
```

Out[7]: 88326.755

In [8]:
```
dlambda_per_pix = d_groove *  np.cos(degrees20) / f_cam
dlambda_per_pix
```

Out[8]: 1.1321598×10^{-5}

Type *Markdown* and LaTeX: α^2

What is the resolution limit from our slit?

Recall, to be diffraction-limited, the spectrograph must have

$$\sin \theta_{slit} < \frac{n\Delta\lambda}{D}$$

In [9]:
```
thing = (dlambda_per_pix * pixel_size )/ d_groove
thing = thing.decompose()
slit_resolution_limit = np.arcsin(thing)
slit_resolution_limit = slit_resolution_limit.to(u.arcsec)
slit_resolution_limit
```

Out[9]: $19.382552\,''$

What is the effect if our slit is bigger than this?

If the slit is bigger then our resolution limit gets smaller approaching 0

Assume that the size of the fiber pinholes is 100 μm.

Our telescope is a 14" f/10.

Is the fiber slit limiting our resolution?

In [10]:
```
focal_length = (14 * u.imperial.inch) * 10
plate_scale = 1 / focal_length

thing2 = (100*u.um) * plate_scale
thing2 = thing2.decompose()

angsize_of_slit = thing2 *(u.rad)
angsize_of_slit = angsize_of_slit.to(u.arcsec)
angsize_of_slit
```

Out[10]: $5.8004726''$

In [11]:
```
lim_slit = np.sin(slit_resolution_limit)
lim_fiber = np.sin(angsize_of_slit)

print(lim_slit)
print(lim_fiber)
```

```
9.396926207859084e-05
2.81214848106917e-05
```

What is the expected resolution of Pepito at 5000 Angstroms??

In [12]:
```
# R = lambda/ del lambda
resolution = (5000 * u.AA)/dlambda_per_pix
resolution = resolution.decompose()
resolution
```

Out[12]: 0.044163378 m

What if we use the other grating?

Our second grating has 1200 grooves/mm.

Assume it's operating at the same angle. What wavelength is at the center?

$$\sin \theta_{slit} < \frac{n\Delta\lambda}{D}$$

In [13]:
```
grooves_per_mm_2 = 1200*u.mm**-1
```

In [14]:
```
D2 = 1/grooves_per_mm_2

wavelength_20_D2 = (D2) * np.sin(degrees20)
wavelength_20_D2 = wavelength_20_D2.decompose()
wavelength_20_D2 = wavelength_20_D2.to(u.AA)
wavelength_20_D2
```

Out[14]: 2850.1679 Å

What angle do we need to position our camera at if we want the same central wavelength as the first grating?

In [15]:
```
angle_to_match_800gmm_grating = np.arcsin(wavelength_20 / D2)
angle_to_match_800gmm_grating = angle_to_match_800gmm_grating.to(u.deg)
angle_to_match_800gmm_grating
```

Out[15]: 29.635925 °

What's the effective resolution (at the same central wavelength)?

In [16]:
```
dispersion_1200gmm_grating = f_cam / D2 /  np.cos(angle_to_match_800gmm_grating)
dispersion_1200gmm_grating
```

Out[16]: 138060.36

In [17]:
```
dlambda_per_pix_1200gmm_grating = (D2 *  np.cos(angle_to_match_800gmm_grating) /
dlambda_per_pix_1200gmm_grating ## should be in AA ?
```

Out[17]: 7.2432088×10^{-6}

What is the resolution?

In [18]:
```
resolution2 = (5000 * u.AA)/dlambda_per_pix_1200gmm_grating
resolution2 = resolution2.decompose()
resolution2
```

Out[18]: 0.069030179 m

What is the slit size required for our system to be grating-limited?

In [19]:
```
thing2 = (dlambda_per_pix_1200gmm_grating * pixel_size )/ D2
thing2 = thing2.decompose()
slit_resolution_limit_1200gmm_grating = np.arcsin(thing2)
slit_resolution_limit_1200gmm_grating = slit_resolution_limit_1200gmm_grating.to(
slit_resolution_limit_1200gmm_grating
```

Out[19]: 17.928229 ″

Redesign questions

Remember that the incident angle of light can be nonzero, resulting in the modified grating equation

$$n\lambda = D\left(\sin\theta_{in} + \sin\theta_{out}\right)$$

Our spectrograph has a fixed angle, which you've measured (but hope to measure even more precisely).

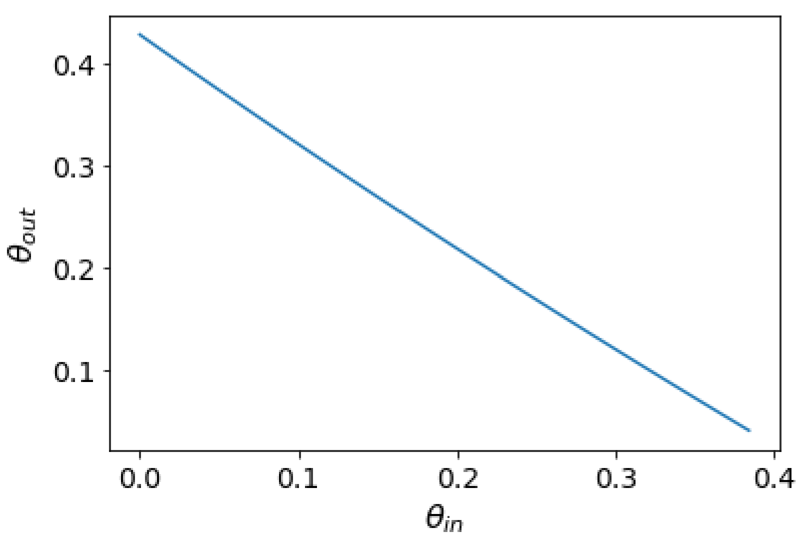
Given that measured angle of the *whole system*, at what angles is it possible to center $\lambda = 500$ nm, assuming we're using the grating in its first order?

(Exercise for students)

In [20]:
```
import pylab as pl
pl.rcParams['figure.facecolor'] = 'w'
pl.rcParams['font.size'] = 14
```

In [21]:
```
angle = (22*u.degree).to(u.rad)
theta_in = np.linspace(0,angle)
wavelength_500 = 500 * u.nm

theta_out = np.arcsin((wavelength_500 / D ) - np.sin(theta_in))
pl.plot(theta_in, theta_out)
pl.xlabel(r"$\theta_{in}$", fontsize=16)
pl.ylabel(r"$\theta_{out}$", fontsize=16);
```



Say we want to position 500 nm at the center of our detector, and the angle of the system is still the same we assumed above.

Can we rotate the grating such that 500nm will be centered on the detector? If so, at what angle?

In [22]:
```
theta_in1 = 22 * u.deg
theta_out = np.arcsin((wavelength_500 / D) - np.sin(theta_in1))
theta_out.to(u.deg)
```

Out[22]: $2.3150015\,^{\circ}$

Can we rotate the grating such that 400nm will be centered on the detector? If so, at what angle?

In [23]:
```
theta_in1 = 22 * u.deg
wavelength_400 = 400 * u.nm
theta_out = np.arcsin((wavelength_400 / D) - np.sin(theta_in1))
theta_out.to(u.deg)
```

Out[23]: $-2.4419172\,^{\circ}$

Can't be roated

Can we rotate the grating such that 656.3nm will be centered on the detector? If so, at what angle? Why is this a significant wavelength?

In [24]:
```
theta_in1 = 22 * u.deg
wavelength_400 = 656.3 * u.nm
theta_out = np.arcsin((wavelength_400 / D) - np.sin(theta_in1))
theta_out.to(u.deg)
```

Out[24]: $9.7949361\,^{\circ}$

first balmer line of hydrogen

What is the actual angle of the grating? And what central wavelength does it imply?

In [25]:
```
theta_in1 = 20 * u.deg
theta_out = 0 * u.deg

central_wavelength = D * (np.sin(theta_in1) + np.sin(theta_out))
central_wavelength.to(u.nm)
```

Out[25]: $412.07246\ \mathrm{nm}$

can assume angle is zero with respect to incoming light

Can we rotate the grating such that 400nm will be centered on the detector? If so, at what angle?

In [26]:
```
theta_in1 = 22 * u.deg
wavelength_420 = 420 * u.nm
theta_out = np.arcsin((wavelength_420 / D) - np.sin(theta_in1))
theta_out.to(u.deg)
```

Out[26]: $-1.4902361\ ^{\circ}$

What range of wavelengths can we center by rotating the grating?

In [27]:
```
theta_in1 = 20 * u.deg
theta_out_range = (0,91,1) * u.deg

central_wavelength = D * (np.sin(theta_in1) + np.sin(theta_out_range))

print("min'",np.min(central_wavelength.to(u.nm)))
print("max'",np.max(central_wavelength.to(u.nm)))
```

```
min' 412.0724618381551 nm
max' 1616.708239135012 nm
```

Lab Measurements Part II

Each group will need to take turns performing these measurements, since we have only 1 Pepito.

Set up Pepito to take in-lab spectra. Obtain spectra of:

- The overhead fluorescent bulbs
- Helium
- Hydrogen
- Neon

Ensure that the spectra are obtained in-focus and properly aligned on the detector.

Be careful that the fiber does not rotate between observations!

Examine the spectra. Start with hydrogen.

Recall from your quantum class that the wavelength of hydrogen lines is given by:

$$\frac{1}{\lambda} = Ry\left(\frac{1}{n_l}^2 - \frac{1}{n_u}^2\right)$$

What lines are in the spectrum?

CCD is sensitive to ~ 400 to ~1000 nm

what hydrogen lines could be in our spectrum

In [28]:
```
#R = 109677.57 *(u.cm)**-1
#R = R.to(u.nm**-1)
#
#n1 = [1,2,3,4,5,6,7,8]
#n2 = [2,3,4,5,6,7,8,9]
#
#lam_inv = []
#lam = []
#
#for i in n1:
#    lam_inv.append( * (1/n1)**2 - (1/n2)**2)
#    lam.append(1/lam_inv)

#lam_inverse = R * ((1/n1)**2 - (1/n2)**2)
#lam = 1/lam_inverses
```

n = 2 Balmer

n = 3 6562.6 h alpha

n = 4 4861 beta

n = 5 4339.4 gamma

n = 6 4100.7 delta

below 4000 angstroms blue cutoff, atmosphere is non transmissive. These lines are the primary ones we'll see, h alpha always the brightest. These are the only visible hydrogen lines

In [29]:
```
from PIL import Image as PILImage
import numpy as np
import pylab as pl
pl.rcParams['image.origin'] = 'lower'  # we want to show images, not matrices, sc
pl.matplotlib.style.use('dark_background')  # Optional configuration: if run, thi
```

In [30]:
```
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
```

In [31]:
```
import os
from astropy.io import fits
```

In [32]:
```
hg_filename = "\\Users\\Sydnee O'Donnell\\OneDrive\\UF\\Obs Tech 2\\BestGroup_Aug
hy_filename = "\\Users\\Sydnee O'Donnell\\OneDrive\\UF\\Obs Tech 2\\BestGroup_Aug
he_filename = "\\Users\\Sydnee O'Donnell\\OneDrive\\UF\\Obs Tech 2\\BestGroup_Aug
ne_filename = "\\Users\\Sydnee O'Donnell\\OneDrive\\UF\\Obs Tech 2\\BestGroup_Aug
sun_filename = "\\Users\\Sydnee O'Donnell\\OneDrive\\UF\\Obs Tech 2\\BestGroup_Au
hy1_filename = "\\Users\\Sydnee O'Donnell\\OneDrive\\UF\\Obs Tech 2\\BestGroup_Au
hy2_filename = "\\Users\\Sydnee O'Donnell\\OneDrive\\UF\\Obs Tech 2\\BestGroup_Au
```

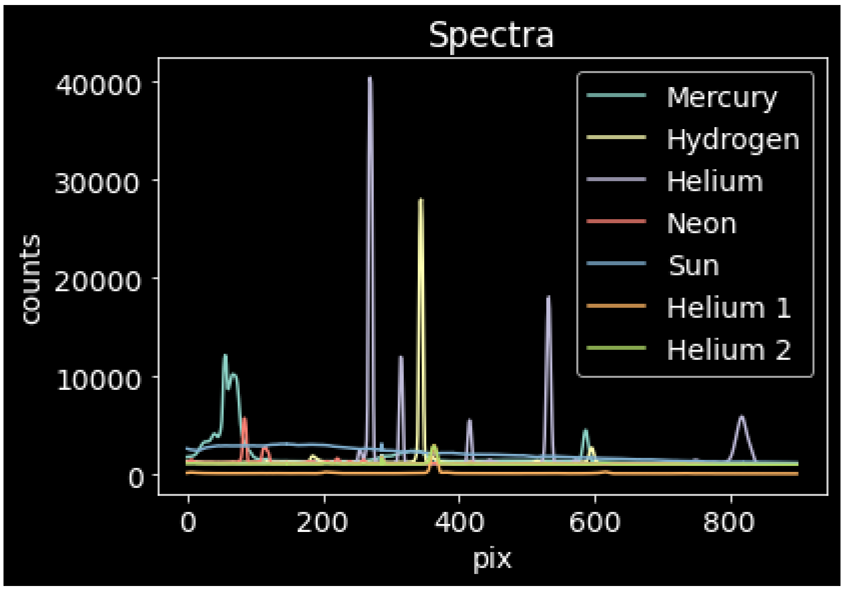
In [33]:
```
hg_image = fits.getdata(hg_filename)
hy_image = fits.getdata(hy_filename)
he_image = fits.getdata(he_filename)
ne_image = fits.getdata(ne_filename)
sun_image = fits.getdata(sun_filename)
hy1_image = fits.getdata(hy1_filename)
hy2_image = fits.getdata(hy2_filename)
```

Crop images to filter out any hot pixles and background noise so our mean isn't poluted

In [34]:
```
hg_spectrum = hg_image[350:450,0:900].mean(axis=0)
hy_spectrum = hy_image[350:450,0:900].mean(axis=0)
he_spectrum = he_image[350:450,0:900].mean(axis=0)
ne_spectrum = ne_image[350:450,0:900].mean(axis=0)
sun_spectrum = sun_image[350:450,0:900].mean(axis=0)
hy1_spectrum = hy1_image[350:450,0:900].mean(axis=0)
hy2_spectrum = hy2_image[350:450,0:900].mean(axis=0)
```

In [35]:
```
xaxis = np.arange(hg_image[350:450,0:900].shape[1])
pl.plot(xaxis, hg_spectrum, label='Mercury')
pl.plot(xaxis, hy_spectrum, label='Hydrogen')
pl.plot(xaxis, he_spectrum, label='Helium')
pl.plot(xaxis, ne_spectrum, label='Neon')
pl.plot(xaxis, sun_spectrum, label='Sun')
pl.plot(xaxis, hy1_spectrum, label='Helium 1')
pl.plot(xaxis, hy2_spectrum, label='Helium 2')
pl.legend(loc='best');
pl.xlabel("pix")
pl.ylabel('counts')
pl.title("Spectra")
```

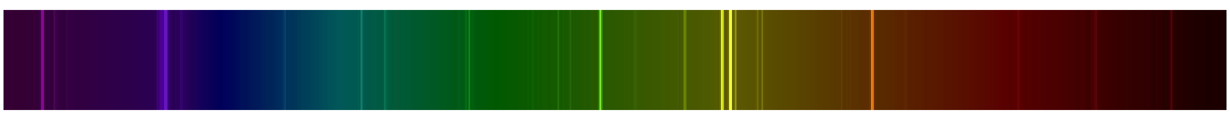
Out[35]:
```
Text(0.5, 1.0, 'Spectra')
```



Mercury

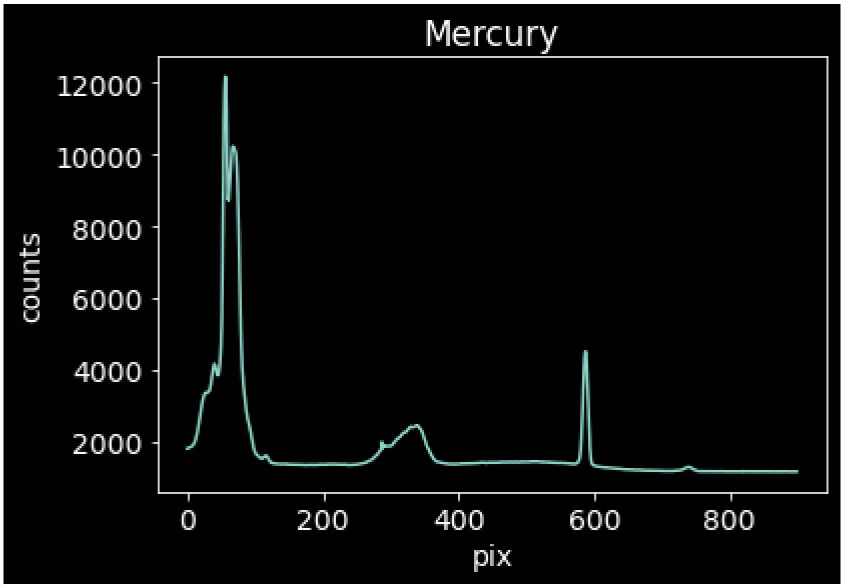
In [36]:
```
tableHg = Nist.query(4000 * u.AA, 7000 * u.AA, linename="Hg")
print(tableHg)
```

```
Spectrum  Observed    Ritz   Transition ... Type   TP    Line
-------- --------- --------- ---------- ... ---- ----- --------
    Hg I   4027.09   4027.15    24831.8 ...   --    -- L7588c86
   Hg II  4027.339        --   24830.29 ...   --    --   L11760
   Hg II  4041.599 4041.6036   24742.69 ...   -- T7226   L11760
   Hg II  4041.718        --  24741.956 ...   --    --   L11760
    Hg I 4047.7081 4047.7074  24705.339 ...   -- T5292    L7247
   Hg II  4048.883  4048.886  24698.167 ...   -- T7226   L11760
    Hg I 4078.9883 4078.9883  24515.883 ...   -- T5292    L7247
    Hg I  4109.213  4109.214   24335.56 ...   -- T3462    L3451
   Hg II  4121.602 4121.6094  24262.408 ...   -- T7226   L11760
  Hg III   4123.23        --    24252.8 ...   --    --    L3332
     ...       ...       ...        ... ...  ...   ...      ...
    Hg I    6872.6        --    14550.6 ...   --    -- L7590c86
    Hg I    6882.3   6882.71    14530.0 ...   --    -- L7590c86
    Hg I  6890.465  6890.463   14512.81 ...   --    --    L3499
    Hg I   6909.37   6909.37    14473.1 ...   -- T3462    L7394
    Hg I    6917.5        --    14456.1 ...   --    -- L7590c86
    Hg I    6926.2   6926.04    14437.9 ...   --    -- L7590c86
    Hg I    6935.9        --    14417.7 ...   --    -- L7590c86
    Hg I    6946.0        --    14396.7 ...   --    -- L7590c86
   Hg II   6947.39  6947.389  14393.895 ...   -- T7226   L11760
    Hg I    6952.6   6952.76    14383.0 ...   --    -- L7590c86
    Hg I    6979.1        --    14328.5 ...   --    --    L7590
Length = 206 rows
```



In [37]:
```
pl.plot(xaxis, hg_spectrum);
pl.xlabel("pix")
pl.ylabel('counts')
pl.title("Mercury")
```

Out[37]:
```
Text(0.5, 1.0, 'Mercury')
```



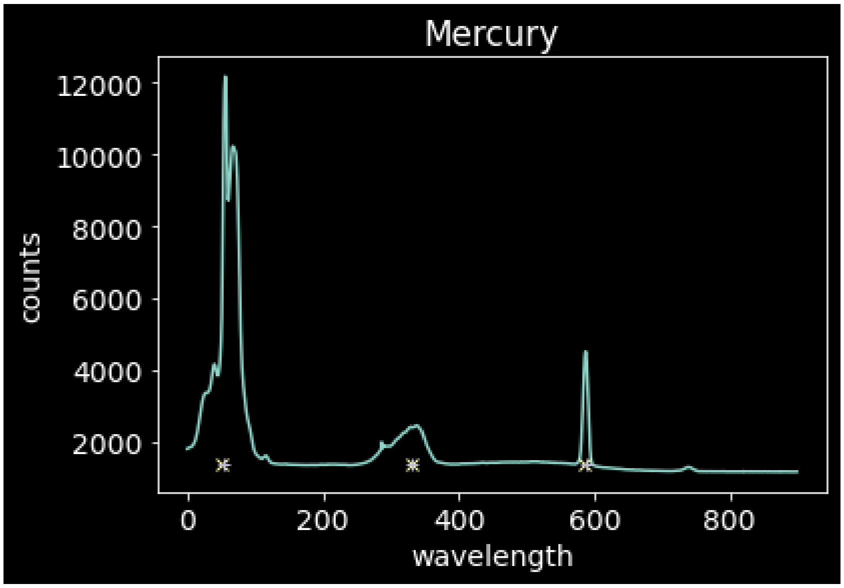
In [38]:
```
guessed_wavelengths_hg = [2280, 1300, 1590]
guessed_xvals_hg = [50, 330, 586]
```

In [39]:
```
npixels = 15
improved_xval_guesses_hg = [np.average(xaxis[g-npixels:g+npixels],
                                       weights=hg_spectrum[g-npixels:g+npixels] - np
                           for g in guessed_xvals_hg]
improved_xval_guesses_hg
```

Out[39]:
```
[53.66590615169603, 330.1679708239513, 587.1014495476982]
```

In [40]:
```
pl.plot(xaxis, hg_spectrum)
pl.plot(guessed_xvals_hg, [1400]*3, 'x')
pl.plot(improved_xval_guesses_hg, [1400]*3, '+');
pl.xlabel("wavelength")
pl.ylabel('counts')
pl.title("Mercury")
```

Out[40]:
```
Text(0.5, 1.0, 'Mercury')
```



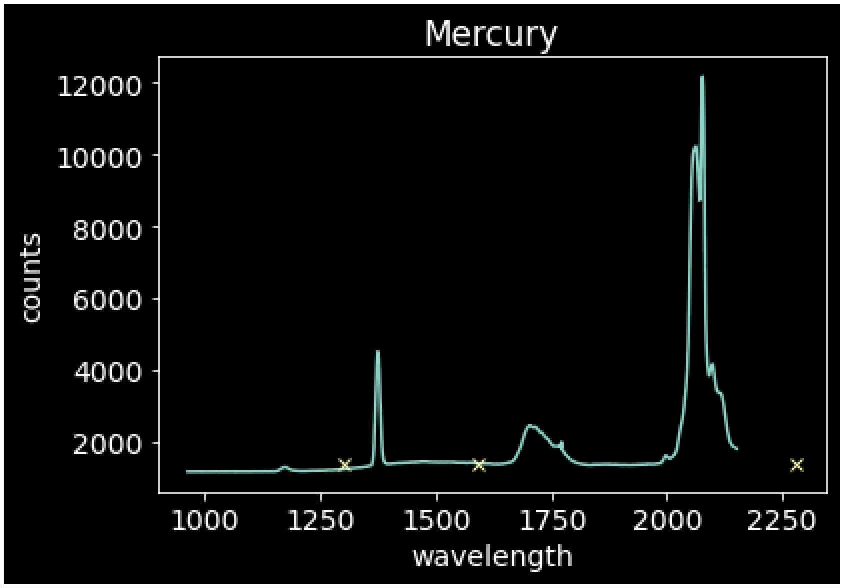
In [41]:
```
linfitter = LinearLSQFitter()
```

In [42]:
```
wlmodel = Linear1D()
linfit_wlmodel = linfitter(model=wlmodel, x=improved_xval_guesses_hg, y=guessed_w
wavelengths = linfit_wlmodel(xaxis) * u.nm
linfit_wlmodel
```

Out[42]:
```
<Linear1D(slope=-1.32202148, intercept=2151.19912052)>
```

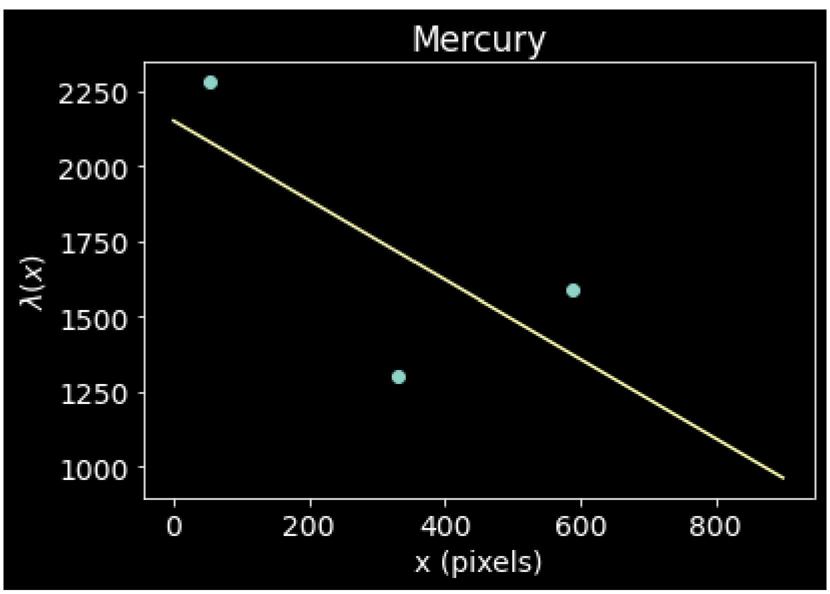
In [43]:
```
pl.plot(wavelengths, hg_spectrum)
pl.plot(guessed_wavelengths_hg, [1400]*3, 'x');
pl.xlabel("wavelength")
pl.ylabel('counts')
pl.title("Mercury")
```

Out[43]:
```
Text(0.5, 1.0, 'Mercury')
```



In [44]:
```
pl.plot(improved_xval_guesses_hg, guessed_wavelengths_hg, 'o')
pl.plot(xaxis, wavelengths, '-')
pl.ylabel("$\lambda(x)$")
pl.xlabel("x (pixels)")
pl.title("Mercury")
```

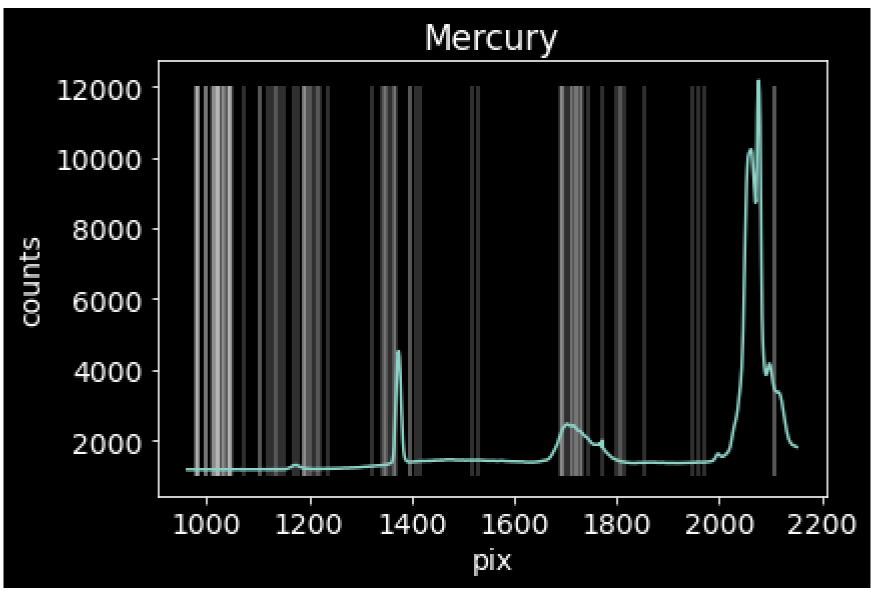
Out[44]:
```
Text(0.5, 1.0, 'Mercury')
```



In [45]:
```
# we adopt the minimum/maximum wavelength from our linear fit
minwave = wavelengths.min()
maxwave = wavelengths.max()
# then we search for atomic lines
# We are only interested in neutral lines, assuming the lamps are not hot enough
mercury_lines = Nist.query(minwav=minwave,
                           maxwav=maxwave,
                           linename='Hg I')
hydrogen_lines = Nist.query(minwav=minwave,
                            maxwav=maxwave,
                            linename='H I')
helium_lines = Nist.query(minwav=minwave,
                          maxwav=maxwave,
                          linename='He I')
neon_lines = Nist.query(minwav=minwave,
                        maxwav=maxwave,
                        linename='Ne I')
```

In [46]:
```
pl.plot(wavelengths, hg_spectrum)
pl.vlines(mercury_lines['Observed'], 1000, 12000, 'w', alpha=0.25);
pl.xlabel("pix")
pl.ylabel('counts')
pl.title("Mercury")
```

Out[46]:
```
Text(0.5, 1.0, 'Mercury')
```



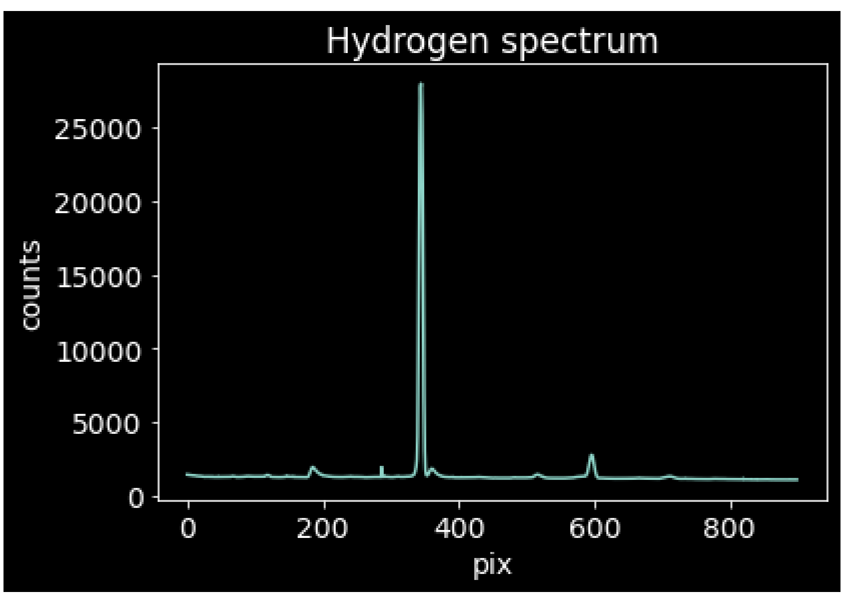
Hydrogen

In [47]:
```
tableHI = Nist.query(4000 * u.AA, 7000 * u.AA, linename="H I")
print(tableHI)
```

```
  Observed        Ritz       Transition   Rel.  ... Type  TP    Line
------------- ------------- ------------- ------ ... ---- ----- --------
           -- 4102.85985517 24373.2429403     -- ...   -- T8637       --
           -- 4102.86191087 24373.2307283     -- ...   -- T8637       --
           --     4102.8632    24373.223     -- ...   --    --      c57
4102.86503481 4102.86503481 24373.2121704     -- ...   E2    --   L11759
           -- 4102.86579132 24373.2076763     -- ...   -- T8637       --
4102.86785074 4102.86785074 24373.1954423     -- ...   M1    --   L11759
           --  4102.8680725 24373.1941249     -- ...   -- T8637       --
     4102.892     4102.8991     24373.05  70000 ...   -- T8637 L7436c29
           --     4102.8922    24373.051     -- ...   --    --      c58
           -- 4102.92068748 24372.8815683     -- ...   -- T8637       --
          ...           ...           ...    ... ...  ...   ...      ...
           --   6564.564672  15233.302588     -- ...   -- T8637       --
           --   6564.579878  15233.267302     -- ...   M1    --       --
           --      6564.583     15233.26     -- ...   --    --      c66
  6564.584404   6564.584403  15233.256799     -- ...   -- T8637 L6891c38
       6564.6      6564.632     15233.21 500000 ...   -- T8637 L7400c29
           --      6564.608    15233.202     -- ...   --    --      c69
   6564.66464    6564.66466   15233.07061     -- ...   -- T8637    L2752
           --     6564.6662    15233.067     -- ...   --    --      c71
           --      6564.667    15233.065     -- ...   --    --      c70
           --   6564.680232  15233.034432     -- ...   -- T8637       --
           --   6564.722349    15232.9367     -- ...   -- T8637       --
Length = 53 rows
```

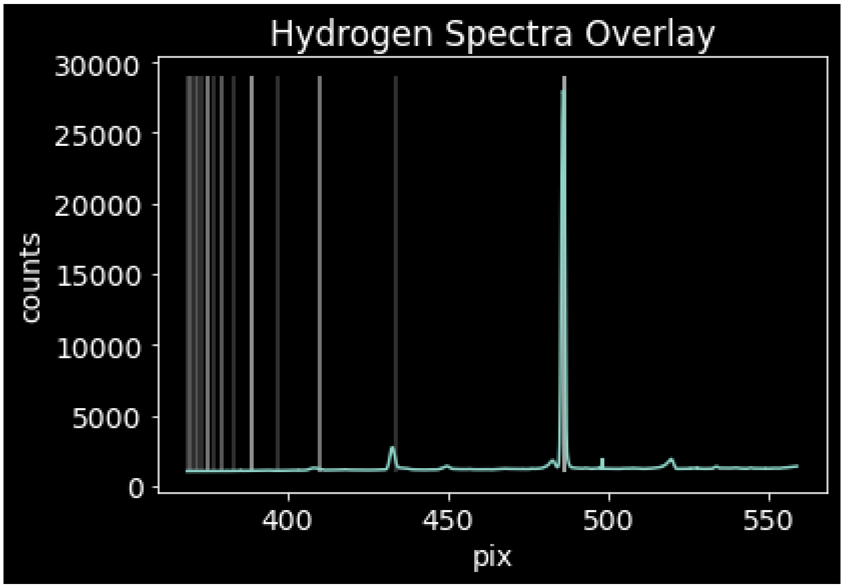
In [48]:
```
pl.plot(xaxis, hy_spectrum);
pl.xlabel("pix")
pl.ylabel('counts')
pl.title("Hydrogen spectrum")
```

Out[48]: Text(0.5, 1.0, 'Hydrogen spectrum')



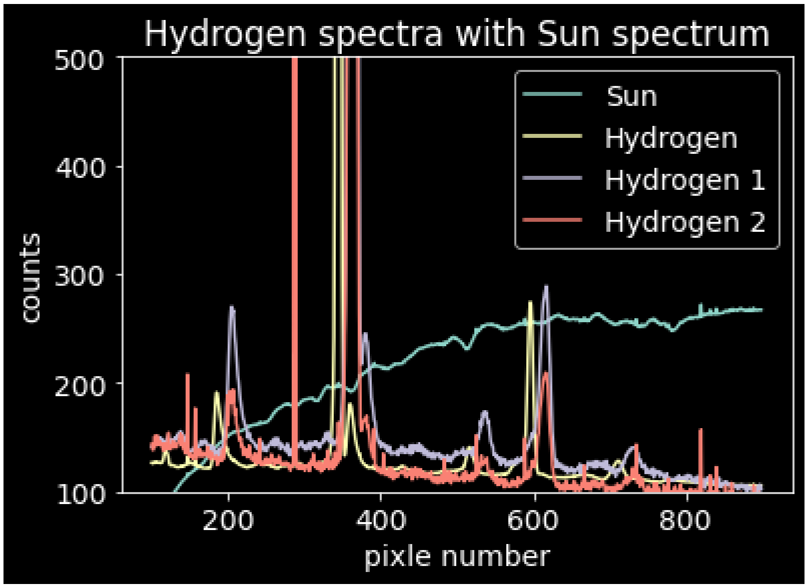
In [73]:
```
pl.plot(wavelengths, hy_spectrum)
pl.vlines(hydrogen_lines['Observed'], 1000, 29000, 'w', alpha=0.25);
pl.xlabel("pix")
pl.ylabel('counts')
pl.title('Hydrogen Spectra Overlay')
```

Out[73]:
```
Text(0.5, 1.0, 'Hydrogen Spectra Overlay')
```



In [74]:
```
pl.plot(xaxis[100:], xaxis[100:]*sun_spectrum[100:]/4000, label = 'Sun')
pl.plot(xaxis[100:], hy_spectrum[100:]/10, label = 'Hydrogen');
pl.plot(xaxis[100:], hy1_spectrum[100:], label = 'Hydrogen 1');
pl.plot(xaxis[100:], hy2_spectrum[100:]-950, label = 'Hydrogen 2');
pl.ylim(100,500)
#pl.xlim(10,500)
pl.legend(loc='best');
pl.xlabel("pixle number")
pl.ylabel('counts')
pl.title("Hydrogen spectra with Sun spectrum")
```

Out[74]:
```
Text(0.5, 1.0, 'Hydrogen spectra with Sun spectrum')
```

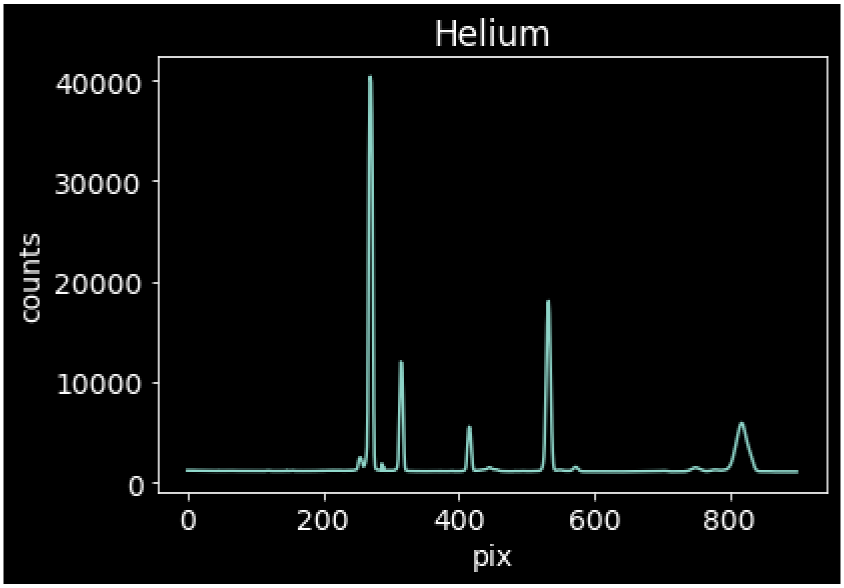


Helium

In [ ]:
```
tableHe = Nist.query(4000 * u.AA, 7000 * u.AA, linename="He")
print(tableHe)
```

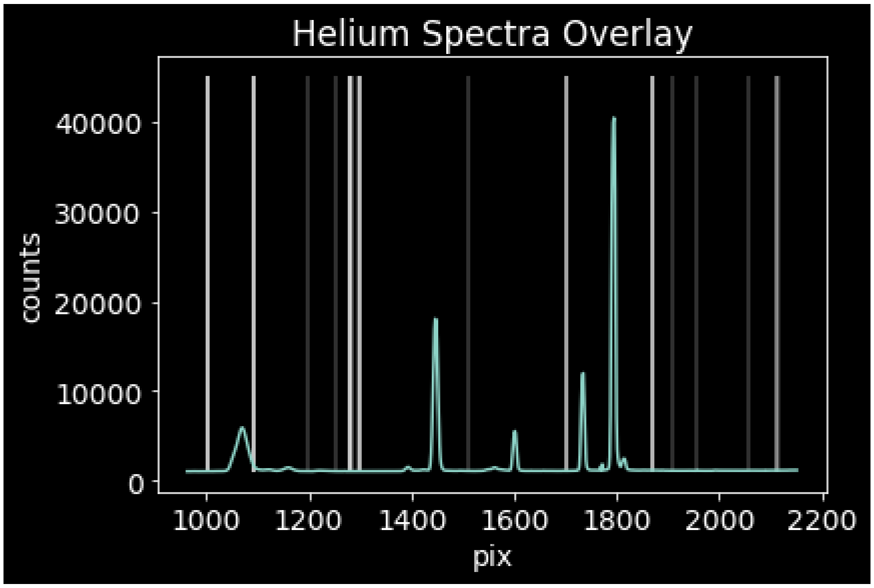
In [50]:
```
pl.plot(xaxis, he_spectrum);
pl.xlabel("pix")
pl.ylabel('counts')
pl.title("Helium")
```

Out[50]: Text(0.5, 1.0, 'Helium')



In [55]:
```
pl.plot(wavelengths, he_spectrum)
pl.vlines(helium_lines['Observed'], 1000, 45000, 'w', alpha=0.25);
pl.xlabel("pix")
pl.ylabel('counts')
pl.title('Helium Spectra Overlay')
```

Out[55]: Text(0.5, 1.0, 'Helium Spectra Overlay')



Neon

In [56]:
```
import pandas as pd
tableNe = Nist.query(4000 * u.AA, 7000 * u.AA, linename="Ne")
tableNe
```

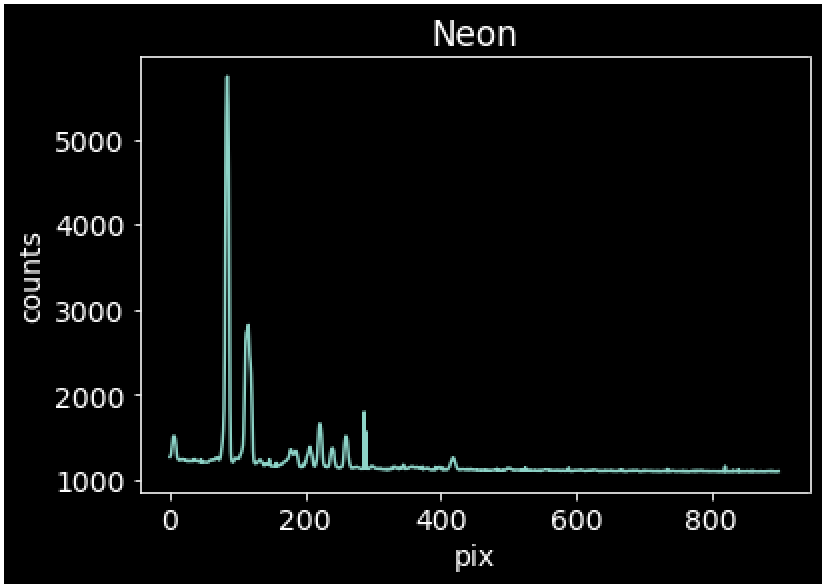
Out[56]: *Table length=992*

Spectrum	Observed	Ritz	Transition	Rel.	Aki	fik	Acc.	Ei Ek
str7	float64	float64	float64	str10	float64	float64	str3	str30
Ne I	4000.394	4000.408	24997.54	10	--	--	--	18.38162323 - 21.480912
Ne II	4000.602	4000.60273	24996.24	100	370000.0	0.0018	D	27.85915908 - 30.95829705
Ne VIII	4003.0	4004.0	24981.0	100bl	65100000.0	0.47	A	[225.1915] - [228.2878]
Ne II	4011.81	4011.827	24926.38	10	--	--	--	36.4926649 - 39.583132
Ne I	4014.887	4014.887	24907.3	10	--	--	--	18.38162323 - 21.469735
Ne I	4015.13	4015.11	24905.8	20	--	--	--	18.38162323 - 21.469565
Ne I	4021.151	4021.179	24868.5	20	--	--	--	18.38162323 - 21.464903
Ne II	4023.56	4023.5644	24853.63	5	--	--	--	36.46207411 - 39.5435260
Ne II	4025.18	4025.1792	24843.61	80	--	--	--	35.19841919 - 38.2786348
Ne I	4038.403	4038.402	24762.27	50	--	--	--	18.38162323 - 21.451753
...	...	...	...	...	...	...	...	...
Ne I	6718.8974	6718.8974	14883.3945	700	21700000.0	0.147	B+	16.84805369 - 18.69335943
Ne I	6722.9897	6722.9902	14874.335	20	51500.0	0.000349	B+	18.72638145 - 20.57056379

Spectrum	Observed	Ritz	Transition	Rel.	Aki	fik	Acc.	Ei Ek
Ne I	6739.892	6739.8924	14837.033	700	1250000.0	0.0255	C+	18.96595369 - 20.80551122
Ne I	6761.4479	6761.4513	14789.732	150	160000.0	0.00183	B+	18.72638145 - 20.56007355
Ne III	6888.84	6888.85	14516.24	70	60000000.0	0.43	D	52.051957 - 53.851739
Ne VIII	--	6896.0	14501.0	--	16700000.0	0.237	B	[182.2092] - [184.007]
Ne III	6897.31	6897.31	14498.4	100	60000000.0	0.71	D	52.051957 - 53.8495312
Ne VII	--	6920.0	14460.0	--	4300000.0	0.031	B	179.375 - 181.168
Ne I	6931.3787	6931.3788	14427.1441	100000	17400000.0	0.209	B+	16.84805369 - 18.63679156
Ne VIII	--	6995.0	14296.0	--	15900000.0	0.117	B	[182.2092] - [183.982]

In [57]:
```
pl.plot(xaxis, ne_spectrum);
pl.xlabel("pix")
pl.ylabel('counts')
pl.title("Neon")
```

Out[57]:
```
Text(0.5, 1.0, 'Neon')
```

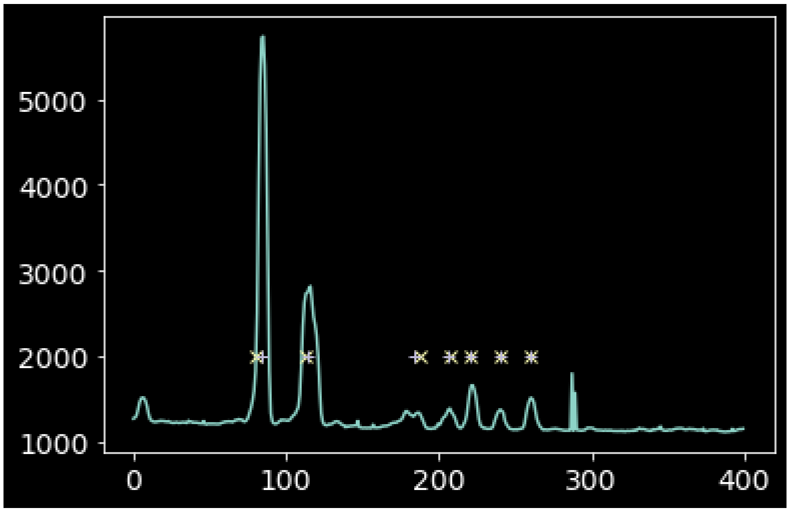


In [58]:
```
guessed_wavelengths_ne = [540, 535, 520, 518, 510, 507, 504]
guessed_xvals_ne = [80, 113, 187, 207, 220, 240, 260]

npixels = 10
improved_xval_guesses_ne = [np.average(xaxis[g-npixels:g+npixels],
                                       weights=ne_spectrum[g-npixels:g+npixels] - np
                            for g in guessed_xvals_ne]
improved_xval_guesses_ne
```

Out[58]:
```
[84.04518316319938,
 114.78478093524011,
 183.78829039174053,
 206.40975505276685,
 220.63915380859257,
 239.6429633695977,
 260.1708826893014]
```

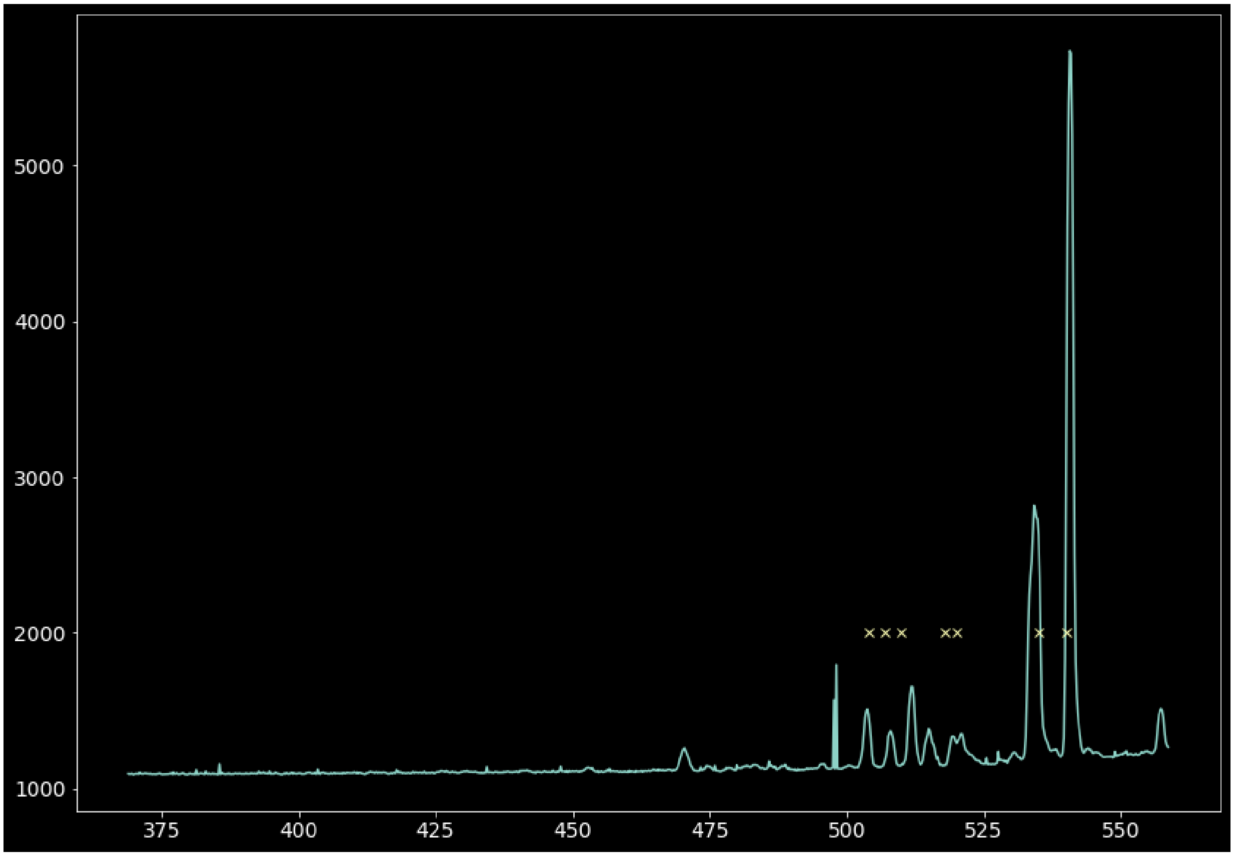
In [59]:
```
pl.plot(xaxis[0:400], ne_spectrum[0:400])
pl.plot(guessed_xvals_ne[0:400], [2000]*7, 'x')
pl.plot(improved_xval_guesses_ne[0:400], [2000]*7, '+');
```



In [60]:
```
linfitter = LinearLSQFitter()
wlmodel = Linear1D()
linfit_wlmodel = linfitter(model=wlmodel, x=improved_xval_guesses_ne, y=guessed_w
wavelengths = linfit_wlmodel(xaxis) * u.nm
linfit_wlmodel
```

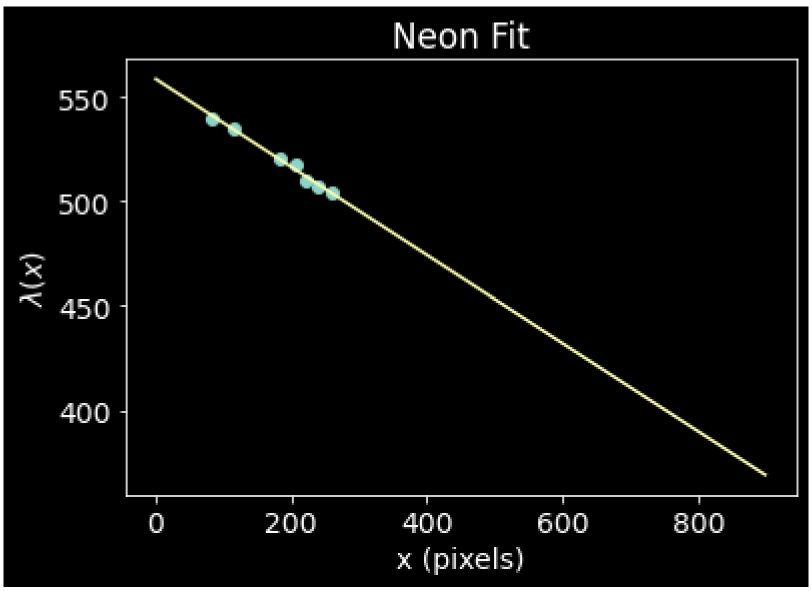
Out[60]:
```
<Linear1D(slope=-0.21101769, intercept=558.61766524)>
```

In [61]:
```
pl.figure(figsize = (14,10))
pl.plot(wavelengths, ne_spectrum)
pl.plot(guessed_wavelengths_ne, [2000]*7, 'x');
```



In [64]:
```
pl.plot(improved_xval_guesses_ne, guessed_wavelengths_ne, 'o')
pl.plot(xaxis, wavelengths, '-')
pl.ylabel("$\lambda(x)$")
pl.xlabel("x (pixels)")
pl.title("Neon Fit")
```

Out[64]: `Text(0.5, 1.0, 'Neon Fit')`

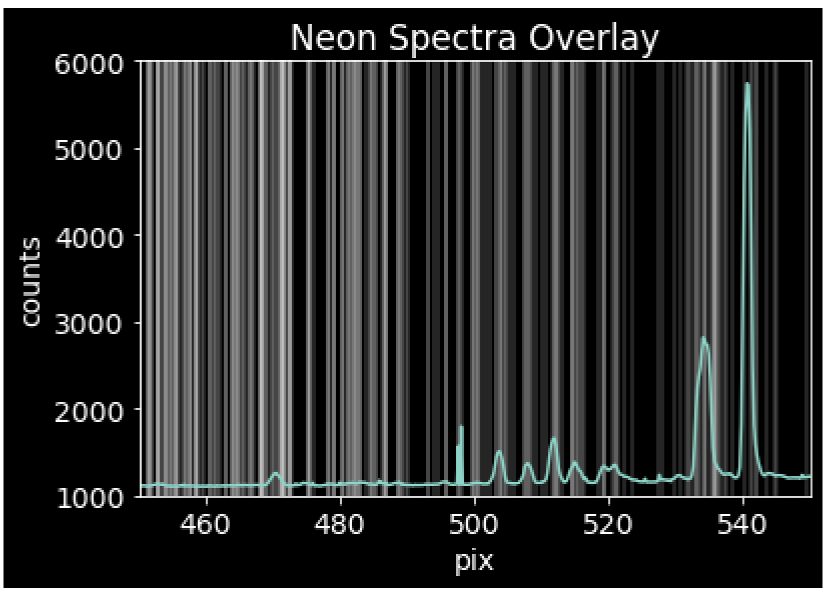


In [65]:
```
minwave = wavelengths.min()
maxwave = wavelengths.max()

hydrogen_lines = Nist.query(minwav=minwave,
                            maxwav=maxwave,
                            linename='H I')
helium_lines = Nist.query(minwav=minwave,
                          maxwav=maxwave,
                          linename='He I')
neon_lines = Nist.query(minwav=minwave,
                        maxwav=maxwave,
                        linename='Ne I')
```

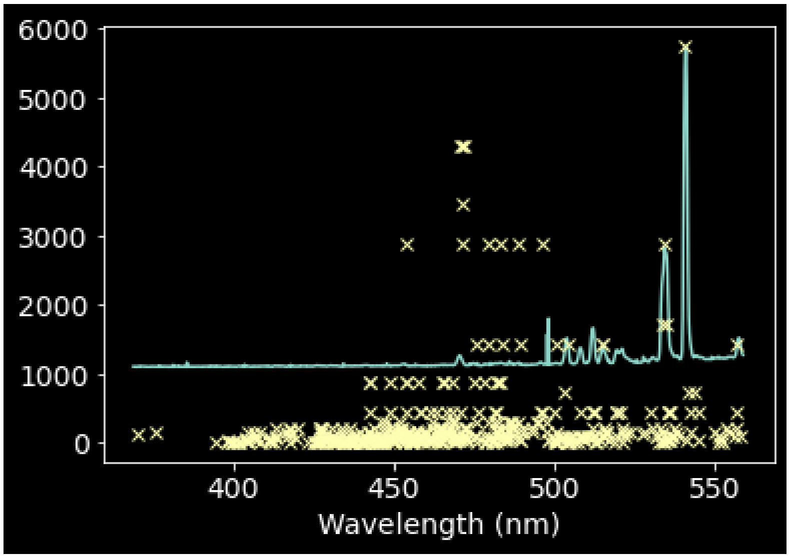
In [66]:
```
pl.plot(wavelengths, ne_spectrum)
pl.vlines(neon_lines['Observed'], 6000, 250, 'w', alpha=0.20);
pl.axis([450, 550, 1000, 6000])
pl.xlabel("pix")
pl.ylabel('counts')
pl.title('Neon Spectra Overlay')
```

Out[66]: Text(0.5, 1.0, 'Neon Spectra Overlay')

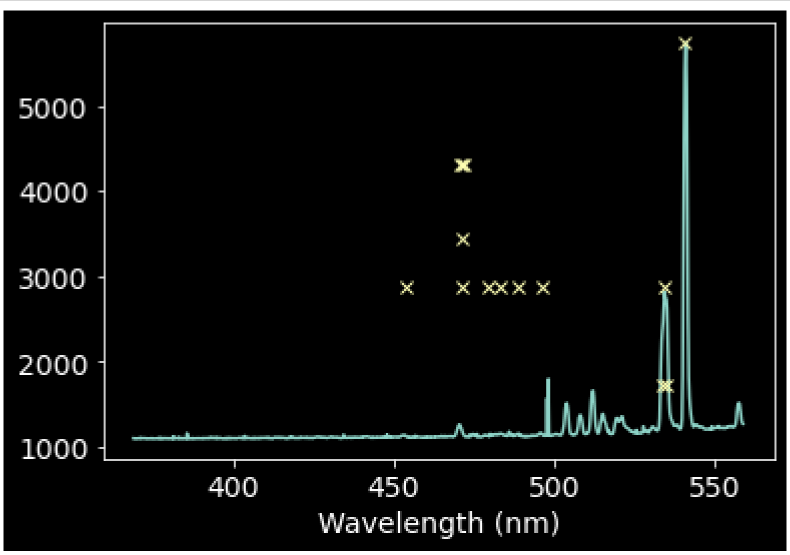


In [67]:
```
ne_keep = np.array([('*' not in x) and ('f' not in x) for x in neon_lines['Rel.']
ne_wl_tbl = neon_lines['Observed'][ne_keep]
ne_rel_tbl = np.array([float(x) for x in neon_lines['Rel.'][ne_keep]])
```

In [68]:
```
ne_rel_intens = ne_rel_tbl / ne_rel_tbl.max() * ne_spectrum.max()
pl.plot(wavelengths, ne_spectrum)
pl.plot(ne_wl_tbl, ne_rel_intens, 'x')
pl.xlabel('Wavelength (nm)');
```



In [69]:
```
ne_keep_final = ne_rel_intens > 1500
pl.plot(wavelengths, ne_spectrum)
pl.plot(ne_wl_tbl[ne_keep_final], ne_rel_intens[ne_keep_final], 'x')
pl.xlabel('Wavelength (nm)');
```



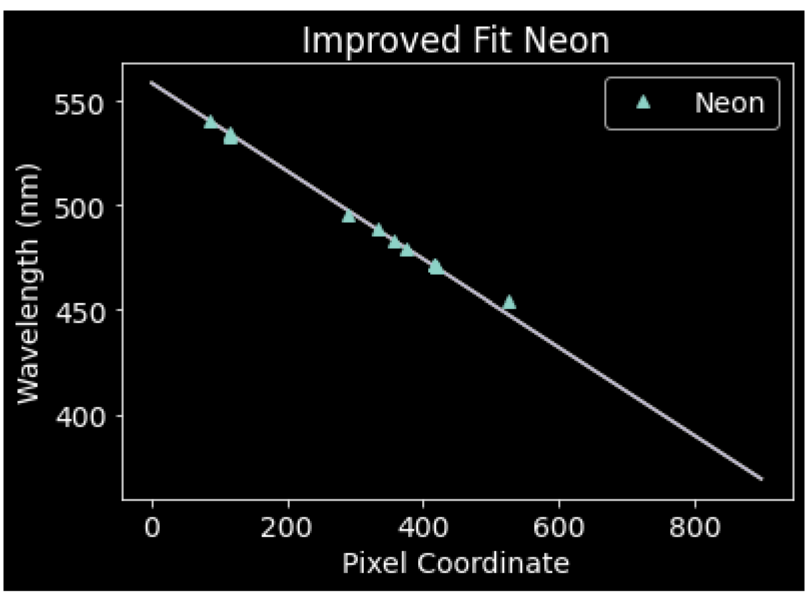
In [70]:
```
ne_wl_final = ne_wl_tbl[ne_keep_final]
ne_pixel_vals = linfit_wlmodel.inverse(ne_wl_final)
```

In [71]:
```
npixels = 10
improved_xval_guesses_ne = [np.average(xaxis[g-npixels:g+npixels],
                                       weights=ne_spectrum[g-npixels:g+npixels] - np
                           for g in map(int, ne_pixel_vals)]
improved_xval_guesses_ne
```

Out[71]:
```
[525.9785306943977,
 418.1729633027523,
 418.12914795699726,
 417.7485068438266,
 417.5125463331328,
 417.0405190404364,
 374.0953070683661,
 358.58622830067657,
 332.5209137373527,
 290.4229791661179,
 115.9923643827471,
 115.12967868360643,
 114.97258363509087,
 84.6300646285281]
```

In [72]:
```
pl.plot(improved_xval_guesses_ne, ne_wl_final, '^', label='Neon')
#pl.plot(improved_xval_guesses, guessed_wavelengths, '+', label='Hydrogen')
pl.plot(xaxis, wavelengths, zorder=-5)
pl.plot(xaxis, linfit_wlmodel(xaxis), zorder=-5)
pl.legend(loc='best')
pl.xlabel("Pixel Coordinate")
pl.ylabel("Wavelength (nm)")
pl.title("Improved Fit Neon")
```

Out[72]: Text(0.5, 1.0, 'Improved Fit Neon')



In [ ]: