Physics 391

Patrick Selep

Laboratory 1

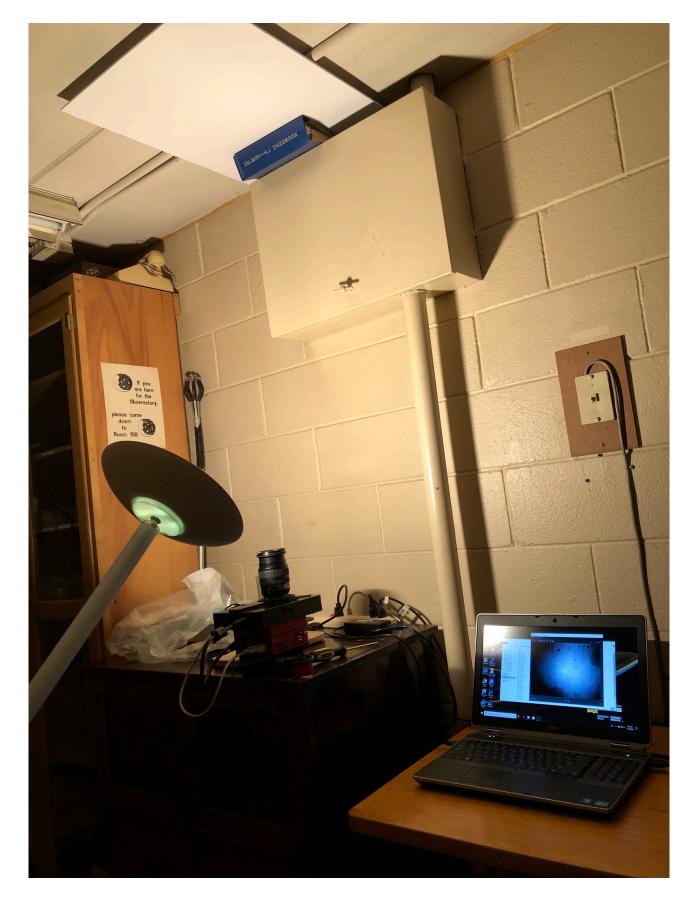
CCD Performance Characteristics

This laboratory covered assessing several performance characteristics of the CCD. Specifically, the Gain, Read Noise, Dark Current and Linearity were measured.

Experimental Apparatus

The SBIG STT 8300 was tested on a bench through a camera lens. A white board was mounted to the ceiling above the camera. The board illuminated with a halogen lamp with a dimmer control. Linearity was tested at varying exposure times up to saturation. Intensity was also varied as was the lighting arrangement. Bias and dark images were taken with the lense cap on. Bias images were taken with an exposure time of zero. Dark images were taken with varying exposure times up to 16 minutes.

Here is a picture of the arrangement of the equipment.



Performance Characteristics

Gain and Read Noise were calculated from a pair of matched flat and bias images. The formulas used to determine them are shown below:

Gain =
$$\frac{(\bar{F}_1 + \bar{F}_2) - (\bar{B}_1 + \bar{B}_2)}{\sigma_{F_1 - F_2}^2 - \sigma_{B_1 - B_2}^2}$$

Read Noise =
$$\frac{\text{Gain} \cdot \sigma_{B_1 - B_2}}{\sqrt{2}}$$

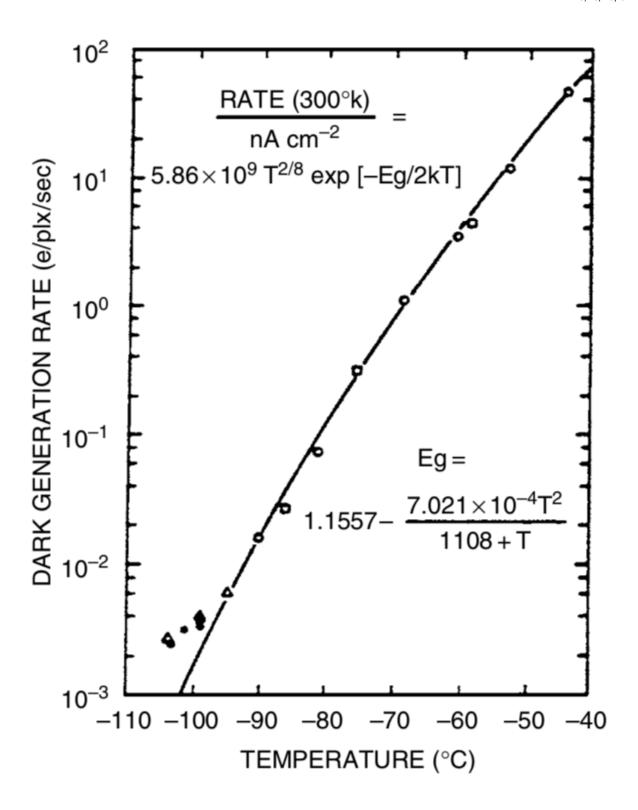
The results were as follows:

The Gain was found to be 0.379 e-/ADU compared to the published 0.37 e-/adu.

The Read Noise was found to be 10.663 e- rms compared to the published <10 e- rms.

The Dark Current was found to be 0.328 e-/pixel/sec at -1 C compared to 0.02 e-/pixel/sec

Dark Current is highly dependent on temperature. From the Handbook of CCD Astronomy "Typical values for properly cooled devices range from 2 electrons per second per pixel down to very low levels of approximately 0.04 electrons per second for each pixel."



The Linearity was found to be linear with and R = .999974 up to saturation.

The residuals of this plot show that counts are below predictions at high counts.

Manufacturer's Specifications

Model	STT-1603ME	STT-3200ME	STT-8300M
CCD	Kodak KAF-1603ME	Kodak KAF-3200ME	Kodak KAF-8300
Pixel Array	1536 x 1024 pixels @ 9u	2184 x 1510 pixels @ 6.8u	3326 x 2504 pixels
CCD Size	13.8 x 9.2 mm	14.85 x 10.26 mm	17.96 x 13.52 mm
Total Pixels	1.6 million	3.2 million	8.3 million
Full Well Capacity	100,000 e-	55,000 e-	25,500 e-
Dark Current	0.1e-/pixel/sec @ -20C.	0.06e-/pixel/sec at -20C.	0.02e-/pixel/sec at -15C.
Antiblooming	NABG only	NABG Only	1000X
Shutter	Mechanical, Even-illumination	Mechanical, Even-illumination	Mechanical, Even-illumination
Exposure	0.12 to 3600 seconds, 10ms	0.12 to 3600 seconds, 10ms	0.12 to 3600 seconds, 10ms
A/D Converter	16 bits	16 bits	16 bits
Gain	2.3e-/ADU	1.3e-/ADU	0.37e-/ADU
Read Noise	< 15e- rms	10e- rms	< 10e- rms
Binning Modes	1x1, 2x2, 3x3, 9x9, x n	1x1, 2x2, 3x3, 9x9, x n	1x1, 2x2, 3x3, 9x9, x n
Digitization Rate	10 Megapixels per second	8.33 Megapixels per second	10 Megapixels per second
Full Frame Download	< 1 second	< 1 second	< 1 second
Max Cooling Delta	-55C with air only	-55C with air only	-55C with air only
Temp. Regulation	±0.1°C	±0.1°C	±0.1°C
Power	12VDC at 3.5 amps	12VDC at 3.5 amps	12VDC at 3.5 amps
Interface	USB 2.0 and Ethernet	USB 2.0 and Ethernet	USB 2.0 and Ethernet
Computer Compatibility	Windows 32 / 64 bit	Windows 32 / 64 bit	Windows 32 / 64 bit
	Mac OSX, 3rd party Linux	Mac OSX, 3rd party Linux	Mac OSX, 3rd party Linux
Camera Body Size	4.9 x 4.9 x 2.9 in.	4.9 x 4.9 x 2.9 in.	4.9 x 4.9 x 2.9 in.
	124 x 124 x 74mm	124 x 124 x 74mm	124 x 124 x 74mm
Mounting	T-Thread, 2" nosepiece	T-Thread, 2" nosepiece	T-Thread, 2" nosepiece
Weight	2.7 pounds / 1.2kg	2.7 pounds / 1.2kg	2.7 pounds / 1.2kg
Backfocus	0.69 inches / 17.5 mm	0.69 inches / 17.5 mm	0.69 inches / 17.5 mm

```
In [1]: 1     from astropy.io import fits
2     from os import walk
3     from matplotlib import pyplot as plt
4     import numpy as np
6     from scipy import stats
7
```

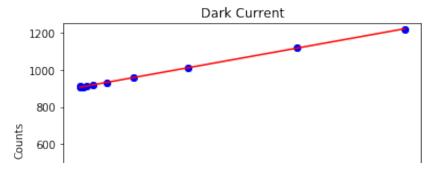
```
In [3]:
            #print(scidata.shape)
            #print(scidata.dtype)
In [4]:
         1
            #f[0].header
In [5]:
            #f[0].header['EXPTIME']
In [6]:
          1
            # Find GAIN
          2
          3
            # Get the data from an HDU.
            Fldata = fits.getdata('2020-01-23/Gain/2sFF1.fit').astype(float)
            #print(f"min: {Fldata.min()}, max: {Fldata.max()}, mean: {Fldata.mea
            Flat1Mean = Fldata.mean()
            Flat1Std = Fldata.std()
         7
            F2data = fits.getdata('2020-01-23/Gain/2sFF2.fit').astype(float)
         9
            #print("min: {}, max: {}, mean: {:.3f}, std: {:.3f}, Flat2".format(F
            Flat2Mean = F2data.mean()
         10
         11 Flat2Std = F2data.std()
         12 | Z1data = fits.getdata('2020-01-23/Gain/00sBS1.fit').astype(float)
         13
            #print("min: {}, max: {}, mean: {:.3f}, std: {:.3f}, Dark1".format(Z
         14
            ZerolMean = Zldata.mean()
         15
            Zero1Std = Z1data.std()
         16 | Z2data = fits.getdata('2020-01-23/Gain/00sBS2.fit').astype(float)
         17
            #print("min: {}, max: {}, mean: {:.3f}, std: {:.3f}, Dark2".format(Z
            Zero2Mean = Z2data.mean()
         18
         19
            Zero2Std = Z2data.std()
         20
         21
            flatdif = (F1data - F2data)
         22
         23 | flatstd = flatdif.std()
            #print("flatstd: {:.3f}".format(flatstd))
         24
            zerodif = Z1data - Z2data
         25
         26 zerostd = zerodif.std()
         27
            #print("zerostd: {:.3f}".format(zerostd))
         28
            gain = ((Flat1Mean + Flat2Mean) - (Zero1Mean + Zero2Mean)) / ((flats
         29
         30
         31
            readnoise = gain * zerostd / np.sqrt(2)
            print("gain: {:.3f}, readnoise: {:.3f}".format(gain, readnoise))
         32
         33
```

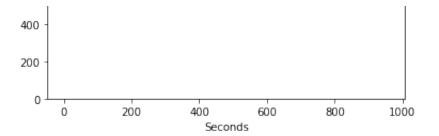
gain: 0.379, readnoise: 10.663

Dark Current

```
In [13]: # READ Files
```

```
3
   from pathlib import Path
4
   Means = []
5
   Expos = []
6
7
   basepath = Path('2020-02-04/')
8
   files in basepath = basepath.iterdir()
9
   for item in files in basepath:
10
       if item.is file():
            with fits.open(item) as f:
11
12
                 f.info()
            print(item.name, (f[0].header['EXPTIME']))
13
14
                data = fits.getdata(basepath/item.name)
                Means.append(data.mean())
15
16
                Expos.append(f[0].header['EXPTIME'])
                 print("mean: {}, exp: {}".format(data.mean(), (f[0].hea
17
                 plt.scatter((f[0].header['EXPTIME']),data.mean())
18
19
20
   plt.scatter(Expos, Means)
21
   plt.title('Dark Current')
   plt.xlabel('Seconds')
22
23
   plt.ylabel('Counts')
24
   plt.ylim(0,1250)
25
26
27
   gradient, intercept, r_value, p_value, std_err = stats.linregress(Ex
28
29
   mn=np.min(Expos)
30
   mx=np.max(Expos)
31 x1=np.linspace(mn, mx, 500)
32
   y1=gradient*x1+intercept
   plt.plot(Expos, Means, 'ob')
33
34
   plt.plot(x1,y1,'-r')
   plt.savefig('Lab1 DarkCurrent.png')
35
36
   plt.show()
37
   print("slope: {:.6f}, R: {:.6f}, P: {:.6f}, Std err: {:.6f}".format(
38
39
40
   with fits.open('2020-02-04/640sDK.fit') as f:
41
       print("Dark Current: {:.3f}, at Temp: {:.3f}".format(gradient,f[
42
```





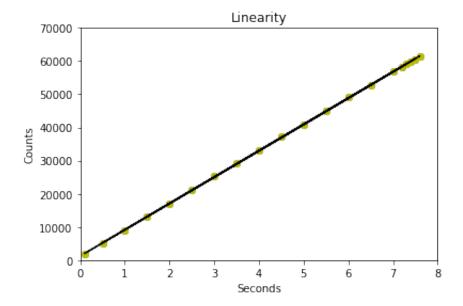
slope: 0.328064, R: 0.999794, P: 0.000000, Std_err: 0.002221
Dark Current: 0.328, at Temp: -0.888

Linearity

```
In [14]:
           1
              # READ Files
           2
           3
             from pathlib import Path
           4
             Means = []
           5
             Expos = []
           6
           7
             basepath = Path('2020-01-23/Fast/')
           8
             files in basepath = basepath.iterdir()
           9
              for item in files in basepath:
          10
                  if item.is file():
                      with fits.open(item) as f:
          11
          12
                           f.info()
          13
                       print(item.name, (f[0].header['EXPTIME']))
          14
                          data = fits.getdata(basepath/item.name)
          15
                          Means.append(data.mean())
                          Expos.append(f[0].header['EXPTIME'])
          16
          17
                           print("mean: {}, exp: {}".format(data.mean(), (f[0].hea
          18
                           plt.scatter((f[0].header['EXPTIME']),data.mean())
          19
          20
             plt.scatter(Expos, Means)
          21
             plt.title('Linearity')
             plt.xlabel('Seconds')
          2.2
          23
             plt.ylabel('Counts')
          24
             plt.ylim(0,70000)
          25
          26
          27
             gradient, intercept, r_value, p_value, std_err = stats.linregress(Ex
          28
             mn=np.min(Expos)
          29
             mx=np.max(Expos)
          30
             x1=np.linspace(mn, mx, 500)
          31
             y1=gradient*x1+intercept
          32
          33
          34
             coef = np.polyfit(Expos, Means, 1)
          35
             poly1d_fn = np.poly1d(coef)
```

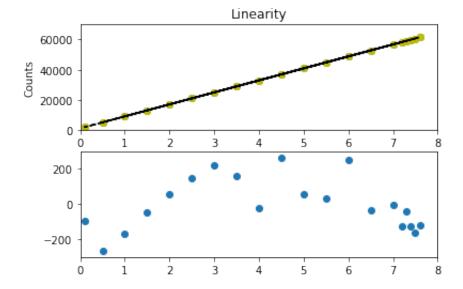
```
# polyld fn is now a function which takes in x and returns an estima
36
37
   plt.plot(Expos, Means, 'yo', Expos, poly1d_fn(Expos), '--k')
38
39
   #plt.plot(x1,y1+1000,'-r')
40
41
42
   plt.xlim(0, 8)
   plt.ylim(0, 70000)
43
   plt.savefig('Lab1 Linearity.png')
44
45
   plt.show
46
47
   print("slope: {:.6f}, R: {:.6f}, P: {:.6f}, Std err: {:.6f}".format(
48
49
```

slope: 7934.225589, R: 0.999974, P: 0.000000, Std_err: 13.591658



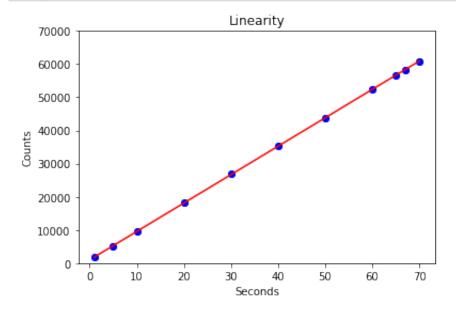
```
In [9]:
          1
          2
             #PLOT
          3
          4
             plt.subplot(211)
          5
            plt.scatter(Expos, Means)
          6
             plt.title('Linearity')
             plt.xlabel('Seconds')
             plt.ylabel('Counts')
          8
          9
            plt.ylim(0,70000)
         10
         11
             coef = np.polyfit(Expos, Means, 1)
         12
            poly1d fn = np.poly1d(coef)
         13
             # polyld fn is now a function which takes in x and returns an estima
         14
         15
             plt.plot(Expos, Means, 'yo', Expos, poly1d_fn(Expos), '--k')
         16
             plt.xlim(0, 8)
```

```
plt.yllm(U, /UUUU)
⊥ /
18
   plt.show
19
20
21
   #Residual plot
22
23
   plt.subplot(212)
24
   plt.scatter(Expos, Means-poly1d_fn(Expos))
25
   #plt.scatter(x1,Means-y1)
26
27
28
   plt.xlim(0, 8)
29
   plt.ylim(-300, 300)
30
   plt.savefig('Lab1_DC_Residuals.png')
31
   plt.show()
32
33
```



```
In [16]:
              # READ Files
           2
              from pathlib import Path
           4
             Means = []
           5
             Expos = []
           6
           7
             basepath = Path('2020-01-23/Slow/')
           8
              files in basepath = basepath.iterdir()
           9
              for item in files in basepath:
          10
                  if item.is file():
          11
                      with fits.open(item) as f:
          12
                           f.info()
                       print(item.name, (f[0].header['EXPTIME']))
          13
                          data = fits.getdata(basepath/item.name)
          14
          15
                          Means.append(data.mean())
                          Expos.append(f[0].header['EXPTIME'])
          16
```

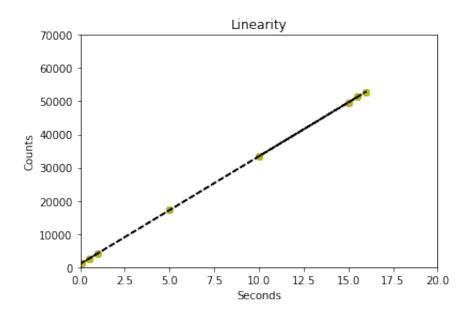
```
17
                 print("mean: {}, exp: {}".format(data.mean(), (f[0].hea
18
                 plt.scatter((f[0].header['EXPTIME']),data.mean())
19
20
   plt.scatter(Expos, Means)
   plt.title('Linearity')
21
22
   plt.xlabel('Seconds')
   plt.ylabel('Counts')
23
24
   plt.ylim(0,70000)
25
26
27
28
29
   gradient, intercept, r value, p value, std err = stats.linregress(Ex
30
   mn=np.min(Expos)
31
   mx=np.max(Expos)
32 \times 1 = np.linspace(mn, mx, 500)
   y1=gradient*x1+intercept
33
34
   plt.plot(Expos, Means, 'ob')
35
   plt.plot(x1,y1,'-r')
36
   plt.savefig('Lab1 DarkCurSlow.png')
37
   plt.show()
38
39
   print("slope: {}, R: {}, P: {}, Std err: {}".format(gradient,r value
40
```



slope: 855.9002635086536, R: 0.9999802673596974, P: 3.942172701229881e
-21, Std_err: 1.7923215654218903

```
6
 7
   basepath = Path('2020-01-23/Med/')
8
   files in basepath = basepath.iterdir()
9
   for item in files in basepath:
10
        if item.is file():
11
            with fits.open(item) as f:
12
                 f.info()
13
             print(item.name, (f[0].header['EXPTIME']))
                data = fits.getdata(basepath/item.name)
14
                Means.append(data.mean())
15
16
                Expos.append(f[0].header['EXPTIME'])
                 print("mean: {}, exp: {}".format(data.mean(), (f[0].hea
17
18
                 plt.scatter((f[0].header['EXPTIME']),data.mean())
19
20
   plt.scatter(Expos, Means)
21
   plt.title('Linearity')
   plt.xlabel('Seconds')
22
23
   plt.ylabel('Counts')
   plt.ylim(0,70000)
24
25
26
27
   coef = np.polyfit(Expos, Means, 1)
28
   poly1d fn = np.poly1d(coef)
29
   # polyld fn is now a function which takes in x and returns an estima
30
31
   plt.plot(Expos, Means, 'yo', Expos, poly1d fn(Expos), '--k')
   plt.xlim(0, 20)
32
33
   plt.ylim(0, 70000)
34
35
36
   plt.savefig('Lab1 DarkCurMed.png')
   plt.show
37
```

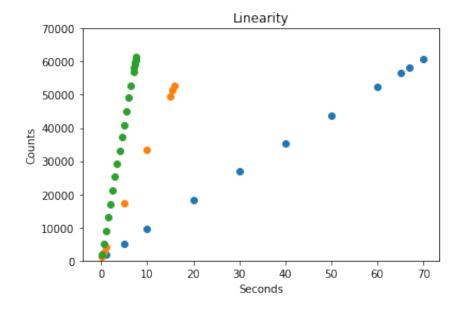
Out[15]: <function matplotlib.pyplot.show(*args, **kw)>



```
In [17]:
              # READ Slow Files
           1
           2
           3
             from pathlib import Path
           4
             SMeans = []
           5
             SExpos = []
           6
           7
             basepath = Path('2020-01-23/Slow/')
             files in basepath = basepath.iterdir()
           8
           9
             for item in files in basepath:
          10
                  if item.is file():
          11
                      with fits.open(item) as f:
          12
                           f.info()
                       print(item.name, (f[0].header['EXPTIME']))
          13
          14
                          data = fits.getdata(basepath/item.name)
          15
                          SMeans.append(data.mean())
          16
                          SExpos.append(f[0].header['EXPTIME'])
          17
                           print("mean: {}, exp: {}".format(data.mean(), (f[0].hea
          18
                           plt.scatter((f[0].header['EXPTIME']),data.mean())
          19
          20
             plt.scatter(SExpos, SMeans)
          21
             plt.title('Linearity')
          22
             plt.xlabel('Seconds')
          23
             plt.ylabel('Counts')
          24
             plt.ylim(0,70000)
          25
          26
          27
          28
             # READ Med Files
          29
          30
             #from pathlib import Path
          31
             MMeans = []
          32
             MExpos = []
          33
          34
             basepath = Path('2020-01-23/Med/')
          35
             files in basepath = basepath.iterdir()
          36
             for item in files in basepath:
          37
                  if item.is file():
          38
                      with fits.open(item) as f:
          39
                           f.info()
          40
                       print(item.name, (f[0].header['EXPTIME']))
          41
                          data = fits.getdata(basepath/item.name)
          42
                          MMeans.append(data.mean())
          43
                          MExpos.append(f[0].header['EXPTIME'])
          44
                           print("mean: {}, exp: {}".format(data.mean(), (f[0].hea
          45
                           plt.scatter((f[0].header['EXPTIME']),data.mean())
          46
          47
             plt.scatter(MExpos, MMeans)
          48
             #plt.title('Linearity')
          49 #plt.xlabel('Seconds')
```

```
#plt.ylabel('Counts')
50
51
   #plt.ylim(0,70000)
52
   # READ Fast Files
53
54
55
   #from pathlib import Path
56
   FMeans = []
57
   FExpos = []
58
59
   basepath = Path('2020-01-23/Fast/')
   files in basepath = basepath.iterdir()
60
61
   for item in files in basepath:
        if item.is file():
62
            with fits.open(item) as f:
63
                 f.info()
64
65
             print(item.name, (f[0].header['EXPTIME']))
66
                data = fits.getdata(basepath/item.name)
67
                FMeans.append(data.mean())
                FExpos.append(f[0].header['EXPTIME'])
68
                 print("mean: {}, exp: {}".format(data.mean(), (f[0].hea
69
70
                 plt.scatter((f[0].header['EXPTIME']),data.mean())
71
72
   plt.scatter(FExpos,FMeans)
73
   #plt.title('Linearity')
74
   #plt.xlabel('Seconds')
75
   #plt.ylabel('Counts')
76
   #plt.ylim(0,70000)
   plt.savefig('Lab1 DarkCurComp.png')
77
78
   plt.show
79
```

Out[17]: <function matplotlib.pyplot.show(*args, **kw)>



In []: 1