PHYS-4007/5007: Computational Physics

Python Tutorial
Making Plots of Spectra in Python

1 Introduction: Download the Sample Code and Data

Log into your Linux account and open the web browser. Open the course web page at:

```
http://faculty.etsu.edu/lutter/courses/phys4007/
```

Scroll down to the Useful Python Programs web page and click the link. Once on this Python web page, click on the "specplot.py" link in the table under the Python Plotting Tutorials and Programs section heading. This will show the program in the web browser GUI.

Now, open a terminal window and change directories to your python subdirectory. Create a new subdirectory of the name specplot and cd to that directory. Open a new file called "specplot.py" (without the double quotes) with emacs at the Linux prompt:

```
emacs specplot.py &
```

where the ampersand symbol (&) puts the emacs session in background so that you can still enter commands at the Linux prompt. Go back to the web browser GUI highlight all of the text in the GUI, and copy and paste this program into the emacs GUI. Finally, save this file and exit the emacs GUI.

Go back to the Python Plotting Tutorials and Programs web page and repeat the actions above for the following spectrum data files, saving each into their own <u>individual</u> ASCII files.

Filename	**Star**	**Wavelength Area**
sun981.wvc	The Sun	Ca II H & K lines
sun997.wvc	The Sun	Ca II IR-triplet
betleo634.wvc	β Leo	Ca II H & K lines
betleo540.wvc	β Leo	Hα
betleo543.wvc	β Leo	Ca II IR-triplet
alpboo638.wvc	α Boo	Ca II H & K lines
alpboo560.wvc	α Boo	Ca II IR-triplet
delvir636.wvc	δ Vir	Ca II H & K lines
uuaur620.wvc	UU Aur	Ca II IR-triplet

Once you have saved each of these spectrum files, exit the emacs GUI session and proceed to the next section below.

2 Examining the Python Code

Let's now examine the code an highlight what it is doing. Note that this code needs to be ran using python3.

```
# This program will read in one of my spectrum files obtained
# with the McMath-Pierce Solar Telescope on Kitt Peak, AZ and
# make a plot of the spectrum.  In the future, I will add a
# line ID function and an equivalent width of a spectral line
# function.  This program will also investigate the use of
# the Matplotlib Pyplot savefig function to make hardcopy
# files of the plots without going through the interactive
# GUI.

# When running this program, one needs to pass the name of
# the spectrum data file (in this example ‘sun981.wvc’ to be
# read and plotted:
#       python3 specplot sun981.wvc

# Always include these next import commands.

import sys
import numpy as np
import matplotlib.pyplot as plt

# Retrieve the data filename and make a root-name for the
# plot output files.  Then make an array of plot output
# filenames.  Assume a maximum of 20 plot files.

nfmax = 20  # Maximum number of encapsulated postscript files.

if (len(sys.argv) == 2):
        spcname = sys.argv[1]

iperiod = spcname.find('.')
rootname = spcname[0:iperiod]
fsuffix = '.eps'    # Assume encapsulated postscript files.
```

```
# Make filenames for the postscript plot files.
plname = []

for i in range(0, nfmax):
    plname.append(rootname+str(i).zfill(2)+fsuffix)

# Read in the data from the spectrum file.

rdline = ''
starname = ''

fspc = open(spcname, 'r')

# Look for the name of the object that was observed.

while rdline != '  \n':
    rdline = fspc.readline()
    qstar = rdline.find('Star:')
    if qstar > -1:
        ieql = rdline.find(' = ')
        if ieql > 0:
            starname = rdline[7:ieql]

rdline = ''
sodate = ''
snccd = ''

# Look for the obs date, CCD #, and number of data points in the spectrum.

while rdline != '  \n':
    rdline = fspc.readline()
    qnpix = rdline.find('Number of pixels:')
    qdate = rdline.find('Obs-Date:')
    qccdn = rdline.find('CCD Picture Number:')
    slen = len(rdline)
    if qnpix > -1:
        snp = rdline[19:slen]
        np = int(snp)
    if qccdn > -1:
        snccd = rdline[qccdn+20:slen]
    if qdate > -1:
        sodate = rdline[11:22]

# Look for the number of data points in the spectrum.
```

```
rdline = ''
sfocus = ''

while rdline != '  \n':
    rdline = fspc.readline()
    qfocus = rdline.find('Telescope focus:')
    if qfocus > -1:
        sfocus = rdline[24:31]
        fwhm = float(sfocus)

rdline = ''

while rdline != '> Wavelength (Angstroms)\n':
    rdline = fspc.readline()

# Read in the wavelength data.

qaduflux = -1
swave = ''

while qaduflux == -1:
    rdline = fspc.readline()
    qaduflux = rdline.find('ADU-Flux:')
    if qaduflux == -1:
        swave = swave + rdline

# Read in the flux data.

qston = -1
sflux = ''

while qston == -1:
    rdline = fspc.readline()
    qston = rdline.find('Signal-to-Noise:')
    if qston == -1:
        sflux = sflux + rdline

fspc.close()

print('Object observed: "'+starname+'"')
print('Date of Observation: '+sodate)
print('CCD Picture #: '+snccd)
print('Number of pixels in spectrum: ', np)
print('Telescope FWHM: '+sfocus+' Angstroms')
```

```
# Convert wavelengths to floats.

sswave = swave.split()
wvlen = len(sswave)
wave = []

for i in range(0, wvlen):
    wave.append(float(sswave[i]))

# Convert fluxes to floats.

ssflux = sflux.split()
fllen = len(ssflux)
flux = []

for i in range(0, fllen):
    flux.append(float(ssflux[i]))

# Show the spectrum.

plt.plot(wave, flux, 'k-')
plt.xlabel('Wavelength (A)', labelpad=10)
plt.ylabel('Flux (ADU)', labelpad=10)
plt.title(starname+' Observation')
plt.figtext(0.18, 0.8, 'CCD Picture #'+snccd)
plt.figtext(0.7, 0.83, sodate)

plt.show()
```

Remember that any text written after the pound-sign (#) symbol tells Python that the following text is a comment. The first three executable lines of the code imports the sys and the NumPy libraries followed by the pyplot utility from the Matplotlib library and relabels the last two as np and plt respectively. Note that the sys library allows additional text to be passed from Linux into the code using the sys.argv[] function.

I will highlight each part of this code verbally during the tutorial session.

3 Running This Python Spectrum Code

For this code that you have saved on your Linux account (*i.e.*, specplot.py), make sure that you are in the subdirectory where you saved this file and the spectrum files and enter the following from the Linux prompt:

```
python3 specplot.py sun981.wvc
```

You will see five line of text printed to your terminal window letting you know the target observed, the date of the observation, the CCD picture number, the number of pixels (*i.e.*, data points) in the spectrum, and the Full-Width-at-Half-Max (FWHM) of the spectrum. Following this, a GUI pop up containing a plot of the spectrum (see Figure 1). At the bottom of this GUI, you will see 7 control buttons that will do a variety of different options to this figure. Here we will just use the last button which will allow us to make a 'hardcopy' version of this figure. Click this button now. You will now see a second GUI pop up with the title "Save the figure". Follow this sequence to make an encapsulated postscript file of this figure:

- From the window containing all of the subdirectories in your login directory, scroll over to the directory where you want to save this figure file (typically this will be the directory where your Python code is located and double clikck that directory.
- Note that the Potable Network Graphics (*.png) format is the default for saved picture files. Go to the Filies of type: pulldown menu bar and select the second entry, Encapsulated Postscript (*.eps).
- Note however that the filename is still listed as a '.png' file. Change this name to sun981.eps and click the Save button.
- Following this, quit this plot GUI by clicking the red-x button on the top-left of this GUI.

Now repeat these steps for each of the spectrum files that you have downloaded. For each spectrum, use the root filename of the spectrum (*e.g.*, sun997 for sun997.wvc, betleo634 for betleo634.wvc, etc.) such that you have unique filenames for all of the '.eps' files you have created.

Once you have finished running this code, you can view these files on the computer screen by using the Linux Ghostview command:

```
gv sun981.eps
```

From the Ghostview GUI, you can carry out a variety of options pertaining to the figure in the GUI. For now, just quit the GUI by depressing the 'File' button at the top left of the GUI and scroll down to 'Quit' and release the mouse button (note, keep depressing the right mouse button until you have selected the 'Quit' option).

Feel free to investigate all of the figures that you have saved to your subdirectory.

For our next tutorial session, we will be making modifications to this specplot.py code to do a variety of other things to these spectrum plots.