ABSTRACT

This paper presents Part II of my RIA training data project. This paper discusses the results of HST ACS photometry of 47 Tuc as observed in F435W (B), F606W (V) and F814W (I).

Introduction

47 Tuc (NGC 104) is the second largest and second brightest globular cluster in the skies (Frommert 2006). It serves as a good candidate for the purpose of a photometry training exercise as well as creating a color magnitude diagram (CMD) due to the large number of stars it contains. The goal of this training exercise is to become familiar with photometry using the Hubble Space Telescope (HST) and to create a CMD of 47 Tuc.

This photometry project involves several steps in order to produce the final CMD. These steps include retrieving HST Advanced Camera for Surveys (ACS) data from the archive, identifying stars on the image to use for photometry, computing aperture photometry, applying aperture corrections, computing photometric transformations and plotting the CMD. Each of these steps are discussed in more detail in the following sections.
Data & Observations

The dataset of 47 Tuc used for this project was taken in May 2002 with HST WFC/ACS (Proposal ID 9018, PI Guido De Marchi). The first step is to retrieve the raw data product and the calibrated data products of datasets j8c0a1011, j8c0c1011 and j8c0d1011 from MAST\(^1\) (Multimission Archive at STScI). Knowing the dataset numbers is all that is needed to retrieve this data and this can be entered in the “Dataset” entry on the HST search form in MAST. The raw data files were re-calibrated with the IRAF task CALACS using the reference files on centralized storage. In order to run CALACS the PHOTCORR switch needs to be changed from PERFORM to OMIT using the IRAF task HEDIT. CALACS needs to be run on each ∗.asn file separately in order to create the ∗.flt’s and ∗.crj’s.

Using the IRAF task GSTAT on the calibrated raw data and on the calibrated data product from the archive produces identical results. Table 1 shows the results from GSTAT.

<table>
<thead>
<tr>
<th>Image</th>
<th>NPIX</th>
<th>MEAN</th>
<th>STDDEV</th>
<th>MIN</th>
<th>MAX</th>
</tr>
</thead>
<tbody>
<tr>
<td>j8c0a1011.crj.fits</td>
<td>8388608</td>
<td>9.7843</td>
<td>526.904</td>
<td>-3876.86</td>
<td>135092.</td>
</tr>
<tr>
<td>j8c0c1011.crj.fits</td>
<td>8388608</td>
<td>37.139</td>
<td>1544.33</td>
<td>-2691.99</td>
<td>174208.</td>
</tr>
<tr>
<td>j8c0d1011.crj.fits</td>
<td>8388608</td>
<td>42.561</td>
<td>1482.98</td>
<td>-3626.64</td>
<td>133633.</td>
</tr>
</tbody>
</table>

Table 1: GSTAT results.

Various keywords in the image header can be queried in IRAF by using the HSELECT task. Table 2 shows the results of querying selected keywords in the image header such as the filter, exposure time, PHOTFLAM and the ST magnitude zero point.

In order to make calibration corrections it is important to know some information about the units of the data. Raw ACS images are in units of Data Numbers (DN), calibrated images are in units of electrons and drizzled images are in units of electrons/second. The pipeline calibration tasks do not alter the units of the pixels in the image. Instead they calculate and write the inverse sensitivity conversion factor (PHOTFLAM) and the ST magnitude scale zero point (PHOTZPT) into header keywords in the calibrated data. The PHOTFLAM is defined as the mean flux density $Flam$ in units of erg cm$^{-2}$s$^{-1}$Å$^{-1}$ that produces 1 count per second in the HST observing mode (HST Data Handbook for ACS 2006). The PHOTFLAM value can be found either in the image header of the ∗.drz image or it can be derived using the bandpar task in SYNPHOT. Table 3 shows the results of comparing the PHOTFLAM values

\(^1\)http://archive.stsci.edu/
Table 2: *HSELECT* results.

from the header with those derived using the *bandpar* task in SYNPHOT. The smallest difference is seen in F435W with a difference of 0.074 erg cm\(^{-2}\)s\(^{-1}\)Å\(^{-1}\), for F606W the difference is 0.096 erg cm\(^{-2}\)s\(^{-1}\)Å\(^{-1}\) and the largest difference between the values is 0.11 erg cm\(^{-2}\)s\(^{-1}\)Å\(^{-1}\) for F814W.

Table 3: *PHOTFLAM* values

Using the *PHOTFLAM* value from the header the zeropt is computed. For example, for F435W\(_{drz}.fits\) the zero point is:

\[
zpt = -2.5 \times \log(3.1096343E^{-19}) - 21.1 = 25.168
\]  

(1)

For F606W\(_{drz}.fits\) the zero point is:

\[
zpt = -2.5 \times \log(7.7664046E^{-20}) - 21.1 = 26.674
\]  

(2)
For F814W_drz.fits the zero point is:

\[
zpt = -2.5 \times \log(6.9255574E^{-20}) - 21.1 = 26.799 \tag{3}\]

Using the IRAF task IMHEADER on the *.drz images allows one to see that not each image is identical due to filter-specific distortions. Table 4 shows the IMHEADER results which show a slight change in the image size due to filter-specific distortions.

<table>
<thead>
<tr>
<th>Image</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>j8c0a1011_drz.fits[1]</td>
<td>[4219,4233][real]:</td>
</tr>
<tr>
<td>j8c0c1011_drz.fits[1]</td>
<td>[4219,4233][real]:</td>
</tr>
<tr>
<td>j8c0d1011_drz.fits[1]</td>
<td>[4218,4232][real]:</td>
</tr>
</tbody>
</table>

Table 4: IMHEADER results.

Each image then needs to be multiplied by the exposure time so that the IRAF task PHOT will compute the photometric errors correctly. At the same time, I want to make the images the same size. This can be done by using the IRAF task IMCALC.

\begin{verbatim}
imcalc j8c0a1011\_drz.fits[1][1:4218, 1:4232] f435w\_drz.fits 'im1*60.'
imcalc j8c0c1011\_drz.fits[1][1:4218, 1:4232] f435w\_drz.fits 'im1*45.'
imcalc j8c0d1011\_drz.fits[1][1:4218, 1:4232] f435w\_drz.fits 'im1*60.'
\end{verbatim}

I displayed the cropped images and measured the position of the stars to be sure they are roughly equal from one filter to the next. I determined that the stars are indeed roughly equal from each filter.

**Aperture Photometry**

A python script was created to run the photometry of 47 Tuc. Appendix A includes a python photometry script I created for this assignment. In order to perform photometry the stars in the field need to be identified. The next step involves creating a combined color image in order to find stars in the field. This simplifies matters because I only have one source list and won’t need to cross correlate my lists for each filter after doing photometry. By comparing the values of PHOTFLAM I can determine how to weight the images when coadding (Step 6A in my python code). The images can be added by using IMCALC for example,
The stars in the color.drz.fits image were detected with the IRAF task DAOFIND (Step 6B in my python code). The DAOFIND parameters used are shown in Table 5. Using these parameters DAOFIND found 4,460 objects however, it should be noted that some of these “objects” include saturated points. Because DAOFIND was having trouble with excluding saturated points and it was finding multiple points within a single star it was suggested to me to try to use SExTractor \(^2\) to determine the stars on the image. This proved to be very worthwhile because SExTractor did a much better job at finding stars and excluding saturated points than DAOFIND. My script includes using DAOFIND as an example of how this can be done within a script however, in the end I did not use the DAOFIND results and instead I used the SExTractor results. Figure 6 shows the parameters used in the default.sex file used for SExTractor. Using these parameters SExTractor found 9,253 stars. Using this list produced a much deeper and cleaner CMD than the DAOFIND results.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>findpar.thresh</td>
<td>50</td>
</tr>
<tr>
<td>findpar.sharplo</td>
<td>0.2</td>
</tr>
<tr>
<td>findpar.sharphi</td>
<td>0.8</td>
</tr>
<tr>
<td>findpar.roundlo</td>
<td>-1.0</td>
</tr>
<tr>
<td>findpar.soundhi</td>
<td>1.0</td>
</tr>
<tr>
<td>datapar.sigma</td>
<td>3.6</td>
</tr>
<tr>
<td>datapar.fwhmpsf</td>
<td>2.0</td>
</tr>
</tbody>
</table>

Table 5: DAOFIND Parameters.

The next step is to compute aperture photometry on a list of stars using the PHOT task in IRAF (Step 7 in my python code). The parameters used for F435W are shown in Table 7.

Using the task TDUMP I can extract the following columns from the APPHOT output including x,y,sky,flux3,mag3,err3,flux10,mag10 and err10 (Step 8 in python code). Next I merge the photometry table for the 3 filters using the task TMERGE and change the column names to something meaningful using the task TCHOL (Step 9 in my python code). Changing the column names is very useful for keeping track of columns as well as for needing to call those columns for any sort of calculation which we will need to do later in this example.

\(^2\)http://sextractor.sourceforge.net/
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>DETECT_MINAREA</td>
<td>5.0</td>
</tr>
<tr>
<td>DETECT_THRESH</td>
<td>5.0</td>
</tr>
<tr>
<td>ANALYSIS_THRESH</td>
<td>5.0</td>
</tr>
<tr>
<td>DEBLEND_NTHRESH</td>
<td>32</td>
</tr>
<tr>
<td>DEBLEND_MINCONT</td>
<td>0.01</td>
</tr>
<tr>
<td>CLEAN_PARAM</td>
<td>1.0</td>
</tr>
</tbody>
</table>

Table 6: SExTractor Parameters for extraction in default.sex

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>datapar.exposure</td>
<td>exptime</td>
</tr>
<tr>
<td>datapar.gain</td>
<td>ccdgain</td>
</tr>
<tr>
<td>datapar.sigma</td>
<td>3.6</td>
</tr>
<tr>
<td>centerpars.calgorithm</td>
<td>centroid</td>
</tr>
<tr>
<td>centerpars.cbox</td>
<td>10</td>
</tr>
<tr>
<td>centerpars.maxshift</td>
<td>6</td>
</tr>
<tr>
<td>fitskypars.salgorithm</td>
<td>mode</td>
</tr>
<tr>
<td>fitskypars.annulus</td>
<td>10</td>
</tr>
<tr>
<td>fitskypars.dannulus</td>
<td>5</td>
</tr>
<tr>
<td>fitskypars.skyvalue</td>
<td>0</td>
</tr>
<tr>
<td>photpars.apertures</td>
<td>3,10</td>
</tr>
<tr>
<td>photpars.zmag</td>
<td>25.1682</td>
</tr>
<tr>
<td>phot.image</td>
<td>f435_drz.fits</td>
</tr>
<tr>
<td>phot.coords.</td>
<td>color_drz.fits</td>
</tr>
</tbody>
</table>

Table 7: PHOT Parameters.

**Aperture Correction**

Aperture corrections need to be applied to my photometry. Accurate aperture corrections are a function of time and location on the chip and should be derived for each frame. First we need to compute the apcor for each filter which is simply mag3 - mag10 (Step 10 in my python code). Next we find the ac05 value (eventually needed to calculate STmag) which can be found “by eye” by looking at a graph of the apcor of each filter versus the mag3 of that filter and estimating the center of the graph. This can be done in IRAF with
Fig. 1.— 435_apcor vs. mag3.

the task **SGRAPH**. For example,

```
sgraph ‘phot_all.tab 435_apcor mag3_435’ wl=-0.5 wr=1 wt=15 wb=23 pointmode=yes
```

Figures 1, 2 and 3 show the results of using **SGRAPH**. Table 8 compares the apcor values estimated from the graphs to what is predicted from the WFC encircled energy from Sirianni (2005).

<table>
<thead>
<tr>
<th>Filter</th>
<th>Siriani (2005) Apcor Value</th>
<th>My Apcor Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>F415W</td>
<td>0.131</td>
<td>0.15</td>
</tr>
<tr>
<td>F606W</td>
<td>0.124</td>
<td>0.14</td>
</tr>
<tr>
<td>F814W</td>
<td>0.156</td>
<td>0.2</td>
</tr>
</tbody>
</table>

Table 8: *Comparison of aperture correction measured from 3 pixels to 0.5”* (ac05)
In order to go from aperture photometry to absolute magnitude the STmag needs to be calculated (Step 12 of my python code). The STmag is defined as the magnitude system based on constant flux per unit wavelength. STmag can be calculated using the following equation:

$$STMAG = zpt - 2.5\log(\text{flux/exptime}) - 2.5\log(\text{photflam}) - \text{ac05} - \text{AC05} - \text{CTE}$$  (4)

where flux is the counts in aperture, exptime is the exposure time, photflam is the inverse sensitivity (erg sec$^{-1}$ cm$^{-2}$ Å$^{-1}$ per DN), ac05 is the apcorr from r = 3 to 10 (0.5") (average value measured for each image), AC05 is apcorr from r=10 (0.5") to infinity (filter specific value taken from ACS table) and CTE is the charge transfer efficiency loss. The next step is to apply the aperture correction measured from 3 pixels to 0.5" (ac05) as well as the aperture correction from 0.5" to infinity (AC05) using Table 5 from Sirianni (2005). These values are listed in Table 9 of this document.

I have also included the CTE correction in this photometry exercise. The CTE per pixel is defined as the fraction of charge transferred from one pixel to the next during readout. In an ideal CCD, the CTE would be exactly 1.0, with no charge being lost. However, due to imperfections on a CCD which are caused by either the manufacturing process or the
Filter | AC05 from Siriani (2005)  
--- | ---  
F415W | 0.107  
F606W | 0.088  
F814W | 0.087

Table 9: AC05 Values from Siriani (2005)

The space radiation environment can act as charge traps. The amount of charge lost per pixel is typically a very small fraction of the total charge. The total CTE becomes increasingly significant as larger CCD arrays are manufactured and flown in space (Chiaberge 2005). The CTE is calculated using the following equation:

$$
\Delta mag = 10^A \times SKY^B \times FLUX^C \times Y/2000 \times (MJD - 52333)/365
$$

where $A=0.15 \ (\pm 0.04)$, $B=-0.25 \ (\pm 0.01)$,$C=-0.44 \ (\pm 0.02)$. SKY is the sky in electrons measured near the star, FLUX is the flux of the star in a 3 pixel aperture radius, Y is the y-transfers where $Y = y$ for WFC2 and $Y = 4096 - y$ for WFC1. It should be noted that MultiDrizzle (which was run through the pipeline) has removed the sky in the drizzled...
Fig. 4.— CTE vs. Y position on CCD of WFC/ACS.

product but not in the *flt’s or *crj products. For this exercise we use the *flt’s and *crj sky values by plotting an image histogram on the *flt’s which gives a value of approx 5.5 electrons for SKY (this is the exactly the case for F814W).

Applying the CTE, however, only caused a very small difference to the overall photometry. This is shown in Figure 4 which plots the CTE vs. Y position on the CCD. The biggest change occurs in the center of the CCD chip just slightly under 0.01 CTE. Thus, the overall effect of the CTE is very small, however, the effect is probably much greater now as the camera has been degraded by the high-radiation space environment longer.
The next step is to compute the transformation to VEGAmag, using the spectrum of an F5V star. VEGAmag is the standard magnitude system for which Vega has magnitude 0 at all wavelengths. This can be done by executing this command for both the STmag and VEGAmag:

```
calphot "band(acs,wfc1,f435w)" /Volumes/grp/hst/cdbs/grid/bz77/bz$\_20.fits"
```

The correction for each filter is thus the VEGAmag - STmag using the spectrum of an F5V star as seen below:

\[ F435W = -12.5389 + 13.163 = +0.624 \]  \hspace{0.5cm} (6)
\[ F606W = -13.096 + 12.8409 = -0.255 \]  \hspace{0.5cm} (7)
\[ F814W = -13.5356 + 12.2734 = +1.262 \]  \hspace{0.5cm} (8)

This correction is then applied to all of my STmags and the new VEGAmag value now represents my calibrated magnitudes. These magnitudes can now be used to create my CMD (the last part of Step 12 - 14 in my python script writes out files for plotting). These files can be read into a plotting program and my CMD is now complete.

**Results**

Figure 5 and Figure 6 shows the CMD of 47 Tuc in V - I vs. I and the B - V vs. V filters respectively. There are a number of basic features we can see from the CMD including the main sequence (MS), main sequence turn-off (MSTO), blue stragglers (BS), the red giant branch (RGB) and even some white dwarfs (WD). One of the key features of globular clusters is the well-defined main sequence extending from the turn-off to fainter magnitudes and redder colors. The sharp main-sequence turn-off is typical of globular cluster color-magnitude diagrams, indicating that the stars within an individual globular cluster all formed at roughly the same time. The turnoff of the cluster is at \(~17\) in V and \(~0.5\) in V - I which agrees with the CMD from Piotto et al. (2004). The narrowness of the CMD also indicates that all the stars in the globular cluster have very similar chemical composition. The RGB extends from the subgiants which connect it to the main sequence to brighter magnitudes and redder colors until the tip of the RGB is reached. One oddity of the CMD of globular clusters is the presence of ‘blue stragglers’ in which several are seen in my CMD of 47 Tuc. The blue stragglers give the appearance of being an extension of the main sequence beyond the turn-off point. Since BS lie along an extension of the MS in a region which is
brighter and bluer than the MSTO, they may undergo abnormal stellar evolution. The cause of this is not yet clearly known, but one of the leading hypotheses is that they are current or former binary stars that are in the process of merging or have already done so (Ferraro et al. 2003). Also, in the fainter magnitude and bluer colors of the CMD is the presence of WD’s.

From the CMD we can determine the cluster’s approximate age and evolutionary status and get a relative idea of the age of the cluster. Of key importance is the turn off point on the diagram where the cluster turns-off the main sequence. The further down the main sequence the turnoff point is, the older the cluster. Since the turnoff is $\sim 0.5$ V - I we know that the cluster is relatively old. In a paper by Kaluzny et al. (2007) they used spectroscopic observations of the eclipsing binary V228 to derive an upper limit to the age of 47 Tuc as 14 Gyr. This can be compared with another recent age estimate of the cluster by Gratton et al. (2003) with an estimated age of 10.8 Gyr found using standard isochrones and incorporating microscopic diffusion in their models. The CMD allows one to visually get a rough idea whether or not the cluster is young or old and in this case we can see on the CMD that 47 Tuc is an old cluster which is also verified in literature. Of course, it is not surprising that 47 Tuc is old given that globular clusters are among the oldest objects in our Galaxy and have been used to provide lower limits to the age of the Universe.

**Color Image**

I have created a color image of 47 Tuc using Adobe Photoshop and FITS Liberator. FITS Liberator is an Adobe Photoshop plugin which helps streamline the process of creating a color image. Another advantage of using FITS Liberator allows one to import a .fits file into Photoshop directly.

FITS Liberator allows you to select a function from the stretch function menu in order to scale the gray brightness levels in the image. The image that I created uses an ArcSin(x) stretch function. The basic idea is to combine our 3 images using Photoshop’s “layers” which permits multiple images to be combined in a single document. In this case we will “colorize” the image by applying a different hue to each layer. This is accomplished by “projecting” each layer onto the virtual screen of a displayed image using Photoshop’s “Screen layer blend” mode. You can assign a color and name to each layer using the “Layer properties”. In this case for the F434W image we apply a blue hue, F606W a green hue and F814W a red hue. The resulting color composite typically appears dull and a number of “Adjustment Layers” can be used to help improve the brightness, contrast and color. Either you can adjust the
Fig. 5.— CMD $V - I$ vs $I$

composite as a whole inserting adjustment layers or add adjustment layers to each layer individually.

In general the colors will not be “real” as in what we would see visually. But the colors do represent actual physical properties of the subject. Figure 7 shows the color image I have made of the WFC1 and Figure 8 shows the color image of WFC2. Since this image is huge I decided to crop the image and separate each chip. It should be noted that some of the resolution of the color image is lost in this LaTeX document. The cropped portions of the images are also huge so Figure 9 shows a zoomed in portion of the WFC2 color image. The image looks much better when zooming in and you can see much more detail. This cropped
image allows one to see the color a bit more. It always helpful to know some information about the image before creating the color image and in this case we know 47 Tuc is old so we expect to see more stars with a reddish hue. Taking this into consideration allowed me to get a feel of how to make adjustments and how to make my stars look more realistic. It should be kept in mind that some of this work can be subjective and because Photoshop allows you to make many adjustments one can play with these adjustment parameters for a very long time until the creator of the color image deems it “pretty”. 
Fig. 7.— *Color Image of WFC1.*

Fig. 8.— *Color Image of WFC2.*

**Acknowledgments**

Thanks to Jennifer Mack for providing this photometry exercise which has been insightful as well as a great experience to learn HST photometry. I would like to thank Brain York
for sharing his python wisdom. I would also like to thank Lisa Frattare for teaching me about FITS Liberator and working with me on the creation of my color image.
References

HST Data Handbook for ACS, Version 5.0 (Baltimore: STScI)
Mack, J., HST Photometry RIA Training Presentation, STScI
Appendix

A. Title

# Photometry Script
# Steps 6 - 14 of Photometry Training Excercise
# Goal: Create a CMD of HST ACS data as observed by filters F435W,F606W,and F814W.
# Tiffany Borders- borders@stsci.edu
# Space Telescope Science Institute
# Last Updated 10/13/08

# Load the packages
import pyraf,os
from numarray import *

from pyraf import iraf
from iraf import digiphot,apphot,daofind
from iraf import imgtools,imcalc
from iraf import noao,digiphot,apphot,phot
from iraf import ttools,tcalc
from iraf import ttools
from iraf import tmerge
from iraf import tchcol

# Multiply the images by the exposure time so that PHOT will compute the
# photometric errors correctly. At the same time this makes all the images the
# same size.
# Step 6A
iraf.imgtools.imcalc(
    input='j8c0a1011_drz.fits[1][1:4218,1:4232]',
    output='f435w_drz.fits',
    equals='im1*60')
iraf.imgtools.imcalc(
    input='j8c0c1011_drz.fits[1][1:4218,1:4232]',
    output='f606w_drz.fits',
    equals='im1*45')

iraf.imgtools.imcalc(
    input='j8c0d1011_drz.fits[1][1:4218,1:4232]',
    output='f814w_drz.fits',
    equals='im1*60')

# Coadding images so that we can run DAOFIND on 1 image
iraf.imgtools.imcalc(
    input='f435w_drz.fits,f606w_drz.fits,f814w_drz.fits',
    output='color_drz.fits',
    equals='(4*im1+im2+im3)/6')

# Run DAOFIND on the color image.
# Step 6B
iraf.unlearn(daofind)

iraf.apphot.daofind.findpar(
    thresh='50',
    sharplo='0.2',
    sharphi='0.8',
    roundlo='-1.0',
    roundhi='1.0')

iraf.apphot.daofind.datapar(
    exposure='exptime',
    gain='ccdgain',
    sigma='3.6',
    fwhmpsf='2.0')

iraf.apphot.daofind(
    image='color_drz.fits[1]',
    output='color_drz.cdt',
    inter='no',
    19
verify='no')

# Compute aperture photometry
#
phot.datapars.sigma = 3.6
phot.datapars.exposure = "EXPTIME"
phot.datapars.gain = "ccdgain"

phot.centerpars.calgorithm="centroid"
phot.centerpars.cbox=10.0
phot.centerpars.maxshift=6.0

phot.fitskypars.salgorithm="mode"
phot.fitskypars.annulus=10
phot.fitskypars.dannulus=5
phot.fitskypars.skyvalue=0.0

phot.photpars.apertures="3,10"

phot.photpars.zmag=25.1682267
phot(image="f435w_drz.fits[1]", coords="color_drz.cdt", interac="no")

phot.photpars.zmag=26.67445
phot(image="f606w_drz.fits[1]", coords="color_drz.cdt", interac="no")

phot.photpars.zmag=26.7988632
phot(image="f814w_drz.fits[1]", coords="color_drz.cdt", interac="no")

# Using the task 'tdump' to extract the following columns from the
# apphot output:
# x,y,sky,flux3,mag3,err3,flux10,mag10,err10
# Step 8
iraf.ttools.tdump(
  table='f435w_drz.fits1.mag.1',
  columns='c7,8,15,29,30,31,38,39,40',
datafile='f435w.phota')

20
iraf.ttools.tdump(
    table='f606w_drz.fits1.mag.1',
    columns='c7,8,15,29,30,31,38,39,40',
    datafile='f606w.phota')

iraf.ttools.tdump(
    table='f814w_drz.fits1.mag.1',
    columns='c7,8,15,29,30,31,38,39,40',
    datafile='f814w.phota')

# Merges the photometry table for the 3 filters
# Step 9
iraf.ttools.tmerge(
    intable='f435w.phota,f606w.phota,f814w.phota',
    outtable='phot_all.tab',
    option='merge')

# Changes the column names to something meaningful.
# For F435
iraf.ttools,tchcol(
    table='phot_all.tab',
    oldname='c1',
    newname='x_435',
    newfmt='%8.6g',
    newunits='')

iraf.ttools,tchcol(
    table='phot_all.tab',
    oldname='c2',
    newname='y_435',
    newfmt='%8.6g',
    newunits='')

iraf.ttools,tchcol(
    table='phot_all.tab',
    oldname='c1',
    newname='x_435',
    newfmt='%8.6g',
    newunits='')
oldname='c3',
newname='sky_435',
newfmt='\%6.5g',
newunits="")

iraf.ttools,tchcol(
  table='phot_all.tab',
  oldname='c4',
  newname='flux3_435',
  newfmt='\%11.9g',
  newunits="")

iraf.ttools,tchcol(
  table='phot_all.tab',
  oldname='c5',
  newname='mag3_435',
  newfmt='\%7.6g',
  newunits="")

iraf.ttools,tchcol(
  table='phot_all.tab',
  oldname='c6',
  newname='err3_435',
  newfmt='\%6.5g',
  newunits="")

iraf.ttools,tchcol(
  table='phot_all.tab',
  oldname='c7',
  newname='flux10_435',
  newfmt='\%11.9g',
  newunits="")

iraf.ttools,tchcol(
  table='phot_all.tab',
  oldname='c8',
  newname='mag10_435',
  newfmt='\%7.6g',
  newunits="")
newunits=""
"
iraf.ttools,tchcol(\
table='phot_all.tab',\
oldname='c9',\
newname='err10_435',\
newfmt='\%6.5g',\
newunits=""
"
# # For F606
#
iraf.ttools,tchcol(\n\table='phot_all.tab',\n\oldname='c10',\n\newname='x_606',\n\newfmt='\%8.6g',\n\newunits=""
)

iraf.ttools,tchcol(\n\table='phot_all.tab',\n\oldname='c11',\n\newname='y_606',\n\newfmt='\%8.6g',\n\newunits=""
)

iraf.ttools,tchcol(\n\table='phot_all.tab',\n\oldname='c12',\n\newname='sky_606',\n\newfmt='\%6.5g',\n\newunits=""
)

iraf.ttools,tchcol(\n\table='phot_all.tab',\n\oldname='c13',\n\newname='flux3_606',\n\newfmt='\%11.9g',\n\newunits=""
)
iraf.ttools,tchcol(
  table='phot_all.tab',
  oldname='c14',
  newname='mag3_606',
  newfmt='%.7g',
  newunits="")

iraf.ttools,tchcol(
  table='phot_all.tab',
  oldname='c15',
  newname='err3_606',
  newfmt='%.6.5g',
  newunits="")

iraf.ttools,tchcol(
  table='phot_all.tab',
  oldname='c16',
  newname='flux10_606',
  newfmt='%.11.9g',
  newunits="")

iraf.ttools,tchcol(
  table='phot_all.tab',
  oldname='c17',
  newname='mag10_606',
  newfmt='%.7.6g',
  newunits="")

iraf.ttools,tchcol(
  table='phot_all.tab',
  oldname='c18',
  newname='err10_606',
  newfmt='%.6.5g',
  newunits="")

#
# For F814
#
iraf.ttools.tchcol(
    table='phot_all.tab',
    oldname='c19',
    newname='x_814',
    newfmt='%8.6g',
    newunits=''
)

iraf.ttools.tchcol(
    table='phot_all.tab',
    oldname='c20',
    newname='y_814',
    newfmt='%8.6g',
    newunits=''
)

iraf.ttools.tchcol(
    table='phot_all.tab',
    oldname='c21',
    newname='sky_814',
    newfmt='%6.5g',
    newunits=''
)

iraf.ttools.tchcol(
    table='phot_all.tab',
    oldname='c22',
    newname='flux3_814',
    newfmt='%11.9g',
    newunits=''
)

iraf.ttools.tchcol(
    table='phot_all.tab',
    oldname='c23',
    newname='mag3_814',
    newfmt='%7.6g',
    newunits=''
)

iraf.ttools.tchcol(
    table='phot_all.tab',
    oldname='c24',
    newname='mag4_814',
    newfmt='%7.6g',
    newunits=''
)
oldname='c24',
newname='err3_814',
newfmt='%6.5g',
newunits="")

iraf.ttools,tchcol(
  table='phot_all.tab',
  oldname='c25',
  newname='flux10_814',
  newfmt='%11.9g',
  newunits="")

iraf.ttools,tchcol(
  table='phot_all.tab',
  oldname='c26',
  newname='mag10_814',
  newfmt='%7.6g',
  newunits="")

iraf.ttools,tchcol(
  table='phot_all.tab',
  oldname='c27',
  newname='err10_814',
  newfmt='%6.5g',
  newunits="")

# adds a column to phot_all.tab calculating apcor
# Step 10
iraf.ttools.tcalc(
  table='phot_all.tab',
  outcol='435_apcor',
  equals='mag3_435-mag10_435',
  colfmt='%6.5g')

iraf.ttools.tcalc(
  table='phot_all.tab',
  outcol='606_apcor',
  equals='mag3_606-mag10_606',
  colfmt='%6.5g')
equals='mag3_606-mag10_606',
colfmt='%.6g')

iraf.ttools.tcalc(\
table='phot_all.tab',\
outcol='814_apcor',\
equals='mag3_814-mag10_814',\
colfmt='%.6g')

# Calculating CTE, STMAG, VEGAMAG and writing out files for plotting
# Extract the MJD from the ACS header
# Step 12 - 14
mjd_435= []
mjd_606= []
mjd_814= []

my_tmp1 = imheader(images="f435w_drz.fits[0]", longheader="yes", Stdout=1)
for item in my_tmp1:
    if item[0:8] == "EXPSTART":
        mjd_435 = int(math.floor(float(item.strip().split()[1])))

my_tmp2 = imheader(images="f606w_drz.fits[0]", longheader="yes", Stdout=1)
for item in my_tmp2:
    if item[0:8] == "EXPSTART":
        mjd_606 = int(math.floor(float(item.strip().split()[1])))

my_tmp3 = imheader(images="f814w_drz.fits[0]", longheader="yes", Stdout=1)
for item in my_tmp3:
    if item[0:8] == "EXPSTART":
        mjd_814 = int(math.floor(float(item.strip().split()[1])))

phot_file=open('phot_all.tab','r')
my_data=phot_file.readlines()
phot_file.close()

# y_435 is at column 2
for line in my_data:
    if line[0] != "#" and line.find("INDEF") == -1:
        # skip comments as "INDEF's"
items = line.strip().split()
tmp_y_435.append(float(items[1]))
# testing value of last element in array
# don’t need to know how many elements in array
if tmp_y_435[-1] > 2048:
    tmp_y_435[-1] = 4096 - tmp_y_435[-1]
# append needs things to exist
if tmp_y_606[-1] > 2048:
    tmp_y_606[-1] = 4096 - tmp_y_606[-1]
tmp_y_814.append(float(items[19]))
if tmp_y_814[-1] > 2048:
    tmp_y_814[-1] = 4096 - tmp_y_814[-1]
tmp_flux_435.append(float(items[3]))
tmp_flux_606.append(float(items[12]))
tmp_flux_814.append(float(items[21]))
tmp_mag3_435.append(float(items[4]))
tmp_mag3_606.append(float(items[13]))
tmp_mag3_814.append(float(items[22]))
tmp_y_435_all.append(float(items[1]))
tmp_y_606_all.append(float(items[10]))
tmp_y_814_all.append(float(items[19]))

y_435 = asarray(tmp_y_435, 'Float64')
y_606 = asarray(tmp_y_606, 'Float64')
y_814 = asarray(tmp_y_814, 'Float64')

flux_435 = asarray(tmp_flux_435, 'Float64')
flux_606 = asarray(tmp_flux_606, 'Float64')
flux_814 = asarray(tmp_flux_814, 'Float64')

mag3_435 = asarray(tmp_mag3_435, 'Float64')
mag3_606 = asarray(tmp_mag3_606, 'Float64')
mag3_814 = asarray(tmp_mag3_814, 'Float64')

y_435_all = asarray(tmp_y_435_all, 'Float64')
y_606_all = asarray(tmp_y_435_all, 'Float64')
y_814_all = asarray(tmp_y_435_all, 'Float64')
A=-0.15
B=-0.25
C=-0.44
SKY=5.5

delmag_435 = 10**A * SKY**B * (flux_435/2)**C * y_435/2000 * (mjd_435 - 52333)/365

delmag_606 = 10**A * SKY**B * (flux_606/2)**C * y_606/2000 * (mjd_606 - 52333)/365

delmag_814 = 10**A * SKY**B * (flux_814/2)**C * y_814/2000 * (mjd_814 - 52333)/365

# print len(delmag_814)
print delmag_814

# need to get mag3's
stmag_435 = mag3_435 - 0.15 - 0.107 - delmag_435
stmag_606 = mag3_606 - 0.14 - 0.088 - delmag_606
stmag_814 = mag3_814 - 0.20 - 0.087 - delmag_814

vegamag_435 = stmag_435 + 0.624
vegamag_606 = stmag_606 - 0.256
vegamag_814 = stmag_814 - 1.262

b = vegamag_435
b_v = vegamag_435 - vegamag_606
# want to plot b_v, b

v = vegamag_606
v_i = vegamag_606 - vegamag_814
# want to plot v_i, v

# open files for plotting
vi_v = open("vi_v.dat",'w')
bv_b = open("bv_b.dat",'w')
cte_435 = open("cte435.dat","w")
cte_606 = open("cte606.dat",'w')
cte_814 = open("cte814.dat","w")

# write out files for plotting CMD
for j in range(len(b)):
    vi_v.write("%5.5g %5.5g\n" % (v_i[j], v[j]))
    # vi_v.write(str(v_i[j])+' '+str(v[j])+'\n')
    # bv_b.write(str(b_v[j])+' '+str(b[j])+'\n')
    bv_b.write("%5.5g %5.5g\n" % (b_v[j], b[j]))
    cte_435.write("%5.5g %5.5g\n" % (y_435_all[j], delmag_435[j]))
    cte_606.write("%5.5g %5.5g\n" % (y_606_all[j], delmag_606[j]))
    cte_814.write("%5.5g %5.5g\n" % (y_814_all[j], delmag_814[j]))

vi_v.close()
bv_b.close()
cte_435.close()
cte_606.close()
cte_814.close()

# The END