KRITTIKA SUMMER PROJECTS 2023

Faintest of the Brightest

Ravi Kumar, Gaurav Waratkar, and Meghna Dixit

KRITTIKA SUMMER PROJECTS 2023

Faintest of the Brightest

Ravi Kumar[1,2], Gaurav Waratkar[1,2,3], and Meghna Dixit[2,3]

[1]Krittika - The Astronomy Club of IIT Bombay, Powai, Mumbai - 400076, India

[2]Indian Institute of Technology Bombay, Mumbai - 400076, India

[3]Mentor for the Project

PUBLISHED BY KRITTIKA: THE ASTRONOMY CLUB OF IIT BOMBAY

GITHUB.COM/KRITTIKAIITB

First Release, July 2023

Abstract

Contents

1. Introduction

1.1 Gamma Ray Bursts

1.1.1 What are Gamma Ray Bursts?

Gamma-ray bursts (GRBs) are massively large explosions caused when a star dies resulting in a supernova, when two neutron stars collide, or when a neutron star collides with a black hole, releasing a large amount of energy.

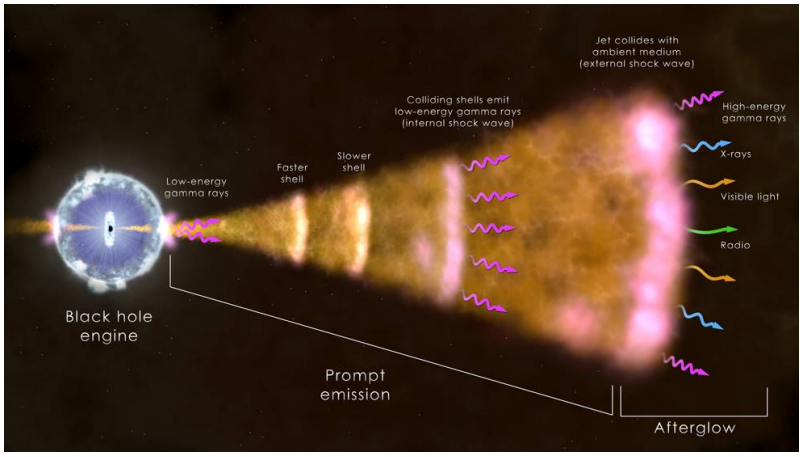


Figure 1.1: Emissions of a GRB (Source: NASA's Goddard Space Flight Center)

This illustration shows the ingredients of a long Gamma-ray burst, the most common type. The core of a massive star (left) has collapsed, forming a black hole that sends a jet of particles moving through the collapsing star and out into space at nearly the speed of light. Radiation across the spectrum arises from hot ionized gas (plasma) in the vicinity of the newborn black hole, collisions among shells of fast-moving gas within the jet (internal shock waves), and from the leading edge of the jet as it sweeps up and interacts with its surroundings (external shock).

The presence of a prompt emission and a much longer afterglow (in multiple wavelengths) is a common feature of GRBs.

1.1.2 Classification of GRBs

GRBs are classified into 3 main classes, short, long and ultra-long, according to the duration for which they are detected.

Short GRBs are events with a duration of less than 2 seconds, and the Gamma rays from short bursts lean toward the high-energy end of the spectrum, while long GRBs emit lower-energy Gamma rays. They are thought to be caused by the merger of two neutron stars or a neutron star and a black hole and account for about 30% of detected GRBs.

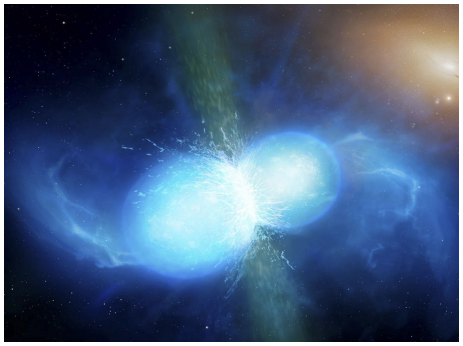


Figure 1.2: Artist depiction of a GRB created by two merging neutron stars (Source: Mark A. Garlick)

Long GRBs are events with a duration of more than 2 seconds. Although the Gamma rays emitted are of lower energy than short GRBs, the total energy output of long GRBs is generally higher owing to their longer duration. They are thought to be caused by the core collapse of rapidly rotating massive stars

Ultra-Long GRBs are events that last more than 10,000 seconds. They have been proposed to be caused by the collapse of a Blue Supergiant star, a tidal disruption event or a newborn magnetar

1.2 Astrosat and CZTI

1.2.1 Astrosat - India's first multiwavelength space telescope

AstroSat is the first dedicated Indian astronomy mission aimed at studying celestial sources in X-ray, optical and UV spectral bands simultaneously. The payloads cover the energy bands of Ultraviolet (Near and Far), limited optical and X-ray regime (0.3 keV to 100keV). One of the unique features of AstroSat mission is that it enables the simultaneous multi-wavelength observations of various astronomical objects with a single satellite.

AstroSat with a lift-off mass of 1515 kg was launched on September 28, 2015 into a 650 km orbit inclined at an angle of 6 deg to the equator by PSLV-C30 from Satish Dhawan Space Centre, Sriharikota. The minimum useful life of the AstroSat mission is expected to be 5 years.

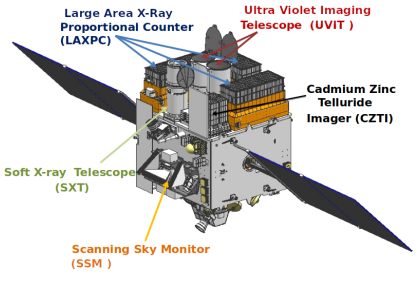


Figure 1.3: Various Payloads on Astrosat (Source: ISSDC)

The spacecraft control centre at Mission Operations Complex (MOX) of ISRO Telemetry, Tracking and Command Network (ISTRAC), Bengaluru manages the satellite during its entire mission life. The science data gathered by five payloads of AstroSat are telemetered to the ground station at MOX. The data is then processed, archived and distributed by Indian Space Science Data Centre (ISSDC) located at Bylalu, near Bengaluru.

1.2.2 Cadmium-Zinc-Telluride Imager (CZTI)

It is a hard X-ray imaging instrument covering the energy band from 10 to 100 keV, has a detector area of 976 cm^2 constructed using CZT modules, and uses a Coded Aperture Mask (CAM) for imaging. The CZTI carries a Cesium Iodide (TI) based scintillator detector for Veto measurements. This is located just under the CZT detector modules. Further, there is a gap of about 8 cm between the base of the collimator slats and the detector plane, in order to accommodate a radioactive calibration source module in each quadrant. This source shines alpha-tagged 60 keV photons on the CZT detector in order to help calibrate the energy response.

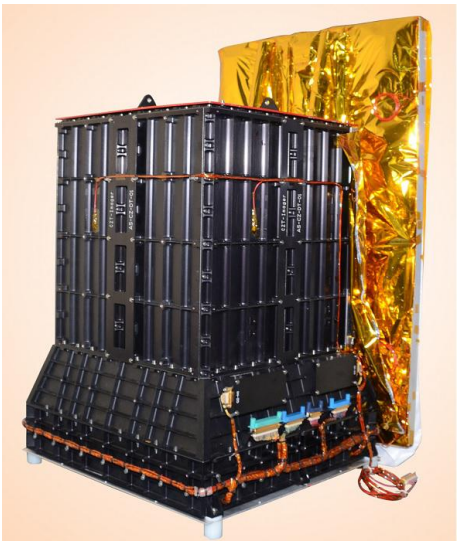


Figure 1.4: CZTI (Source: IUCAA)

Veto Detector - a detector (CsI) covers the large area of 256 cm2 and the light collection is done using two photomultipliers (PMT), viewed from sides. On registering an event, a signal from the detector is sent to a pre-amplifier. This signal is processed and sent to the FEB for further analysis. After amplification of the signal from pre-amplifier, the signal is sent to a comparator via stretcher along with LLD level signal. If LLD is triggered the pulse is digitized to an 8-bit output by ADC through control circuit. This output is used to differentiate Compton scattered events and hence the background in main detector can be reduced.

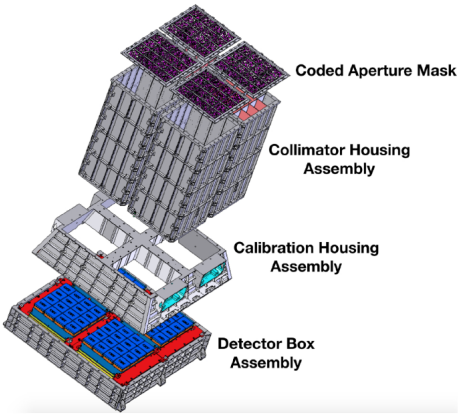


Figure 1.5: CZTI Assembly (Source: Image Calibration Paper)

The full CZTI detector is illuminated only by the on-axis source. Off-axis sources illuminate only a part of the detector depending on the position of the source, this is called "partial coding". The coded mask for CZTI is designed with seven patterns based on 255-element pseudo-noise Hadamard set Uniformly Redundant Arrays (URA). Each pattern has 16×16 mask elements and used as a mask for an individual detector module. A random arrangement of these patterns into a 4 × 4 array results in the mask pattern for the first quadrant (quadrant A). The coded masks for the other quadrants (B, C, and D) were obtained by rotating the mask pattern of quadrant A by 90°, 180° and 270° respectively.

A passive collimator wall of height 400 mm separates any two adjacent modules, restricting the view of each detector module to the coded mask directly above it. The collimator thus restricts the Field of View (FOV) of CZTI to 4.6° × 4.6° FWHM at energies below 100 keV. For energies above 100 keV, the collimator walls and the coded mask become progressively transparent, and allows the detection of Gamma-Ray Bursts from all over the sky

1.2.3 Coded Aperture Mask Imaging

Coded apertures or coded-aperture masks are grids, gratings, or other patterns of materials opaque to various wavelengths of electromagnetic radiation. The wavelengths are usually high-energy radiation such as X-rays and Gamma rays. By blocking radiation in a known pattern, a coded "shadow" is cast upon a plane. The properties of the original radiation sources can then be mathematically reconstructed from this shadow. Coded apertures are used in X- and Gamma ray imaging systems, because these high-energy rays cannot be focused with lenses or mirrors that work for visible light.

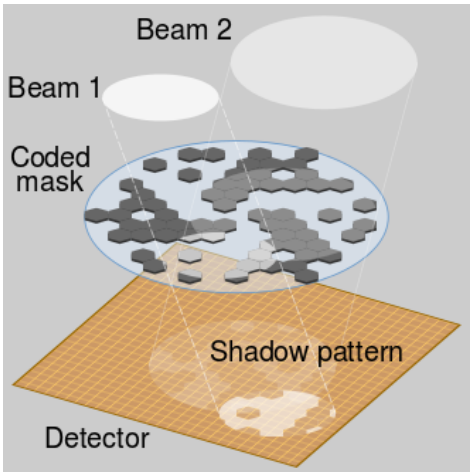


Figure 1.6: Working of a CAM (Source: Wikipedia)

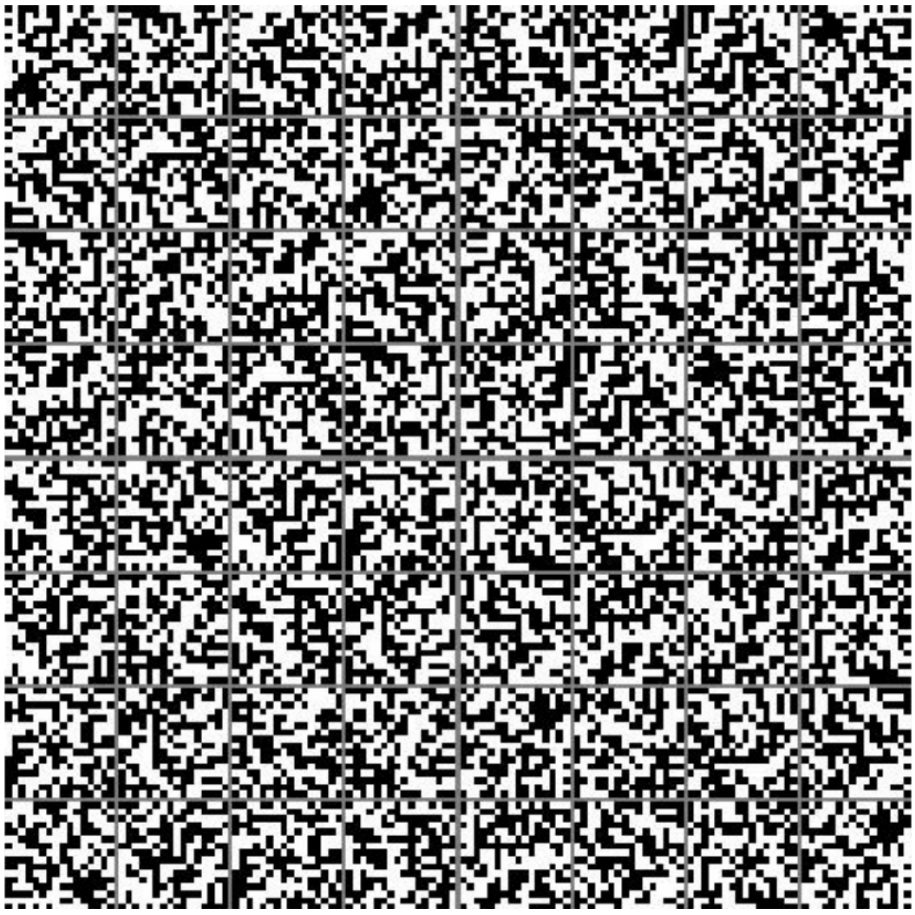


Figure 1.7: CZTI coded aperture mask for all four quadrants designed using 255 element pseudo-noise Hadamard set Uniformly Redundant Arrays. One extra closed element is added to each URA to obtain a square pattern. Here black areas represent closed mask elements and white areas represent open ones. (Source: Image Calibration Paper)

1.3 CZT Pipeline

1.3.1 How It Works

The data received from the CZTI payload passes through a series of steps in a processing pipeline. Three major data levels have been designated for long-term storage. These are:

Level 0

This is the raw data received from satellite telemetry, which is segregated by instrument, along with auxiliary data. This data is archived internally and not distributed for public use.

Level 1

This is reorganized raw data, written in FITS format for Astronomical use. All auxiliary information necessary for further processing of this data are collated at this level and packed along with the respective science data. This data is released via Astrosat data archive, at first to the Principal Investigator (PI) of the corresponding observing proposal and, after a specified initial lock in period, to anyone interested in the data.

Level 2

This data contains standard science products derived from Level 1 data. Level 2 data is also in FITS format and is available for science use, with the same lock-in criteria and release mechanism as the Level 1 data.

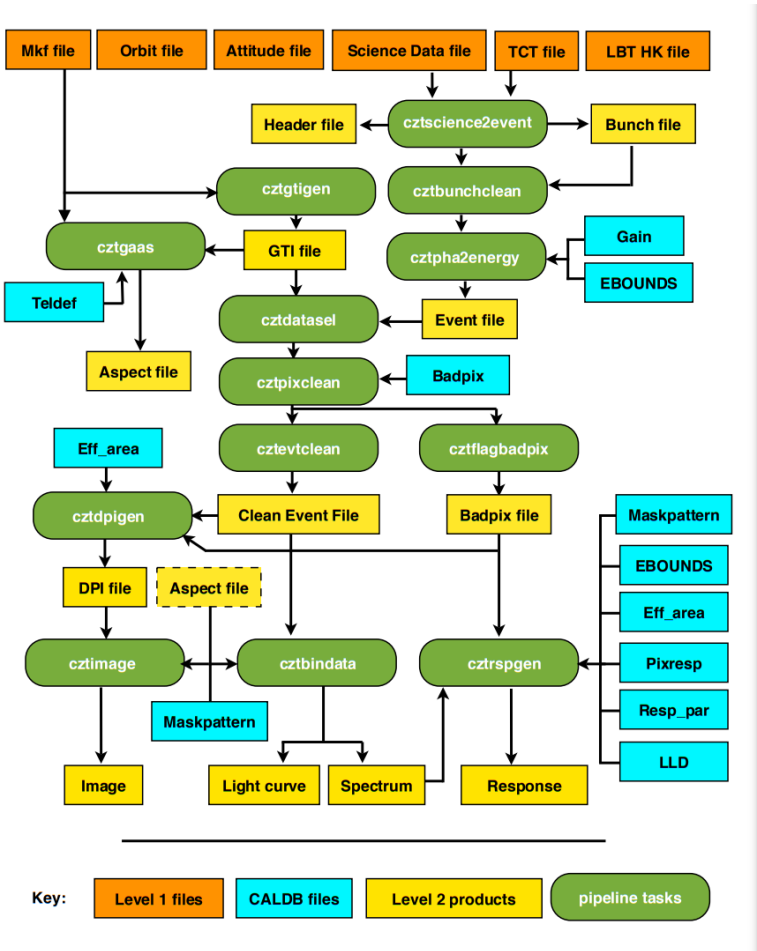


Figure 1.8: Pipeline Work Flow (Source: CZT Software Userguide)

1.3.2 Work Flow

Data reduction for CZTI is performed in three stages. Each of these stages involves execution of several tasks of the pipeline.

- Stage 1: Generation of event file and calibration. In this stage event file is generated
- Stage 2: Selection of data and cleaning. In this stage events are selected based on Good Time Interval and noisy pixel events are removed from the data.
- Stage 3: Generation of science products. In this last stage,DPI, image, spectrum, light curve and response matrix are generated.

The pipeline tasks can either be executed individually or by using the cztpipeline module which allows the user to run required stages of the pipeline tasks.

1.3.3 Pipeline Installation

I found Ubuntu 20.04 LTS to work best with the pipeline on WSL2. Other distros (including other versions of Ubuntu) gave various compilation errors.

To start off, update the system by running

```
$ sudo apt update && sudo apt upgrade
```

After this, we must install some packages that are essential to ensuring the pipeline installs smoothly. Install them by running

```
$ sudo apt install gcc g++ gfortran perl make
```

Now download the pipeline from here and the CALDB from here.
Untar these files, ideally in your home directory by running the following:

```
$ tar -xvf czti_pipeline_20180308_V2.1.tar
$ tar -xvf caldb_goodfiles_as1_czti_20180308_V1.1.tar.gz
```

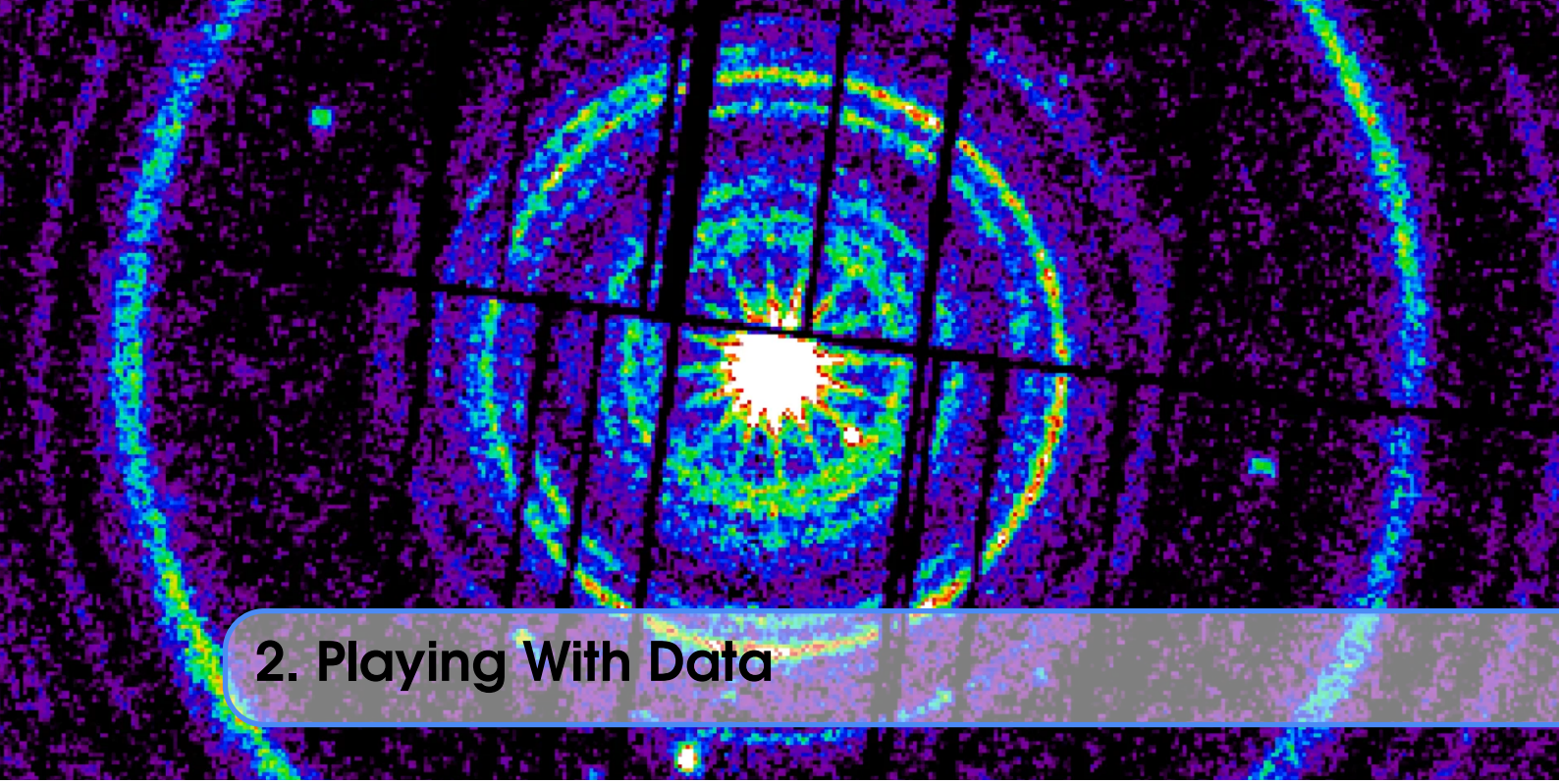
Now add the following to your `.bashrc` file (by running `$ sudo nano ~/.bashrc`)

```
export as1czt=~/czti_pipeline/CZTI/czti/
export PFILES="$PFILES:$as1czt/paramfiles"
export PATH=$as1czt/bin:$as1czt/scripts:$PATH
export GLOG_log_dir=$as1czt/log
export CZTI_templates=$as1czt/templates
export PERL5LIB="$as1czt/lib/":$PERL5LIB
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:$as1czt/lib
export PYTHONPATH="$PYTHONPATH:$as1czt/scripts"
export CXXFLAGS="-fpermissive"
ulimit -s 65532
export CALDB=~/
```

Finally execute the following commands

```
$ source ~/.bashrc
$ cd $as1czt
$ cd ../
$ ./InstallLibs
$ cd $as1czt
$ make
$ cd scripts
$ chmod +x cztpipeline
```

1.3.4 Automation of the Pipeline



2. Playing With Data

2.1 Introduction

2.1.1 FITS Files

FITS (Flexible Image Transport System) is a standard format for storing astronomical data. FITS is much more than an image format (such as JPG or GIF) and is primarily designed to store scientific data sets consisting of multi-dimensional arrays (1-D spectra, 2-D images or 3-D data cubes) and 2-dimensional tables containing rows and columns of information.

Headers

The FITS header is a block of text at the beginning of the file that contains information about the data contained within the file. The header is arranged in a series of keyword-value pairs, known as header cards, that each consist of a keyword, a value, and an optional comment. The keyword tells the type of information that the header card contains, the value is the actual value of the information, and the comment describes the information or how it was derived. The header is terminated by the keyword END.

Data

The data in a FITS file is stored in a series of N-dimensional arrays, where N is any number between 0 (empty) and 999. The data arrays are preceded by a series of header keywords that describe the size (NAXIS), location (NAXISn), data type (BITPIX), and other characteristics of the data arrays. The data arrays are stored in the file in the same order that they are listed in the header.

2.1.2 Accessing Files via Astropy

Astropy is a community Python library for Astronomy. It contains among other things:

- A powerful N-dimensional array object

- Sophisticated (broadcasting) functions
- Tools for integrating Fortran code
- Useful linear algebra, Fourier transform, and random number capabilities

Opening a FITS file is as simple as:

```
from astropy.io import fits
hdul = fits.open('file.fits')
hdul.info()
```

This will print out the information about the file, including the number of HDUs in the file and the name and dimensions of each extension.

To obtain the data and header from the file, we can use:

```
data = hdul[0].data
header = hdul[0].header
```

2.1.3 Creating the Light Curve Files

The light curve files are created by running the `cztbindata` command in the terminal. The command takes in the following input files along with time bin size and produces the .lc and .pha (spectrum) file:

- `inevtfile` - The input evt file
- `mkffile` - The input mkf file
- `badpixfile` - The input badpix file
- `livetimefile` - The input livetime file

Finally the .lc file can be opened using `astropy.io.fits` and the required columns can be plotted. In our case we will be plotting the `COUNTS` column against the `TIME` column.

Here is a sample light curve file plotted for the duration of the GRB190928A:

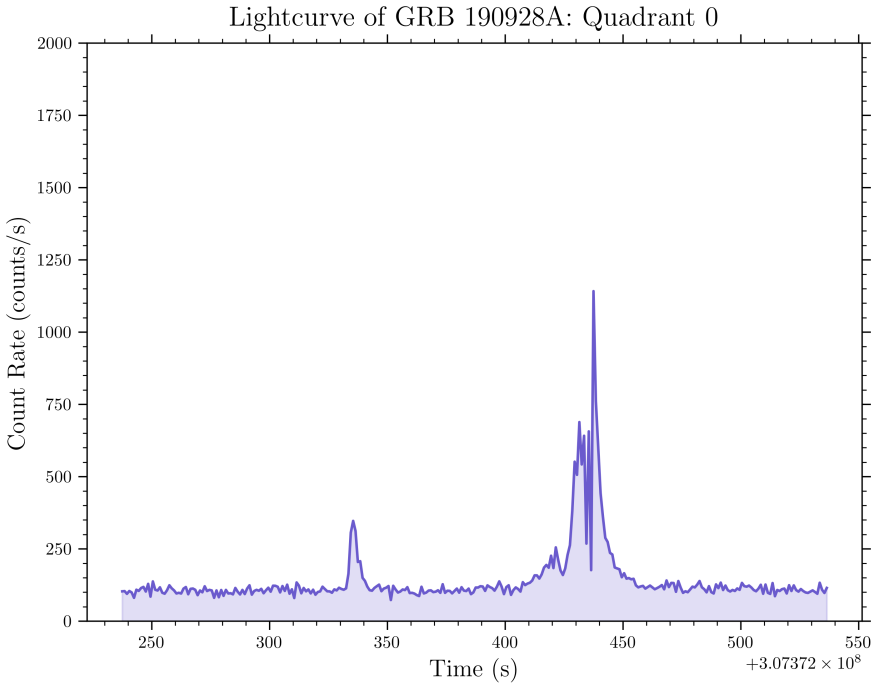


Figure 2.1: Light Curve for GRB190928A

2.1.4 Creating the Spectrum Files

The process used in creating the light curve will automatically produce a .pha file which contains the spectrum of the GRB. We can plot the spectrum using `astropy.io.fits` and `matplotlib.pyplot`.

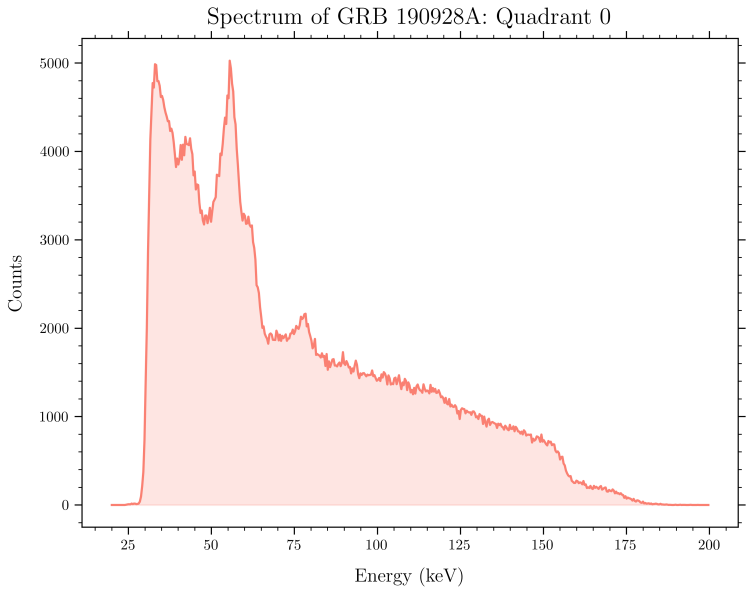


Figure 2.2: Spectrum for GRB190928A

2.1.5 Some Anomalies in the Data

The above light curve was plotted around the time of the GRB, but if we plot the full light curve, we can see some anomalies in the data where the counts suddenly fall to zero.

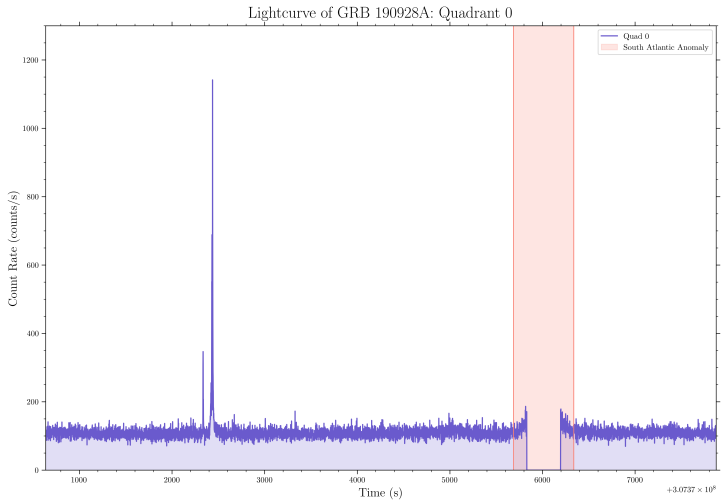


Figure 2.3: Anomaly in the Light Curve

The region shaded in red is the region where the counts start rising then fall to 0. This is because there is a high influx of cosmic rays in a particular region in the South Atlantic Ocean. This region is known as the South Atlantic Anomaly (SAA).

The SAA is a consequence of the fact that the Earth's inner Van Allen radiation belt comes closest to the Earth's surface at the south magnetic pole. This leads to an increased flux of energetic particles in this region and exposes orbiting satellites to higher than usual levels of radiation.

The detectors on Astrosat are shut off when the satellite passes through the SAA, and hence the counts fall to 0.

2.2 Analysis of GRB190928A

2.2.1 Finding the GRB

We notice two peaks in the light curve. The first peak is the Prompt Emission, and the second peak is the Afterglow. We can find the GRB by looking at the `COUNTS` column and finding the maximum value. We can then use the `TIME` column to find the time at which the maximum value occurs. In this case, the GRB occurs at 2019-09-28 13:12:17 UTC.

The duration of the GRB would be the time difference between the time at which the GRB starts and the time at which it ends. The start and end times are taken roughly before the prompt emission and after the afterglow respectively. In this case, the duration of the GRB is 132 seconds.

A thing to note is that `cztbindata` produces .lc files for each quadrant of the CZT detectors (namely Quad 0, Quad 1, Quad 2 and Quad 3). We must plot all four quadrants to get a good idea of the GRB.

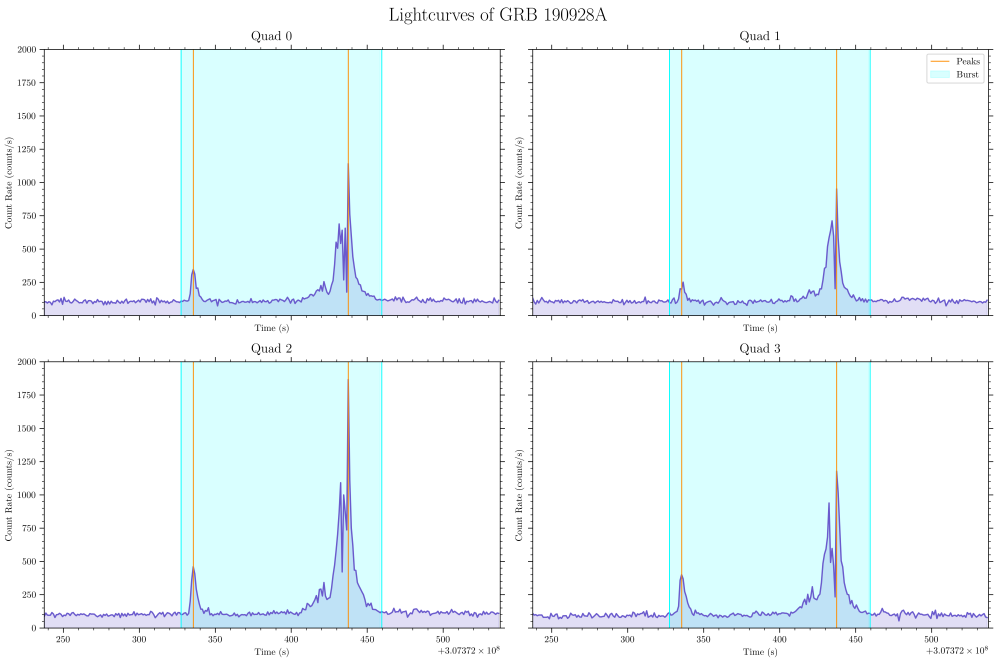


Figure 2.4: Light Curves for all four quadrants

The above plot shows the light curves for all four quadrants along with the peaks of the prompt emission and afterglow and markers for the start and end of the GRB.

2.2.2 Filtering and Detrending

From Figure 2.3, we can see that the "backgorund" data (the signal apart from the GRB) has a mean value somewhere around 100, along with a lot of variation around that mean value. This variation is known as *noise* and is caused by the inherent physics of the working of the CZTI detectors. There is also an upward slope right before the SAA region which causes a "trend" in the data. We need to "detrend" the data to make any GRB in that region stand out.

We can do this by using the `savgol_filter` function from `scipy.stats` to filter the data. The `savgol_filter` function applies a Savitzky-Golay filter to the data.

The `savgol_filter` function takes in the data, the window length and the polynomial order as input. The filter fits a polynomial of a given order to the datapoints in the window, then we subtract this polynomial from the data. This is done for all the datapoints in the window and further for all windows in the data.

The above process is done for the data points *outside* of the GRB region, so as to not detrend the GRB itself. We also have to subtract the mean of the original background (≈ 100) from the GRB to get the final GRB data. This step can easily be done before the detrending.

In python, filtering and detrending can be done as follows:

```
1  from scipy.signal import savgol_filter
2  import numpy as np
3  # Subtracting the mean of the original background
4  data[grb] -= np.mean(data[noise])
5  # Filtering and Detrending
6  data[noise] -= savgol_filter(data[noise], 101, 3)
```

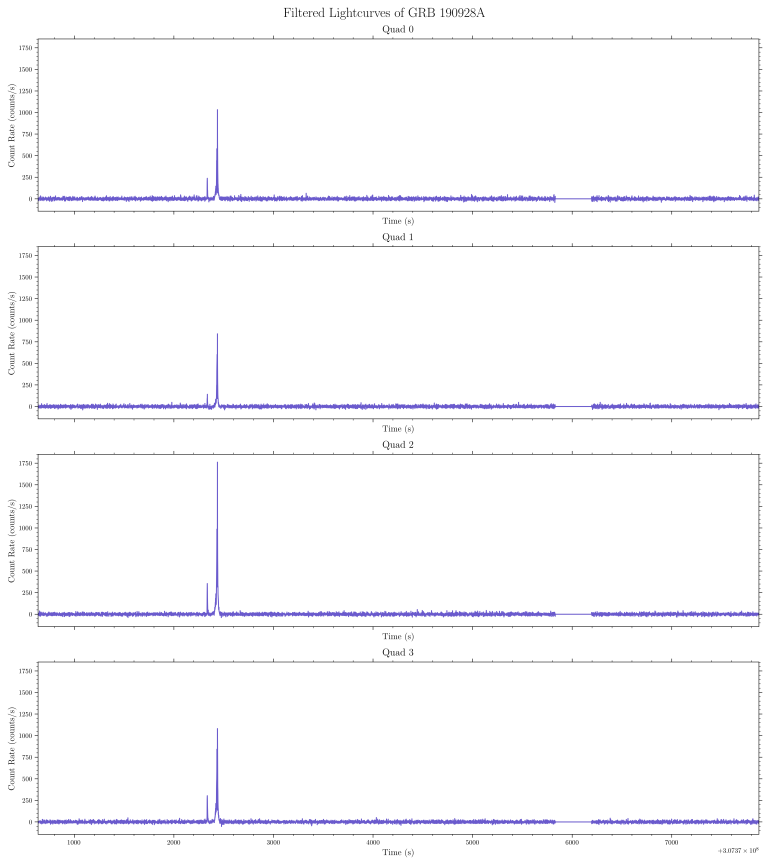


Figure 2.5: Detrended Light Curve

For the purposes of our analysis, we will be using a window length of 101s and a polynomial order of 3. The results of the filtering and detrending are shown below:

To the left we can see that the slope next to the SAA region has been removed, the background has been reduced to nearly 0, and the GRB has been brought down from the original background of ≈ 100 to ≈ 0.

We can also see that the "dip" that occurs during the SAA region is now a flat line with values equal to 0.

2.3 Different Bin Sizes

Uptil now we have analysed the data using a bin size of 1s. We can also analyse the data using different bin sizes to see how the results change. For the purposes of this analysis, we will be using bin sizes of 0.1s, 1s, 10s.

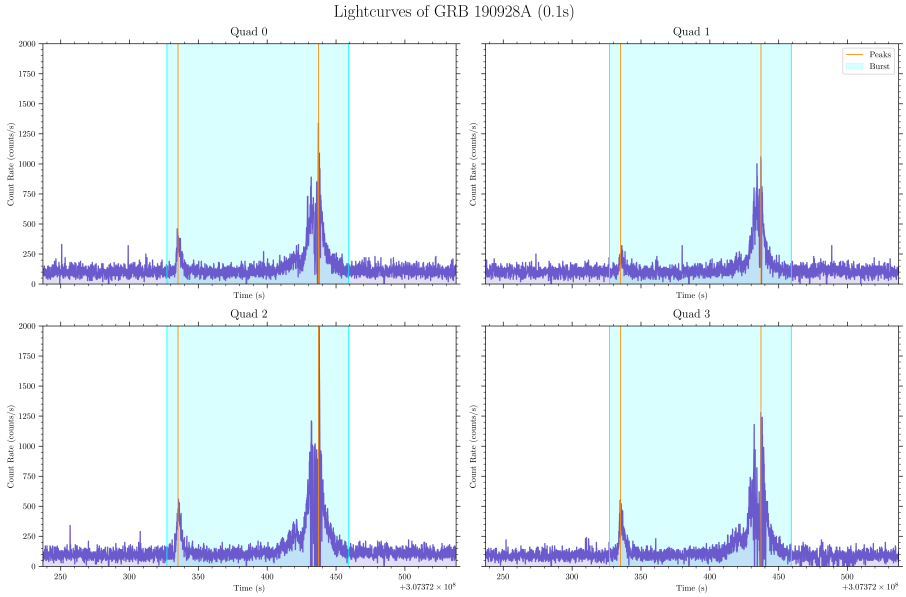


Figure 2.6: Light Curves for bins of 0.1s

Here we have the light curves of all four quadrants during the interval of the GRB. We can see that the smaller binsize causes the variations in the background to be more prominent.

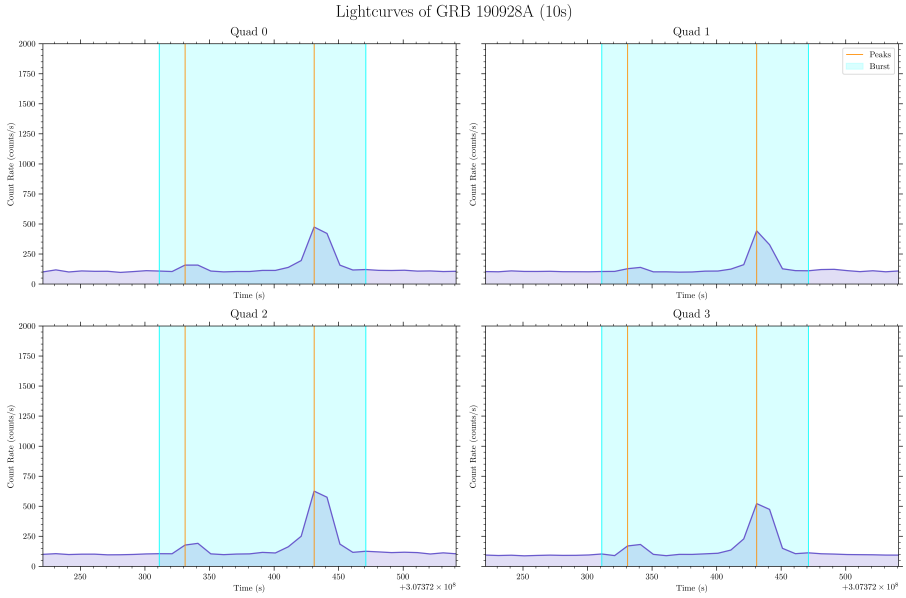


Figure 2.7: Light Curves for bins of 10s

Here we can see that the larger binsize causes the variations in the background to be less prominent, as more of the noise gets smooothened (averaged) out.

3. Signal To Noise Ratio

3.1 Introduction

The signal-to-noise ratio (SNR) is a measure used in science and engineering that compares the level of a desired signal to the level of background noise. SNR is defined as the ratio of signal power to the noise power, often expressed in decibels. A ratio higher than 1:1 (greater than 0 dB) indicates more signal than noise. While SNR is commonly quoted for electrical signals, it can be applied to any form of signal, in out case, we will be finding the SNR of a single in a timeseries.

Now, comparing the levels of the signal and noise are easier said than done. The signal is the GRB, and the noise is the background. The background is not constant, and varies a lot. This particular variation can be seen most clearly in Figure 2.6. GRB 190928A is a particularly bright GRB, so it is easy to see the GRB in the light curve. But what about fainter GRBs? How do we find the SNR of a faint GRB? We'll look at the various methods used to determine the SNR in the upcoming sections.

3.1.1 Methodology

There are a plethora of methods that can be used to determine the SNR of a GRB. There are methods that involve integrating the signal and noise separately and then dividing the two, there are methods that involve fitting a distribution to the noise and taking a certain percentile of the distribution as the noise and so on. Regardless of the methods, the crux of the matter is to find quantities that best describe the signal and the noise.

3.2 Signal

Describing the signal is a fairly straightforward task as opposed to describing the noise.

The signal is the GRB, and the GRB is a peak in the light curve. So, we can describe the signal as the maximum value of the light curve textitin the GRB window. This is the simplest way of describing the signal, and is the method we will be using for the majority of this project and will be referred to as M_s.

Another method of describing the signal is to integrate the light curve over the duration of the GRB. This method will be taken as a baseline for the other methods.

3.3 Noise

Describing the noise is a much more difficult task than describing the signal. The noise is the background, and the background is not constant. The background varies a lot, and the variation is slightly different for each orbit of the satellite, as well as each quadrant of the CZT detectors.

3.3.1 Method 1 - Time Averaged Integrals

Since we are dealing with data that is already time binned, our "Integral" simply becomes the sum of the counts for both the noise and the signal. An important thing to note is that the duration of the GRB is not the same as the duration of the we sample from the lightcurve, so it's important to normalise the values of the signal and noise by dividing by their respective durations.

Mathematically speaking, we can write this down as:

$$\text{Signal} = \frac{\sum_{i=0}^{n} \text{Counts}_i}{\text{Duration of GRB}} \qquad \text{Noise} = \frac{\sum_{i=0}^{m} \text{Counts}_i}{\text{Duration of Noise}}$$

Where n is the number of data points in the GRB, and m is the number of data points in the noise.

3.3.2 Method 2 - Fitting a Distribution to the Noise

This method involves fitting a distribution to the noise and then taking a certain sigma clip of the distribution as the noise. This method is more robust than the previous method as it takes into account the variation in the noise. Say for example, the noise is a Gaussian distribution, then we fit a Gaussian distribution by varying the mean (μ) and standard deviation (σ) till we get the best fit. We then take the value of noise as $\mu + 3\sigma$.

There are quite a few different distributions that are of interest to us, namely the Gaussian, Poisson, Gamma, Weibull and Rayleigh distributions. We will be looking at each of these distributions and the troubles faced in fitting them in the upcoming sections.

An important note: *When we fit the noise to some distrbution, according to the nature of the distribution, we get a value for the optimal mean and standard deviation that will fit our noise. Since the data has been detrended, our noise is centered around 0, so our entire data must be shifted by the mean obtained by fitting, and our signal becomes $M_s + \mu$.*

3.3.3 Gaussian Distribution

A Gaussian distribution is a continuous function representing the distribution of many random variables as a symmetrical bell-shaped graph. The Gaussian distribution is also known as the normal distribution. The equation for a Gaussian distribution is given by:

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \tag{3.1}$$

Where μ is the mean and σ is the standard deviation of the distribution. The Gaussian is of some interest to us because the noise in the light curves follows a rough bell-shaped distribution.

Fitting a Gaussian is pretty straight forward using the `scipy.optimize.curve_fit` function, and defining a function that produces a purely Gaussian equation either by using the equation (3.1) or the `scipy.stats.norm.pdf` function. In either case, the function takes in the data as input and returns the mean and standard deviation of the distribution.

We can then take the noise as $\mu + 3\sigma$ and the signal as $M_s + \mu$ (Recall the note at the end of Section 3.3.2).

$$\therefore \text{SNR} = \frac{M_s + \mu}{\mu + 3\sigma} \tag{3.2}$$

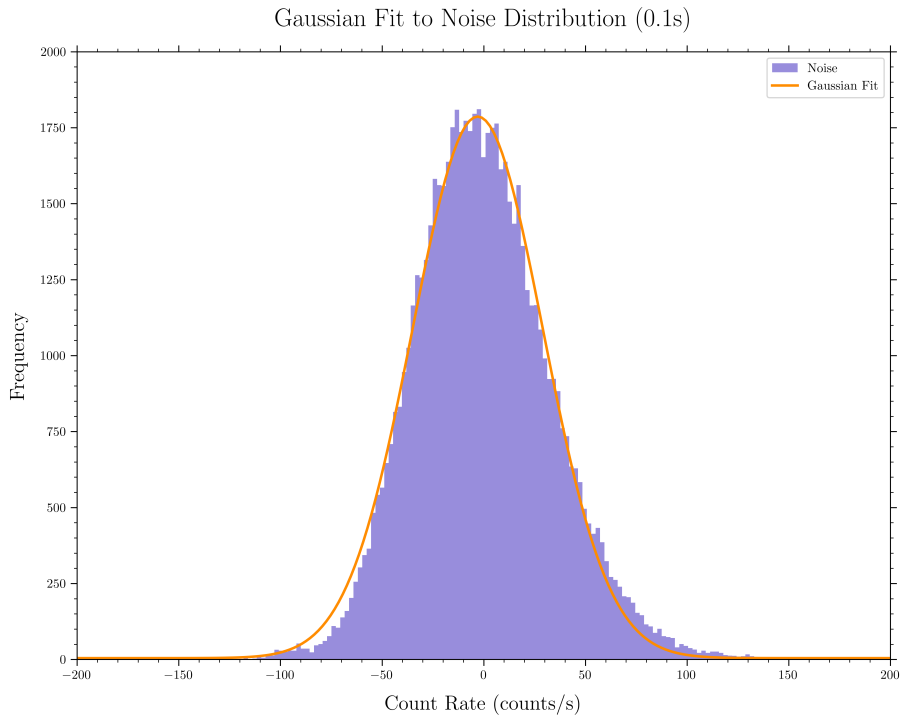


Figure 3.1: Fitting a Gaussian to the Noise (BinSize 0.1s) of Quadrant 0

It's clear from the above picture that our noise is *not* Gaussian but a Gaussian is a good approximation of the noise.

3.3.4 Poisson Distribution

The Poisson distribution is a discrete probability distribution that expresses the probability of a given number of events occurring in a fixed interval of time or space if these events occur with a known constant mean rate and independently of the time since the last event. The Poisson distribution can also be used for the number of events in other specified intervals such as distance, area or volume. The Poisson distribution is defined as:

$$f(k;\lambda) = \frac{\lambda^k e^{-\lambda}}{k!} \tag{3.3}$$

Where k is the number of events, and λ is the average number of events per interval. An interesting thing to note is that the mean and variance of the Poisson distribution are both equal to λ. The Poisson distribution is of interest to us because the noise in the light curves seems to have a skewed distribution, and the Poisson distribution, in general, approximates a skewed distribution well.

Fitting a Poisson distribution is a bit more complicated than fitting a Gaussian distribution. We use the `scipy.optimize.curve_fit` function and defining a function that returns a Poisson distribution by using `scipy.stats.poisson.pmf`. In this case we obtain only the mean of the distribution (λ).

We can now take the noise as $\lambda + 3\sqrt{\lambda}$ and the signal as $M_s + \lambda$.

$$\therefore \text{SNR} = \frac{M_s + \lambda}{\lambda + 3\sqrt{\lambda}} \tag{3.4}$$

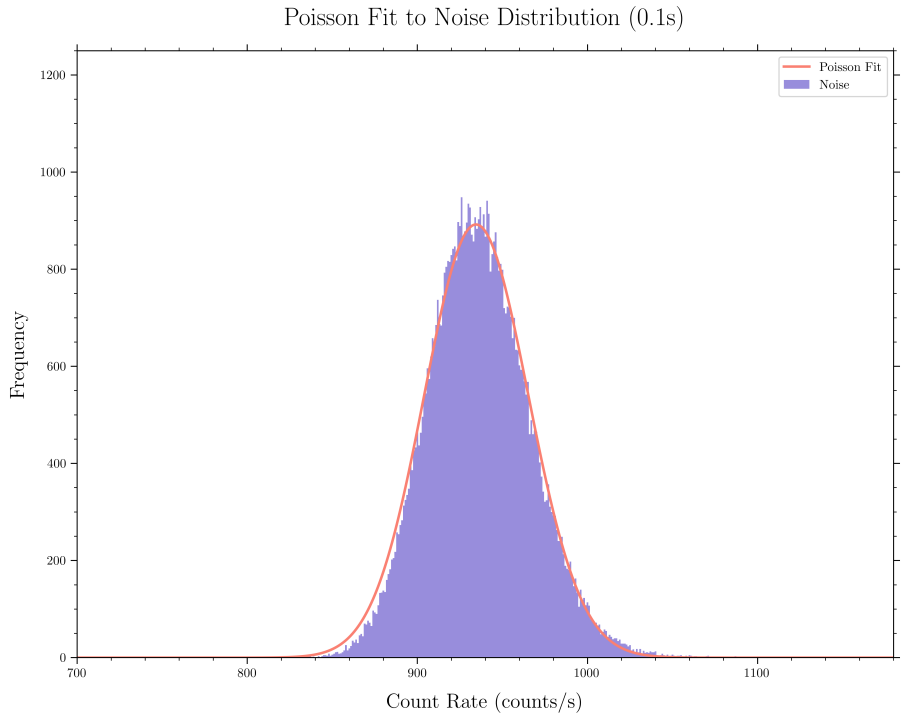


Figure 3.2: Fitting a Poisson Distribution to the Noise (BinSize 0.1s) of Quadrant 0

We can see that the Poisson distribution is a better fit to the noise than the Gaussian distribution.

3.3.5 Gamma Distribution

The Gamma distribution is a two-parameter family of continuous probability distributions. The exponential distribution, Erlang distribution, and chi-squared distribution are special cases of the Gamma distribution. There are two different parameterisations in common use:

- With a shape parameter k and a scale parameter θ.
- With a shape parameter $\alpha = k$ and an inverse scale parameter $\beta = 1/\theta$, called a rate parameter.

In either case, the probability density function of the Gamma distribution is given by:

$$f(x;k,\theta) = \frac{x^{k-1}e^{-x/\theta}}{\theta^k \Gamma(k)} \tag{3.5}$$

Where $\Gamma(k)$ is the Gamma function, and the mean is given by $k\theta$ and variance is $k\theta^2$. The Gamma distribution is of interest to us because the noise in the light curves seems to have a skewed distribution, and the Gamma distribution can also approximate a skewed distribution well.

Fitting a Gamma distribution is a bit simpler than fitting a Poisson distribution. We use the `scipy.stats.gamma.fit` function that can directly give us the shape and scale parameters of the distribution.

We can now take the noise as $k\theta + 3\sqrt{k\theta^2}$ and the signal as $M_s + k\theta$.

$$\therefore \text{SNR} = \frac{M_s + k\theta}{k\theta + 3\sqrt{k\theta^2}} \tag{3.6}$$

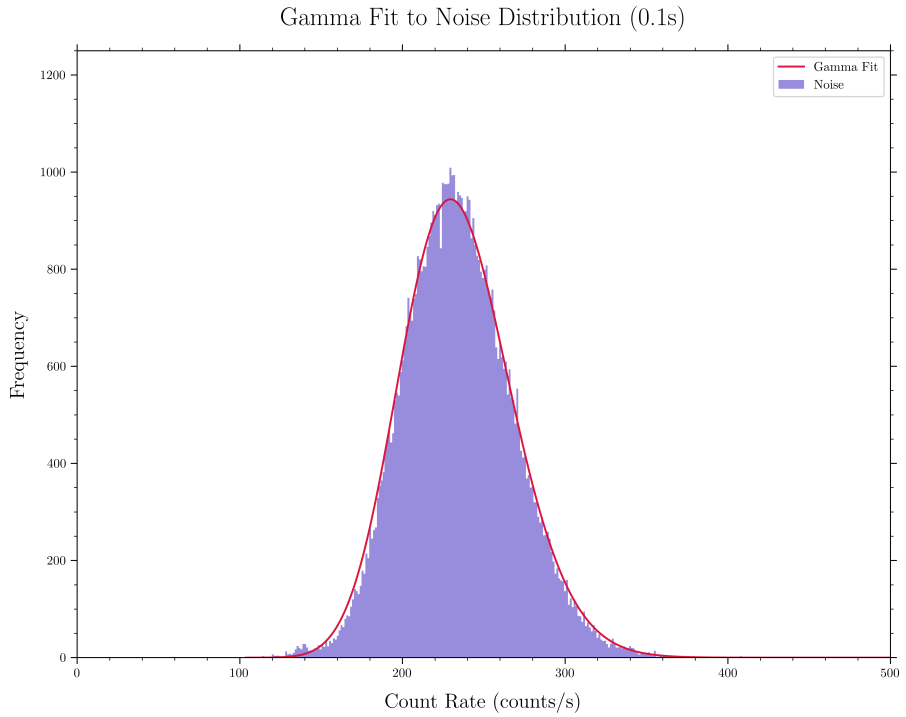


Figure 3.3: Fitting a Gamma Distribution to the Noise (Binsize 0.1s) of Quadrant 0

We can see that the Gamma distribution is very close to the Poisson distribution in terms of the fit, maybe even better. We will quantify the fits in future sections.