In [1]:
```
%matplotlib inline
import numpy as np
from astropy.io import fits
from astropy.table import Table
import pylab as pl
import glob
from astropy.visualization import simple_norm
pl.style.use('dark_background')
pl.rcParams['figure.dpi'] = 150
import datetime
from astropy import coordinates
from astroplan import Observer
import pytz
from astropy import units as u
gainesville_location = coordinates.EarthLocation.from_geodetic(lon=-82.3*u.deg, lat=29.6*u.deg, height=100*u.m)
```

In [2]:
```
from astropy.time import Time, TimeDelta
from astropy.wcs import WCS
from astropy.visualization.wcsaxes import SphericalCircle
from astropy.utils.data import get_pkg_data_filename
from astropy.visualization.wcsaxes.frame import EllipticalFrame
from astropy.visualization import simple_norm
from astropy.table import Table
import pytz
import datetime
```

Loading in all files

In [3]:
```
cd RadioLabData2022/
```

```
C:\Users\Sydnee O'Donnell\OneDrive\UF\Obs Tech 2\Labs\RadioTestLab\RadioLabData2022
```

In [4]:
```
ls
```

```
 Volume in drive C has no label.
 Volume Serial Number is EC8D-ED0B

 Directory of C:\Users\Sydnee O'Donnell\OneDrive\UF\Obs Tech 2\Labs\RadioTestLab\RadioLabData2022

11/17/2022  12:08 PM    <DIR>          .
11/30/2022  08:25 PM    <DIR>          ..
11/17/2022  12:08 PM               156 ipython_log_2022-11-15.py
11/17/2022  12:08 PM               261 ipython_log_2022-11-15.py append
11/17/2022  12:05 PM    <DIR>          JohnDixon
11/17/2022  12:08 PM    <DIR>          KristinaBerkova
11/17/2022  12:06 PM    <DIR>          MorganHimes
11/17/2022  12:07 PM    <DIR>          MorganHimes-2
11/17/2022  12:06 PM    <DIR>          PaeSwanson
11/17/2022  11:56 AM     1,570,758,459 RadioLabData2022.zip
11/17/2022  12:08 PM    <DIR>          SaganSutherland
11/17/2022  12:08 PM    <DIR>          SydneeODonnell
11/17/2022  12:07 PM    <DIR>          WillMacKenzie
11/17/2022  12:05 PM    <DIR>          zip_files
               3 File(s)  1,570,758,876 bytes
              11 Dir(s)  144,986,345,472 bytes free
```

In [5]:
```
tp1_data = {}
tp2_data = {}
fsw_data = {}
freq1_data = {}
all_coords = coords = {}
for student in glob.glob("*/"):
    if 'zip_files' not in student:
        print(student)
        tp1_data[student.strip("/").strip("\\")] = np.array([
            Table.read(fn)['power1'] for fn in glob.glob(f'{student}/psd*fits')]
            )
        tp2_data[student.strip("/").strip("\\")] = np.array([
            Table.read(fn)['power2'] for fn in glob.glob(f'{student}/psd*fits')]
            )
        fsw_data[student.strip("/").strip("\\")] = np.array([
            Table.read(fn)['fsw_pow'] for fn in glob.glob(f'{student}/psd*fits')]
            )
        freq1_data[student.strip("/").strip("\\")] = np.array([
            Table.read(fn)['freq1'] for fn in glob.glob(f'{student}/psd*fits')]
            )
        tables = [Table.read(fn) for fn in glob.glob(f'{student}/psd*fits')]
        coords[student.strip("/").strip("\\")] = [
            coordinates.AltAz(alt=float(tb.meta['--altitude'])*u.deg,
                              az=-184*u.deg if 'Kristina' in student else float(tb.meta['--azimuth'])*u.deg,
                              location=coordinates.EarthLocation.from_geodetic(
                                  lon=float(tb.meta['--obs_lon'])*u.deg,
                                  lat=float(tb.meta['--obs_lat'])*u.deg,
                                  height=100*u.m),
                              obstime=pytz.timezone('US/Eastern').localize(
                                  datetime.datetime.strptime(tb.meta['DATE-OBS'], "%y%m%d_%H%M%S"))
                              )
            if np.abs(float(tb.meta['--altitude'])) <= 90
            else "Calibrator"
            for tb in tables]
        freq1 = tables[0]['freq1']
        freq2 = tables[0]['freq2']
        fsw_rvel1 = tables[0]['fsw_rvel1']
        fsw_rvel2 = tables[0]['fsw_rvel2']
```

```
JohnDixon\
KristinaBerkova\
MorganHimes\
MorganHimes-2\
PaeSwanson\
SaganSutherland\
SydneeODonnell\
WillMacKenzie\
```

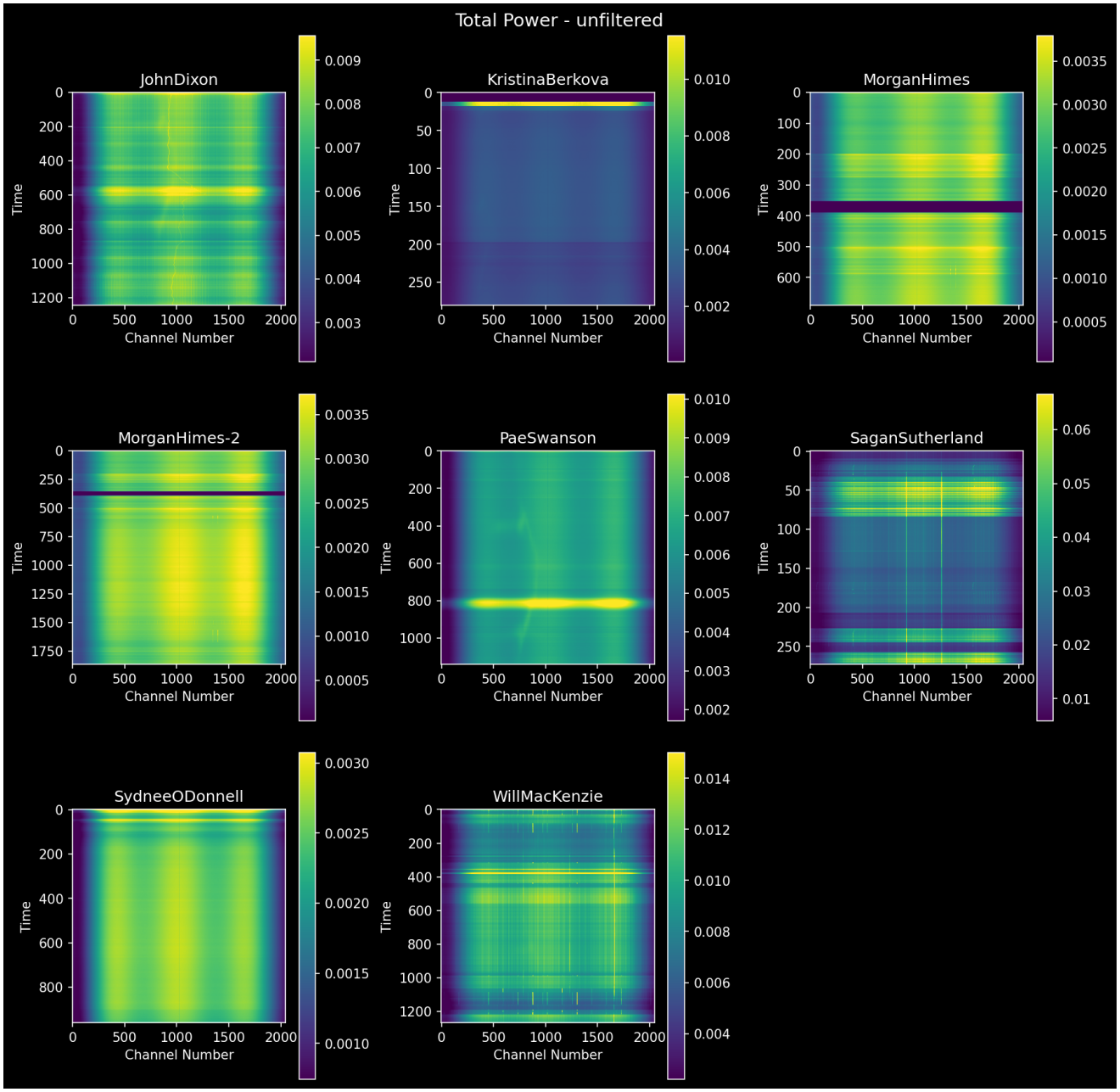
In [6]:
```
assert 'zip_files/' not in coords
```

In [7]:
```
for student in tp1_data:
    print(student, tp1_data[student].shape)
```

```
JohnDixon (1246, 2048)
KristinaBerkova (281, 2048)
MorganHimes (691, 2048)
MorganHimes-2 (1864, 2048)
PaeSwanson (1139, 2048)
SaganSutherland (273, 2048)
SydneeODonnell (960, 2048)
WillMacKenzie (1268, 2048)
```

Plotting Total Power

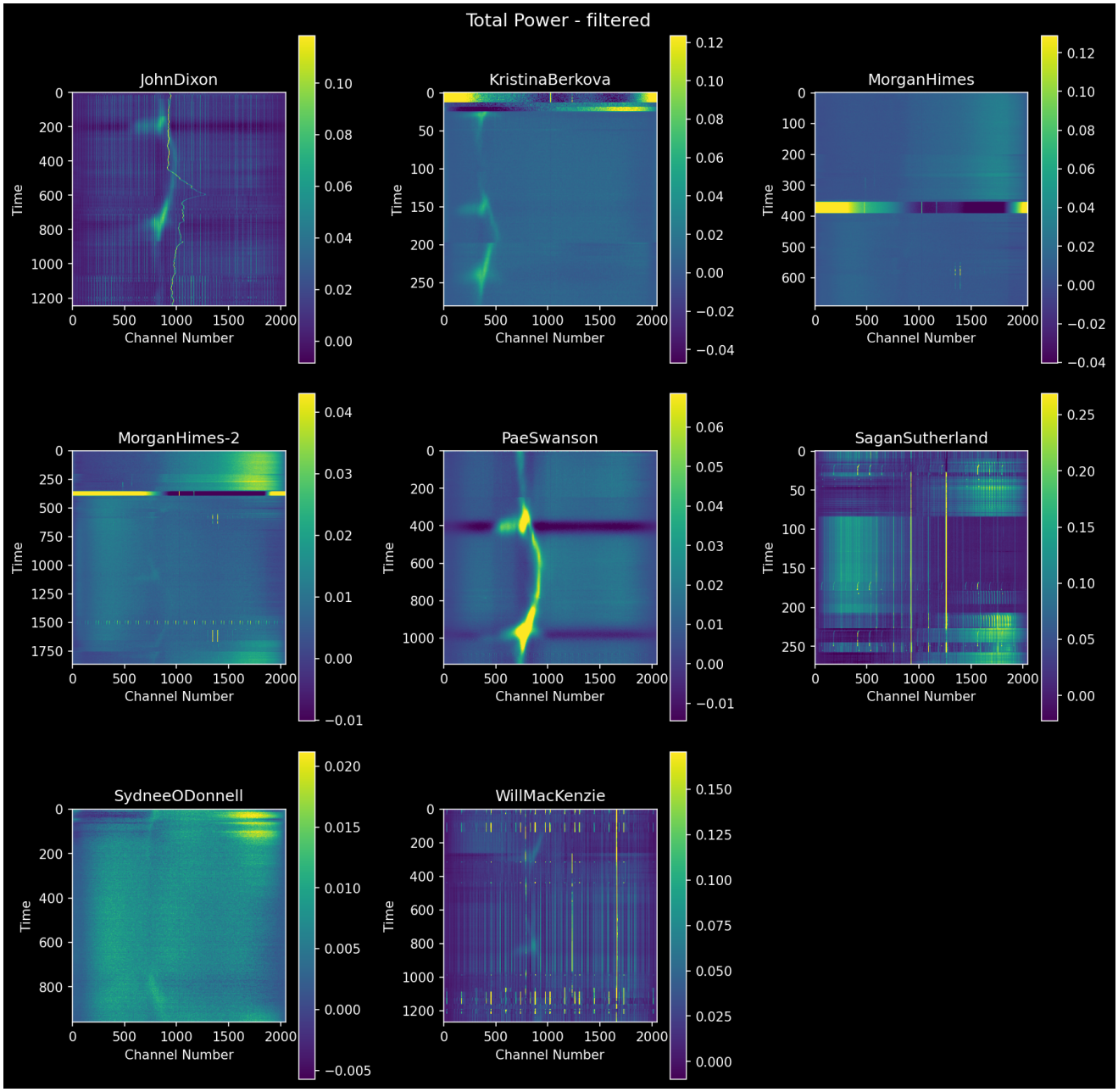
In [8]:
```
pl.figure(figsize=(12, 12), dpi=150)
for ii, (student, data) in enumerate(tp1_data.items()):
    ax = pl.subplot(3, 3, ii+1)
    im = ax.imshow(data, norm=simple_norm(data, max_percent=99, min_percent=1))
    ax.set_title(student)
    ax.set_aspect(data.shape[1]/data.shape[0])
    ax.set_xlabel("Channel Number")
    ax.set_ylabel("Time")
    pl.colorbar(mappable=im)

pl.suptitle("Total Power - unfiltered", fontsize=14);
pl.tight_layout();
```



Normalizing

In [9]:
```
normed = {student: data/np.median(data, axis=1)[:,None] for student, data in tp1_data.items()}
normsubbed = {student: data - np.percentile(data, 10, axis=0) for student, data in normed.items()}
```

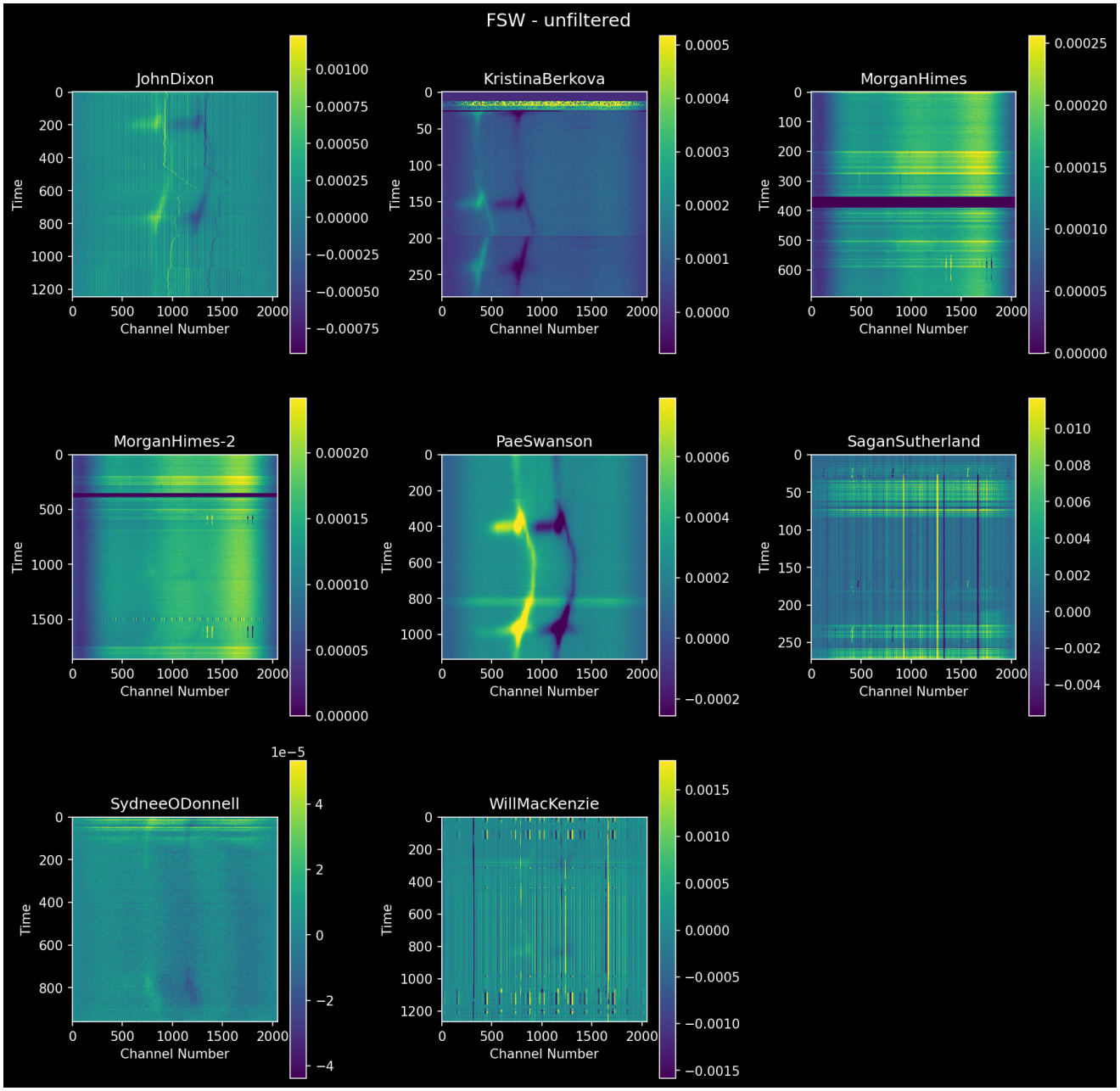
In [10]:
```
pl.figure(figsize=(12, 12), dpi=150)
for ii, (student, data) in enumerate(normsubbed.items()):
    ax = pl.subplot(3, 3, ii+1)
    im = ax.imshow(data, norm=simple_norm(data, max_percent=99, min_percent=1))
    ax.set_title(student)
    ax.set_aspect(data.shape[1]/data.shape[0])
    ax.set_xlabel("Channel Number")
    ax.set_ylabel("Time")
    pl.colorbar(mappable=im)

pl.suptitle("Total Power - filtered", fontsize=14);
pl.tight_layout();
```



Plotting Normalized fsw

In [11]:
```
fsw_normed = {student: data/np.median(data, axis=1)[:,None] for student, data in fsw_data.items()}
fsw_normsubbed = {student: data - np.median(data, axis=0) for student, data in fsw_normed.items()}
```

In [12]:
```
pl.figure(figsize=(12, 12), dpi=150)
for ii, (student, data) in enumerate(fsw_data.items()):
    ax = pl.subplot(3, 3, ii+1)
    im = ax.imshow(data, norm=simple_norm(data, max_percent=99, min_percent=1))
    ax.set_title(student)
    ax.set_aspect(data.shape[1]/data.shape[0])
    ax.set_xlabel("Channel Number")
    ax.set_ylabel("Time")
    pl.colorbar(mappable=im)
pl.suptitle("FSW - unfiltered", fontsize=14);
pl.tight_layout();
```



Mapping Area of Sky Covered by Class

In [13]:
```
for student in coords:
    print(student)
    for c in coords[student]:
        pass
```

```
JohnDixon
KristinaBerkova
MorganHimes
MorganHimes-2
PaeSwanson
SaganSutherland
SydneeODonnell
WillMacKenzie
```

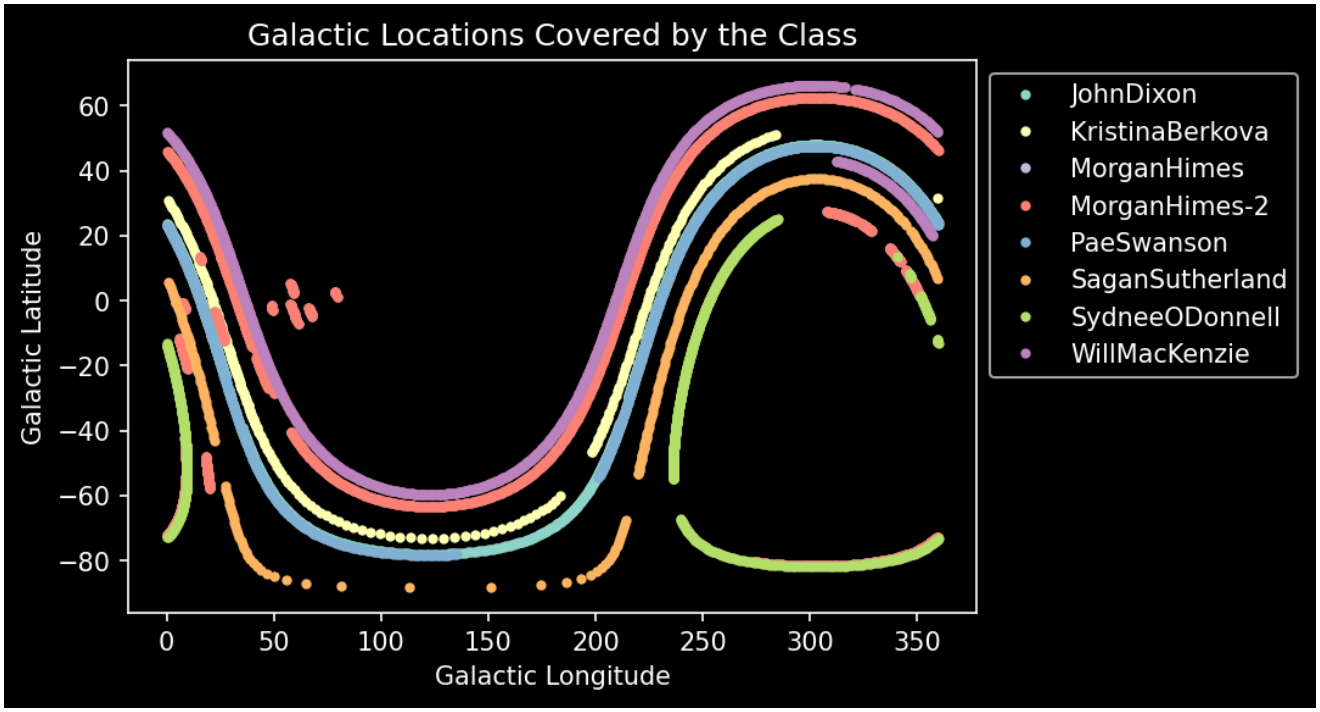
In [14]:
```
assert 'zip_files/' not in coords
galcoords = {student:
             coordinates.SkyCoord([cc.transform_to(coordinates.Galactic()) for cc in coordlist
                                   if isinstance(cc, coordinates.AltAz)])
             for student, coordlist in coords.items()
            }
print(coords.keys())
```

```
dict_keys(['JohnDixon', 'KristinaBerkova', 'MorganHimes', 'MorganHimes-2', 'PaeSwanson', 'SaganSutherland', 'SydneeODonnell',
'WillMacKenzie'])
```

In [15]:
```
for student, gcs in galcoords.items():
    print(student)
    pl.plot(gcs.l, gcs.b, '.', label=student)
pl.legend(bbox_to_anchor=(1,1,0,0))
pl.xlabel("Galactic Longitude")
pl.ylabel("Galactic Latitude")
pl.title("Galactic Locations Covered by the Class")
```

```
JohnDixon
KristinaBerkova
MorganHimes
MorganHimes-2
PaeSwanson
SaganSutherland
SydneeODonnell
WillMacKenzie
```

Out[15]: Text(0.5, 1.0, 'Galactic Locations Covered by the Class')

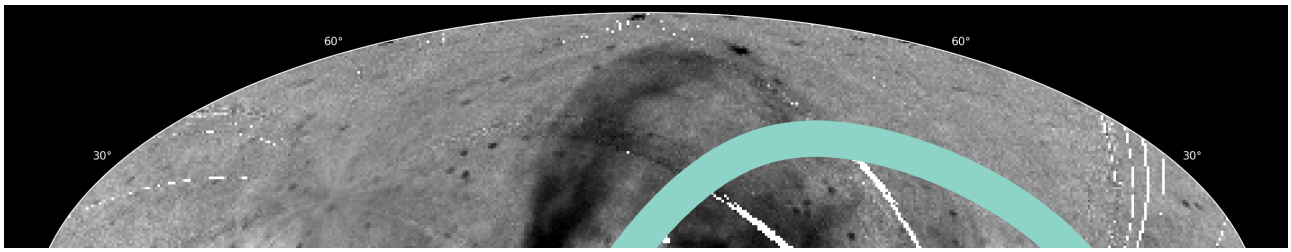


In [16]:
```
filename = get_pkg_data_filename('allsky/allsky_rosat.fits')
hdu = fits.open(filename)[0]
wcs = WCS(hdu.header)
data = hdu.data

# let's also grab the HI data
from reproject import reproject_from_healpix, reproject_to_healpix
hihdu = fits.open('https://lambda.gsfc.nasa.gov/data/foregrounds/ebv_2017/mom0_-90_90_1024.hpx.fits')
hiarray, hifootprint = reproject_from_healpix(hihdu[1], hdu.header)
```

```
C:\ProgramData\Anaconda3\lib\site-packages\astropy_healpix\core.py:586: RuntimeWarning: invalid value encountered in bilinear_i
nterpolation_weights
  result = _core.bilinear_interpolation_weights(lon, lat, nside)
```

In [17]:
```
for student, gcs in galcoords.items():
    print(student)
    pl.figure(figsize=(20,15))
    ax = pl.subplot(projection=wcs, frame_class=EllipticalFrame)
    im = ax.imshow(data, origin='lower', cmap='gray_r', interpolation='none',
               norm=simple_norm(data, stretch='asinh', max_percent=99.5, min_percent=1))
    ax.scatter(gcs.l, gcs.b, s=1000, transform=ax.get_transform('galactic'), label=student)
```

```
JohnDixon
KristinaBerkova
MorganHimes
MorganHimes-2
PaeSwanson
SaganSutherland
SydneeODonnell
WillMacKenzie
```

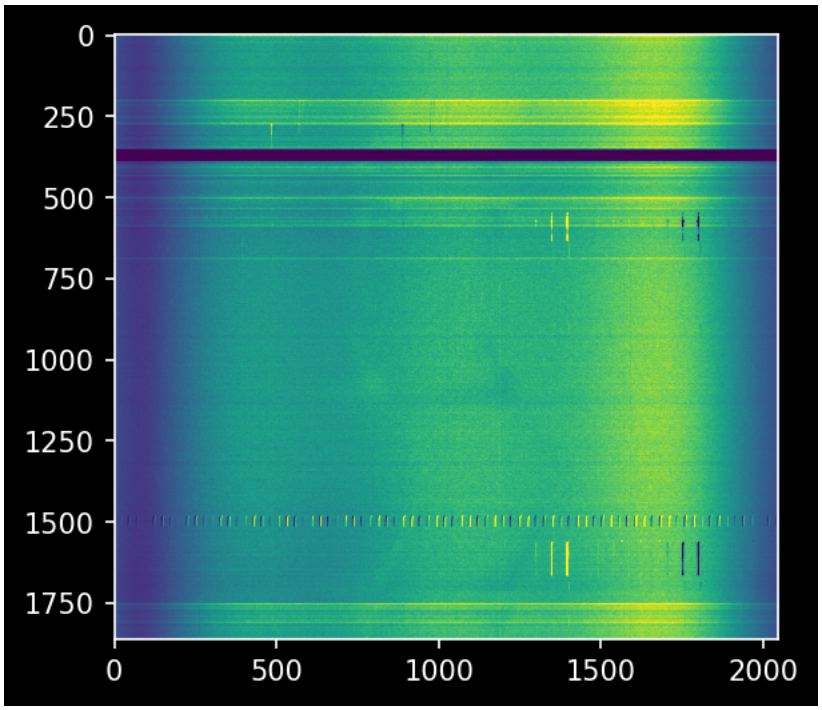


Working with Morgan's Data

In [18]:
```
morganfsw = fsw_data['MorganHimes-2']
morganfsw_normed = morganfsw/np.median(morganfsw, axis=1)[:,None]
morganfsw_normsubbed = morganfsw_normed - np.median(morganfsw_normed, axis=0)
```

In [19]:
```
pl.imshow(morganfsw, norm=simple_norm(morganfsw, min_percent=1, max_percent=99))
```

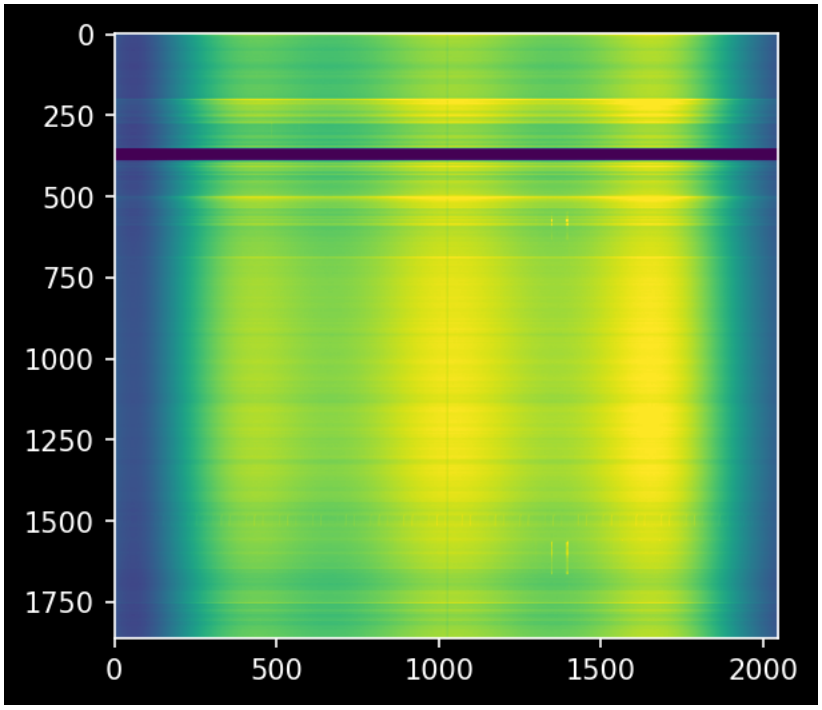
Out[19]: <matplotlib.image.AxesImage at 0x210298713d0>



In [20]:
```
morgantp1 = tp1_data['MorganHimes-2']
morgantp1_normed = morgantp1/np.median(morgantp1, axis=1)[:,None]
morgantp1_normsubbed = morgantp1_normed - np.median(morgantp1_normed, axis=0)
```

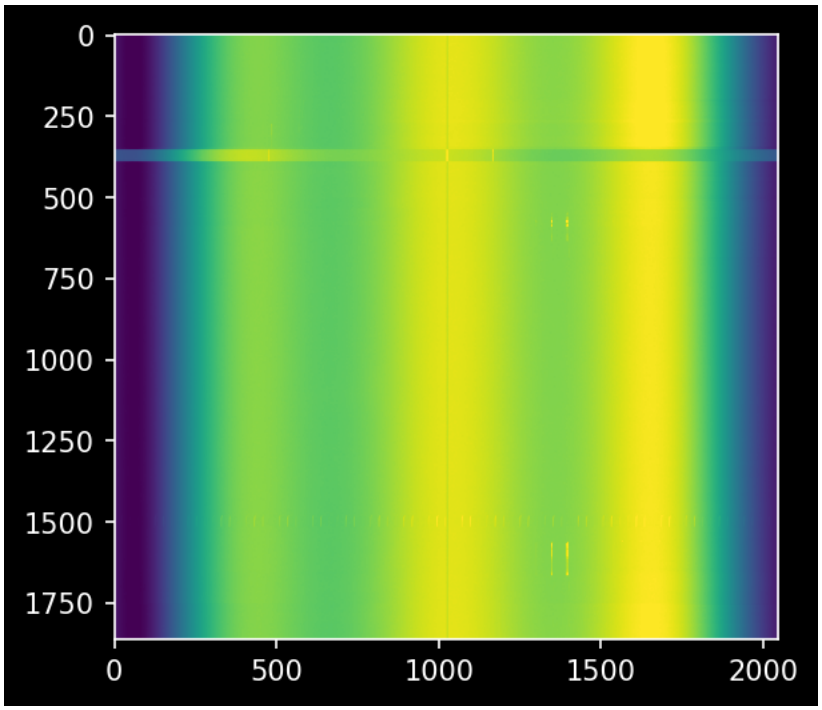
In [21]:
```
pl.imshow(morgantp1, norm=simple_norm(morgantp1, min_percent=1, max_percent=99))
```

Out[21]: <matplotlib.image.AxesImage at 0x210298b7dc0>



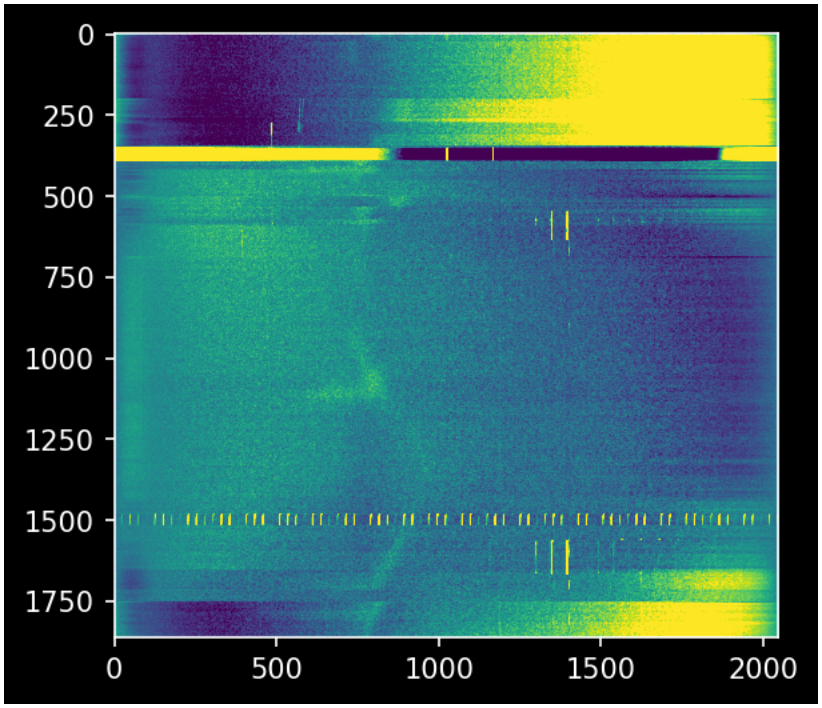
In [22]:
```
pl.imshow(morgantp1_normed, norm=simple_norm(morgantp1_normed, min_percent=1, max_percent=99))
```

Out[22]: <matplotlib.image.AxesImage at 0x2102990a520>



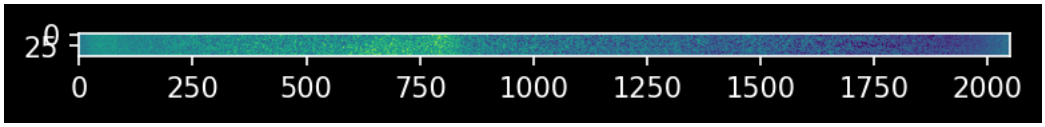
In [23]:
```
pl.imshow(morgantp1_normsubbed, norm=simple_norm(morgantp1_normsubbed, min_percent=10, max_percent=90))
```

Out[23]: <matplotlib.image.AxesImage at 0x21029af4bb0>



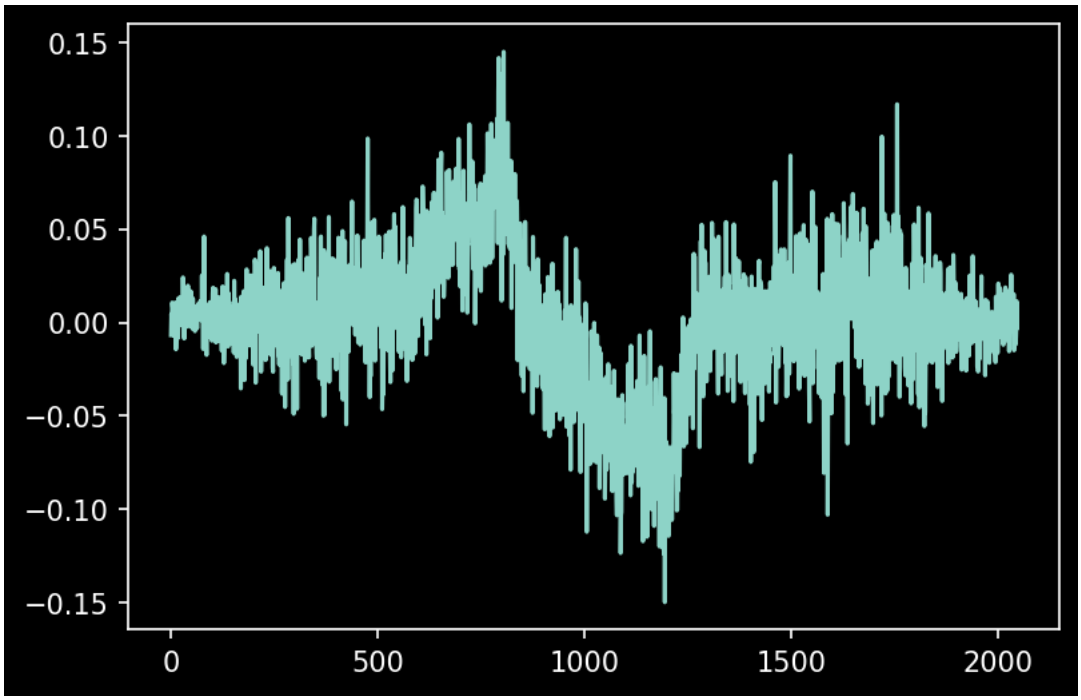
In [24]:
```
pl.imshow(morgantp1_normsubbed[1075:1125,:], norm=simple_norm(morgantp1_normsubbed, min_percent=10, max_percent=90))
```

Out[24]: <matplotlib.image.AxesImage at 0x2102a2064f0>



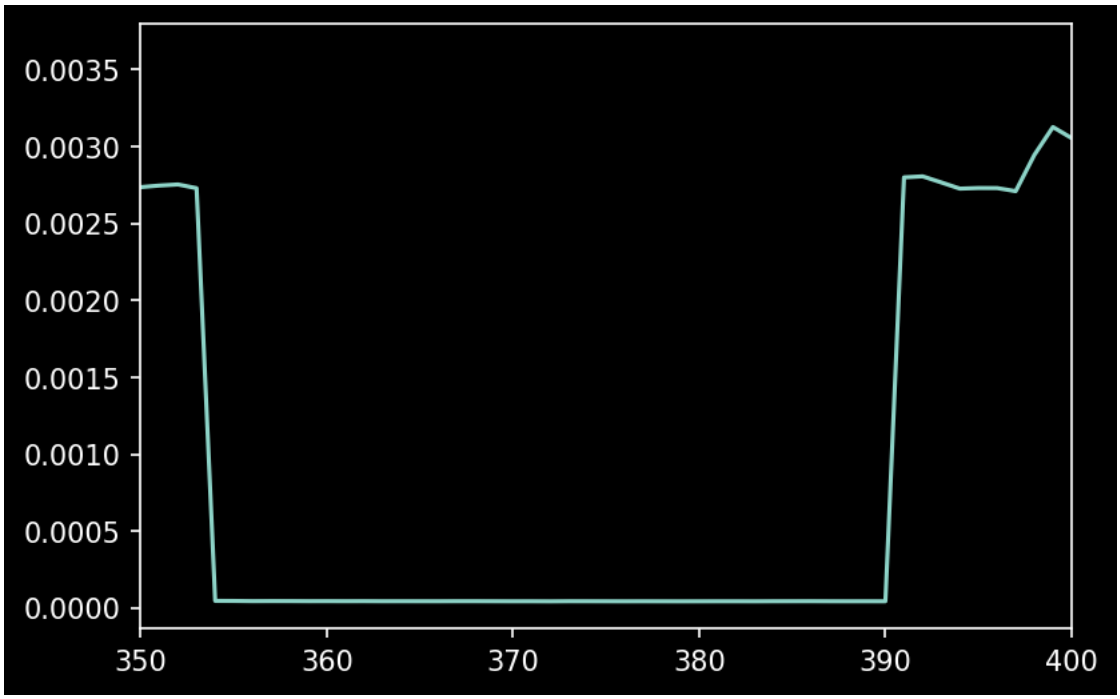
In [25]:
```
pl.plot(morganfsw_normsubbed[1075:1125,:].mean(axis=0))
```

Out[25]:
```
[<matplotlib.lines.Line2D at 0x2102a24d460>]
```



In [26]:
```
pl.plot(morgantp1[:, 600])
pl.xlim(350,400)
```

Out[26]:
```
(350.0, 400.0)
```

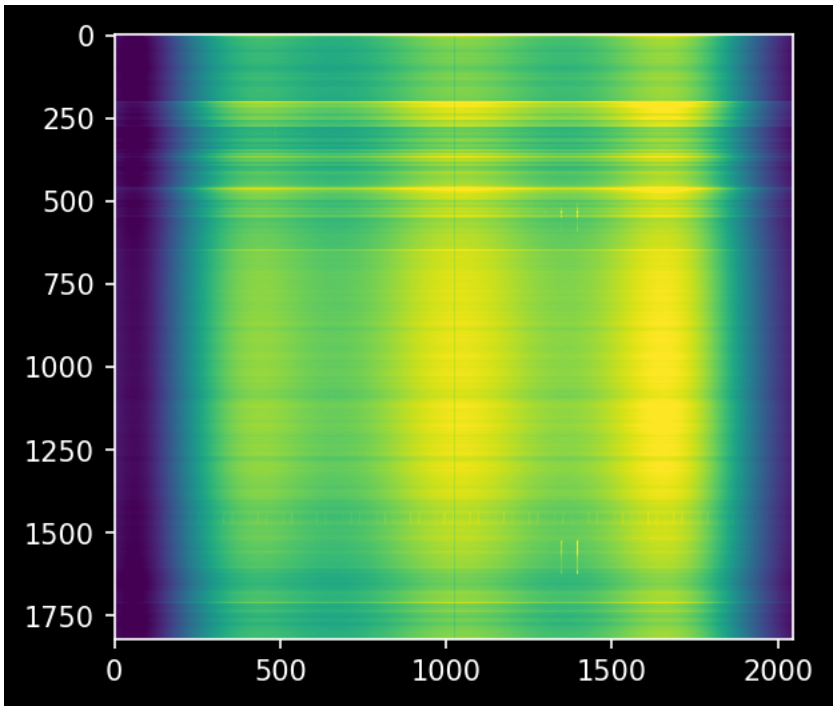


In [27]:
```
morgantp1b = np.vstack([morgantp1[:350], morgantp1[391:]])
morgantp1b.shape
```

Out[27]:
```
(1823, 2048)
```

In [28]:
```
pl.imshow(morgantp1b, norm=simple_norm(morgantp1b, min_percent=1, max_percent=99))
```

Out[28]:
```
<matplotlib.image.AxesImage at 0x2102a5acbe0>
```

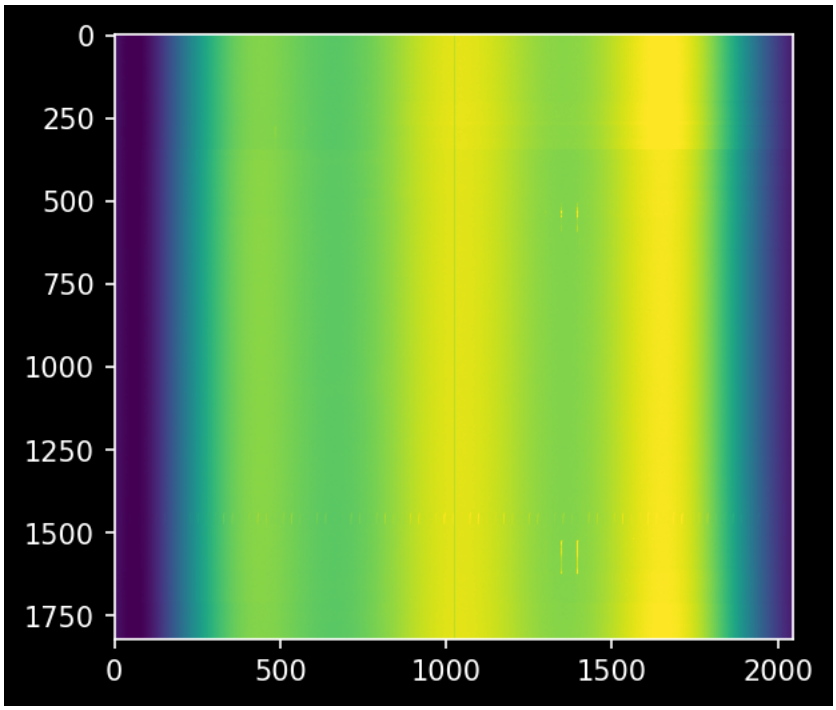


In [29]:
```
morgantp1b_normed = morgantp1b/np.median(morgantp1b, axis=1)[:,None]
morgantp1b_normsubbed = morgantp1b_normed - np.median(morgantp1b_normed, axis=0)
```

In [ ]:
```

```

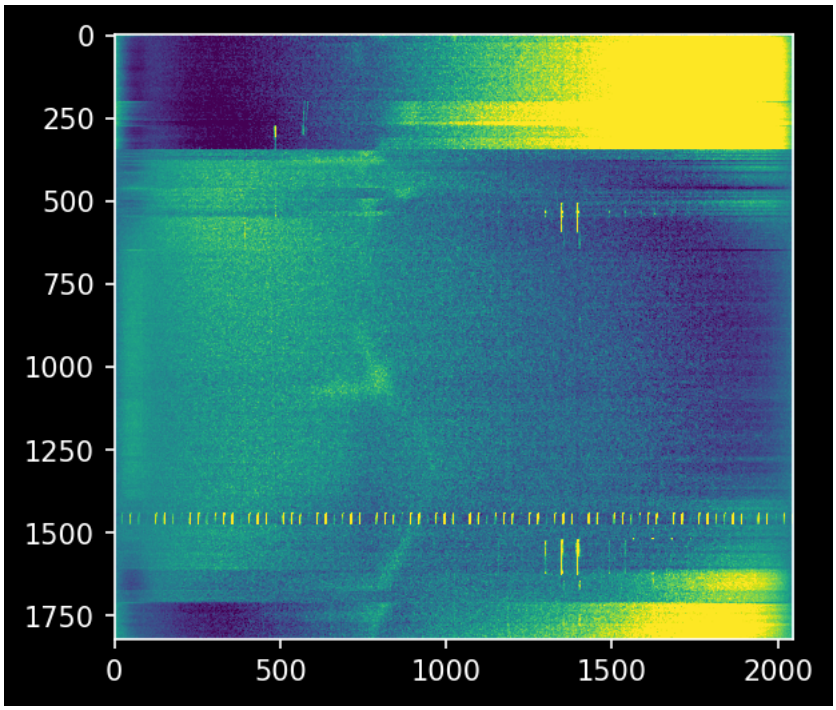
In [30]:
```
pl.imshow(morgantp1b_normed, norm=simple_norm(morgantp1b_normed, min_percent=1, max_percent=99))
```

Out[30]:
```
<matplotlib.image.AxesImage at 0x2102a5fe520>
```



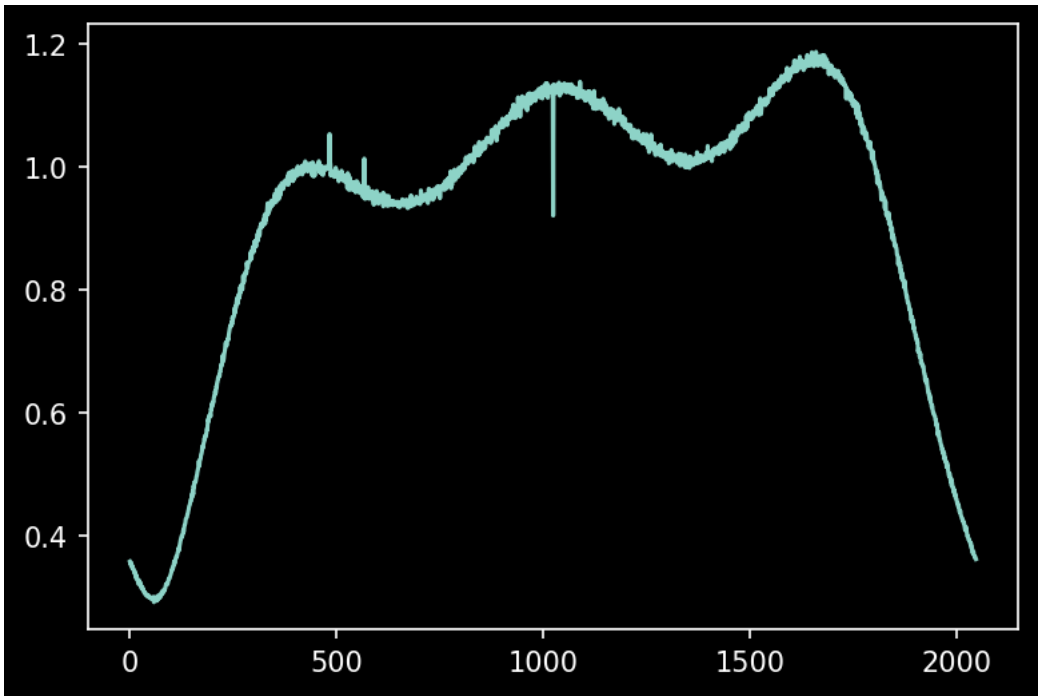
In [31]:
```
pl.imshow(morgantp1b_normsubbed, norm=simple_norm(morgantp1b_normsubbed, min_percent=10, max_percent=90))
```

Out[31]:
```
<matplotlib.image.AxesImage at 0x2102b376b20>
```



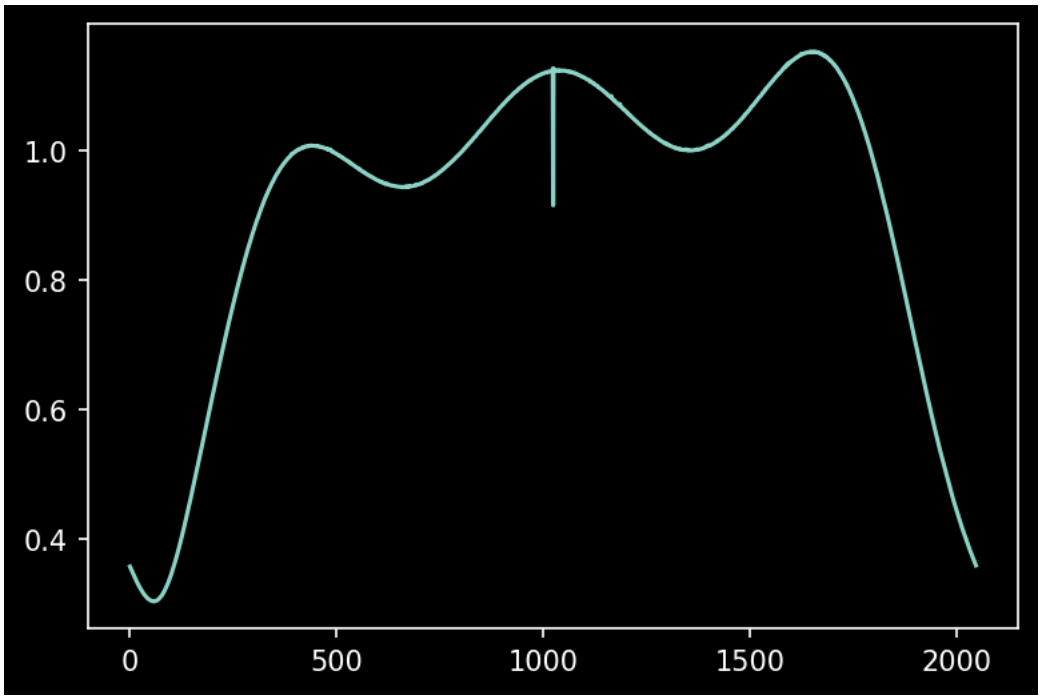
In [32]:
```
pl.plot(morgantp1[300,:] / np.median(morgantp1[300,:]))
```

Out[32]:
```
[<matplotlib.lines.Line2D at 0x2101e9c20d0>]
```



In [33]:
```
pl.plot(np.median(morgantp1b_normed, axis=0))
```

Out[33]:
```
[<matplotlib.lines.Line2D at 0x2102a25e2b0>]
```



Notes from class

two bright sources on wills graph

FSW data

- x axis in km/s
- earths rotation speed 0.5 km/s at equatior
- curve of signal not due to Earth's rotation - it's due to physcal motion of gas we're observing
- first and fourth quadrent are inner galaxt - first signal for will is inner galaxy 36 degrees
- second is around 209 degrees - outer galaxy

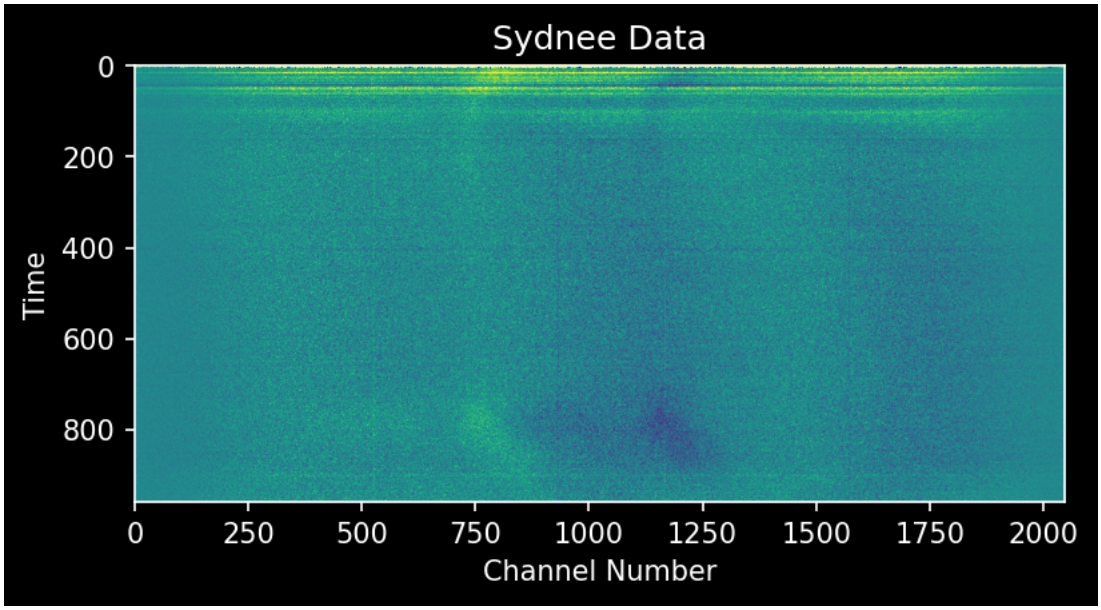
- if you have rfi use multi component fitting - can't do moment analysis
- how determine galctrocentric radius we're seeing when looking at innerr galaxy - tangent line assumption

My data

In [34]:
```
fsw = fsw_data['SydneeODonnell']
fsw_normed = fsw/np.median(fsw, axis=1)[:,None]
fsw_normsubbed = fsw_normed - np.median(fsw_normed, axis=0)
```

In [35]:
```
pl.imshow(fsw_data['SydneeODonnell'], norm=simple_norm(fsw_data['SydneeODonnell'], max_percent=99, min_percent=1) )
pl.title('Sydnee Data')
pl.xlabel("Channel Number")
pl.ylabel("Time")
```

Out[35]:
```
Text(0, 0.5, 'Time')
```

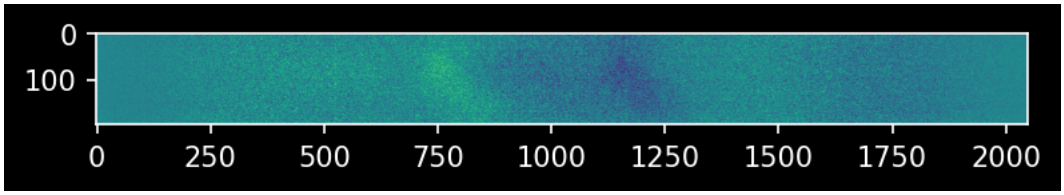


In [36]:
```
badcoords = [isinstance(c, str) for c in coords['SydneeODonnell']]
sum(badcoords)
```

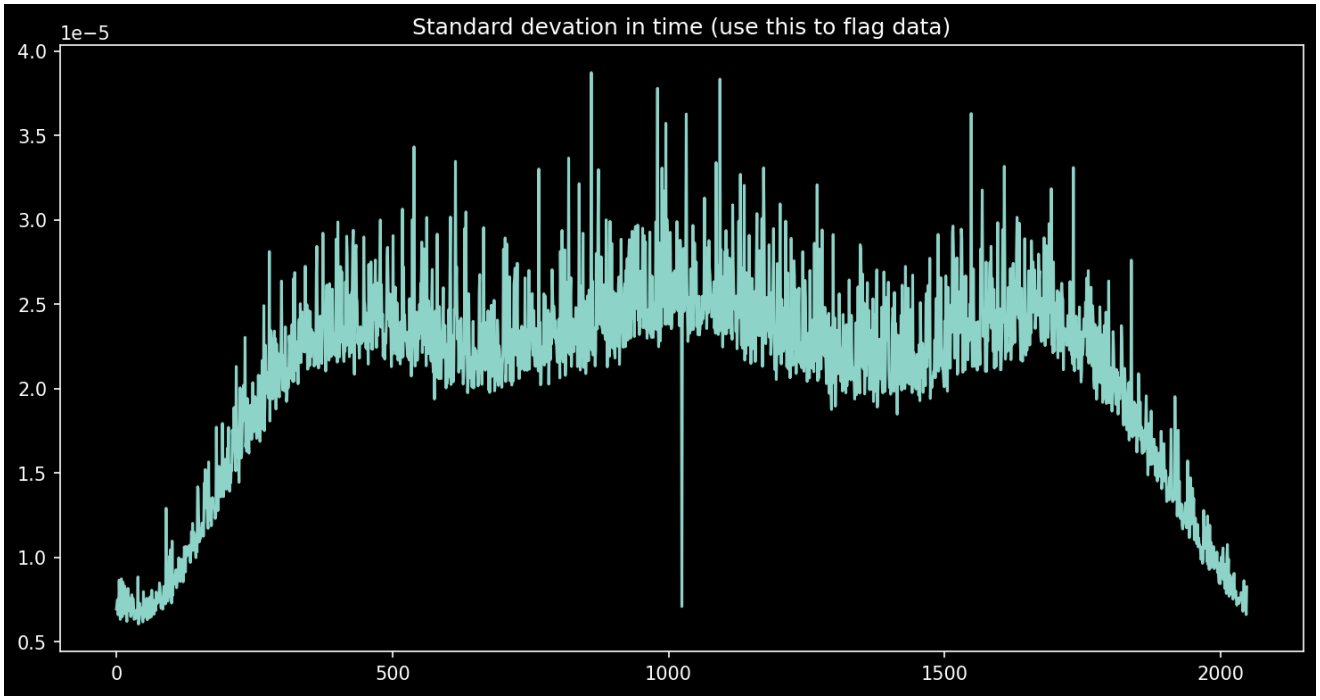
Out[36]:
```
0
```

In [37]:
```
pl.imshow(fsw[700:900, :], norm=simple_norm(fsw, max_percent=99, min_percent=1) )
```

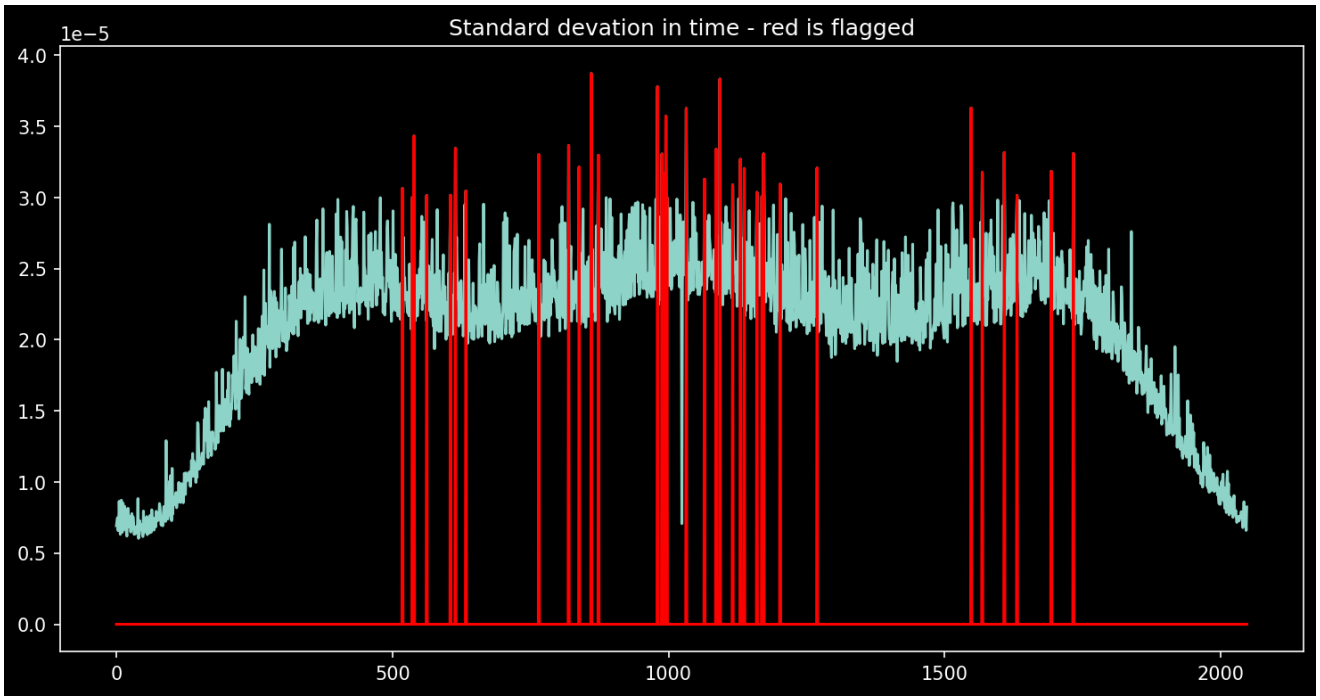
Out[37]:
```
<matplotlib.image.AxesImage at 0x21028f66370>
```



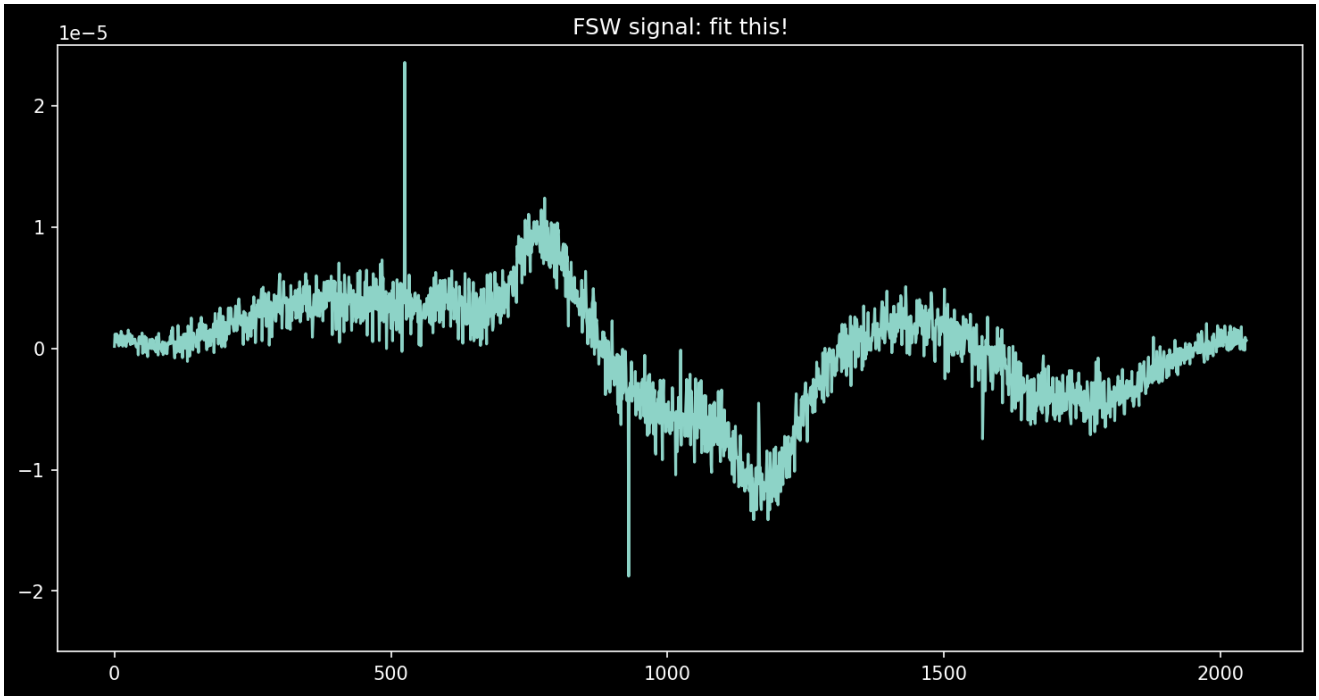
In [38]:
```
pl.figure(figsize=(12,6))
pl.plot(fsw.std(axis=0))
pl.title("Standard devation in time (use this to flag data)");
```



In [39]:
```
pl.figure(figsize=(12,6))
high_stddev = fsw.std(axis=0) > 0.00003
pl.plot(fsw.std(axis=0))
pl.plot(fsw.std(axis=0) * (high_stddev), color='r')
pl.title("Standard devation in time - red is flagged");
```



In [40]:
```
pl.figure(figsize=(12,6))

meanspec = np.mean(fsw[700:900, :], axis=0)
pl.plot(meanspec)
pl.ylim(-0.000025,0.000025)
pl.title("FSW signal: fit this!");
```



In [41]:
```
from astropy.modeling.models import Gaussian1D
from astropy.modeling.fitting import LevMarLSQFitter
```

In [42]:
```
# create a mask to eliminate data we don't want
high_stddev = fsw.std(axis=0) > 0.0003
low_stddev = fsw.std(axis=0) == 0
bad_mask = high_stddev | low_stddev

# average over some time range
meanspec = np.mean(fsw[700:900,:], axis=0)

# estimate the RMS
rms_estimate = meanspec[~bad_mask].std()
big_number = rms_estimate * 10000

# put in a guess (you will likely need a two-component fit!)
guess = (Gaussian1D(0.00001, 0*u.km/u.s, 25*u.km/u.s) + Gaussian1D(0.00005, 50*u.km/u.s, 55*u.km/u.s) +
         Gaussian1D(-0.00015, -100*u.km/u.s, 25*u.km/u.s) + Gaussian1D(0.00005, -50*u.km/u.s, 55*u.km/u.s))

def tie_vel0(model):
    return model.mean_0.value - 100
def tie_vel1(model):
    return model.mean_1.value - 100
guess.mean_2.tied = tie_vel0
guess.mean_3.tied = tie_vel1

fitter = LevMarLSQFitter()
fitted = fitter(guess, fsw_rvel1*u.km/u.s, meanspec, weights=1/rms_estimate + (1/big_number)*bad_mask)
fitted
```

```
WARNING: The fit may be unsuccessful; check fit_info['message'] for more information. [astropy.modeling.fitting]
```

Out[42]:
```
<CompoundModel(amplitude_0=0.00000635 , mean_0=11.05945752 km / s, stddev_0=9.98808563 km / s, amplitude_1=0.0000039 , mean_1=5
4.81556151 km / s, stddev_1=83.61215199 km / s, amplitude_2=-0.00001117 , mean_2=-88.94054248 km / s, stddev_2=14.1598989 km /
s, amplitude_3=-0.00000752 , mean_3=-45.18443849 km / s, stddev_3=22.16345341 km / s)>
```

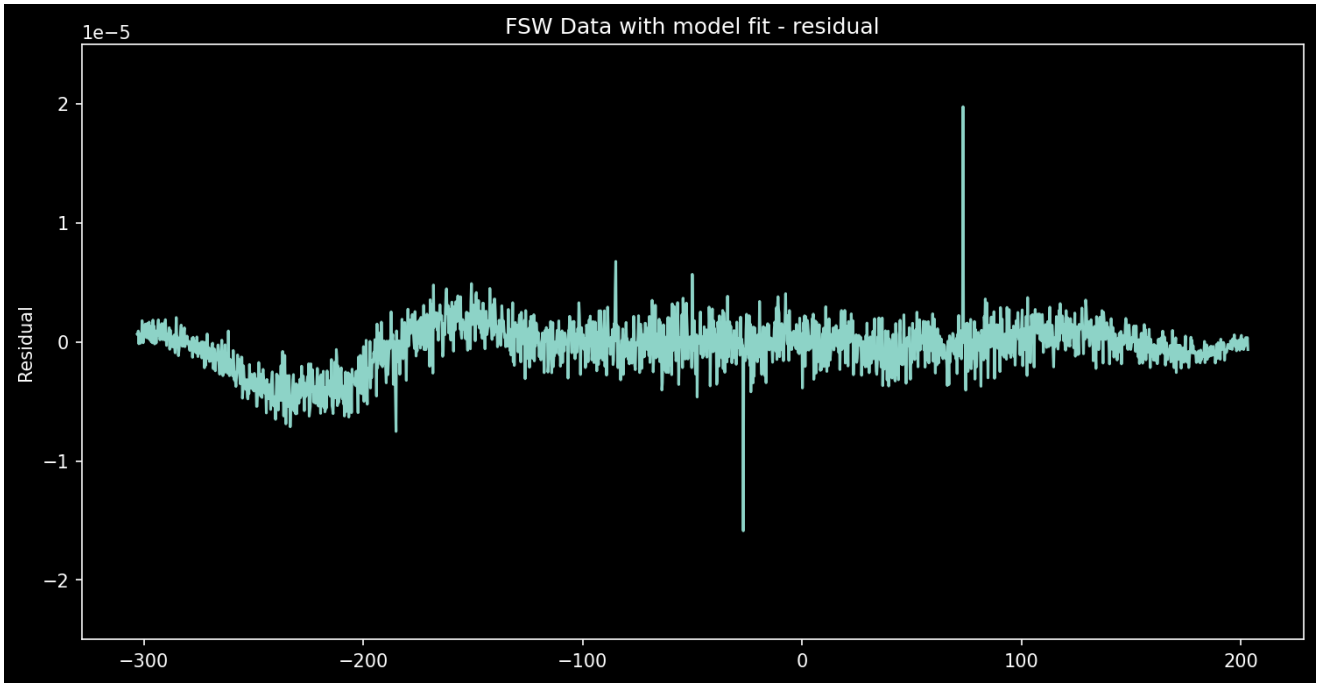
In [43]:
```
pl.figure(figsize=(12,6))
pl.plot(fsw_rvel1, meanspec)
pl.plot(fsw_rvel1, fitted(fsw_rvel1*u.km/u.s))


pl.ylim(-0.000025,0.000025)
pl.title("FSW Data with model fit");
```

In [44]:
```
# uncertantie on parameters
fitter.fit_info['param_cov'].diagonal()
```

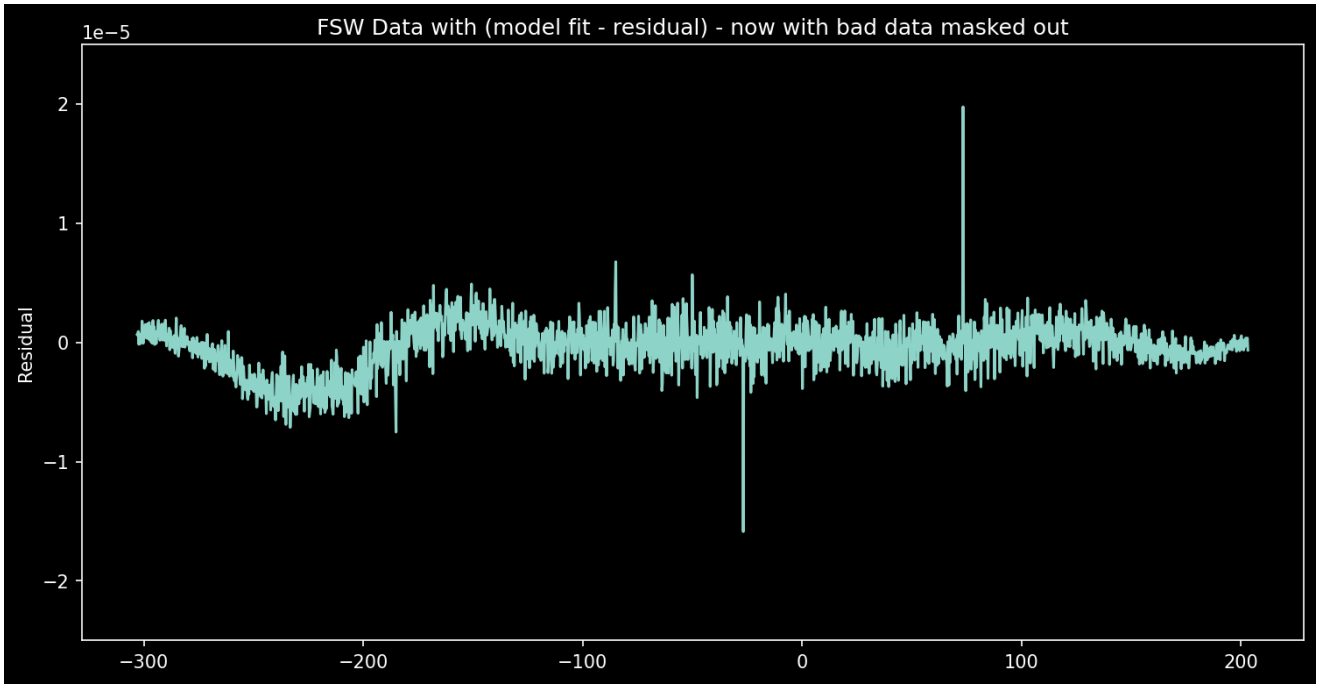
```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
<ipython-input-44-b406adee489b> in <module>
      1 # uncertantie on parameters
----> 2 fitter.fit_info['param_cov'].diagonal()

AttributeError: 'NoneType' object has no attribute 'diagonal'
```

In [ ]:
```

```

In [45]:
```
pl.figure(figsize=(12,6))
pl.plot(fsw_rvel1, (meanspec - fitted(fsw_rvel1*u.km/u.s)))
pl.ylim(-0.000025,0.000025);
pl.ylabel("Residual");
pl.title("FSW Data with model fit - residual");
```



In [46]:
```
pl.figure(figsize=(12,6))
resid = (meanspec - fitted(fsw_rvel1*u.km/u.s))
resid[bad_mask] = np.nan
pl.plot(fsw_rvel1, resid)
pl.ylim(-0.000025,0.000025);
pl.ylabel("Residual");
pl.title("FSW Data with (model fit - residual) - now with bad data masked out");
```



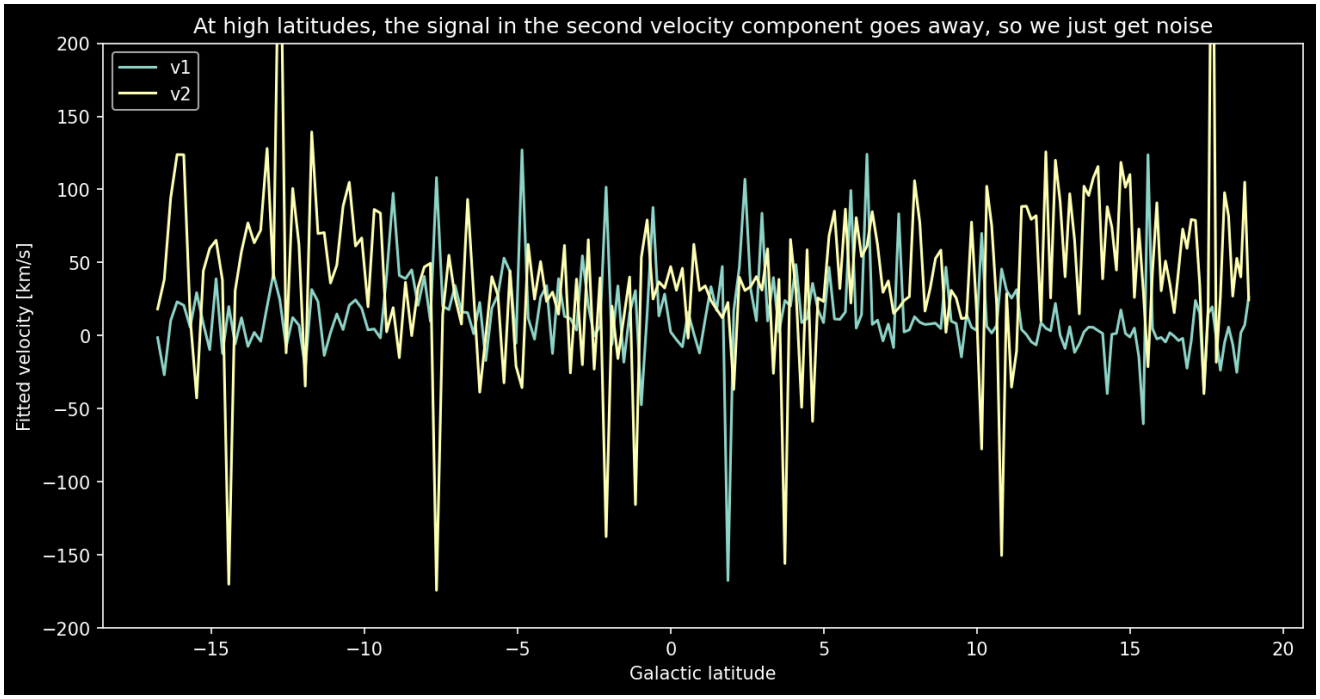
Repeat for all spectra

In [47]:
```
galcoords = coordinates.SkyCoord([c.transform_to(coordinates.Galactic()) for c in coords['SydneeODonnell']
                                  if not isinstance(c, str)])
```

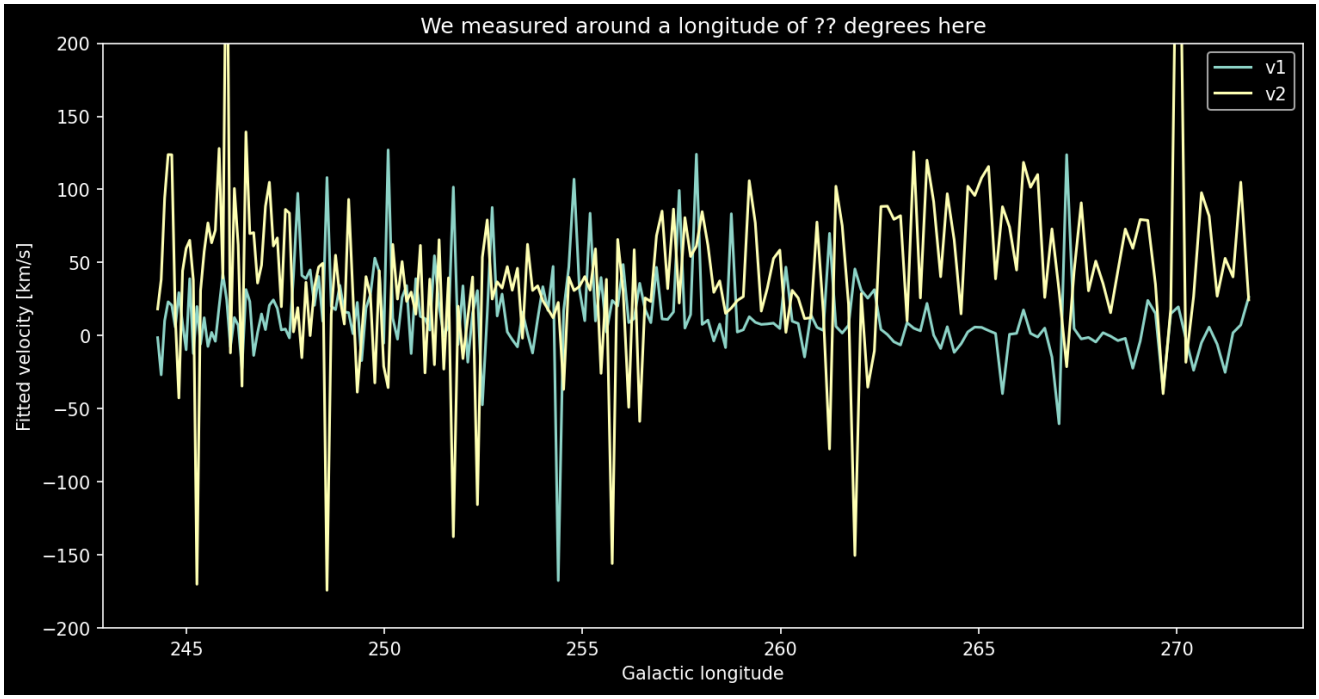
In [48]:
```
fit_list = []
#for index in range(250, 350):
for index in range(700, 900):
    spec = fsw[index,:]

    # estimate the RMS
    rms_estimate = spec[~bad_mask].std()
    big_number = rms_estimate * 10000

    # put in a guess (you will likely need a two-component fit!)
    guess = (Gaussian1D(0.00001, 0*u.km/u.s, 25*u.km/u.s) + Gaussian1D(0.00005, 50*u.km/u.s, 55*u.km/u.s) +
         Gaussian1D(-0.00015, -100*u.km/u.s, 25*u.km/u.s) + Gaussian1D(0.00005, -50*u.km/u.s, 55*u.km/u.s))

    def tie_vel0(model):
        return model.mean_0.value - 100
    def tie_vel1(model):
        return model.mean_1.value - 100
    guess.mean_2.tied = tie_vel0
    guess.mean_3.tied = tie_vel1

    fitter = LevMarLSQFitter()
    fitted = fitter(guess, fsw_rvel1*u.km/u.s, spec, weights=1/rms_estimate + (1/big_number)*bad_mask)
    fit_list.append(fitted)
```

```
WARNING: The fit may be unsuccessful; check fit_info['message'] for more information. [astropy.modeling.fitting]
```

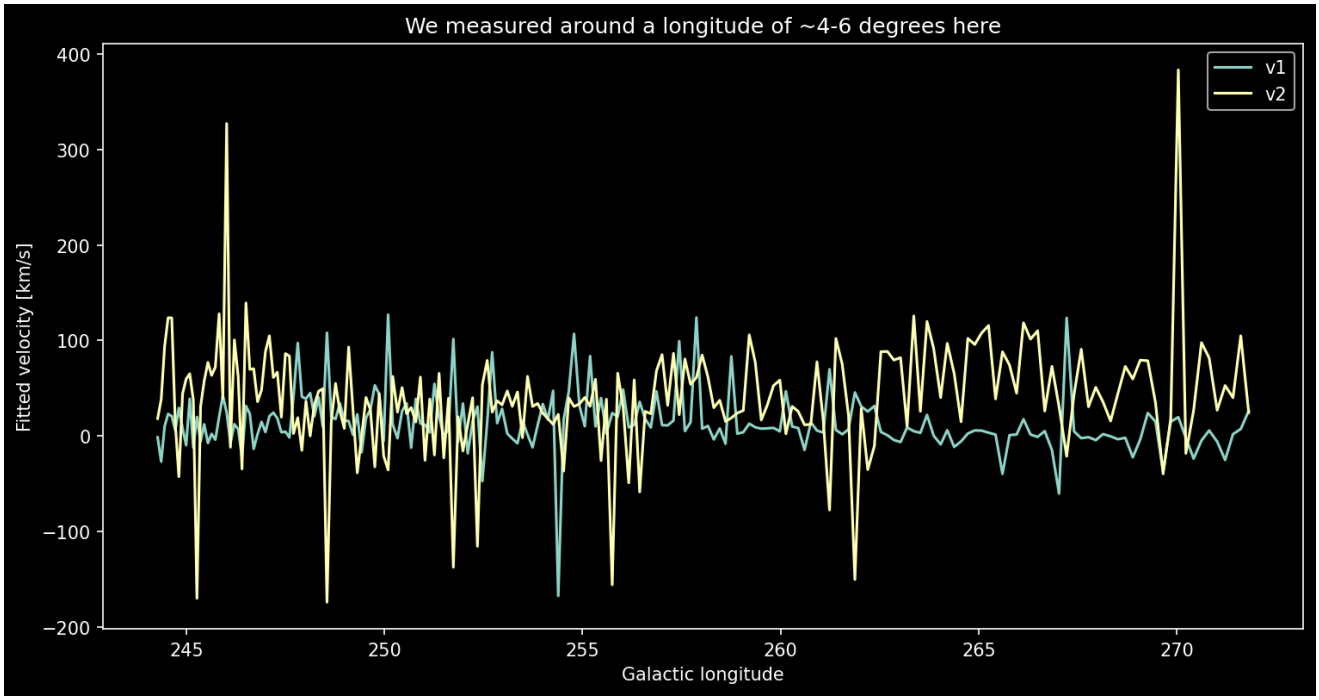
In [49]:
```
pl.figure(figsize=(12,6))
pl.plot(galcoords.b[700:900], [fl.mean_0.value for fl in fit_list], label='v1') # change for midplane
pl.plot(galcoords.b[700:900], [fl.mean_1.value for fl in fit_list], label='v2')
pl.legend(loc='best');
pl.ylim(-200,200)
pl.ylabel("Fitted velocity [km/s]")
pl.xlabel("Galactic latitude");
pl.title("At high latitudes, the signal in the second velocity component goes away, so we just get noise");
```



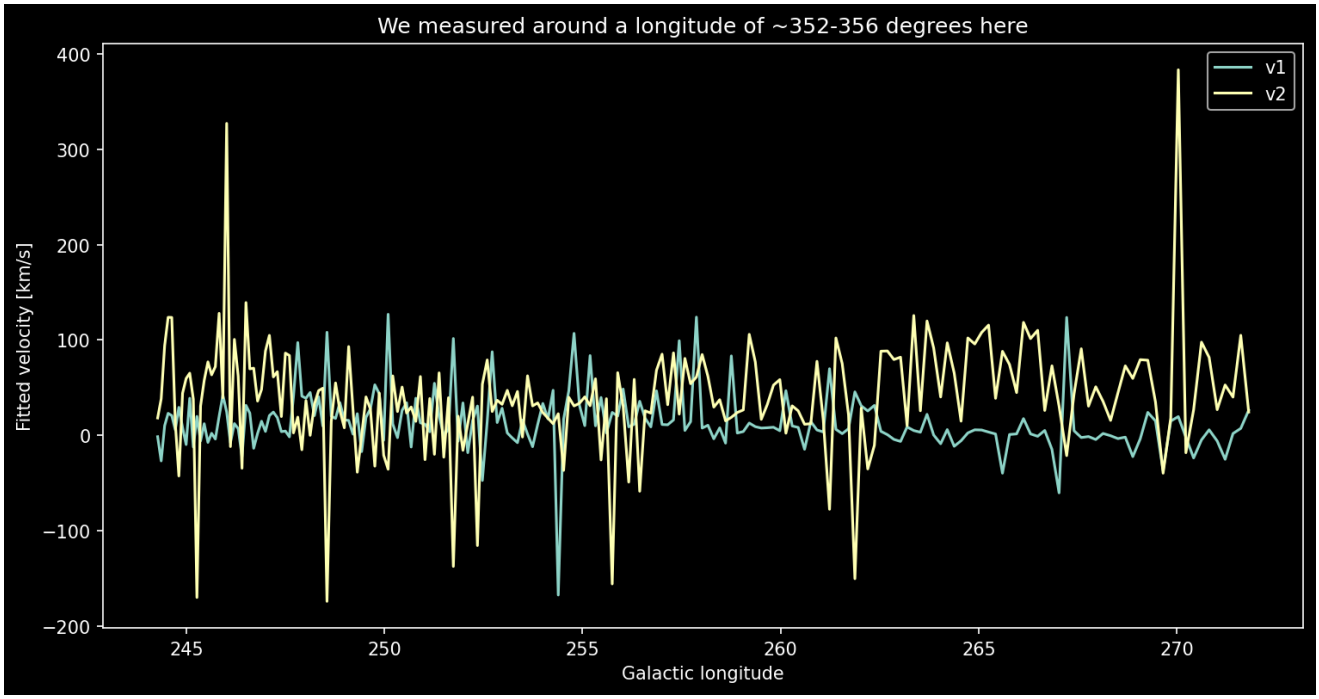
In [50]:
```
pl.figure(figsize=(12,6))
pl.plot(galcoords.l[700:900], [fl.mean_0.value for fl in fit_list], label='v1')
pl.plot(galcoords.l[700:900], [fl.mean_1.value for fl in fit_list], label='v2')
pl.ylim(-200,200)
pl.legend(loc='best');
pl.ylabel("Fitted velocity [km/s]")
pl.xlabel("Galactic longitude");
pl.title("We measured around a longitude of ?? degrees here");
```



In [51]:
```
pl.figure(figsize=(12,6))
pl.plot(galcoords.l[700:900], [fl.mean_0.value for fl in fit_list], label='v1')
pl.plot(galcoords.l[700:900], [fl.mean_1.value for fl in fit_list], label='v2')
#pl.xlim(0,10)
pl.legend(loc='best');
pl.ylabel("Fitted velocity [km/s]")
pl.xlabel("Galactic longitude");
pl.title("We measured around a longitude of ~4-6 degrees here");
```



In [52]:
```
pl.figure(figsize=(12,6))
pl.plot(galcoords.l[700:900], [fl.mean_0.value for fl in fit_list], label='v1')
pl.plot(galcoords.l[700:900], [fl.mean_1.value for fl in fit_list], label='v2')
#pl.xlim(345,362)
pl.legend(loc='best');
pl.ylabel("Fitted velocity [km/s]")
pl.xlabel("Galactic longitude");
pl.title("We measured around a longitude of ~352-356 degrees here");
```



In [117]:
```
# average long and lat over good spectra
```

In [ ]:
```
galcoords = coordinates.SkyCoord([c.transform_to(coordinates.Galactic()) for c in coords['SydneeODonnell'][]
                                  if not isinstance(c, str)])
```