Sagan Summer Workshop 2016
Is There a Planet in My Data?

Detection and Spectroscopic Characterization of Transiting Exoplanet Observations with the James Webb Space Telescope (JWST) Hands-on Session

Facilitator: Nikole Lewis
Contributors: Michael Line, Jason Rowe, Jeff Valenti, Laura Kreidberg

1 Introduction

The James Webb Space Telescope (JWST) will be host to an array of instruments and modes that are suited to the study of transiting exoplanets in wavelength range from 0.6 to 28 μm (see excellent review in Beichman et al. (2014)) The Single-Object Slitless Spectroscopy (SOSS) mode of the Near-InfraRed Image and Slitless Spectrograph (NIRISS) instrument on the James Webb Space Telescope (JWST) was specifically designed to enable high-precision observations of transiting exoplanets. SOSS was specifically designed to have a very wide ($\sim$25 pix) cross-dispersion point-spread function (PSF) that should limit its susceptibility to jitter induced sources of noise and allow for the observation of bright exoplanet host stars (J>6). The SOSS mode operates in the wavelength range between 0.6 and 2.8 μm, which encompasses spectral features of key atmospheric species such as water, carbon dioxide, and methane. The R$\sim$700 spectra from SOSS will be critical in the characteriation of exoplanet atmospheres with JWST.

2 Objectives

The objective of this session is to give you hands-on experience with processing and analyzing synthetic JWST transiting exoplanet datasets. It is expected that the JWST pipeline will produce a range of data products that will allow both novice and expert users to rapidly extract 'science' from their transiting exoplanet observations. Here we will start with data products that are expected to enter the brach of the JWST pipeline specific to time-series observations like those of transiting exoplanets. We will then step you through the spectral extraction, transit fitting, and atmospheric retrieval processes required to extract meaningful contraints on the composition (and complexities) of exoplanet atmospheres from transit observations with JWST. A full analysis typically takes many days, so here we will provide precomputed shortcuts between many steps. However, many of the tools and lessons presented here can be adapted to serve as robust analysis pipelines for future JWST data. All of the code and tools presented here are Python based, which will serve as 'common language' for JWST.

3 Background

Exoplanets that transit their host star as viewed from earth represent our best opportunity for atmospheric characterization studies given observational facilities like JWST. These transiting exoplanets allow their atmospheres and/or surfaces to be probed through high-precision (<1%) relative spectrophotometric observation obtained throughout the planet's

orbit Figure 1. Atmospheric transmission spectra obtained as the planet passes in front of the host star probe the chemical composition of the planet?s atmosphere. The shape of the planetary transit can be used to constrain some of the orbital properties of the planet such semi-major axis (in stellar units, $a/R_\star$) and orbital inclination (i). The orbital period of the planet (P) can be estimated if multiple transit events are measured for the same target. Critically, transits provide constraints on the radius of the planet (in stellar units, $R_p/R_\star$) through the depth of the transit ($R_p^2/R_\star^2$). Further discussion of transit geometry is provided by Winn (2010) and many other references.

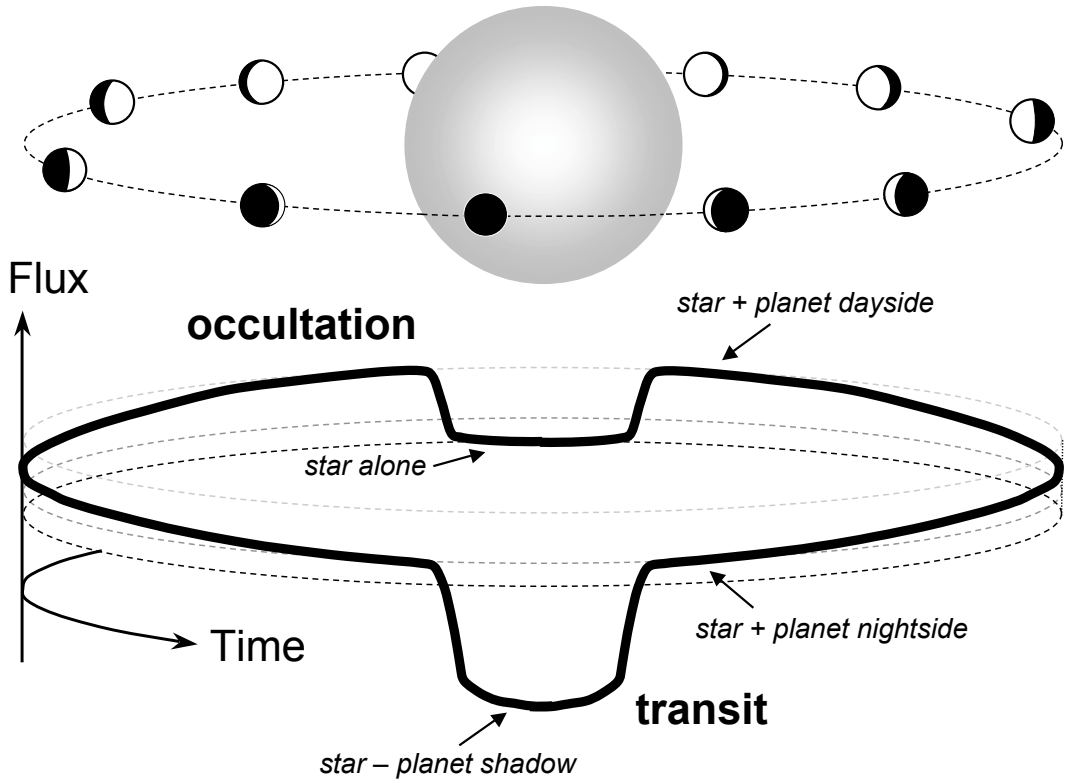


Figure 1: Schematic of transits and occultations from Winn (2010). During transit the planet blocks a portion of the stellar disk creating a drop in the flux from the system. The transit can be used to constrain some of the orbital properties of the planet, but critically the transit yields an estimate of the radius of the planet.

Limb darkening is the term used to describe the ratio of stellar intensity to central intensity from disk center to the limb. At disk center this ratio is unity by definition. For simple geometric reasons, as one views progressively closer to the limb, the spectrum forms progressively higher in the stellar atmosphere. Temperature decreases with height in the photosphere, so forming higher in the atmosphere means forming in cooler layers. Cooler layers are fainter. Thus, stellar intensity decreases (get "darker") towards the limb. The effects of limb darkening on the shape of transits can be seen in the left panel of Figure 2. Further discussion of limb darkening in the context of planetary transits can be found in Mandel and Agol (2002); Knutson *et al.* (2007); Espinoza and Jordán (2015, 2016) (not an exhaustive list).

Measuring planetary transits at multiple wavelengths produces a planetary spectrum (see right panel of Figure 2). The planet will appear larger or smaller depending on the opacity of the atmosphere at a given wavelength. The amplitude of these spectral features will depend on planet size, temperature, and atmospheric composition. Roughly speaking the level of absorption will scale as the transit depth ($R_p^2/R_\star^2$) times the ratio of the planetary scale height (H) to the planetary bulk radius (R_p). The planetary scale height is determined by both the planetary temperature and atmospheric composition, $H = kT_p/\mu g$, where k is Boltzmann's constant, T_p is the temperature of the planet, μ is the mean molecular weight of the atmosphere, and g is the average planetary gravitational acceleration. An excellent reference for the topic of exoplanet atmospheres is the book entitled "Exoplanet Atmospheres: Physical Processes" by Sara Seager (Seager, 2010)

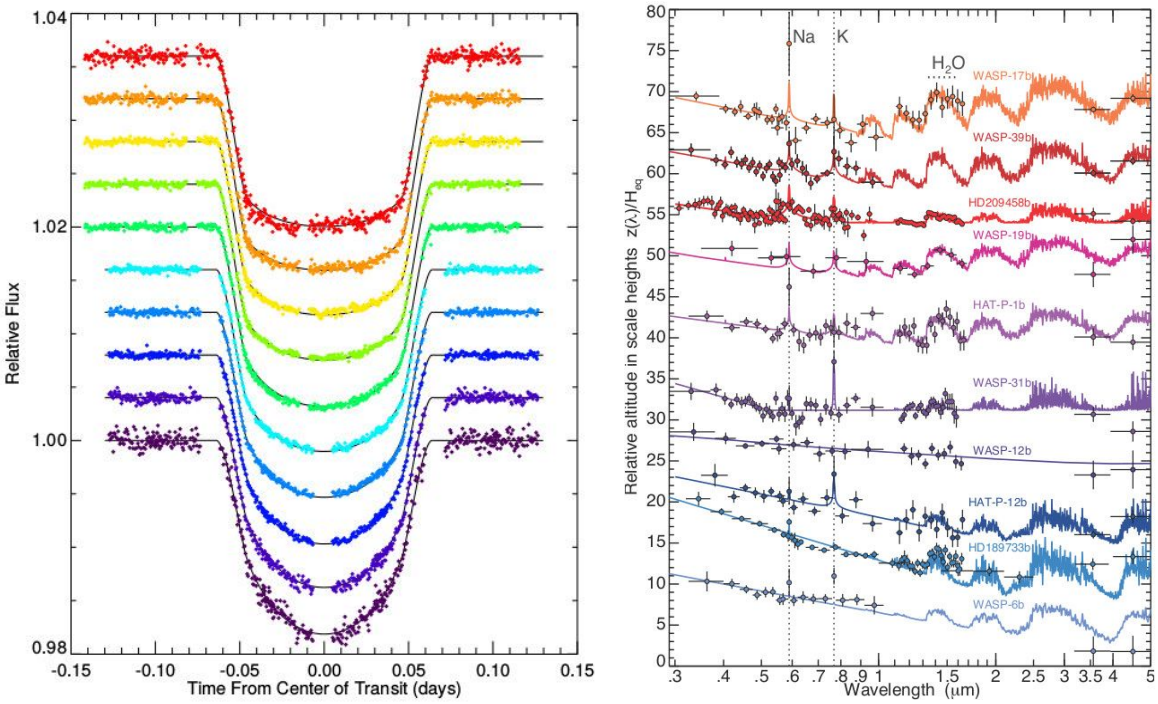


Figure 2: Left panel: Normalized light curves of the transiting exoplanet HD 209458b in ten spectroscopic channels Knutson *et al.* (2007). The change in limb darkening from 300 nm (purple) to 1000 nm (red) is readily noticeable. Right panel: Transmission spectra of ten exoplanets Sing *et al.* (2016). These example spectra illustrate the current state of the field. JWST will provide more precise, higher-resolution transmission spectra over a broader wavelength range (0.6 - 28 μm).

Atmospheric retrieval refers to the coupling between Bayesian parameter estimation and radiative transfer. In general, the purpose of an atmospheric retrieval is to extract the absorber abundances and thermal structures from a spectrum with the ultimate goal of better understanding the physical and chemical processes occurring in planetary atmospheres. The concept of atmospheric retrieval is by no means new, with its origin in the Earth remote sensing community (Rodgers, 1976; Twomey *et al.*, 1977; Rodgers, 2000; Crisp *et al.*, 2004)

developed to interpret satellite based sounding data to improve weather forecasting. Retrieval is often seen as a mysterious "dark art" but is little more than a simple parameter estimation problem. Most of the challenges in atmospheric retrieval involve the complex nature of planetary atmospheres and how to parameterize them in a computationally feasible manner.

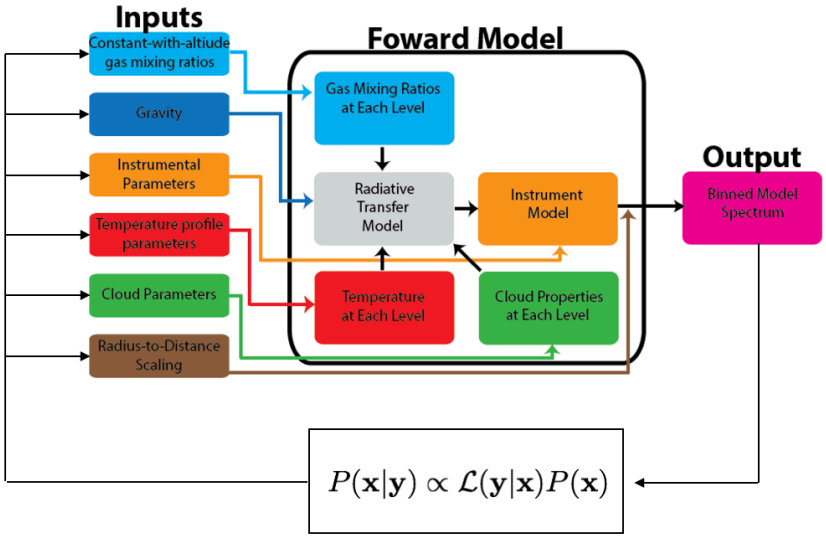


Figure 3: Schematic of retrieval process.

As remote sensing data of the solar system planets improved via orbiters and probes, atmospheric retrieval techniques were necessarily required. The techniques usually included a radiative transfer model (also refereed to as the forward model) that would determine the top-of-atmosphere fluxes as a function of wavelength given the vertical thermal profile, cloud properties, and molecular abundances coupled with a Levenberg-Marquardt solver (Conrath *et al.*, 1998; Irwin *et al.*, 2008; Fletcher *et al.*, 2007; Greathouse *et al.*, 2011). This classic inversion approach, Optimal Estimation, makes use of local gradient information, or how the flux at each wavelength varies with a perturbation on each parameter (the Jacobian) to minimize a cost function that is some combination of the data-model fit and deviation from the prior (if incorporated). This is a fast, iterative approach and is generally applicable when the signal-to-noise and spectral resolutions are sufficiently high and systematic uncertainties are negligible such that the parameter space is well approximated by a multi-dimensional Gaussian, typical of earth and solar system data. Gaussian parameter uncertainties are estimated via a covariance matrix about the best fit. The additional complexity of retrieval over a simple "linear-regression" is the inclusion of prior information. For well observed objects strong priors on the thermal structure and abundances could be used.

The above approaches work well for solar system and Earth data where data quality is high, e.g., the signal-to-noise, spectral resolution, and spectral coverage are all very high. In this regime parameter uncertainties are well approximated by Gaussian statistics, and strong priors exist. In the realm of exoplanet atmosphere retrievals, neither of these exist. In this case, classic retrieval approaches can break down. In this case, more sophisticated Bayesian

methods must be employed. The most common are grid search methods (e.g. Madhusudhan and Seager, 2009) or Markov Chain Monte Carlo (Madhusudhan *et al.*, 2011; Benneke and Seager, 2012; Line *et al.*, 2013; Waldmann *et al.*, 2015). Furthemore, atmospheric model parameterizations are ill defined and the answers can depend strongly upon model assumptions. Model selection approaches become increasingly more important to justify the use of one set of model assumptions over the other.

Figure 3 illustrates the basic components of a retrieval algorithm. The forward model, F(x), is usually some type of radiative transfer model that maps the atmospheric state (e.g., temperatures, abundances, cloud properties) onto the data, e.g., transmission, emission, or reflection spectra. The forward model parameters can then be adjusted with various Bayesian methods.

4 Outline

This exercise will consist of three main modules:

1. First we will work with the three-dimensional synthetic NIRISS SOSS spectral data cubes to extract a time-series of the stellar spectrum and produce a light curve(s).

2. Next we will fit planetary transit(s) to the light curve(s) to derive a planetary spectrum.

3. Finally we will perform an atmospheric retrieval exercise to place constraints on the composition of the planetary atmosphere.

Fully executed python notebooks are located in appendix of this document.

5 Exercises

5.1 Spectral Extraction

Here we will take a look at synthetic set of NIRISS SOSS 2D and 1D spectra timeseries. These data are similar to what will be produced by level 2 and 3 of the TSO branch of the JWST pipeline. It is important to note that the JWST pipeline will deliver high-quality science ready products, such as extracted spectra and white-light curves. However, understanding how those spectra are produced is essential for furthering the science yield of JWST. High-level questions for this exercise are:

1. What are the potential complications with extracting JWST time-series spectra?

2. With the time-series spectra in hand, how do you find your planetary transit?

3. What is the potential utility of the white-light curve data JWST pipeline data product?

The python notebook and supporting codes/files are contained in the spec_extract/ folder. To begin the exercise, ensure that you are in the correct folder and type:
jupyter notebook Spectral_Extraction_Exercise.ipynb

5.2 Light-curve Fitting

Here we will step through the process of fitting theoretical planetary light-curves as a function of wavelength to produce a planetary spectrum (Rp(?)/R?). High-level questions for this exercise are:

1. How do each of the key planetary parameters ($a/R_\star$, inclination, and R_p/R_{star}) affect the shape of the transit?

2. How does your choice of limb-darkening functional form affect the shape of your light curve? What potential errors could be introduced into your planetary spectrum from poor limb-darkening choices (functional form, wavelength dependence, and fixed vs. fitted)?

3. How do you extract your planetary spectrum from the wavelength dependent transit data?

The python notebook and supporting code are contained in the transit/ folder. To begin this exercise, ensure that you are in the correct folder and type:
jupyter notebook Transit_Fitting_with_BATMAN.ipynb

5.3 Spectral Retrieval

In the retrieval component of this workshop we will play with a simple transit transmission spectrum forward model. We will first explore the important physical parameters that govern the shape of a transmission spectrum. We will then analyze the results of a pre-computed atmospheric retrieval (due to time constraints) to determine the degree to which we can constrain various atmospheric properties from a representative JWST spectra. High-level questions for this exercise are:

1. What are the primary constituents in your exoplanet atmosphere based on the derived spectrum?

2. What compositional information can be robustly determined from the NIRISS SOSS data alone?

3. How does the possible presence of clouds affect your compositional solution?

The python notebook and supporting codes and opacity tables are contained in the retrievals/ folder. To begin this exercise, ensure that you are in the correct folder and type:
jupyter notebook Spectral_Retrieval_Exercise.ipynb

References

Benneke, B., and S. Seager 2012. Atmospheric Retrieval for Super-Earths: Uniquely Constraining the Atmospheric Composition with Transmission Spectroscopy. *ApJ* **753**, 100.

Conrath, B. J., P. J. Gierasch, and E. A. Ustinov 1998. Thermal Structure and Para Hydrogen Fraction on the Outer Planets from Voyager IRIS Measurements. *Icarus* **135**, 501–517.

Crisp, D., R. M. Atlas, F.-M. Breon, L. R. Brown, J. P. Burrows, P. Ciais, B. J. Connor, S. C. Doney, I. Y. Fung, D. J. Jacob, C. E. Miller, D. O'Brien, S. Pawson, J. T. Randerson, P. Rayner, R. J. Salawitch, S. P. Sander, B. Sen, G. L. Stephens, P. P. Tans, G. C. Toon, P. O. Wennberg, S. C. Wofsy, Y. L. Yung, Z. Kuang, B. Chudasama, G. Sprague, B. Weiss, R. Pollock, D. Kenyon, and S. Schroll 2004. The Orbiting Carbon Observatory (OCO) mission. *Advances in Space Research* **34**, 700–709.

Espinoza, N., and A. Jordán 2015. Limb darkening and exoplanets: testing stellar model atmospheres and identifying biases in transit parameters. *MNRAS* **450**, 1879–1899.

Espinoza, N., and A. Jordán 2016. Limb darkening and exoplanets - II. Choosing the best law for optimal retrieval of transit parameters. *MNRAS* **457**, 3573–3581.

Fletcher, L. N., P. G. J. Irwin, N. A. Teanby, G. S. Orton, P. D. Parrish, R. de Kok, C. Howett, S. B. Calcutt, N. Bowles, and F. W. Taylor 2007. Characterising Saturn's vertical temperature structure from Cassini/CIRS. *Icarus* **189**, 457–478.

Greathouse, T. K., M. Richter, J. Lacy, J. Moses, G. Orton, T. Encrenaz, H. B. Hammel, and D. Jaffe 2011. A spatially resolved high spectral resolution study of Neptune's stratosphere. *Icarus* **214**, 606–621.

Irwin, P. G. J., N. A. Teanby, R. de Kok, L. N. Fletcher, C. J. A. Howett, C. C. C. Tsang, C. F. Wilson, S. B. Calcutt, C. A. Nixon, and P. D. Parrish 2008. The NEMESIS planetary atmosphere radiative transfer and retrieval tool. *J. Quant. Spec. Radiat. Transf.* **109**, 1136–1150.

Knutson, H. A., D. Charbonneau, R. W. Noyes, T. M. Brown, and R. L. Gilliland 2007. Using Stellar Limb-Darkening to Refine the Properties of HD 209458b. *ApJ* **655**, 564–575.

Line, M. R., A. S. Wolf, X. Zhang, H. Knutson, J. A. Kammer, E. Ellison, P. Deroo, D. Crisp, and Y. L. Yung 2013. A Systematic Retrieval Analysis of Secondary Eclipse Spectra. I. A Comparison of Atmospheric Retrieval Techniques. *ApJ* **775**, 137.

Madhusudhan, N., J. Harrington, K. B. Stevenson, S. Nymeyer, C. J. Campo, P. J. Wheatley, D. Deming, J. Blecic, R. A. Hardy, N. B. Lust, D. R. Anderson, A. Collier-Cameron, C. B. T. Britt, W. C. Bowman, L. Hebb, C. Hellier, P. F. L. Maxted, D. Pollacco, and R. G. West 2011. A high C/O ratio and weak thermal inversion in the atmosphere of exoplanet WASP-12b. *Nature* **469**, 64–67.

Madhusudhan, N., and S. Seager 2009. A Temperature and Abundance Retrieval Method for Exoplanet Atmospheres. *ApJ* **707**, 24–39.

Mandel, K., and E. Agol 2002. Analytic Light Curves for Planetary Transit Searches. *ApJL* **580**, L171–L175.

Rodgers, C. D. 1976. Retrieval of Atmospheric Temperature and Composition From Remote Measurements of Thermal Radiation. *Reviews of Geophysics and Space Physics* **14**, 609.

Rodgers, C. D. 2000. Inverse Methods for Atmospheric Sounding - Theory and Practice. *Inverse Methods for Atmospheric Sounding - Theory and Practice. Series: Series on Atmospheric Oceanic and Planetary Physics, ISBN: ¡ISBN¿9789812813718¡/ISBN¿. World Scientific Publishing Co. Pte. Ltd., Edited by Clive D. Rodgers, vol. 2* **2**.

Seager, S. 2010. *Exoplanet Atmospheres: Physical Processes*.

Sing, D. K., J. J. Fortney, N. Nikolov, H. R. Wakeford, T. Kataria, T. M. Evans, S. Aigrain, G. E. Ballester, A. S. Burrows, D. Deming, J.-M. Désert, N. P. Gibson, G. W. Henry, C. M. Huitson, H. A. Knutson, A. L. D. Etangs, F. Pont, A. P. Showman, A. Vidal-Madjar, M. H. Williamson, and P. A. Wilson 2016. A continuum from clear to cloudy hot-Jupiter exoplanets without primordial water depletion. *Nature* **529**, 59–62.

Twomey, S., B. Herman, and R. Rabinoff 1977. An Extension to the Chahine Method of Inverting the Radiative Transfer Equation. *Journal of Atmospheric Sciences* **34**, 1085–1090.

Waldmann, I. P., M. Rocchetto, G. Tinetti, E. J. Barton, S. N. Yurchenko, and J. Tennyson 2015. Tau-REx II: Retrieval of Emission Spectra. *ApJ* **813**, 13.

Winn, J. N. 2010. *Exoplanet Transits and Occultations*, pp. 55–77.

A Appendix

Welcome! In this notebook we will step you through how to work with high-level data products that will be produced by the time-series branch of the JWST pipeline. This exercise was compiled by Nikole Lewis (STScI) with significant inputs from Jason Rowe (UdeM) and Jeff Valenti (STScI).

In [1]:
```
#First let's load some useful modules
import numpy as np
import matplotlib
import matplotlib.pyplot as plt
from astropy.io import fits
%matplotlib inline
```

In [2]:
```
#Let's first investigate the 2D SOSS trace images as a function of tim
e and get some info about what's in the fits file
hdulist2D= fits.open('planet1_2Dim.fits')
hdulist2D.info()
```

```
Filename: planet1_2Dim.fits
No.    Name         Type      Cards   Dimensions   Format
0    PRIMARY     PrimaryHDU      44   ()
1    SCI         ImageHDU        10   (2048, 256, 144)   float32
2    JDMID       ImageHDU         8   (144,)       float64
```

The primary starting point for all "Level 3" processing of time-series data (spectra and photometry) are datacubes that contain the 2D "slope" images, which have had basic corrections/calibrations applied (bias, dark, flat, etc.). The fits file here contains primary header information, the 2D images, and a table of julien dates (JD) that correspond to the center of each integration (time standards beyond JD such as BJD_UTC, BJD_TT, and HJD will be provided).

In [3]:
```
#First we'll check on the information in the primary header
primary = hdulist2D[0]
primary.header
```

```
Out[3]: SIMPLE  =                    T / Fits standard
        BITPIX  =                   16 / Bits per pixel
        NAXIS   =                    0 / Number of axes
        EXTEND  =                    T / FITS dataset may contain extensions
        IRAF-TLM= '2016-05-29T04:16:56' / Time of last modification
        DATE    = '2016-05-29T04:16:56.00' / Date this file was created (UTC
        )
        FILETYPE= 'UNCALIBRATED'       / Type of data in the file
        TELESCOP= 'JWST    '           / Telescope used to acquire the data

                Programmatic information

        TITLE   = 'Exoplanet characterization in the JWST era' / Proposal ti
        tle
        PI_NAME = 'John Mather'        / Principal investigator name

                Instrument configuration information

        INSTRUME= 'NIRISS  '           / Instrument used to acquire the data

                Exposure parameters

        NINTS   =                  144 / Number of integrations in exposure
        NGROUPS =                    7 / Number of groups in integration
        NFRAMES =                    1 / Number of frames per group
        GROUPGAP=                    0 / Number of frames dropped between gr
        oups
        NSAMPLES=                    1 / Number of A/D samples per pixel
        TSAMPLE =                   10 / Time between samples (microsec)
        TFRAME  =              5.49130 / Time between frames (sec)
        TGROUP  =              5.49130 / Time between groups (sec)
        EFFINTTM=              32.9478 / Effective integration time (sec)
        EFFEXPTM=              4744.48 / Effective exposure time (sec)
        NRSTSTRT=                    1 / Number of resets at start of exposu
        re

                Subarray parameters

        SUBARRAY= 'SUBSTRIP256'        / Subarray used
        SUBSTRT1=                    1 / Number of pixels in axis 1 directio
        n
        SUBSIZE1=                 2048 / Number of pixels in axis 1 directio
        n
        SUBSIZE2=                  256 / Number of pixels in axis 2 directio
        n
        FASTAXIS=                    2 / Fast readout axis direction
        SLOWAXIS=                    1 / Slow readout axis direction

                Aperture information

        APERNAME= 'NIS-SOSSTA'         / science aperture used
```

In [4]:
```
#Now let's look at the science images and header information
sci = hdulist2D[1]
sci.data.shape
```

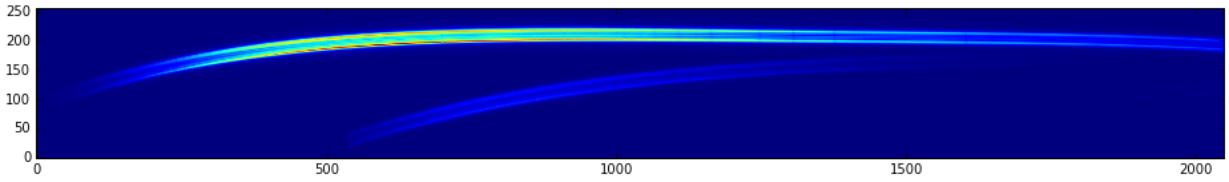
Out[4]:
```
(144, 256, 2048)
```

In [5]:
```
sci.header
```

Out[5]:
```
XTENSION= 'IMAGE   '           / Image extension
BITPIX  =                  -32 / array data type
NAXIS   =                    3 /
NAXIS1  =                 2048 /
NAXIS2  =                  256 /
NAXIS3  =                  144 /
PCOUNT  =                    0 / number of random group parameters
GCOUNT  =                    1 / number of random groups
EXTNAME = 'SCI     '           / extension name
BUNIT   = 'DN/s    '           / physical units of the data array va
lues
```

In [6]:
```
fig, ax = plt.subplots(figsize=(18, 2))
ax.imshow(sci.data[0,:,:], origin='lower')
```

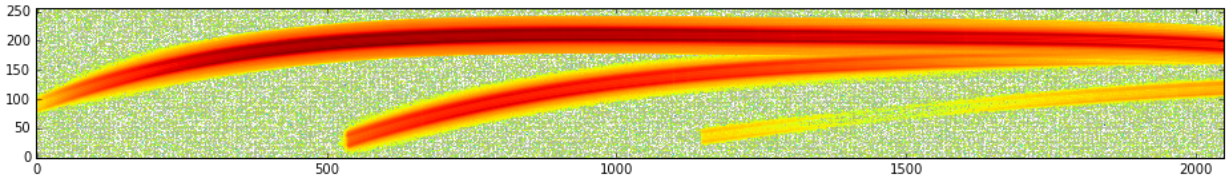
Out[6]:
```
<matplotlib.image.AxesImage at 0x1100ef590>
```



Above should be an image of the SOSS spectral traces in the 256x2048 subarray. What do you notice about these spectral traces?

In [7]:
```
#Let's use a log scale to get another view of this 2D SOSS image
from matplotlib.colors import LogNorm
fig, ax = plt.subplots(figsize=(18, 2))
ax.imshow(sci.data[0,:,:], origin='lower', norm=LogNorm())
```

Out[7]:
```
<matplotlib.image.AxesImage at 0x1290db590>
```



The image above should clearly show the three-orders of the NIRISS SOSS spectra (1st, 2nd, and 3rd from top to bottom). The 1st order covers roughly 0.7-2.8 microns, the 2nd order covers roughly 0.6-1.3 microns, and the 3rd order covers roughly 0.6-0.85 microns. Wavelength increases to the right in the image above, with the 1st and 2nd order overlapping at long wavelength end. What benefits and challenges do you see with having multiple orders available for NIRISS SOSS spectra?

We will not work with extracting spectra from these 2D images in this hands-on session, but encourage you to think about/develop spectral extraction techniques given this synthetic data. The level 3 of the time-series branch of the JWST pipeline will perform spectral extraction and deliver spectra as function of time. The fidelity of the extract spectra will likely evolve over time as techniques are developed/improved and integrated into the pipeline. Keep in mind that the entire python-based JWST pipeline will be available to all users and modules like the time-series spectral extraction can be swapped for user-defined versions.

In [8]:
```
#Before we leave the 2D image data, let's get our JD values
jdmid = hdulist2D[2]
jdmid.data.shape
```

Out[8]:
```
(144,)
```

In [9]:
```
#Now let's move to the extracted spectra as a function of time
hdulist1D= fits.open('planet1_1Dspec.fits')
hdulist1D.info()
```

```
Filename: planet1_1Dspec.fits
No.    Name         Type      Cards   Dimensions   Format
0    PRIMARY     PrimaryHDU      44   ()
1    EXTRACT1D   BinTableHDU     17   2048R x 2C   [1E, 1E]
2    EXTRACT1D   BinTableHDU     17   2048R x 2C   [1E, 1E]
3    EXTRACT1D   BinTableHDU     17   2048R x 2C   [1E, 1E]
4    EXTRACT1D   BinTableHDU     17   2048R x 2C   [1E, 1E]
5    EXTRACT1D   BinTableHDU     17   2048R x 2C   [1E, 1E]
6    EXTRACT1D   BinTableHDU     17   2048R x 2C   [1E, 1E]
7    EXTRACT1D   BinTableHDU     17   2048R x 2C   [1E, 1E]
8    EXTRACT1D   BinTableHDU     17   2048R x 2C   [1E, 1E]
9    EXTRACT1D   BinTableHDU     17   2048R x 2C   [1E, 1E]
10   EXTRACT1D   BinTableHDU     17   2048R x 2C   [1E, 1E]
11   EXTRACT1D   BinTableHDU     17   2048R x 2C   [1E, 1E]
12   EXTRACT1D   BinTableHDU     17   2048R x 2C   [1E, 1E]
13   EXTRACT1D   BinTableHDU     17   2048R x 2C   [1E, 1E]
14   EXTRACT1D   BinTableHDU     17   2048R x 2C   [1E, 1E]
15   EXTRACT1D   BinTableHDU     17   2048R x 2C   [1E, 1E]
16   EXTRACT1D   BinTableHDU     17   2048R x 2C   [1E, 1E]
17   EXTRACT1D   BinTableHDU     17   2048R x 2C   [1E, 1E]
18   EXTRACT1D   BinTableHDU     17   2048R x 2C   [1E, 1E]
19   EXTRACT1D   BinTableHDU     17   2048R x 2C   [1E, 1E]
20   EXTRACT1D   BinTableHDU     17   2048R x 2C   [1E, 1E]
21   EXTRACT1D   BinTableHDU     17   2048R x 2C   [1E, 1E]
22   EXTRACT1D   BinTableHDU     17   2048R x 2C   [1E, 1E]
```

```
129  EXTRACT1D   BinTableHDU     17   2048R x 2C   [1E, 1E]
130  EXTRACT1D   BinTableHDU     17   2048R x 2C   [1E, 1E]
131  EXTRACT1D   BinTableHDU     17   2048R x 2C   [1E, 1E]
132  EXTRACT1D   BinTableHDU     17   2048R x 2C   [1E, 1E]
133  EXTRACT1D   BinTableHDU     17   2048R x 2C   [1E, 1E]
134  EXTRACT1D   BinTableHDU     17   2048R x 2C   [1E, 1E]
135  EXTRACT1D   BinTableHDU     17   2048R x 2C   [1E, 1E]
136  EXTRACT1D   BinTableHDU     17   2048R x 2C   [1E, 1E]
137  EXTRACT1D   BinTableHDU     17   2048R x 2C   [1E, 1E]
138  EXTRACT1D   BinTableHDU     17   2048R x 2C   [1E, 1E]
139  EXTRACT1D   BinTableHDU     17   2048R x 2C   [1E, 1E]
140  EXTRACT1D   BinTableHDU     17   2048R x 2C   [1E, 1E]
141  EXTRACT1D   BinTableHDU     17   2048R x 2C   [1E, 1E]
142  EXTRACT1D   BinTableHDU     17   2048R x 2C   [1E, 1E]
143  EXTRACT1D   BinTableHDU     17   2048R x 2C   [1E, 1E]
144  EXTRACT1D   BinTableHDU     17   2048R x 2C   [1E, 1E]
```

What do you notice about the way that this data is stored compared to the 2D images?

In [10]:
```
#Let's take a look at the primary header information
hdulist1D[0].header
```

```
Out[10]: SIMPLE  =                    T / Fits standard
         BITPIX  =                   16 / Bits per pixel
         NAXIS   =                    0 / Number of axes
         EXTEND  =                    T / File may contain extensions
         IRAF-TLM= '2016-05-29T04:16:56' / Time of last modification
         DATE    = '2016-05-29T04:16:56.00' / Date this file was created (UTC
         )
         FILETYPE= 'UNCALIBRATED'       / Type of data in the file
         TELESCOP= 'JWST    '           / Telescope used to acquire the data

                 Programmatic information

         TITLE   = 'Exoplanet characterization in the JWST era' / Proposal ti
         tle
         PI_NAME = 'John Mather'        / Principal investigator name

                 Instrument configuration information

         INSTRUME= 'NIRISS  '           / Instrument used to acquire the data

                 Exposure parameters

         NINTS   =                  144 / Number of integrations in exposure
         NGROUPS =                    7 / Number of groups in integration
         NFRAMES =                    1 / Number of frames per group
         GROUPGAP=                    0 / Number of frames dropped between gr
         oups
         NSAMPLES=                    1 / Number of A/D samples per pixel
         TSAMPLE =                   10 / Time between samples (microsec)
         TFRAME  =              5.49130 / Time between frames (sec)
         TGROUP  =              5.49130 / Time between groups (sec)
         EFFINTTM=              32.9478 / Effective integration time (sec)
         EFFEXPTM=              4744.48 / Effective exposure time (sec)
         NRSTSTRT=                    1 / Number of resets at start of exposu
         re

                 Subarray parameters

         SUBARRAY= 'SUBSTRIP256'        / Subarray used
         SUBSTRT1=                    1 / Number of pixels in axis 1 directio
         n
         SUBSIZE1=                 2048 / Number of pixels in axis 1 directio
         n
         SUBSIZE2=                  256 / Number of pixels in axis 2 directio
         n
         FASTAXIS=                    2 / Fast readout axis direction
         SLOWAXIS=                    1 / Slow readout axis direction

                 Aperture information

         APERNAME= 'NIS-SOSSTA'         / science aperture used
```

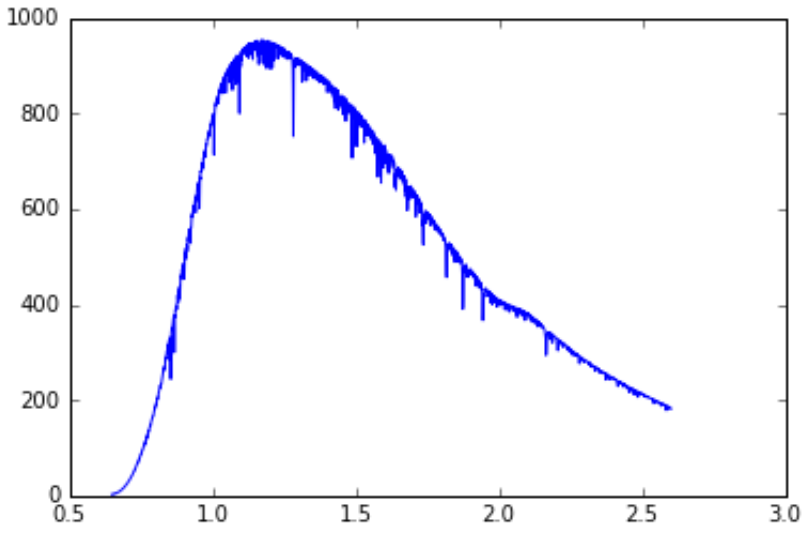
In [11]:
```
#Let's look at the first spectrum
spec1 = hdulist1D[1]
spec1.header
```

Out[11]:
```
XTENSION= 'BINTABLE'           / binary table extension
BITPIX  =                    8 / array data type
NAXIS   =                    2 / number of array dimensions
NAXIS1  =                    8 / length of dimension 1
NAXIS2  =                 2048 / length of dimension 2
PCOUNT  =                    0 / number of group parameters
GCOUNT  =                    1 / number of group
TFIELDS =                    2 / number of table fields
EXTNAME = 'EXTRACT1D'          / extension name
EXTVER  =                    1 / extension value
JDMID   =       2458710.46502 / Julian date at integration midpoint
TFORM1  = '1E      '           /Real*4 (floating point)
TTYPE1  = 'wavelength'         /Label for column 1
TUNIT1  = 'micron  '           /Units of column 1
TFORM2  = '1E      '           /Real*4 (floating point)
TTYPE2  = 'countrate'          /Label for column 2
TUNIT2  = 'DN/s    '           /Units of column 2
```

In [12]:
```
#Let's extract the JD from the header and read in the wavelength and c
ount rate
jd = spec1.header['JDMID']
wavelength = spec1.data.field(0)
countrate = spec1.data.field(1)
```

In [13]:
```
#Let's plot the first (in time) spectrum
plt.plot(wavelength, countrate)
```

Out[13]:
```
[<matplotlib.lines.Line2D at 0x12d016e50>]
```



Now that you see how to read-in/extract the first spectrum you are ready to FIND THE PLANET IN YOUR DATA! Your first goal is to first produce a 'white-light' light curve (relative flux as a function of time). The white-light light curve will be a quick look product for the time-series branch of the JWST pipeline. Then you should try to produce light curves for various wavelengths (unbinned and binned).

TRANSIT FITTING with BATMAN!!!!

All credit goes to Laura Kreidberg for the development of BATMAN (cite: Kriedberg (2015), PASP) and the excellent documentation maintained here: http://astro.uchicago.edu/~kreidberg/batman/index.html (http://astro.uchicago.edu/~kreidberg/batman/index.html) All credit goes to Dan Foreman-Mackey and Contributors for the development of EMCEE (cite: Foreman-Mackey et al (2013), PASP) with documenation maintained here: http://dan.iel.fm/emcee/current/ (http://dan.iel.fm/emcee/current/)

This notebook will walk you through the process of fitting wavelength dependent transits to produce a planetary atmospheric spectrum. This exercise was compiled by Nikole Lewis (STScI) with substantial inputs from Laura Kreidberg (Harvard). First, let's initialize the BATMAN model.

In [1]:
```
#Import required libraries
import numpy as np
import batman #package by Laura Kreidberg:  http://astro.uchicago.edu/~kreidberg/batman/
import emcee  #package by Dan Foreman-Mackey:  http://dan.iel.fm/emcee/current/
import corner #package by Dan Foreman-Mackey:  http://corner.readthedocs.io/en/latest/
import matplotlib
import matplotlib.pyplot as plt
%matplotlib inline
```

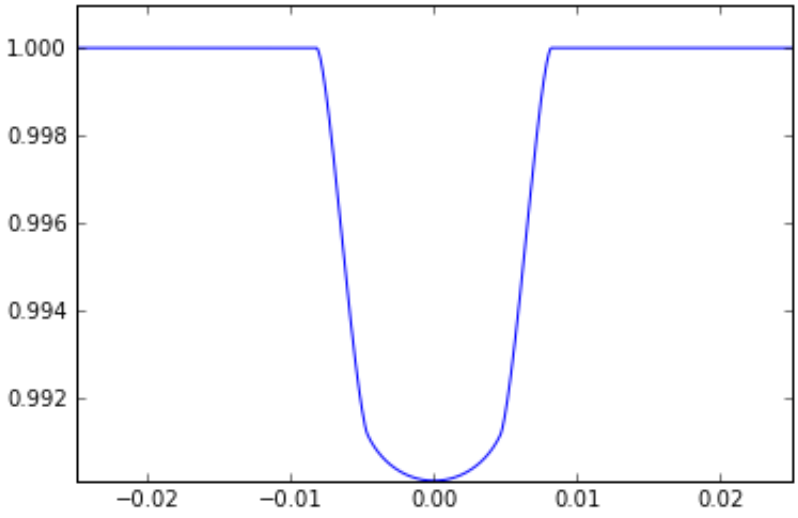
In [2]:
```
#Initialize Parameters
params = batman.TransitParams()       #object to store transit parameters
params.t0 = 0.                        #time of inferior conjunction
params.per = 1.                       #orbital period
params.rp = 0.1                       #planet radius (in units of stellar radii)
params.a = 15.                        #semi-major axis (in units of stellar radii)
params.inc = 87.                      #orbital inclination (in degrees)
params.ecc = 0.                       #eccentricity
params.w = 90.                        #longitude of periastron (in degrees)
params.limb_dark = "nonlinear"        #limb darkening model ->"uniform", "linear", "quadratic", "nonlinear", etc.
params.u = [0.5, 0.1, 0.1, -0.1]      #limb darkening coefficients

t = np.linspace(-0.025, 0.025, 1000)  #times at which to calculate light curve
m = batman.TransitModel(params, t)    #initializes model
```

In [3]:
```
#Now let's make a light curve
flux = m.light_curve(params)                    #calculates light curv
e
```

In [4]:
```
#Let's plot the light curve
plt.plot(t, flux)
plt.axis([t.min(), t.max(), flux.min(), 1.001])
```

Out[4]:
```
[-0.025000000000000001, 0.025000000000000001, 0.99013517123557859, 1
.001]
```

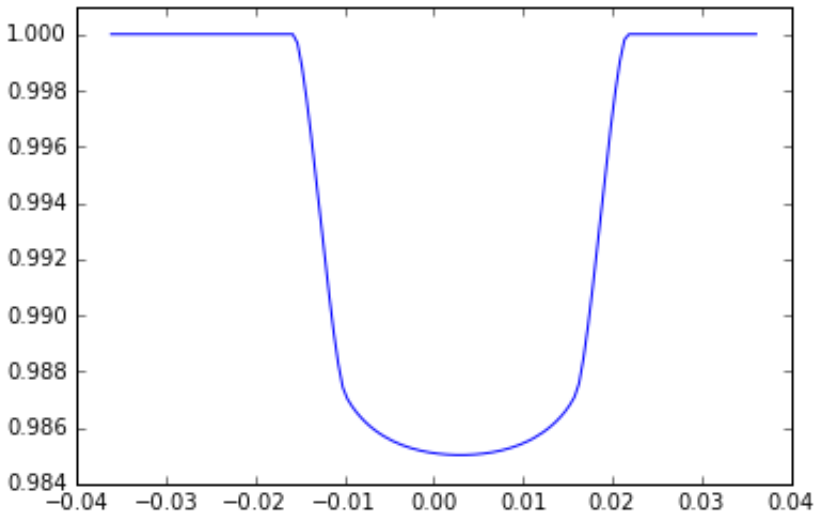


Now take some time to adjust various parameters and see they affect the shape of the tranist light curve. Pay particular attention to how your choices for limb darkening parameters and model affect the transit shape.

In [5]:
```
#We have saved light curves generated from the spectral extraction exe
rcise.  Let's load that data.
import pickle
jdtot, white_lc, bin16_lcs, w16 = pickle.load(open("planet1_lcs.pic","
r"))
```

In [6]:
```
#Let's plot the 'white-light' light curve
plt.plot(jdtot-np.median(jdtot), white_lc)
plt.axis([-0.04, 0.04, 0.984, 1.001])
```

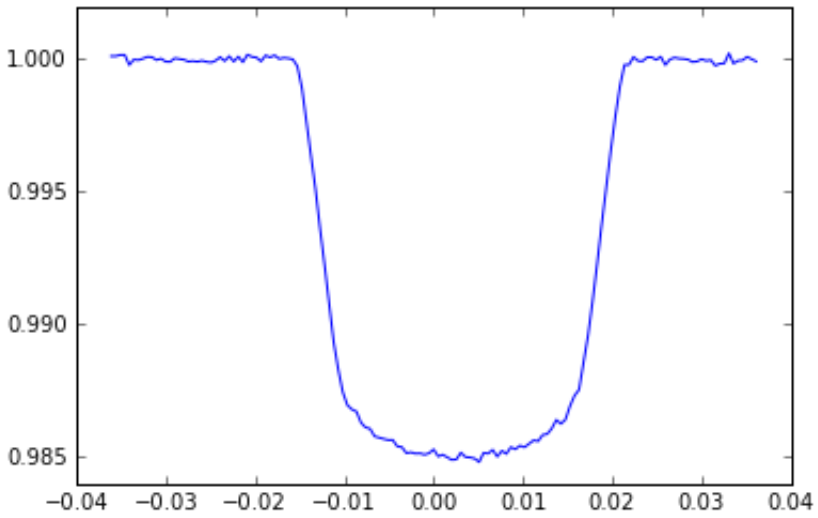
Out[6]: [-0.04, 0.04, 0.984, 1.001]



In [76]:
```
#Above should be a nice clean noise-free light curve.  This is not rea
lstic.
#Let's create some noised up more realistic data and do some MCMC fitt
ing
n = jdtot.size
t = jdtot-np.median(jdtot)
flux=white_lc
err = 100.e-6  #100 ppm noise
flux = flux + np.random.normal(0, err, n)
```

In [77]:
```
#Let's take a look at our noisy creation
plt.plot(t, flux)
```

Out[77]: [<matplotlib.lines.Line2D at 0x1155f1090>]



In the next few cells we're going to define some functions to assist with our fitting and allow us to interface the EMCEE and BATMAN.

In [78]:
```
#intialize a transit model
def initialize_model(t, t0, per, rp, a, inc, ecc, w, u, limb_dark):
        params = batman.TransitParams()
        params.t0 = t0
        params.per = per
        params.rp = rp
        params.a = a
        params.inc = inc
        params.ecc = ecc
        params.w = w
        params.u = u
        params.limb_dark = limb_dark

        model = batman.TransitModel(params, t)

        return params, batman.TransitModel(params, t)    #return param
eters and model objects
```

In [79]:
```
#prior
def lnprior(theta):
        return 0.  #assumes all priors have uniform probability
```

In [162]:
```
#likelihood function
def lnlike(theta, params, model, t, flux, err):
    params.rp, params.u = theta[0], [theta[1]]  #update parameters
    params.t0 = theta[2]
#    params.a, params.inc = theta[3], theta[4]
    lc = model.light_curve(params)
    residuals = flux - lc
    ln_likelihood = -0.5*(np.sum((residuals/err)**2 + np.log(2.0*np.pi
*(err)**2)))

    return ln_likelihood
```

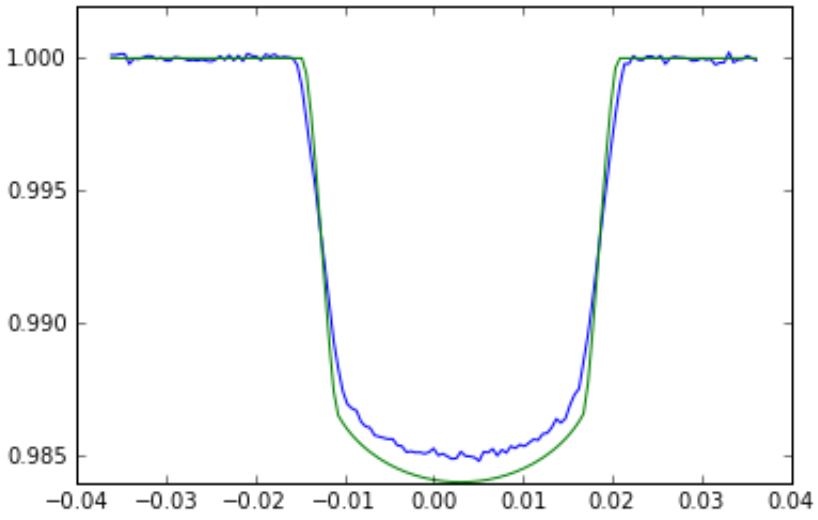
In [142]:
```
#posterior probability
def lnprob(theta, params, model, t, flux, err):
        lp = lnprior(theta)
        if not np.isfinite(lp):
                return -np.inf
        return lp + lnlike(theta, params, model, t, flux, err)
```

In [155]:
```
#Now for some initial parameter guesses
t0 = 0.0  #time of inferior conjunction
per = 1.58040482  #orbital period -> contrained from other observation
s
rp = 0.1  #planet radius (in units of stellar radii)
a = 16.0   #semi-major axis (in units of stellar radii)
inc = 90. #orbital inclination (in degrees)
ecc = 0.  #eccentricity  -> Let's assume for now that the planet is no
t on an eccentric orbit
w = 90.   #longitude of periastron (in degrees)
u = [0.3] #limb darkening coefficients
limb_dark = "linear" #limb darkening model -> Simple limb darkening mo
del, need to consider other models
```

In [156]:
```
#initialize model and parameters
params, m = initialize_model(t, t0, per, rp, a, inc, ecc, w, u, limb_d
ark)
```

In [157]:
```
#compare your initial guess with the data and make adjustments
plt.plot(t, flux) #blue
plt.plot(t, m.light_curve(params)) #green
```

Out[157]:
```
[<matplotlib.lines.Line2D at 0x11c7e3910>]
```

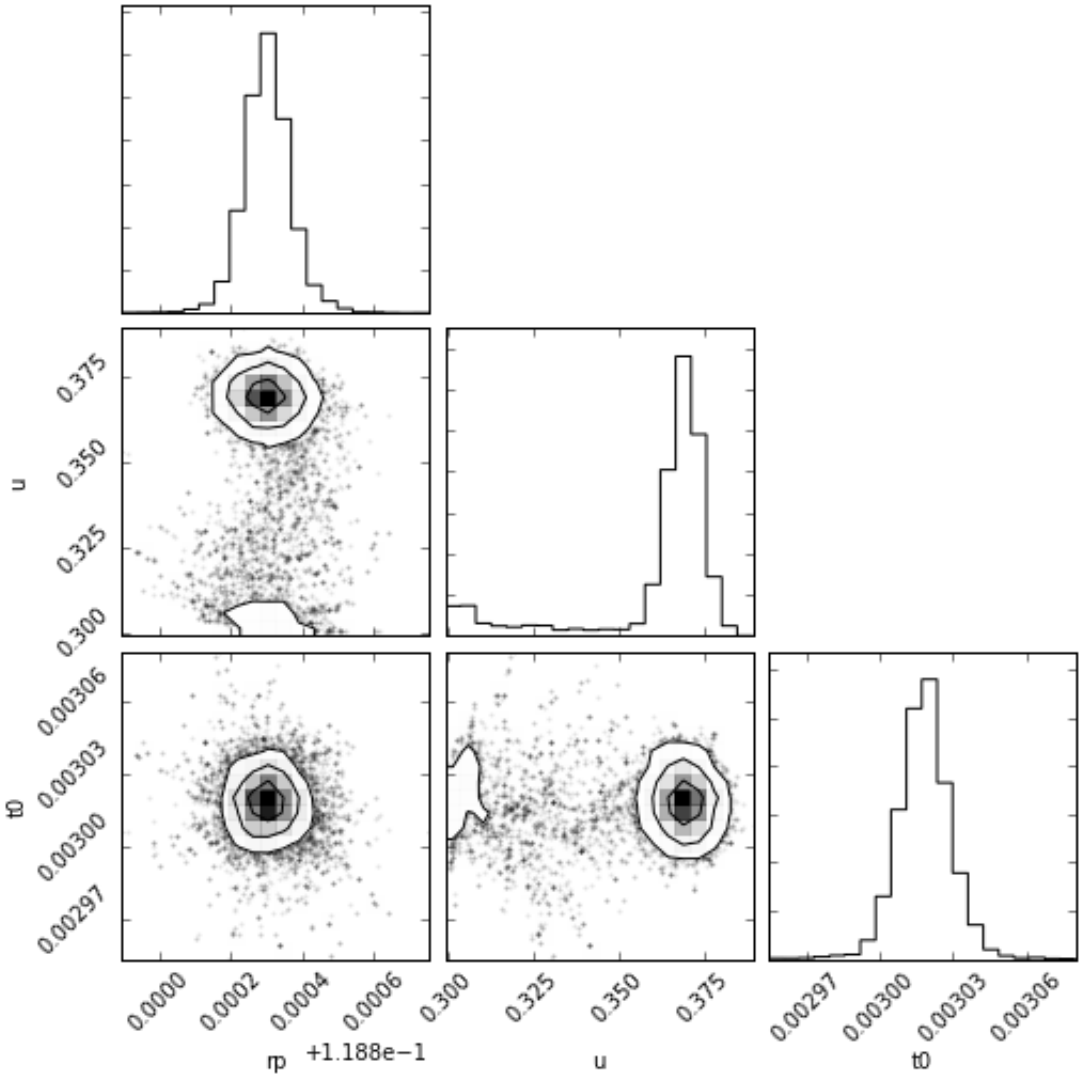


In [163]:
```
#now let's get the MCMC initialized
#initial guesses for MCMC fit parameters from your by eye fits
#for simplicity we will only fit for Rp/R*, limb darkening coefficient
, and center of transit time
guess_rp, guess_u, guess_t0 = 0.12, 0.3, 0.003
theta = [guess_rp, guess_u, guess_t0]
```

In [164]:
```
#initialize sampler
ndim, nwalkers = len(theta), 50
sampler = emcee.EnsembleSampler(nwalkers, ndim, lnprob, args = (params
, m, t, flux, err))
pos = [theta + 1e-5*np.random.randn(ndim) for i in range(nwalkers)]
```

In [165]:
```
#run mcmc
sampler.run_mcmc(pos,500);
```

In [167]:
```
#make a pairs plot from MCMC output
import corner
samples = sampler.chain[:, 50:, :].reshape((-1, ndim)) #discard first
50 samples as burn-in
fig = corner.corner(samples, labels = ["rp", "u", "t0"])
plt.show()
```



Above you will see what is referred to as a 'corner plot', which gives you the posterior probability distributions for your transit parameters and shows the covariances between the parameters. Take a moment to inspect the corner plot and note any strong covariances between parameters. Think about what the histograms are telling you about the 'best-fit' value for each parameter. Go back and increase/decrease the value of the 'err' on your data and see how that affects the parameter distributions.

In [168]:
```
#Now we need to derive the best-fit planet parameters and their 1-sigm
a error bars
rp_mcmc, u_mcmc, t0_mcmc = map(lambda v: (v[1], v[2]-v[1], v[1]-v[0]),
zip(*np.percentile(samples, [16, 50, 84], axis=0)))
print rp_mcmc
print u_mcmc
print t0_mcmc
```

```
(0.119101358754768, 6.0069357746411756e-05, 5.96454386467965e-05)
(0.36831275084557974, 0.0053943474324035789, 0.013640732805393685)
(0.0030180323134686134, 1.0029810048214211e-05, 9.8698143549636949e-
06)
```

Now that you've seen how it's done for the 'white-light' light curve, it's time to find out what kind of planet is in your data. To produce a planetary transit spectrum, the fitting must be done as a function of wavelength. The results from the 'white-light' light curve provide an important initial constraints. Remember that as a function of wavelength only $R_p/R_\star$ and the limb darkening should change, the planetary orbital parameters are not wavelength dependent. The choice of limb darkening model and whether to fix or fit for the limb darkening parameters is widely discussed in the literature. Your choices for limb darkening should be well justified from both a theoretical and statistical perspective. Now, FIND YOUR PLANETARY SPECTRUM!!!

Welcome! In this notebook we will step you through how to work with high-level data products that will be produced by the time-series branch of the JWST pipeline. This exercise was compiled by Nikole Lewis (STScI) with significant inputs from Jason Rowe (UdeM) and Jeff Valenti (STScI).

In [1]:
```
#First let's load some useful modules
import numpy as np
import matplotlib
import matplotlib.pyplot as plt
from astropy.io import fits
%matplotlib inline
```

In [2]:
```
#Let's first investigate the 2D SOSS trace images as a function of tim
e and get some info about what's in the fits file
hdulist2D= fits.open('planet1_2Dim.fits')
hdulist2D.info()
```

```
Filename: planet1_2Dim.fits
No.    Name         Type      Cards   Dimensions   Format
0    PRIMARY     PrimaryHDU      44   ()
1    SCI         ImageHDU        10   (2048, 256, 144)   float32
2    JDMID       ImageHDU         8   (144,)       float64
```

The primary starting point for all "Level 3" processing of time-series data (spectra and photometry) are datacubes that contain the 2D "slope" images, which have had basic corrections/calibrations applied (bias, dark, flat, etc.). The fits file here contains primary header information, the 2D images, and a table of julien dates (JD) that correspond to the center of each integration (time standards beyond JD such as BJD_UTC, BJD_TT, and HJD will be provided).

In [3]:
```
#First we'll check on the information in the primary header
primary = hdulist2D[0]
primary.header
```

```
Out[3]: SIMPLE  =                    T / Fits standard
        BITPIX  =                   16 / Bits per pixel
        NAXIS   =                    0 / Number of axes
        EXTEND  =                    T / FITS dataset may contain extensions
        IRAF-TLM= '2016-05-29T04:16:56' / Time of last modification
        DATE    = '2016-05-29T04:16:56.00' / Date this file was created (UTC
        )
        FILETYPE= 'UNCALIBRATED'       / Type of data in the file
        TELESCOP= 'JWST    '           / Telescope used to acquire the data

                Programmatic information

        TITLE   = 'Exoplanet characterization in the JWST era' / Proposal ti
        tle
        PI_NAME = 'John Mather'        / Principal investigator name

                Instrument configuration information

        INSTRUME= 'NIRISS  '           / Instrument used to acquire the data

                Exposure parameters

        NINTS   =                  144 / Number of integrations in exposure
        NGROUPS =                    7 / Number of groups in integration
        NFRAMES =                    1 / Number of frames per group
        GROUPGAP=                    0 / Number of frames dropped between gr
        oups
        NSAMPLES=                    1 / Number of A/D samples per pixel
        TSAMPLE =                   10 / Time between samples (microsec)
        TFRAME  =              5.49130 / Time between frames (sec)
        TGROUP  =              5.49130 / Time between groups (sec)
        EFFINTTM=              32.9478 / Effective integration time (sec)
        EFFEXPTM=              4744.48 / Effective exposure time (sec)
        NRSTSTRT=                    1 / Number of resets at start of exposu
        re

                Subarray parameters

        SUBARRAY= 'SUBSTRIP256'        / Subarray used
        SUBSTRT1=                    1 / Number of pixels in axis 1 directio
        n
        SUBSIZE1=                 2048 / Number of pixels in axis 1 directio
        n
        SUBSIZE2=                  256 / Number of pixels in axis 2 directio
        n
        FASTAXIS=                    2 / Fast readout axis direction
        SLOWAXIS=                    1 / Slow readout axis direction

                Aperture information

        APERNAME= 'NIS-SOSSTA'         / science aperture used
```

In [4]:
```
#Now let's look at the science images and header information
sci = hdulist2D[1]
sci.data.shape
```

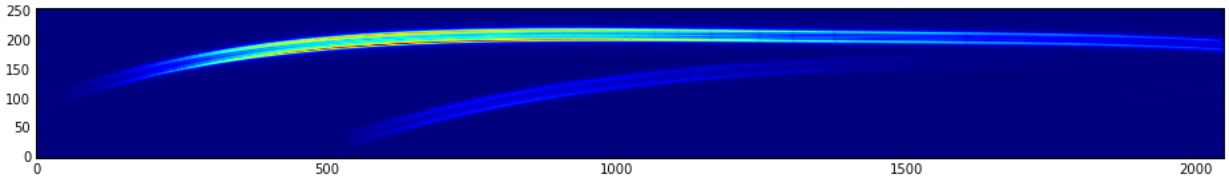
Out[4]:
```
(144, 256, 2048)
```

In [5]:
```
sci.header
```

Out[5]:
```
XTENSION= 'IMAGE   '           / Image extension
BITPIX  =                  -32 / array data type
NAXIS   =                    3 /
NAXIS1  =                 2048 /
NAXIS2  =                  256 /
NAXIS3  =                  144 /
PCOUNT  =                    0 / number of random group parameters
GCOUNT  =                    1 / number of random groups
EXTNAME = 'SCI     '           / extension name
BUNIT   = 'DN/s    '           / physical units of the data array va
lues
```

In [6]:
```
fig, ax = plt.subplots(figsize=(18, 2))
ax.imshow(sci.data[0,:,:], origin='lower')
```

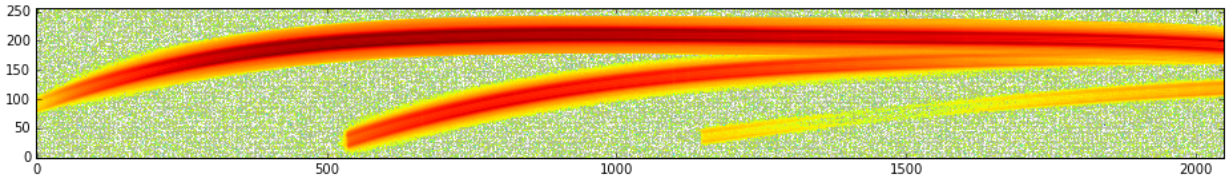
Out[6]:
```
<matplotlib.image.AxesImage at 0x1100ef590>
```



Above should be an image of the SOSS spectral traces in the 256x2048 subarray. What do you notice about these spectral traces?

In [7]:
```
#Let's use a log scale to get another view of this 2D SOSS image
from matplotlib.colors import LogNorm
fig, ax = plt.subplots(figsize=(18, 2))
ax.imshow(sci.data[0,:,:], origin='lower', norm=LogNorm())
```

Out[7]:
```
<matplotlib.image.AxesImage at 0x1290db590>
```



The image above should clearly show the three-orders of the NIRISS SOSS spectra (1st, 2nd, and 3rd from top to bottom). The 1st order covers roughly 0.7-2.8 microns, the 2nd order covers roughly 0.6-1.3 microns, and the 3rd order covers roughly 0.6-0.85 microns. Wavelength increases to the right in the image above, with the 1st and 2nd order overlapping at long wavelength end. What benefits and challenges do you see with having multiple orders available for NIRISS SOSS spectra?

We will not work with extracting spectra from these 2D images in this hands-on session, but encourage you to think about/develop spectral extraction techniques given this synthetic data. The level 3 of the time-series branch of the JWST pipeline will perform spectral extraction and deliver spectra as function of time. The fidelity of the extract spectra will likely evolve over time as techniques are developed/improved and integrated into the pipeline. Keep in mind that the entire python-based JWST pipeline will be available to all users and modules like the time-series spectral extraction can be swapped for user-defined versions.

In [8]:
```
#Before we leave the 2D image data, let's get our JD values
jdmid = hdulist2D[2]
jdmid.data.shape
```

Out[8]:
```
(144,)
```

In [9]:
```
#Now let's move to the extracted spectra as a function of time
hdulist1D= fits.open('planet1_1Dspec.fits')
hdulist1D.info()
```

```
Filename: planet1_1Dspec.fits
No.    Name         Type      Cards   Dimensions   Format
0    PRIMARY     PrimaryHDU      44   ()
1    EXTRACT1D   BinTableHDU     17   2048R x 2C   [1E, 1E]
2    EXTRACT1D   BinTableHDU     17   2048R x 2C   [1E, 1E]
3    EXTRACT1D   BinTableHDU     17   2048R x 2C   [1E, 1E]
4    EXTRACT1D   BinTableHDU     17   2048R x 2C   [1E, 1E]
5    EXTRACT1D   BinTableHDU     17   2048R x 2C   [1E, 1E]
6    EXTRACT1D   BinTableHDU     17   2048R x 2C   [1E, 1E]
7    EXTRACT1D   BinTableHDU     17   2048R x 2C   [1E, 1E]
8    EXTRACT1D   BinTableHDU     17   2048R x 2C   [1E, 1E]
9    EXTRACT1D   BinTableHDU     17   2048R x 2C   [1E, 1E]
10   EXTRACT1D   BinTableHDU     17   2048R x 2C   [1E, 1E]
11   EXTRACT1D   BinTableHDU     17   2048R x 2C   [1E, 1E]
12   EXTRACT1D   BinTableHDU     17   2048R x 2C   [1E, 1E]
13   EXTRACT1D   BinTableHDU     17   2048R x 2C   [1E, 1E]
14   EXTRACT1D   BinTableHDU     17   2048R x 2C   [1E, 1E]
15   EXTRACT1D   BinTableHDU     17   2048R x 2C   [1E, 1E]
16   EXTRACT1D   BinTableHDU     17   2048R x 2C   [1E, 1E]
17   EXTRACT1D   BinTableHDU     17   2048R x 2C   [1E, 1E]
18   EXTRACT1D   BinTableHDU     17   2048R x 2C   [1E, 1E]
19   EXTRACT1D   BinTableHDU     17   2048R x 2C   [1E, 1E]
20   EXTRACT1D   BinTableHDU     17   2048R x 2C   [1E, 1E]
21   EXTRACT1D   BinTableHDU     17   2048R x 2C   [1E, 1E]
22   EXTRACT1D   BinTableHDU     17   2048R x 2C   [1E, 1E]
```

```
129  EXTRACT1D   BinTableHDU     17   2048R x 2C   [1E, 1E]
130  EXTRACT1D   BinTableHDU     17   2048R x 2C   [1E, 1E]
131  EXTRACT1D   BinTableHDU     17   2048R x 2C   [1E, 1E]
132  EXTRACT1D   BinTableHDU     17   2048R x 2C   [1E, 1E]
133  EXTRACT1D   BinTableHDU     17   2048R x 2C   [1E, 1E]
134  EXTRACT1D   BinTableHDU     17   2048R x 2C   [1E, 1E]
135  EXTRACT1D   BinTableHDU     17   2048R x 2C   [1E, 1E]
136  EXTRACT1D   BinTableHDU     17   2048R x 2C   [1E, 1E]
137  EXTRACT1D   BinTableHDU     17   2048R x 2C   [1E, 1E]
138  EXTRACT1D   BinTableHDU     17   2048R x 2C   [1E, 1E]
139  EXTRACT1D   BinTableHDU     17   2048R x 2C   [1E, 1E]
140  EXTRACT1D   BinTableHDU     17   2048R x 2C   [1E, 1E]
141  EXTRACT1D   BinTableHDU     17   2048R x 2C   [1E, 1E]
142  EXTRACT1D   BinTableHDU     17   2048R x 2C   [1E, 1E]
143  EXTRACT1D   BinTableHDU     17   2048R x 2C   [1E, 1E]
144  EXTRACT1D   BinTableHDU     17   2048R x 2C   [1E, 1E]
```

What do you notice about the way that this data is stored compared to the 2D images?

In [10]:
```
#Let's take a look at the primary header information
hdulist1D[0].header
```

```
Out[10]: SIMPLE  =                    T / Fits standard
         BITPIX  =                   16 / Bits per pixel
         NAXIS   =                    0 / Number of axes
         EXTEND  =                    T / File may contain extensions
         IRAF-TLM= '2016-05-29T04:16:56' / Time of last modification
         DATE    = '2016-05-29T04:16:56.00' / Date this file was created (UTC
         )
         FILETYPE= 'UNCALIBRATED'       / Type of data in the file
         TELESCOP= 'JWST    '           / Telescope used to acquire the data

                 Programmatic information

         TITLE   = 'Exoplanet characterization in the JWST era' / Proposal ti
         tle
         PI_NAME = 'John Mather'        / Principal investigator name

                 Instrument configuration information

         INSTRUME= 'NIRISS  '           / Instrument used to acquire the data

                 Exposure parameters

         NINTS   =                  144 / Number of integrations in exposure
         NGROUPS =                    7 / Number of groups in integration
         NFRAMES =                    1 / Number of frames per group
         GROUPGAP=                    0 / Number of frames dropped between gr
         oups
         NSAMPLES=                    1 / Number of A/D samples per pixel
         TSAMPLE =                   10 / Time between samples (microsec)
         TFRAME  =              5.49130 / Time between frames (sec)
         TGROUP  =              5.49130 / Time between groups (sec)
         EFFINTTM=              32.9478 / Effective integration time (sec)
         EFFEXPTM=              4744.48 / Effective exposure time (sec)
         NRSTSTRT=                    1 / Number of resets at start of exposu
         re

                 Subarray parameters

         SUBARRAY= 'SUBSTRIP256'        / Subarray used
         SUBSTRT1=                    1 / Number of pixels in axis 1 directio
         n
         SUBSIZE1=                 2048 / Number of pixels in axis 1 directio
         n
         SUBSIZE2=                  256 / Number of pixels in axis 2 directio
         n
         FASTAXIS=                    2 / Fast readout axis direction
         SLOWAXIS=                    1 / Slow readout axis direction

                 Aperture information

         APERNAME= 'NIS-SOSSTA'         / science aperture used
```

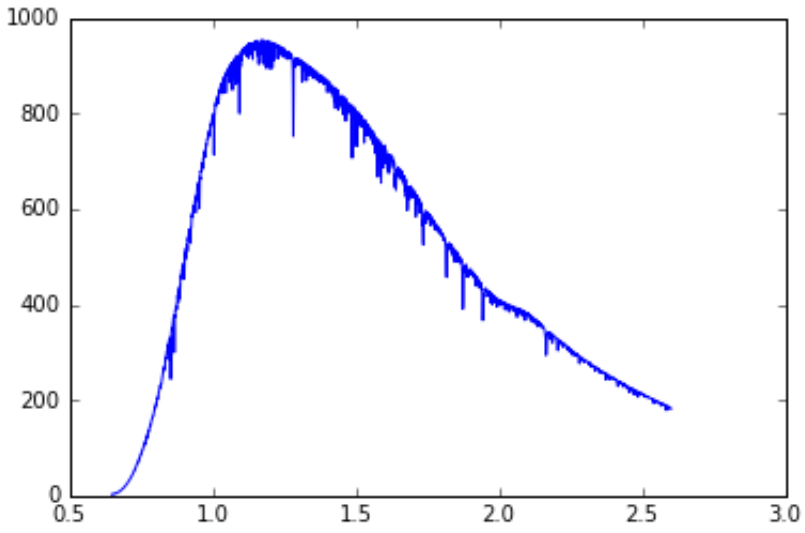
In [11]:
```
#Let's look at the first spectrum
spec1 = hdulist1D[1]
spec1.header
```

Out[11]:
```
XTENSION= 'BINTABLE'           / binary table extension
BITPIX  =                    8 / array data type
NAXIS   =                    2 / number of array dimensions
NAXIS1  =                    8 / length of dimension 1
NAXIS2  =                 2048 / length of dimension 2
PCOUNT  =                    0 / number of group parameters
GCOUNT  =                    1 / number of group
TFIELDS =                    2 / number of table fields
EXTNAME = 'EXTRACT1D'          / extension name
EXTVER  =                    1 / extension value
JDMID   =       2458710.46502 / Julian date at integration midpoint
TFORM1  = '1E      '           /Real*4 (floating point)
TTYPE1  = 'wavelength'         /Label for column 1
TUNIT1  = 'micron  '           /Units of column 1
TFORM2  = '1E      '           /Real*4 (floating point)
TTYPE2  = 'countrate'          /Label for column 2
TUNIT2  = 'DN/s    '           /Units of column 2
```

In [12]:
```
#Let's extract the JD from the header and read in the wavelength and c
ount rate
jd = spec1.header['JDMID']
wavelength = spec1.data.field(0)
countrate = spec1.data.field(1)
```

In [13]:
```
#Let's plot the first (in time) spectrum
plt.plot(wavelength, countrate)
```

Out[13]:
```
[<matplotlib.lines.Line2D at 0x12d016e50>]
```



Now that you see how to read-in/extract the first spectrum you are ready to FIND THE PLANET IN YOUR DATA! Your first goal is to first produce a 'white-light' light curve (relative flux as a function of time). The white-light light curve will be a quick look product for the time-series branch of the JWST pipeline. Then you should try to produce light curves for various wavelengths (unbinned and binned).