Analyzing Optically-Dark Short Gamma Ray Bursts

David Fitzpatrick[1], Alexander van der Horst[2]

[1] *Georgetown University, Department of Physics, 37th and O Streets NW, Washington D.C. 20057, United States*
[2] *The George Washington University, Department of Physics, 725 21st Street NW, Washington D.C. 20052, United States*

1 Abstract

While all gamma-ray bursts (GRBs) emit multi-wavelength afterglow radiation following an initial burst of gamma rays, some GRBs have afterglows which are markedly dark in the visible band of the electromagnetic spectrum compared to others (i.e. X-ray or radio bands). These optically-subluminous or "dark" GRBs could be caused by extinction of optical light, high redshift, or other properties internal to the host galaxy. Although many studies have been done for identifying dark long-duration GRBs (associated with the collapse of massive stars), little work has been done for identifying dark short-duration GRBs (associated with the merger of two compact objects such as neutron stars). As such, we present a comprehensive analysis and identification of optically-dark short GRBs of the 70 detected since the launch of NASA's Neil Gehrels *Swift* Observatory in 2004 through March 2015. We begin by illustrating how we computationally automate the calculation of the optical-to-X-ray spectral index, β_{ox}. We further discuss how we use β_{ox} to automate the classification of dark GRBs according to two different methods that are commonly used within the GRB community. Upon classification of 9 dark short GRBs, we discuss individually the unique properties surrounding each and theorize explanations for their optical darkness. By investigating trends between these 9 bursts' host galaxies, redshifts, UVOT lightcurves, and other relevant factors, it is clear that more comprehensive case studies must be done to pinpoint any distinguishing explanation for optical darkness. However, we find that the proportion of dark short GRBs ($\approx 13\%$) is notably less than that for long GRBs ($\approx 40\% - 60\%$), which agrees with the tenable compact binary merger progenitor model for short GRBs.[1]

2 Introduction

2.1 Overview

In the field of astrophysics, gamma-ray bursts (GRBs) are amongst the most captivating and cataclysmic phenomena in the universe. Primarily arising from supernovae or kilonovae, resulting from massive stellar collapses or compact binary mergers (i.e. when two neutron stars or a neutron star and a black hole merge) respectively, GRBs are extremely energetic explosions that yield very-concentrated beams of highly-energetic electromagnetic radiation (for an extensive review on GRBs, see Gehrels et al. 2009). While the initial burst of gamma rays of a GRB does not last very long, the power of GRBs is still extremely impressive; in sometimes only a few seconds (a GRB with a duration of ≤ 2 seconds is called "short"), the total energy released by a GRB is equivalent to the total amount of energy that would be released by the Sun during its entire 10 billion-year lifetime. After an initial flash of gamma rays, GRBs emit longer-lived afterglows at longer

[1] A recording of the presentation associated with this thesis is available at https://youtu.be/uf4KlX3rFi4.

wavelengths (focus is usually given to X-ray, optical/visible, and radio) which are a crucial point of study for most astrophysicists involved in GRB research.

2.2 What is a GRB?

The electromagnetic spectrum is used to characterize the many forms of electromagnetic (EM) radiation that can appear throughout the universe. The most familiar form of EM radiation is visible light, ranging from wavelengths of 400 [nm] to 700 [nm] and consisting of purple, red, and all the colors in between. Other forms of radiation - such as microwave, radio, or ultraviolet (UV) - might also be familiar, although these forms of radiation cannot be seen by the naked eye (Figure 1).[2]

The EM spectrum categorizes the various forms of EM radiation (or energy that travels and spreads out as it propagates) based on their frequencies (ν), wavelengths (λ), and photon energies (E), as per physicist's Max Planck's equation, $E = h\nu$, where $\nu = \frac{c}{\lambda}$.[3]

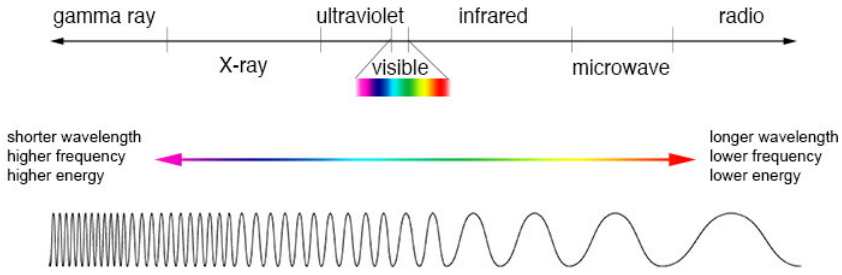


Figure 1: A visual depiction of the electromagnetic spectrum.

Gamma Ray Bursts (Figure 2)[4] derive their name from the form of EM radiation with the highest energy called gamma rays (seen all the way to the left in Figure 1). Lasting from fractions of a second to several minutes, GRBs are bright outbursts of radiation, with spectral energy distributions peaking in the gamma ray region, produced by energetic explosions in distant galaxies. These unpredictable and non-repetitive flashes of gamma rays come from random directions in the sky at random times, and when they do occur, they outshine all other sources of gamma ray radiation in the universe combined (Mészáros et al. 2014). Following the initial flash of gamma rays, GRBs then produce what are called "afterglows," or other forms of EM radiation emitted at longer wavelengths, that appear as point-like sources and can last from hours to months to sometimes even years in duration. These afterglows include X-ray (Costa et al.

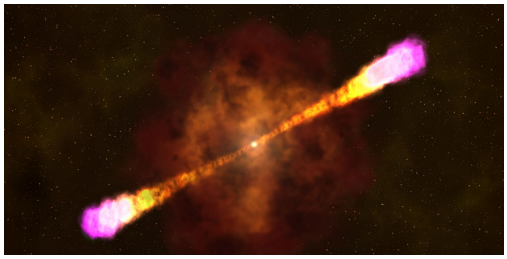


Figure 2: An artist's rendition of a GRB (NASA).

[2]From https://imagine.gsfc.nasa.gov/
[3]c, the speed of light, is equivalent to $299,792,458\ \left[\frac{\text{m}}{\text{s}}\right]$
[4]From https://aasnova.org/2019/04/03/the-variable-jets-of-gamma-ray-bursts/

1997), ultraviolet, optical (van Paradijs et al. 1997), infrared, microwave, and radio components (Frail et al. 1997).[5]

Combined with their afterglows, GRBs are the most luminous events known, capable of yielding directed streams of relativistic matter with kinetic luminosities exceeding 10^{53} [erg·s^{-1}] (Gehrels et al. 2009).[6] Spectroscopic observations of GRB afterglows also allow cosmologists to discern the cosmological redshifts[7] of their host galaxies and, in turn, their distances from Earth. Assuming that the radiation from GRBs is emitted isotropically (uniformly in all directions)[8], their energies E_{iso} range from $\sim 10^{52} - 10^{56}$ [erg], which is on a level comparable to the entire rest energy of the Sun (Atteia et al. 2017).[9] Because very few processes are powerful enough to be as bright as GRBs (or, for that matter, produce gamma rays at all), astrophysicists study GRBs in an effort to understand some of the most distant and violent events in the universe.

2.3 History of GRBs

The Vela Satellites Because the Earth's atmosphere disallows any gamma rays from reaching the surface, the first detection of a GRB did not come until the era of satellites which began during the Cold War between the United States and the Soviet Union. In an effort to detect nuclear detonations and monitor compliance with the 1963 Partial Test Ban Treaty by the Soviet Union, the U.S. launched the Vela satellites, which today are part of the contemporary Integrated Operational NuDet (Nuclear Detonation) Detection System (IONDS). On July 2$^{\text{nd}}$, 1967, the Vela III and IV satellites detected flashes of gamma rays with radiation signatures never before seen. Since it was determined that the gamma rays had not originated from a nuclear explosion on Earth, the leading investigators at Los Alamos Scientific Laboratory noted the data and put it into archives. However, as more Vela satellites were launched, the frequency of these unexpected spikes in gamma ray readings continued. Through analysis of the varying arrival times of bursts as recorded by each satellite, the investigators at the Los Alamos labs were able to work out estimations for the sky positions of 16 bursts (Klebesadel et al. 1973). From this, their 1973 publication *Observations of Gamma-Ray Bursts of Cosmic Origin* presented their conclusion that since these bursts did not correlate with any extreme solar activity, these so-called "GRBs" originated from outside our solar system (hence the phrase, "Cosmic Origin").

BATSE* and *BeppoSAX Following the Vela's serendipitous discovery of GRBs, NASA's Compton Gamma-Ray Observatory (CGRO) with its Burst and Transient Experiment (*BATSE*) made great strides in determining the distance to the origins of these bursts in its nine-year lifespan (1991-2000), recording the measurements for over 2700 GRBs (Fishman et al. 2005, Gomboc 2011). However, it was not until the

[5] Analysis of radio afterglows is important for understanding the Fireball Model (see *2.5 The Fireball Model*) and how relativity effects GRBs.

[6] Kinetic luminosity is the total amount of kinetic energy emitted per unit time (i.e. power) by a celestial body. In astrophysics, it is commonly expressed in terms of [erg·s^{-1}], where 1 [erg] = 10^{-7} [J] and 1 [erg·s^{-1}] = 10^{-7} [W].

[7] Due to the expanding nature of the universe, all wavelengths of light traveling from a source to an observer across the universe are stretched by a factor $\lambda_{\text{observed}} - \lambda_{\text{emitted}} = 1 + z$, where $z = (\lambda_{\text{observed}} - \lambda_{\text{emitted}}) / \lambda_{\text{emitted}} = \Delta\lambda / \lambda_{\text{emitted}}$ is the cosmological redshift describing the factor by which the universe has expanded between the time of emission and reception of light. A larger z refers to objects being farther away.

[8] While GRBs were originally predicted to emit isotropically, it was found later that GRB emissions are instead collimated, as is explained in *2.3 The History of GRBs*.

[9] The rest energy of the Sun $M_{\odot}c^2 \approx 1.8 \times 10^{47}$ [J] = 1.8×10^{54} [erg].

Italian-Dutch satellite *BeppoSAX* that scientists realized that the initial flash of gamma rays for a GRB was accompanied by a longer-lived afterglow of EM radiation at longer wavelengths. Soon after GRB 970228,[10] *BeppoSAX* measured a fading X-ray source coming from the same location as the GRB itself, which in turn led to the discovery of a fading optical component in the same region (Wijers et al. 1997). Using these afterglows with flux F that generally faded according to a power law ($F \propto t^{-\alpha}$, with $\alpha > 0$), astrophysicists were able take more precise spectroscopic measurements of the optical transients (the fading optical component of the afterglows), which allowed them to identify the redshifts of the GRB progenitors (the celestial objects that emit GRBs). Following the first redshift measurement of $z = 0.835$ for GRB 970508 (Metzger et al. 1997), it was found that the GRB redshift measurements had a median value of ≈ 1.0 (Kulkarni et al. 2000 and references therein; Fynbo et al. 2001). Since such a redshift refers only to objects situated outside our own galaxy, it was clear that these GRBs originated from galaxies far away from the Milky Way. This then comfortably confirmed the conclusion from *BATSE* that GRBs originate from outside our solar system and are indeed cosmological entities.

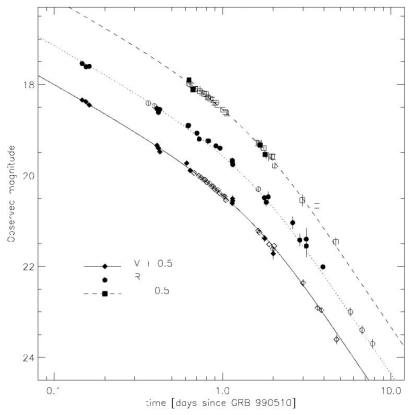


Figure 3: Optical light curves for the GRB 990510 showing the relationship between flux and time [days] and the characteristic jet break using three different filters. (Harrison et al. 1999).

Two years later, however, another measurement by *BeppoSAX* sent astrophysicists scrambling as calculations revealed that GRB 990123 had an $E_{\text{iso}} \approx 4.5 \times 10^{47}$ [J], which exceeded the restrictive quantity of Mc^2 for the progenitor[11] transformed to gamma ray emission (Hjorth et al. 1999). From this, scientists concluded that the belief that GRB emission is isotropic is actually erroneous, instead asserting the contemporary theory that GRB emissions are instead *collimated* (Frail et al. 2001). With GRB emissions restricted to two opposite jets with half-opening angle $\theta_j << 1$ radian, this lowers the required energy to a level within the restriction placed on it by Einstein's mass-energy equivalence.

Further observations of optical transients not only substantiated the notion of collimated GRB emissions, but also provided evidence that the jets were moving at relativistic speeds. Utilizing *light curves* to understand how the afterglow radiation evolves in time, astrophysicists noticed that many GRBs exhibited simultaneous breaks in their optical and X-ray light curves, a feature which is characteristic of collimated and relativistic ejecta (Piran 1999).[12] Due to its relativistic nature, the emission of a GRB is subject to the properties of relativistic beaming, or the process by which relativistic effects alter the apparent luminosity (brightness) of EM radiation moving close to the speed of light (beaming hinges on the properties of aberration as applied

[10]The notation for labeling GRBs is based on the date the burst was measured (i.e. GRB YYMMDD). If more than one burst is detected on a particular day, letters are appended at the end of the list of numbers. For example, GRB 130603B refers to the second of two GRBs recorded on June 3$^{\text{rd}}$, 2013.

[11]This restrictive quantity comes from Einstein's derivation of the mass-energy equivalence, which states that anything which has mass m has energy E given by the relation $E = mc^2$. Thus, GRB 990123 must not have an E which exceeds this bound, which it does when considering isotropic radiation emission.

[12]Typically, astrophysicists use the Jansky [Jy], named after Karl Guthe Jansky, to measure flux. 1 [Jy] = $[10^{-26}$ W/ $(\text{m}^2 \cdot \text{Hz})]$

to electromagnetism).[13] For relativistic ejecta with a Lorentz Factor[14] γ, the beaming angle is $\theta_b \approx \frac{1}{\gamma}$ radians (Piran 1999). Initially, when $\theta_b < \theta_j$, an observer on Earth will be able to view only a small patch of emission that is inside an angle θ_b, rendering both spherical and collimated outflow tantamount to one another. However, as the particles travel through the interstellar medium of space, the ejecta begins to slow down, eventually reaching the point such that γ decreases and $\theta_b = \theta_j$. At this point, the observer can begin to see the edge of the jet cone, thereby receiving less light than they would in the case of spherical outflow. Once $\theta_b > \theta_j$, this results in a steepening in the decay of the light curves for all wavelengths of a GRB afterglow that is called the *achromatic* break or *jet break* (Figures 3 and 4). From these jet breaks, astrophysicists have been able to calculate θ_j for many GRBs, which have values ranging from $\sim 1°$ to $25°$ with a strong concentration near $4°$ (Frail et al. 2001). Moreover, since we can only measure GRBs when they happen to point in the direction of Earth, extensive data on beaming angles has led to the prediction that there are $\sim 100 - 1000$ GRBs occurring in the visible universe on a daily basis (Piran 1999).

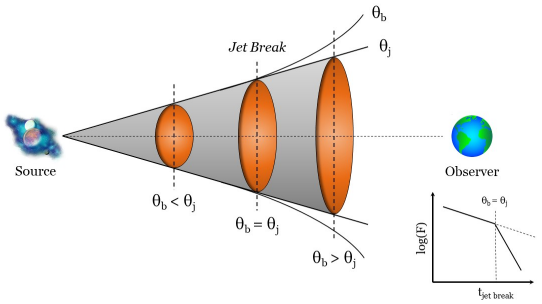


Figure 4: A visual depiction of the jet break as it relates to the relationship between θ_b and θ_j side by side with an approximated graph of an afterglow emission light curve.

The Modern Era: *Swift* and *Fermi* Today, astrophysicists primarily rely on the Neil Gehrels Swift Observatory (commonly referred to as "*Swift*"; Figure 5)[15] and the Fermi Gamma-ray Space Telescope to obtain GRB data. With *Swift*, a NASA mission with international participation launched in 2004 and designed specifically for GRB discovery, scientists are able to detect GRBs within seconds of the first gamma rays reaching Earth. Combining a wide-field hard X-ray burst detection telescope with narrow-field X-ray and UV-Optical telescopes (UVOT), *Swift* contains a powerful burst detection al-

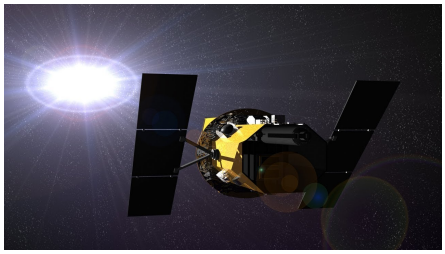


Figure 5: An artist's rendition of the Neil Gehrels Swift Observatory.

[13]Aberration is the change in an object's apparent direction due to the relative transverse motion of an observer. Say, for example, you are stopped at a red light while driving in a rainstorm. Since you are not moving transverse (perpendicular to) the direction of the rainfall, the rain appears to fall exactly vertically on the hood of your car. If, however, you return to driving on the highway, the rain appears to be angled towards you because you are moving transverse to the direction of rainfall at highway speeds. In the context of relativistic ejecta, emission aberration will beam the ejected particles in the direction the jet is traveling due to the expansive nature of the universe. Thus, in the Earth's rest frame, observations will find there to be more energy in an emission than there truly is.

[14]The Lorentz factor, a consequence of Einstein's special relativity, is the factor by which time, length, and relativistic mass change for an object that is moving. Mathematically, for speed v and the speed of light c, this is given by $\gamma = 1/\sqrt{1 - v^2/c^2}$

[15]From https://www.youtube.com/watch?v=eMl4qajkz7E

gorithm which allows it to autonomously repoint itself so that it can begin charting X-ray and optical observations within ≈ 1 minute after the initial burst trigger (Gehrels et al. 2009). Once *Swift* makes such a significant detection of gamma rays indicating a GRB with its Burst Alert Telescope (BAT) instrument, it relays the information location via the Gamma Ray Coordinates Network (GCN) to telescopes on the ground, thereby allowing for both space-based and ground-based stations to measure accurately the event from as early on as possible. This collaboration paints an incredibly precise and desirable picture for scientists of many GRBs, especially the previously-elusive short GRBs.

The Fermi Telescope, originally known as the Gamma-ray Large Area Space Telescope (GLAST) but renamed in honor of high-energy physicist Enrico Fermi, was launched in 2008 to supplement *Swift*. Situated in low-earth orbit, *Fermi* is a joint venture between NASA and many international participants, and is the most sensitive gamma-ray telescope on orbit. While *Fermi* does not have the capability for obtaining UV/Optical data, its instruments do allow for it to measure frequencies at an even higher level than *Swift* can. In addition, *Fermi*'s main instrument is the Large Area Telescope (LAT), which astrophysicists have used to perform an all-sky survey to study cosmological phenomena of many kinds. This wide-field capability has allowed *Fermi* to resolve the gamma-ray sky, as well as advance research regarding supermassive black-hole systems, pulsars, the origin of cosmic rays, and many more astrophysical phenomena.

2.4 Long and Short GRBs

One of the more confounding characteristics of GRBs is that no two GRBs have identical gamma ray light curves. As can be seen in a sampling of gamma ray light curves in Figure 6 for an assortment of GRBs, some graphs seem to follow a discernible power-log relationship, others a roughly normalized distribution, and yet others seem to have no pattern at all. Although some progress has been made on reproducing GRB light curves using modeling, most astrophysicists generally conclude that there is still much to learn about the vast diversity displayed by GRBs.

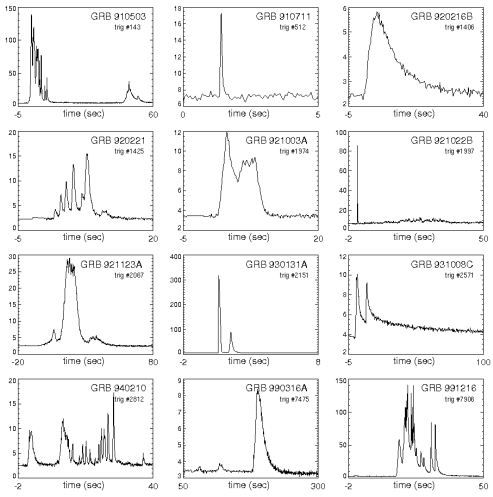


Figure 6: Light curves for an assortment of GRBs illustrating the varying relationship between time and number of gamma rays detected (NASA Marshall Space Flight Center/Space Sciences Laboratory).

Despite the ambiguity which arises from the GRB light curves, one important clue can be gleaned from plotting the flux of gamma rays F_γ against time, which shows rapid variability in F_γ on a timescale of milliseconds. Since variability in Δt cannot be produced in an area which exceeds the distance light travels in this time, the size of the source was originally estimated to be $D \leq c\Delta t \approx 300$ [km] (Rees & Mészáros 1992). However, given the incredible energies involved, relativistic effects must be taken into account when determining the progenitor diameter. Such an inclusion of relativity then reveals that the observed $D \approx 300$ [km] is actually a factor of γ^2 smaller than the real diameter, yielding $D_{\text{real}} \approx 300\gamma^2$ [km] (Rees & Mészáros 1992).

While *BATSE* was in commission, astrophysicists were able to use large sampling of GRBs to determine that they can be divided into two broad categories based on their duration using the parameter T_{90}, the magnitude of the time interval over which a burst emits 90% of its fluence[16] (Kouveliotou 1993). Such a bimodality in T_{90} is clearly seen in Figure 7[17] taken from the *Fourth BATSE Gamma Ray Burst Catalog* (Paciesas et al. 1996).

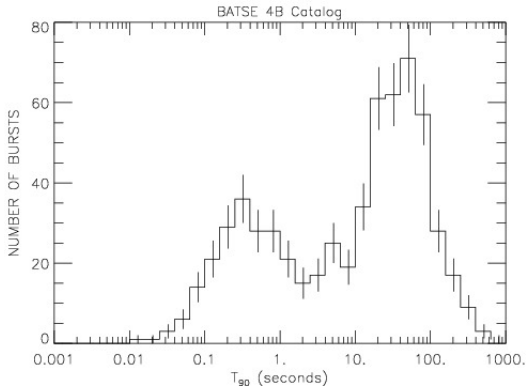


Figure 7: The relationship between number of GRBs and T_{90} clearly displaying a bimodality in this parameter for GRBs (Paciesas et al. 1996).

Today, astrophysicists characterize this bimodality by defining two classes of GRBs based on their T_{90} distribution: **short GRBs**, or those which have a $T_{90} \lesssim 2$ seconds (and an average duration of approximately 0.3 seconds), and **long GRBs**, or those which have a $T_{90} > 2$ seconds (and an average duration of approximately 30 seconds) (Kouveliotou et al. 1993). These two disparate categories have very different causes, which are summarized in Figure 8 and are further discussed below.

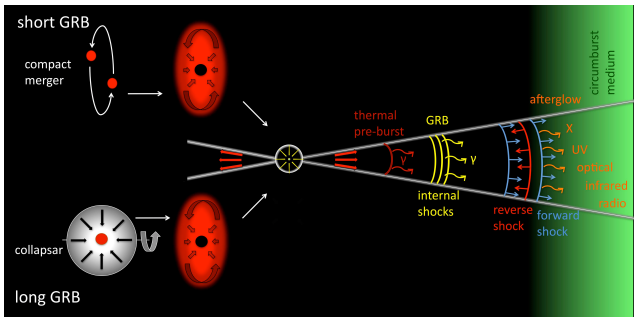


Figure 8: Progenitor models for short and long GRBs combined with a visual depiction of the current understanding of GRB afterglows (Gomboc 2011).

Long GRBs and Collapsars Most GRBs ($\approx$ 70 %) are defined to be long GRBs. Due to their grand temporal range, long GRBs are also those which produce the brightest afterglows, which allows scientists to understand them more fully than their shorter counterparts. Moreover, their abundance has also led scientists to discover that nearly every long GRB is a direct result of a core-collapse supernova, or the death

[16]Fluence (H_e), or radiant exposure, is the burst's irradiance ($F_\nu = \frac{\partial \Phi_\nu}{\partial A}$; in units of flux or power per unit area) integrated over the total time duration of the event (T). Mathematically, this is defined as $H_e = \int_0^T F_\nu dt$

[17]From https://gammaray.msfc.nasa.gov/batse/grb/duration/

of an old massive star ($\gtrsim 25M_\odot$) that explodes after it has collapsed in on itself (predicted by Woosley 1993; observed by Galama et al. 1998, Hjorth et al. 2003). Specifically, these kinds of collapsars (also known as hypernova) which produce long GRBs refer to those stars which have exploded in a Type Ic supernova (which arise when a star's core *collapses*).

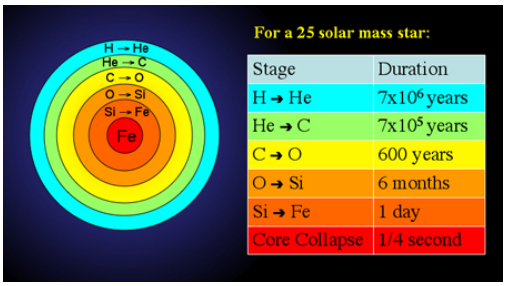


Figure 9: A visual depiction of the shrinking size of a star's core during core collapse for a star with mass $25M_\odot$.

Regardless of the mass of a star, all main sequence stars generally pass through the first stage of their lives in a similar fashion: carrying out nuclear fusion to convert hydrogen into helium. Utilizing their incredible temperatures, stars undergo nuclear fusion in order to have their thermal pressure balance the increasing central gravitational force acting inwardly on the star's core. However, once all the hydrogen is used up, the decrease in the amount of energy released by hydrogen fusion results in gravity forcing the core to contract to a smaller size. This phase is also accompanied by the gas layers around the core expanding dramatically, thereby forming a Red Giant. With the contraction of the core, the temperature of the core increases further, eventually reaching a point which allows for the beginning of Helium fusion into carbon. For small stars with masses less than a few $M_\odot$, the process ends here, and the star will eventually form a white dwarf incapable of producing GRBs. For a larger star, however, this process will continue, leading to nuclear fusion for carbon, oxygen, silicon, and eventually iron (Figure 9).[18] Each fusion stage is shorter than the previous, and with the conclusion of each fusion stage, gravity compresses the star's core into an even smaller amount of space.

Because producing heavier nuclei such as iron requires the input of energy, the onset of iron fusion marks the conclusion of fusion for the star. Therefore, while previously the tremendous mass of the star was supported against gravity by fusing lighter elements into heavier ones, the star then no longer releases energy sufficient to withhold gravity, and the core implodes. The core collapses further until the density exceeds the nuclear density (the density of protons and neutrons in the nucleus). The strong nuclear force then becomes repulsive, reversing the collapsing motion of the star and accelerating it outwards. The shock wave produced by this violent reversal results in the spectacular and colorful cosmic explosions which astrophysicists refer to as supernovae (Icko Iben Jr. 2013; Chaisson et al. 2017).

In this mere fraction of a second that the star begins its final collapse sequence, the temperature in the star's core reaches a point capable of producing gamma rays (Woosley et al. 2006). Thus, study of these GRBs can provide much information about the supernovae themselves, as well as about the galaxies from which these supernovae originate.

Short GRBs and Kilonovae Those GRBs that last less than ≈ 2 seconds are classified as short GRBs. These kinds of bursts, now known to account for $\approx 30\%$ of measured GRBs, were at first very difficult for scientists to measure due to their afterglows being uniformly and markedly fainter than those of long GRBs. However, once the Swift Observatory launched in 2004, scientists were able to garner much more data on

[18] From http://astronomy.swin.edu.au/cosmos/c/core-collapse

these objects and provide substantiated theories surrounding their progenitors.

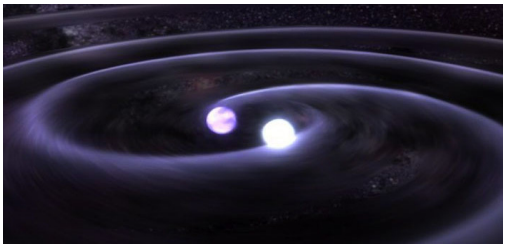


Figure 10: An artist's rendition of a two neutron stars circling each other just before a binary merger.

With a detection rate of around 100 GRBs per year, *Swift* has allowed scientists to come to the general consensus that the majority of short GRB progenitors are compact binary mergers (i.e. when two neutron stars or a neutron star and a black hole merge into one another) that result in kilonovae (Nakar 2007, Metzger et al. 2010). In such binary systems of two neutron stars (the smallest and densest stars arising from the collapse of a giant star) or a neutron star and a black hole, as gravity draws the two objects closer, they begin to circle each other at increasing speeds (Figure 10).[19] The distance between the binary pair continues to decrease until eventually the two objects collide, resulting in a catastrophic collision that generates two highly-concentrated relativistic jets of energy collimated in opposite directions (i.e. a GRB). This collision also results in the devastating outflow of the kilonova burst itself, a term which was introduced to characterize its peak brightness (Metzger et al. 2010). Kilonova have been measured to be 1000 times brighter than a classical nova,[20] and $\frac{1}{10}$ to $\frac{1}{100}$ the brightness of a typical supernova (Chaisson et al. 2017).

Since neutron stars are primarily composed of neutrons, the collision of two neutron stars which results in a kilonova creates conditions that allow for a fascinating process known as rapid neutron capture ("r-process").[21] With all these subatomic building blocks flying around immediately after the binary merger, much of the radioactive neutron star material sticks together, creating new elements. However, before these elements decay, the "r-process" allows for more neutrons to attach themselves to these newly formed elements in the chaos of the collision, thereby creating even more elements with greater masses (Iben Jr. 2013, Chaisson et al. 2017). Kilonovae, therefore, are the main synthesizers of heavy elements such as gold, platinum, lead, uranium, and silver. Formation of such elements has perplexed scientists for a long time, but with spectroscopic measurements from the Hubble Space Telescope, Gemini Observatory, and the ESO's Very Large Telescope, scientists now have conclusive evidence that kilonovae are responsible for forming the heaviest elements in the universe.

In the summer of 2013, the first kilonova was found when a short gamma ray burst GRB 130603B was detected by instruments onboard *Swift* and confirmed by other detectors to have arisen from a compact binary merger (Tanvir et al. 2013). More significantly, however, the kilonovae progenitor theory for short GRBs was bolstered when GRB 170817A was detected only 1.7 seconds after the detection of gravitational wave GW170817, which LIGO and VIRGO measured to have occurred from the merger of two neutron stars that resulted in a kilonova (Abbott et al. 2017). The gravitational waves produced by such a merger immediately preceded *Fermi*'s measurements of a short GRB, thereby affirming that the sources of these short bursts are indeed such compact binary mergers (Abbott et al. 2017).

[19] From www.lanl.gov/org/ddste/aldsc/computer-computational-statistical-sciences/computational-physics-methods/

[20] A nova is an explosion from the surface of a white-dwarf star in a binary star system that results when the white dwarf "steals" gas from its nearby companion star, which yields a dense but shallow atmosphere around the white dwarf. After the white dwarf thermally heats the matter in this atmosphere to a critical temperature, rapid runaway fusion occurs, expelling the atmosphere into interstellar space and creating the bright nova burst.

[21] From https://science.howstuffworks.com/kilonovas-are-biggest-baddest-stellar-blasts-in-space.htm

2.5 The Fireball Model

Since the energy levels of GRBs are so extreme that they cannot be explained by thermal processes, astrophysicists have had to put forth some rather complicated theories to explain how so much energy can originate from such a small area. The Fireball Model (Figure 11)[22], considered to be the most complete theory to date, uses two different shock wave models to explain both the prompt gamma ray emission of the GRB (caused by the internal shocks) and its afterglow (produced by the external shocks). The mechanism by which a GRB occurs is precisely what led to this theory's name: a fireball of ultra-relativistic energy consisting of optically thin material[23] with very few baryons[24] (Rees & Mészáros 1992, Mészáros & Rees 1992). While this fireball ends up masking the identity of the inner engine, or that which expels all the energy, the model does predict that the inner engine must be an extremely compact source, a claim which is consistent with the association of GRB progenitors with collapsars and compact binary mergers.

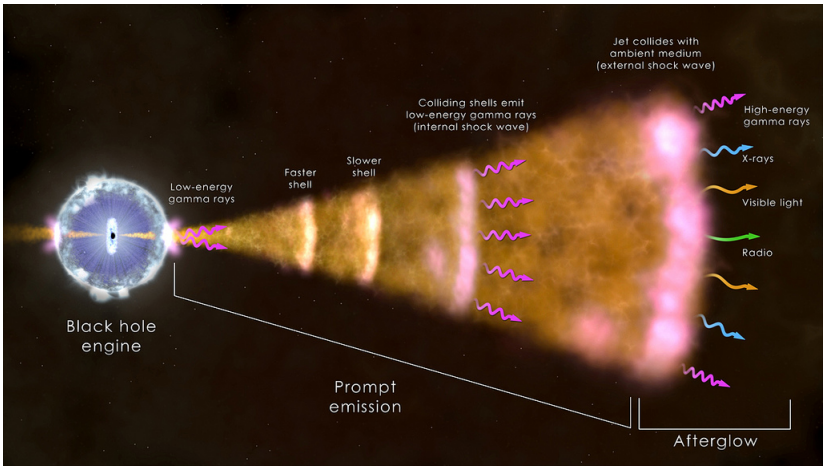


Figure 11: An illustration of the Fireball Model detailing the internal and external shocks in their relation to the prompt gamma ray emission and the proceeding afterglow emission (Daniel Kocevski, NASA GSFC).

Internal Shocks Internal shocks are the mechanism for the production of the prompt gamma ray emission that precedes the afterglow. Following the initial GRB event, shock waves traveling at different relativistic speeds ($\approx 0.99995c$ with $\gamma \sim 100$) emanate from the inner engine, converting raw kinetic energy into gamma-ray photons. The difference in speeds of the shock waves then force them to interact with one another as they travel, producing inverse Compton[25] and synchrotron emission[26] that further increase the energies of the gamma rays (Piran 1999, Rees & Mészáros 1992, Mészáros & Rees 1992). Moreover, the inverse Compton emission causes the earlier shock waves to eventually slow down, thereby adding even more energy to the gamma rays as the amount of interactions between the earlier shock waves and the later ones increases.

[22] From https://asd.gsfc.nasa.gov/conferences/fgo2/program/DKocevski.pdf

[23] A medium is said to be optically thin if, on average, a photon can pass through the medium without absorption.

[24] A baryon is a type of composite subatomic particle defined by the Standard Model of Particle Physics. With properties such as containing an odd number of valence quarks (at least three), belonging to the hadron family, and being fermions (half odd-integer spin), baryonic matter is pretty much all matter encountered in everyday life. The two most common examples of baryons are protons and neutrons.

[25] Inverse Compton emission/scattering is when low energy photons are scattered to high energies by ultrarelativistic electrons so that the photons gain energy and the electrons lose energy.

[26] Synchrotron emission is a type of non-thermal radiation created by electrons spiralling around magnetic field lines at relativistic speeds. Since the spiralling motion of these electrons causes them to always be changing directions, they are accelerating and emitting photons with energies corresponding to the instantaneous kinetic energies of the electrons.

External Shocks The external shock waves are the mechanism for the production of the afterglows such as those first detected by *BeppoSAX* in the late 1990s. As the internal shock waves emanate from the inner engine, they eventually interact with dust/gas in the interstellar medium, transferring their energy into this matter. As the internal shocks sweep up this material, the so-called "external shocks" are produced as these particles in the interstellar medium emit radiation; spectroscopic analysis of these emissions reveals that they correspond to the emission spectra found in the GRB afterglows (Rees & Mészáros 1992, Mészáros & Rees 1992, Piran 1999). The incredible initial amount of energy in the internal shock waves gives rise to the rather lengthy duration of the afterglows, as well as to the reason for why the afterglows cover the entirety of the EM spectrum.

2.6 Dark Bursts

Soon after the first discovery of an optical counterpart to a GRB (van Paradijs et al. 1997), scientists were perplexed when they were unable to find an optical transient for GRB 970828 (Groot et al. 1998). This strange trend continued for many other GRBs, leading to the idea that, mysteriously, not all GRBs are accompanied by optical transients. Specifically, these so-called "dark" or "optically-dark" bursts, or those which lack significant afterglow emission in the visible spectrum (Fynbo et al. 2001), significantly account for $\approx 40\%$ - 60% of all long GRBs (Roming & Mason 2006, Greiner et al. 2010). While a substantial fraction of these might be explained by a lack of observational sensitivity or a large delay between GRB trigger time (when the initial gamma rays of the GRB reach Earth) and the start of actual observations, optically-dark GRBs were (and, to an extent, still are) a puzzle for astrophysicists, leading to many hypotheses about their occurrence. One possible explanation is that the darkness of GRBs is caused by dust in the host galaxy blocking optical afterglow emissions. Another idea is that dark bursts originate at high-redshifts ($z \geq 5$), thereby shifting the UV band into the optical band and masking the trace of optical emission (Jakobsson et al. 2004). Still another hypothesis attributes this darkness to the claim that some GRBs are simply intrinsically faint at optical wavelengths. Despite the many questions which arise from these conjectures, two methods persist today that are widely accepted in the scientific community to define optical darkness, which we shall refer to as the **Jakobsson method** (Jakobsson et al. 2004) and the **Van der Horst method** (Van der Horst et al. 2009).

2.6.1 The Jakobsson Method

With the onset of *Swift*, astrophysicists soon found that all frequency bands of GRB afterglows generally decay according to a simple power law defined by the relationship $F_\nu \propto \nu^\beta$, where ν is the radiation frequency, F_ν is the corresponding radiative flux density[27], and β, a unitless value which is typically < 0 for GRB afterglows, is the spectral index of the source. For both the Jakobsson and Van der Horst methods, the classification of a dark GRB relies heavily upon a specific type of spectral index known as the optical-to-X-ray spectral index β_{ox} defined by

$$\frac{F_x}{F_o} = \left(\frac{\nu_x}{\nu_o}\right)^{\beta_{ox}} \tag{1}$$

where the "o" stands for "optical" and the "x" stands for "X-ray." [28] Note that the optical spectral index

[27] The radiative flux density F_ν is the radiative flux per unit frequency and is commonly given in cgs units of $\left[\text{erg} \cdot \text{cm}^{-2}\text{s}^{-1}\text{Hz}^{-1}\right]$.

[28] See the section titled *Calculating* β_{ox} for a more detailed calculation of β_{ox}.

β_o and the X-ray spectral index β_x[29] are characterized similarly by

$$F_o \propto (\nu_o)^{\beta_o}\,; \qquad F_x \propto (\nu_x)^{\beta_x} \tag{2}$$

Note here that while β_o and β_x are the spectral indices which best fit the flux curve only for the optical and X-ray bands respectively, β_{ox} is the index which best fits the region between the optical and X-ray bands.

According to most variations of the Fireball Model, the distribution of relativistic electrons with a Lorentz factor γ is given by:

$$n_e\,(\gamma)\,\mathrm{d}\gamma \propto \gamma^{-p}\mathrm{d}\gamma \tag{3}$$

where n_e is the number density and p is known as the energy distribution index of electrons (Sari et al. 1998, Li & Chevalier 1999). To keep the energy of the electrons finite and to prevent diverging at large γ, the Jakobsson model places a restriction on the energy distribution of electrons such that $p > 2$ (Sari et al. 1998). According to the Fireball Model, as an electron travels in the afterglow, synchrotron emission scattering causes the electrons within the internal shock waves to lose a small fraction of their energy over time. The magnitude of an electron's energy loss through synchrotron emission is directly proportional to the total energy of the electron, such that those electrons with greater energy lose more energy to synchrotron emission than electrons with less energy. Thus, as the energy distribution loses electrons at the high end from synchrotron emission, it begins to deviate from the simple power law (with slope p) given in Equation 1, resulting in a steeper and more negative slope at higher energies (Region H in Figure 12). This shift in slope, changing from $\frac{p-1}{2}$ to $\frac{p}{2}$, occurs at the cooling break and is marked by the characteristic frequency ν_c.

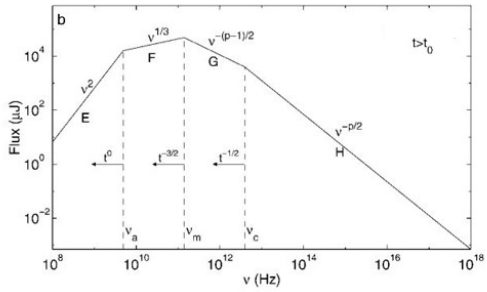


Figure 12: The relationship between frequency and flux for relativistic electrons in GRB afterglows (Sari et al. 1999). We focus on regions G and H.

Although Figure 12 illustrates an afterglow spectrum with a cooling break that occurs somewhere in the infrared region, this break may occur at other frequencies for other afterglows, including in the optical or X-ray bands. As such, it is necessary to take the cooling break into account when considering how we expect the afterglow radiation to decay (i.e. the possible values for the spectral index corresponding to any frequency band β). Thus, analysis of afterglow data reveals that the general spectral index β varies according to

[29] β_x comes directly from the data collected by *Swift*, and is the power law index which best fits the flux curve for the X-rays.

$$\beta = \begin{cases} \frac{p-1}{2}, & \nu < \nu_c \\ \frac{p}{2}, & \nu > \nu_c \end{cases} \tag{4}$$

where ν is the generalized electron frequency. Letting p go to its most extreme allowed value of 2, β is found to be, at a minimum, equal to 0.5. Therefore, the Jakobsson Method concludes that those bursts which are fainter than expected by the Fireball Model and are classified as **optically-dark are those which have $\beta_{\mathbf{ox}} < \mathbf{0.5}$**.

2.6.2 Van der Horst Method

The critical difference between the Van der Horst Method and the Jakobsson Method is that the former derives its requirement for optical darkness without placing any restrictions on p; that is, the Van der Horst Method allows for $p < 2$. This is supported by the fact that while in most cases it is true that p is found to be larger than 2, it has been shown that $p < 2$ is indeed observed, and can be accounted for theoretically by incorporating a high-energy cutoff in the electron energy distribution (Van der Horst et al. 2009).

In addition to letting $p < 2$, the other critical assumptions made by the Van der Horst method are that (1) the primary emission method for both the optical and X-ray transients is synchrotron emission and (2) that both transients originate from the same source. In this way, given a certain value for the X-ray spectral index β_x, the optical spectral index β_o must be either equivalent to β_x or 0.5 less than β_x if there is a cooling break in between the optical and X-ray regions (Van der Horst et al. 2009). For the latter case, the term of 0.5 is derived from:

$$\Delta\beta = \beta_{\nu>\nu_c} - \beta_{\nu<\nu_c} = \frac{p}{2} - \frac{p-1}{2} = 0.5 \tag{5}$$

Since β_{ox} defines the region between the optical and X-ray bands, we then expect that, for all GRBs, β_{ox} has the following characteristics:

$$\beta_{ox} = \begin{cases} \beta_x, & \nu < \nu_c \\ \beta_x - 0.5, & \nu > \nu_c \end{cases} \tag{6}$$

so that $\beta_x - 0.5 < \beta_{ox} < \beta_x$. Therefore, in a plot of β_{ox} vs. β_x, the Van der Horst method expects all bursts to lie in the region between the two parallel lines $\beta_{ox} = \beta_x$ and $\beta_{ox} = \beta_x - 0.5$. However, not all bursts do, and it is those bursts with a value of **$\beta_{\mathbf{ox}} < \beta_{\mathbf{x}} - \mathbf{0.5}$ which are classified as optically dark.**[30]

Graphically, this refers to the shaded regions in Figure 13:

[30] One of the more interesting things of the Van der Horst method is that it also identifies those bursts with $\beta_{ox} > \beta_x$ as optically-superluminous, of which a very small number have been found experimentally.

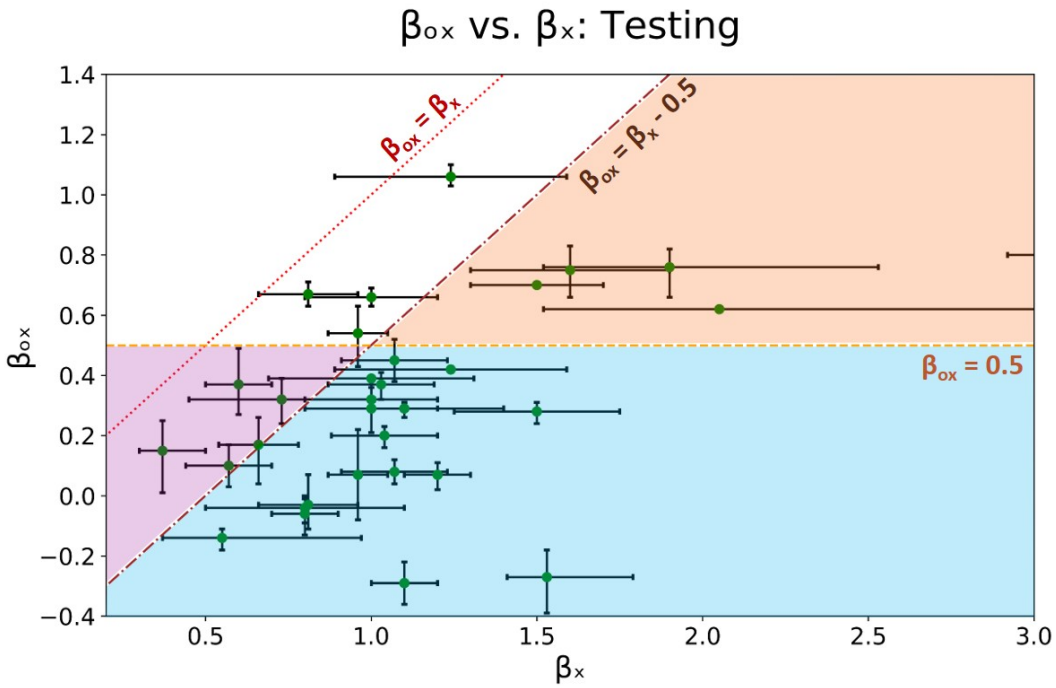


Figure 13: A graph of β_{ox} vs. β_x for various GRB data points in green (with the corresponding error bars in β_x, $\sigma_{\pm x}$, and in β_{ox}, $\sigma_{\pm ox}$) illustrating the optically dark region as defined by the Jakobsson method in purple and blue and the Van der Horst method in orange and blue.

Although much exploration of identifying optically-dark long GRBs has already been done (Greiner et al. 2010), astrophysicists' catalog of optically-dark short GRBs is far less expansive. As such, my research focuses on (1) applying both the Jakobsson and Van der Horst methods to identify optically-dark short GRBs of the 70 which have been detected since the launch of *Swift* in 2004 through March 2015 and (2) analyzing the distribution and the characteristics of dark short GRBs in an effort to glean any unique traits which may be directly related to their darkness.

3 Methodology

3.1 Overview

Calculating β_{ox} and the ensuing darkness classification for a large data set of GRBs required the development of C++ code and Python code, both of which rely on the mathematical theory presented here. Created as a feeder to the Python code, the C++ code directs the user to read in files of optical and X-ray measurements of GRBs, create optical-X-ray pairs of GRB data points within the allowed temporal separation, calculate β_{ox}, and output a file which contains each paired GRB data point with its relevant parameters (including β_{ox} and its corresponding uncertainty, $\sigma_{\pm ox}$[31]). The Python code then reads in this file and provides the user the ability to generate graphs of β_{ox} vs. β_x using either the Jakobsson or Van der Horst methods for

[31]Note that while we use $\sigma_{\pm ox}$ to represent the uncertainty of β_{ox} and σ_x to represent the uncertainty of F_x, we use $\sigma_{\pm x}$ to represent the often-asymmetric uncertainty of β_x.

user-defined assortments of data.

3.2 Automating the Calculation of β_{ox}

3.2.1 Input Files

Because the largest amount of data was taken from the Short GRB catalogs in Fong et al. (2015), the C++ Code is tailored to deal with input delimited text files of identical form to many of the tables presented by Fong et al. (2015), as shown by Table 1.[32] When running the code, the user is prompted to input these file names for loading in the order that they are presented in Table 1. If the user inputs a file name that does not load successfully (i.e. spelling mistake, file not in proper directory, etc.), the user is prompted to try again until a file is successfully loaded.

File Name	Fong et al. (2015) Correspondence	Fields	Description
`XRayData.txt`	Table 6	GRB ID; Δt_x [s]; Exposure Time [s]; F_x [μJy]; σ_x [μJy]	Text file cataloging GRB X-ray afterglow.
`BetaXData.txt`	Table 2	GRB ID; β_x; σ_{+x}; σ_{-x}	Text file cataloging GRB X-ray spectral indices.
`OpticalData.txt`	Table 7	GRB ID; Δt_o [hr]; Telescope; Instrument; Filter; Exposure Time [s]; F_o [μJy]; σ_o [μJy]	Text file cataloging GRB optical afterglow.
`FilterInfo.txt`	N/A	Telescope; Instrument; Filter; Wavelength [nm]; Frequency [Hz]	Text file categorizing optical telescope filters.

Table 1: A summary of the files to be read by the C++ Code (in order of appearance).

3.2.2 GRB Class

For ease of data manipulation, we introduce the GRB class in C++, which is defined by Table 2.
When the first file corresponding to the X-ray measurements is read in, a GRB object is created corresponding to each line of that file, with attributes assigned to the corresponding fields in each line. These GRB objects are then appended to a vector of GRB objects, which provides an organized structure to the data which is read in.

3.2.3 User-Defined Temporal Separation

As soon as the Swift Observatory detects a GRB, it signals the burst's location to many ground-based optical telescopes so that measurements of the afterglow can be made across the EM spectrum around the world. However, since the same instrument cannot measure both F_x and F_o at the same time, astrophysicists are forced to handle and reconcile X-ray data points with optical data points which correspond to different times and different telescopes.

In order to calculate β_{ox}, it is necessary to pair X-ray data from a GRB with optical data for the same GRB because both F_x and F_o and their corresponding uncertainties are required for the calculation. Therefore,

[32] A delimited text file contains lines corresponding to a single entity (in our case, a GRB measurement) which consist of fields separated by the delimiter (usually a tab or a space). All such .txt files were generated from CSV (comma-separated values) files.

C++: class GRB	
Attribute	Description
`string GRB_ID`	GRB ID Number
`double dt_XRay`	X-ray Time Δt_x [hr]
`double ExpT_XRay`	X-Ray Exposure Time [s]
`double F_x`	X-Ray Flux F_x [μJy]
`double sigma_x`	X-Ray Flux Uncertainty σ_x [μJy]
`double Beta_X`	X-Ray Spectral Index β_x
`double Beta_X_upper_sigma`	X-Ray Spectral Index Upper Uncertainty σ_{+x}
`double Beta_X_lower_sigma`	X-Ray Spectral Index Lower Uncertainty σ_{-x}
`string References_XRay`	Fong et al. (2015) X-Ray References
`double dt_Opt`	Optical Time Δt_o [hr]
`string telescope`	Optical Telescope Name
`string instrument`	Telescope Instrument Name
`string filter`	Optical Filter Name
`double ExpT_Opt`	Optical Exposure Time [s]
`double F_o`	Optical Flux F_o [μJy]
`double sigma_o`	Optical Flux Uncertainty σ_o [μJy]
`string References_Opt`	Fong et al. (2015) Optical References
`double frequency_XRay`	X-Ray Frequency [Hz] ν_x
`double frequency_Opt`	Optical Frequency [Hz] ν_o
`double Beta_OX`	Optical-to-X-Ray Spectral Index β_{ox}
`double sigma_OX_upper`	Upper Uncertainty for β_{ox}
`double sigma_OX_lower`	Lower Uncertainty for β_{ox}

Table 2: The attributes for the C++ GRB class used to prepare GRB objects for pairing.

when pairing X-ray data with optical data, it is essential to take into account the temporal separation between the two measurements. For example, if an X-ray measurement was taken at $t_x = 1$ [hr], and optical measurements were taken at $t_o = 1.5$ [hr] and $t_o = 2$ [hr], the user may want to pair the X-ray measurement with the $t_o = 1.5$ [hr] data point and not the $t_o = 2.0$ [hr] depending on the desired temporal separation.

In addition to being prompted to input file names for loading, the user is asked to define the maximum allowed temporal separation for pairing X-ray measurements with optical measurements. We define this temporal separation, $\Delta t_{\%}$, as the percent difference between the X-ray measurement time, Δt_x, and the optical measurement time, Δt_o (both relative to the beginning of afterglow measurements). Specifically, this is given as:

$$\Delta t_{\%} = 100\% \cdot \frac{|\Delta t_x - \Delta t_o|}{\Delta t_o} \tag{7}$$

Therefore, in order for a pair to be made, the calculated $\Delta t_{\%}$ for some set of X-ray data points and optical data points must be within this user-defined bound.

3.2.4 Calculation of β_{ox}

Both the Jakobsson and the Van der Horst methods assert that optical darkness hinges on the relationship between the optical flux F_o and the X-ray flux F_x and, more importantly, the value of the optical-to-X-ray spectral index β_{ox}. To calculate this, we first note again that the relationship for general spectral index β is given as

$$F_\nu \propto \nu^\beta \tag{8}$$

For β_o and β_x, this simply becomes

$$F_o \propto (\nu_o)^{\beta_o}\,; \qquad F_x \propto (\nu_x)^{\beta_x} \tag{9}$$

For β_{ox}, this relationship can be transformed to

$$\frac{F_x}{F_o} = \left(\frac{\nu_x}{\nu_o}\right)^{\beta_{ox}} \tag{10}$$

$$\beta_{ox} = \frac{\log\left(\frac{F_x}{F_o}\right)}{\log\left(\frac{\nu_x}{\nu_o}\right)} \tag{11}$$

Including uncertainties for the flux densities F_x and F_o and simplifying, we then have

$$\begin{aligned}\beta_{ox} \pm \sigma_{\pm ox} &= \frac{\log\left(\frac{F_x \pm \sigma_X}{F_o \pm \sigma_o}\right)}{\log\left(\frac{\nu_x}{\nu_o}\right)} \\ &= \frac{\log\left[\frac{F_x\left(1 \pm \frac{\sigma_x}{F_x}\right)}{F_o\left(1 \pm \frac{\sigma_o}{F_o}\right)}\right]}{\log\left(\frac{\nu_x}{\nu_o}\right)}\end{aligned} \tag{12}$$

$$\beta_{ox} \pm \sigma_{\pm ox} = \frac{\log\left(\frac{F_x}{F_o}\right)}{\log\left(\frac{\nu_x}{\nu_o}\right)} + \frac{\log\left(\frac{1 \pm \frac{\sigma_x}{F_x}}{1 \mp \frac{\sigma_o}{F_o}}\right)}{\log\left(\frac{\nu_x}{\nu_o}\right)} \tag{13}$$

Example 1: Calculation of β_{ox} for Short GRB 050724A To illustrate this calculation process more clearly, we choose to walk though the calculation of β_{ox} for GRB 050724A.

Since it is a viable approximation, all of our data is based on X-ray fluxes which have been converted to flux densities corresponding to energies of 1 [keV]. Thus, knowing that $E = h\nu \Rightarrow \nu = \frac{E}{h}$, we can find the value of the frequency of the X-rays ν_x as

$$\nu_x = \frac{1 \times 10^3 \text{ [eV]}}{1} \cdot \frac{1 \text{ [Hz]}}{4.136 \times 10^{-15} \text{ [eV]}} \tag{14}$$

$$\nu_x \approx 2.418 \times 10^{17} \text{ [Hz]} \tag{15}$$

This value of ν_x is used for all GRBs in our calculations. From Fong et al. (2015), we know that the X-ray emission spectral index for GRB 050724A has been calculated from *Swift*'s data as $\beta_x = -0.81 \pm 0.15$. Next, from the many X-ray flux measurements taken by *Swift*, we choose the value $F_x = 0.32 \pm 0.068$ [μJy] measured at $\Delta t_x = 4.2 \times 10^4$ [s] ≈ 11 [hr]. Written another way, this data point is equivalent to

$$(\Delta t \text{ [s]}, F_x \text{ [}\mu\text{Jy]}): \quad \left(4.2 \times 10^4, 0.32 \pm 0.068\right) \tag{16}$$

We then choose a value for the optical flux density measured around the same time as the above data point. From Fong et al. (2015), we choose the following pair:

$$(\Delta t \text{ [hr]}, F_o \text{ [}\mu\text{Jy]}) : \quad (11.6, 45.6 \pm 1.4) \tag{17}$$

$$(\Delta t \text{ [s]}, F_o \text{ [}\mu\text{Jy]}) : \quad \left(4.18 \times 10^4, 45.6 \pm 1.4\right) \tag{18}$$

This optical measurement was taken using a K Filter from the Magellan Telescopes in Chile and was measured at an optical wavelength of $\lambda_o \approx 390$ [nm].[33] Recalling that $\nu = \frac{c}{\lambda}$, we can then find the value of the frequency of the optical-rays ν_o as

$$\nu_o = \frac{c}{\lambda_o} = \frac{2.998 \times 10^8 \ \left[\frac{\text{m}}{\text{s}}\right]}{390 \times 10^{-9} \text{ [m]}} \tag{19}$$

$$\nu_o \approx 7.69 \times 10^{14} \text{ [Hz]} \tag{20}$$

With all these values determined, we then substitute into our expression for β_{ox} to obtain

$$\begin{aligned} \beta_{ox} \pm \sigma_{\pm ox} &= \frac{\log\left(\frac{0.32 \text{ [}\mu\text{Jy]}}{45.6 \text{ [}\mu\text{Jy]}}\right)}{\log\left(\frac{2.418\times10^{17} \text{ [Hz]}}{7.69\times10^{14} \text{ [Hz]}}\right)} + \frac{\log\left(\frac{1 \pm \frac{0.068 \text{ [}\mu\text{Jy]}}{0.32 \text{ [}\mu\text{Jy]}}}{1 \mp \frac{1.4 \text{ [}\mu\text{Jy]}}{45.6 \text{ [}\mu\text{Jy]}}}\right)}{\log\left(\frac{2.418\times10^{17} \text{ [Hz]}}{7.69\times10^{14} \text{ [Hz]}}\right)} \\ &= -0.803^{+0.0362}_{-0.0436} \end{aligned} \tag{21}$$

3.2.5 Overall Pairing Method

The C++ program carries out four main steps to pair the X-ray data with the optical data:

1. Constructs GRB objects after `XRayData.txt` is loaded. These GRB objects are placed in a vector of GRB objects.
2. Loads `BetaXData.txt`, traverses through the vector of GRB objects, and pairs β_x-data with X-ray measurement data by matching GRB IDs.
3. Asks user for their desired temporal separation between optical and X-ray measurements $\Delta t_{\%}$.
4. Loads `OpticalData.txt` and begins optical pairing.
    - For every line of `OpticalData.txt`, the GRB ID and Δt_o are passed to a function which traverses through the vector of GRB objects with X-ray and (some) β_x data. If the ID from the line of `OpticalData.txt` matches with one from the vector of GRB objects, that GRB object has been populated with β_x data, *and* $\Delta t_{\%}$ for these two measurements is within the user-defined bound, then the location of the GRB object in the vector is returned.
    - Using the returned location in the vector, this GRB object with only X-ray and β_x data is copied to create a new GRB object. This new GRB object is then also populated with the matching optical data and appended to a new vector of GRB objects with X-ray, β_x, and optical data.

[33] Wavelengths corresponding to certain filters and general information about the telescopes relied upon by this paper are publicly available online as well as in Table 7 in the Appendix.

- Once a pairing has been made, rather than breaking out of the loop, the code returns to the vector of GRB objects with X-ray and (some) β_x data and continues making pairs with the same line from `OpticalData.txt`. The code recognizes when a new GRB ID has been reached so as to restart the process.

5. Loads `FilterInfo.txt`, traverses the GRB vector containing GRBs with X-ray, β_x, and optical data, and pairs with frequency data based on the optical telescope's name, instrument, and filter.

6. Traverses the vector of nearly fully-populated GRB objects, calculates $\beta_{ox} \pm \sigma_{\pm ox}$ for each, and accordingly sets corresponding attributes.

7. Traverses the vector and writes a file (to be read by the Python code) with each line corresponding to one fully-populated GRB object. The file's fields are as follows:

   - GRB ID; Δt_x [hr]; Δt_o [hr]; $|\Delta t_x - \Delta t_o|$ [hr]; β_x; σ_{+x}; σ_{-x}; β_{ox}; σ_{+ox}; σ_{-ox}

3.3 Automating the Graphing of β_{ox} vs. β_x

After β_{ox} has been calculated for many GRB objects using the C++ code, the next step in determining optical darkness is generating graphs of β_{ox} vs. β_x so that we can apply the Jakobsson and/or Van der Horst methods. To automate this process and generate graphs directly from the file output of the C++ code, we switch to using Python for its superior ability to generate graphs.

Upon running the Python code, the user is prompted in the terminal for the name of the file which was outputted by the C++ Code. After receiving a valid file name, this Python code reads in each line of the file and uses its corresponding field entries to construct a GRB object, just as was previously done in C++. Likewise, all these paired GRB objects are appended to a vector of paired GRB objects. The GRB class in Python is defined slightly differently, however, and is characterized by Table 3.

Python: class GRB	
Attribute	Description
`string ID`	GRB ID Number
`double dtX`	X-ray Time Δt_x [hr]
`double dtO`	Optical Time Δt_o [hr]
`double del_t`	$\lvert\Delta t_x - \Delta t_o\rvert$ [hr]
`double Beta_X`	X-Ray Spectral Index β_x
`double upper_sigmaX`	β_x Upper Uncertainty σ_{+x}
`double lower_sigmaX`	β_x Lower Uncertainty σ_{-x}
`double BetaOX`	Optical-to-X-Ray Spectral Index β_{ox}
`double upper_sigmaOX`	β_{ox} Upper Uncertainty σ_{+x}
`double lower_sigmaOX`	β_{ox} Lower Uncertainty σ_{-x}
`double D_Jakobsson`	Distance to $\beta_x = 0.5$
`double D_vanderHorst`	Distance to $\beta_{ox} = \beta_x - 0.5$

Table 3: The attributes for the Python GRB class used to create graphs of β_{ox} vs. β_x.

After the data is loaded and the vector of paired GRB objects is created, the user is then prompted to choose from seven options:

1. Graph all data from loaded file.

- Graphs all the GRB objects from the initial vector of paired GRB objects on the $\beta_x - \beta_{ox}$ plane.
- Displays said graph and saves it to the current working directory.

2. Graph those GRB pairings which are optically-dark according to the Jakobsson method.
   - Passes the initial vector of paired GRB objects to a function which identifies those GRB objects in the vector which are dark according to the Jakobsson method.
   - Appends such dark objects to their own vector of Jakobsson-dark GRB objects.
   - Graphs all the Jakobsson-dark GRB objects on the $\beta_x - \beta_{ox}$ plane.
   - Displays said graph and saves it to the current working directory.
   - Traverses through the vector of Jakobsson-dark GRB objects and outputs them to a .csv file with fields corresponding to the GRB objects' attributes.
3. Graph those GRB pairings that are optically-dark according to the Van der Horst method.
   - Passes the initial vector of paired GRB objects to a function which identifies those GRB objects in the vector which are dark according to the Van der Horst method.
   - Appends such dark objects to their own vector of Van der Horst-dark GRB objects.
   - Graphs all the Van der Horst-dark GRB objects on the $\beta_x - \beta_{ox}$ plane.
   - Displays said graph and saves it to the current working directory.
   - Traverses through the vector of Van der Horst-dark GRB objects and outputs them to a .csv file with fields corresponding to the GRB objects' attributes.
4. Graph only the GRB pairings that are the darkest for their unique GRB ID according to the Jakobsson method.
   - Passes the vector of Jakobsson-dark GRB objects to a function which identifies the darkest per each unique GRB ID according to the Jakobsson method.
   - Appends such Jakobsson-darkest objects to their own vector of Jakobsson-darkest GRB objects.
   - Graphs all the Jakobsson-darkest GRB objects on the $\beta_x - \beta_{ox}$ plane.
   - Displays said graph and saves it to the current working directory.
5. Graph only the GRB pairings that are the darkest for their unique GRB ID according to the Van der Horst method.
   - Passes the vector of Van der Horst-dark GRB objects to a function which identifies the darkest per each unique GRB ID according to the Van der Horst method.
   - Appends such Van der Horst-darkest objects to their own vector of Van der Horst-darkest GRB objects.
   - Graphs all the Van der Horst-darkest GRB objects on the $\beta_x - \beta_{ox}$ plane.
   - Displays said graph and saves it to the current working directory.
6. Graph data points for a particular GRB ID.
   - Prompts the user to input a particular GRB ID in the form YYMMDD.

- Passes the initial vector of paired GRB objects to a function which identifies all those GRB objects with the user's ID of interest.
- Appends such GRB objects to a vector corresponding to their unique GRB ID.
- Provides the user with options 1-5 and 7 again.

7. Quit.

After the user chooses one option, the user is returned to the menu and is given the chance to choose again until the user chooses to quit the program. Moreover, if a graph or .csv file is created and saved to the working directory, it is assigned a unique name which includes the $\Delta t_{\%}$ corresponding to the initial vector of paired GRB objects from the C++ code.

3.3.1 Determination of Dark Burst with Uncertainties

In order to be confident that a burst is indeed dark, we choose to incorporate its associated error bars when applying both the Jakobsson method and the Van der Horst method. Consider an arbitrary point in the $\beta_x - \beta_{ox}$ plane, $(\beta'_x \pm \sigma_{\pm x'}, \beta'_{ox} \pm \sigma_{\pm ox'})$. Using the Jakobsson method, incorporating error bars into the calculation of a dark burst is trivial. As is shown graphically in Figure 14, the requirement is simply given as

$$\textbf{Jakobsson:} \qquad 0.5 > \beta_{ox} + \sigma_{+ox} \tag{22}$$

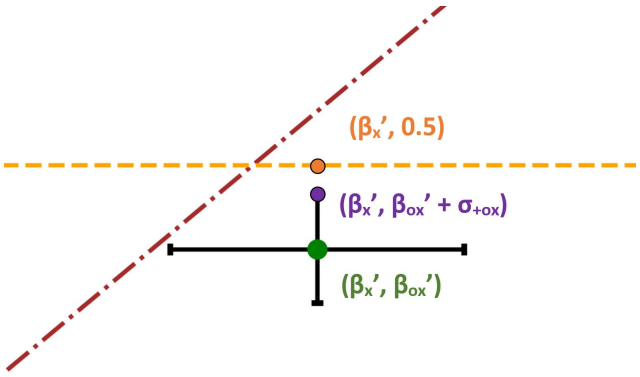


Figure 14: A graph of β_{ox} vs. β_x depicting a single dark burst according to the Jakobsson method located at an arbitrary test point $(\beta'_x \pm \sigma_{\pm x'}, \beta'_{ox} \pm \sigma_{\pm ox'})$.

Incorporating error bars with the Van der Horst method is also fairly trivial, as can be seen in Figure 15. Due to the linear behavior of $\beta_{ox} = \beta_x - 0.5$, the conditions for such a dark burst are given as

$$\textbf{Van der Horst:} \qquad \begin{cases} \beta'_x - 0.5 > \beta'_{ox} + \sigma_{+ox} \\ \beta'_x - \sigma_{-x} > \beta'_{ox} + 0.5 \end{cases} \tag{23}$$

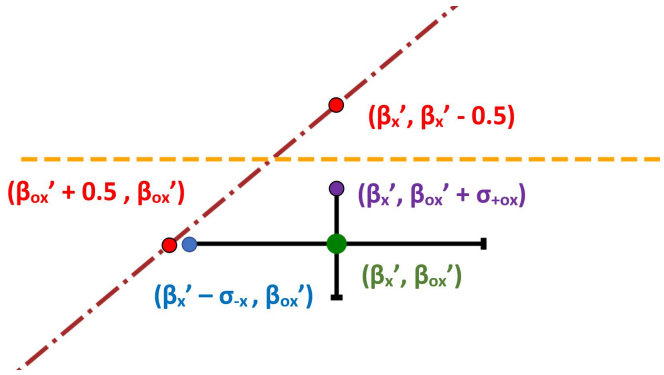


Figure 15: A graph of β_{ox} vs. β_x depicting a single dark burst according to the Van der Horst method located at an arbitrary test point $(\beta'_x \pm \sigma_{\pm x'}, \beta'_{ox} \pm \sigma_{\pm ox'})$.

3.3.2 Determination of Darkest Burst

For each unique GRB ID, there exists a multitude of pairings if there are multiple instances where the temporal separation between Δt_x and Δt_o is within the user-defined bound given by $\Delta t_{\%}$. Specifically, this requirement is given by

$$100\% \cdot \frac{|\Delta t_x - \Delta t_o|}{\Delta t_o} < \Delta t_{\%} \tag{24}$$

However, in an effort to mitigate over-crowding and redundancy when graphing all these pairings, the "darkest" burst per each unique GRB ID can also be calculated. The "darkest" burst is that burst which yields coordinates lying the farthest away from either the line $\beta_{ox} = 0.5$ or the line $\beta_{ox} = \beta_x - 0.5$, depending on definition. Beginning with Jakobsson method, the distance between the arbitrary point (β'_x, β'_{ox}) and the line $\beta_{ox} = 0.5$ is trivially given by:

$$\mathbf{D_{0.5}} = \mathbf{0.5} - \beta'_{\mathbf{ox}} \tag{25}$$

The distance between (β'_x, β'_{ox}) and $\beta_{ox} = \beta_x - 0.5$, as required by the Van der Horst method, is less trivial. The equation of the normal line which passes through (β'_x, β'_{ox}) is given by

$$\beta^{\mathrm{n}}_{ox} - \beta'_{ox} = \frac{-1}{m}(\beta_x - \beta'_x), \qquad m = 1 \tag{26}$$

$$\beta^{\mathrm{n}}_{ox} = \beta'_x - \beta_x + \beta'_{ox} \tag{27}$$

where m is the slope of the position vector from the origin to (β'_x, β'_{ox}). From this, the point at which these two lines intersect is the closest point between (β'_x, β'_{ox}) and the line $\beta_{ox} = \beta_x - 0.5$ (as the shortest distance between two points is a straight line). Thus, setting β^{n}_{ox} and β_{ox} equal, we find that

$$\beta_x - 0.5 = \beta'_x - \beta_x + \beta'_{ox} \tag{28}$$

$$\beta_x = \frac{\beta'_x + \beta'_{ox} + 0.5}{2} \tag{29}$$

Then, inserting this into our expression for β_{ox}, we find that

$$\begin{aligned}\beta_{ox} &= \frac{\beta'_x + \beta'_{ox} + 0.5}{2} - 0.5 \\ &= \frac{\beta'_x + \beta'_{ox} - 0.5}{2}\end{aligned} \tag{30}$$

We then conclude that the distance between (β_x, β_{ox}) and (β'_x, β'_{ox}) is

$$\begin{aligned}D_{\beta_x - 0.5} &= \sqrt{\left(\frac{\beta'_x + \beta'_{ox} + 0.5}{2} - \beta'_x\right)^2 + \left(\frac{\beta'_x + \beta'_{ox} - 0.5}{2} - \beta'_{ox}\right)^2} \\ &= \sqrt{\frac{{\beta'_{ox}}^2 + {\beta'_x}^2 + \beta'_{ox} - \beta'_x - 2\beta'_x\beta'_{ox} + 0.25}{2}} \\ &= \frac{\beta'_x - \beta'_{ox} - 0.5}{\sqrt{2}}\end{aligned} \tag{31}$$

Thus, depending on definition, the darkest burst is one which maximizes either of the following parameters:

$$\begin{cases} D_{0.5} = 0.5 - \beta'_{ox} & \text{(Jakobsson)} \\ D_{\beta_x - 0.5} = \frac{\beta'_x - \beta'_{ox} - 0.5}{\sqrt{2}} & \text{(Van der Horst)} \end{cases} \tag{32}$$

Including error bars (i.e. $\beta'_x = \beta'^{\,+\sigma_{+x'}}_{x-\sigma_{-x'}}$, $\beta'_{ox} = \beta'^{\,+\sigma_{+ox'}}_{ox-\sigma_{-ox'}}$), these specifications become

$$\begin{cases} D_{0.5} = 0.5 - (\beta'_{ox} \pm \sigma_{+ox'}) & \text{(Jakobsson)} \\ D_{\beta_x - 0.5} = \frac{(\beta'_x \pm \sigma_{-x'}) - (\beta'_{ox} \pm \sigma_{+ox'}) - 0.5}{\sqrt{2}} & \text{(Van der Horst)} \end{cases} \tag{33}$$

In an effort to confirm that each GRB is darkest even with its uncertainties, we then utilize our liberty to choose between $\pm$ and make the following choices regarding the uncertainties in each data point to arrive at our final result:

$$\begin{cases} D_{0.5} = 0.5 - \beta'_{ox} - \sigma_{+ox'} & \text{(Jakobsson)} \\ D_{\beta_x - 0.5} = \frac{\beta'_x - \sigma_{-x'} - \beta'_{ox} - \sigma_{+ox'} - 0.5}{\sqrt{2}} & \text{(Van der Horst)} \end{cases} \tag{34}$$

Thus, that burst which maximizes $D_{0.5}$ or $D_{\beta_x - 0.5}$ for a unique GRB ID is considered to be the darkest Jakobsson GRB or the darkest Van der Horst GRB, respectively.

4 Results and Analysis

Although the design of these programs allows for many different arrangements of data, each possibly providing their own conclusions, we present here a determination and analysis of those GRBs which we strongly believe to be optically-dark. Such GRBs are those which consistently yield (β_x, β_{ox}) pairs which are classified as dark by the Jakobsson and/or Van der Horst methods across a range of temporal separations (specifically, $\Delta t_{\%} = 5\%$, $\Delta t_{\%} = 10\%$, and $\Delta t_{\%} = 20\%$) with the inclusion of measurement uncertainty ($\sigma_{\pm x}$; $\sigma_{\pm ox}$) and uncertainty arising from the temporal separation itself (designated $\Delta\beta_{ox}$). The process of identifying dark GRBs is explained visually by Figure 16 and is explained throughout Section 4.

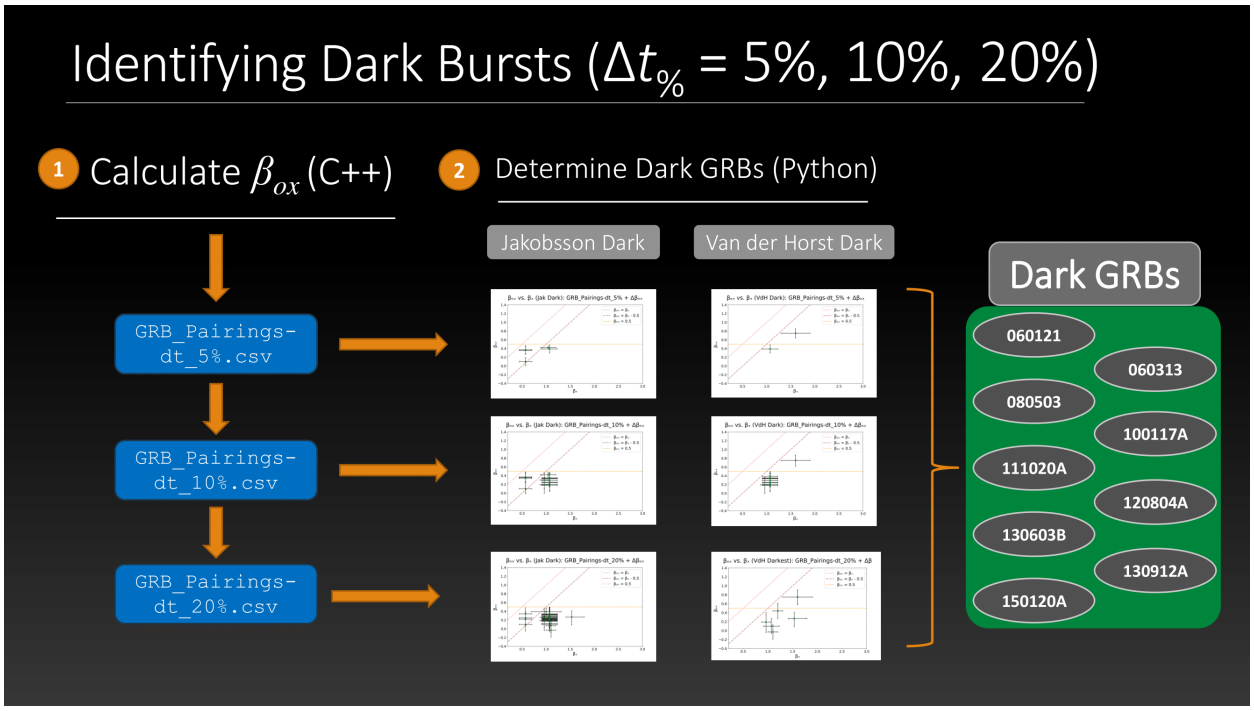


Figure 16: A visual depiction of how we identified dark short GRBs using both the C++ code and the Python code.

4.1 Error in β_{ox} due to Temporal Separation: $\Delta\beta_{ox}$

While increasing $\Delta t_{\%}$ allows for more pairings and therefore analysis of more data, its effect on the error in the calculation of β_{ox} is nonzero and must be taken into account. To understand its influence, we begin by recalling the relationship between the spectral index and the flux:

$$\beta_{ox} \propto \log\left(\frac{F_o}{F_x}\right) \tag{35}$$

We then recall the definitions of F_o and F_x:

$$F_o = F_{n,o}\left(\frac{t_o}{t_{n,o}}\right)^{-\alpha}; \qquad F_x = F_{n,x}\left(\frac{t_x}{t_{n,x}}\right)^{-\alpha} \tag{36}$$

where $F_{n,o}$ and $F_{n,x}$ are the magnitudes of F_o and F_x at some reference times $t_{n,o}$ and $t_{n,x}$, respectively, and α is the decay time parameter. Substituting, we then find that

$$\begin{aligned}\frac{F_o}{F_x} &= \frac{F_{n,o}}{F_{n,x}} \cdot \left(\frac{t_o}{t_{n,o}}\right)^{-\alpha}\left(\frac{t_{n,x}}{t_x}\right)^{-\alpha} \\ &= \frac{F_{n,o}}{F_{n,x}} \cdot \left(\frac{t_o}{t_x}\right)^{-\alpha}\left(\frac{t_{n,x}}{t_{n,o}}\right)^{-\alpha}\end{aligned} \tag{37}$$

Assuming similar reference times such that $t_{n,o} \approx t_{n,x}$, we then have

$$\frac{F_o}{F_x} = \frac{F_{n,o}}{F_{n,x}} \cdot \left(\frac{t_o}{t_x}\right)^{-\alpha} \tag{38}$$

For $t_o = t_x + \Delta t_{\%} \cdot t_x$, this becomes

$$\begin{aligned} \frac{F_o}{F_x} &= \frac{F_{n,o}}{F_{n,x}} \cdot \left(\frac{t_x + \Delta t_{\%} \cdot t_x}{t_x}\right)^{-\alpha} \\ &= \frac{F_{n,o}}{F_{n,x}} \cdot (1 + \Delta t_{\%})^{-\alpha} \end{aligned} \tag{39}$$

Thus, the relationship between β_{ox} and $\Delta t_{\%}$ is then

$$\beta_{ox} \propto \log\left(\frac{F_o}{F_x}\right) \propto -\alpha \log\left(1 + \Delta t_{\%}\right) \tag{40}$$

$$\Delta\beta_{ox} = \alpha \log\left(1 + \Delta t_{\%}\right) \tag{41}$$

Letting $\alpha \approx 1$, Table 4 shows $\Delta\beta_{ox}$ for a few values of $\Delta t_{\%}$. As illustrated, it is clear that greater temporal separation leads to larger error in the calculation of β_{ox}, which is to be expected.

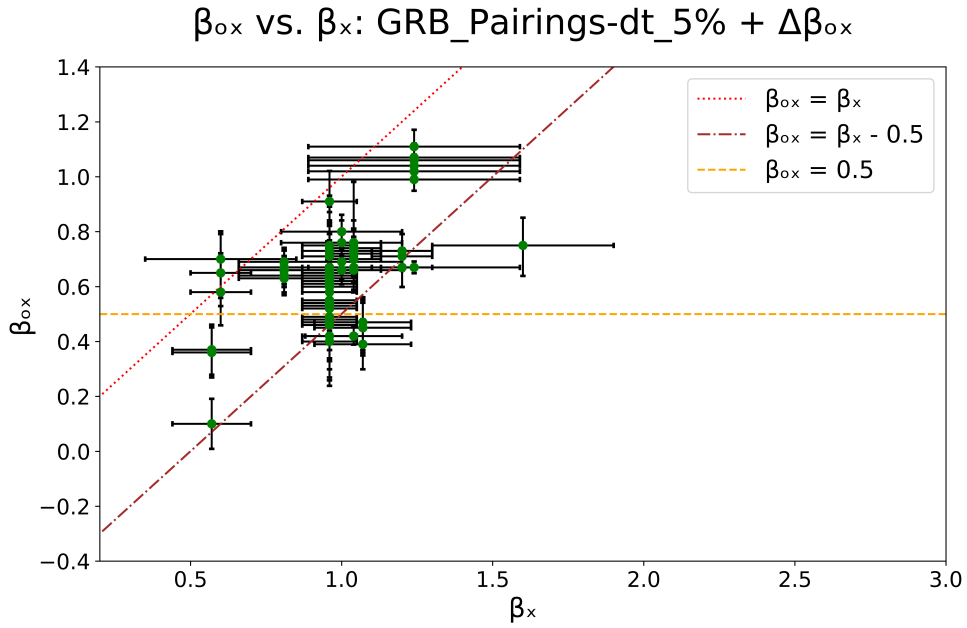
$\Delta t_{\%}$	$\Delta\beta_{ox}$
5%	0.021
10%	0.041
20%	0.079
30%	0.114
40%	0.146
50%	0.176

Table 4: The relationship between $\Delta\beta_{ox}$ and $\Delta t_{\%}$ showing that greater temporal separation leads to greater error in β_{ox}.

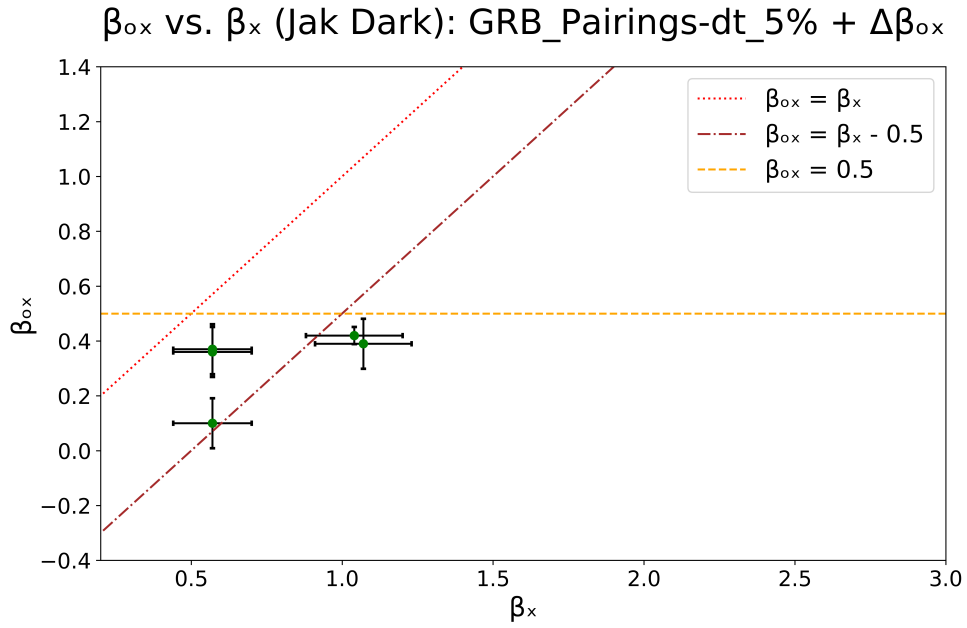
4.2 Graphing β_{ox} vs. β_x

Due to the significant increase in error in β_{ox} for large $\Delta t_{\%}$, we limit our choices of temporal separation to $\Delta t_{\%} = 5\%$, $\Delta t_{\%} = 10\%$, and $\Delta t_{\%} = 20\%$ for our overall classification of optically-dark bursts. As such, we used the program written in C++ to generate files `GRB_Pairings-dt_5%.csv`, `GRB_Pairings-dt_10%.csv`, and `GRB_Pairings-dt_20%.csv`. After altering the Python code so as to include $\Delta\beta_{ox}$ in the total error (i.e. using $\sigma_{\pm ox} \pm \Delta\beta_{ox}$ instead of simply $\sigma_{\pm ox}$), we then fed the three files created by the C++ code into the Python code and generated plots in the $\beta_x - \beta_{ox}$ plane for each loaded file with the aim of cross-compiling the results to identify the optically-dark GRBs.

Figures 17, 18, and 19 respectively compare the distributions of β_x and β_{ox} at $\Delta t_{\%} = 5\%$, $\Delta t_{\%} = 10\%$, and $\Delta t_{\%} = 20\%$, respectively, for (a) all GRB pairings made with the corresponding $\Delta t_{\%}$, (b) only those GRB pairings which are dark according to the Jakobsson method (i.e. those which lie below the line of constant $\beta_{ox} = 0.5$), and (c) only those GRB pairings which are dark according to the Van der Horst method (i.e. those which lie below the line $\beta_{ox} = \beta_x - 0.5$).

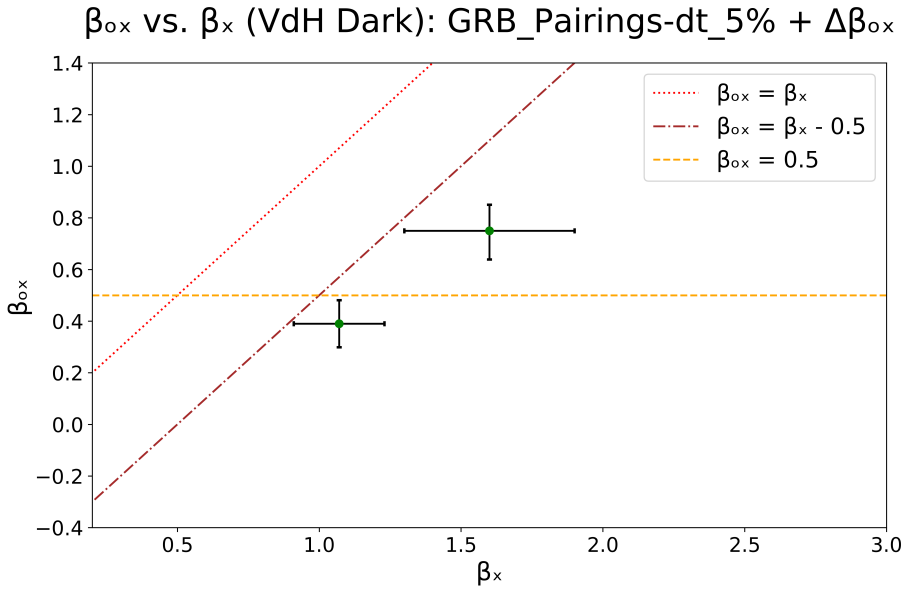


(a)



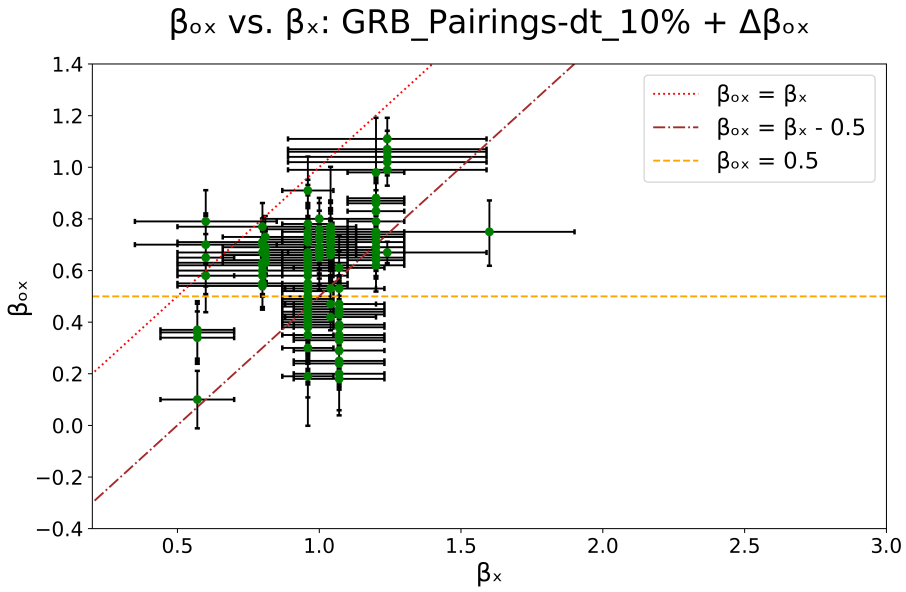
(b)

Figure 17: Graphs of β_{ox} vs. β_x for $\Delta t_{\%} = 5\%$ illustrating (a) all GRB pairings, (b) only those pairings which are dark by the Jakobsson method, and (c) only those pairings which are dark by the Van der Horst method.



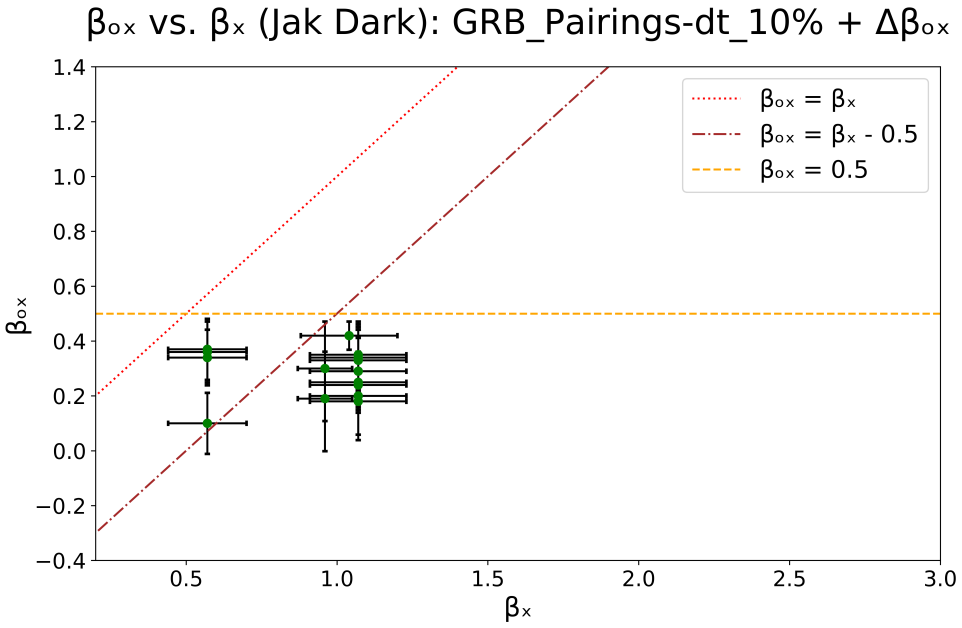
(c)

Figure 17: (Cont.)

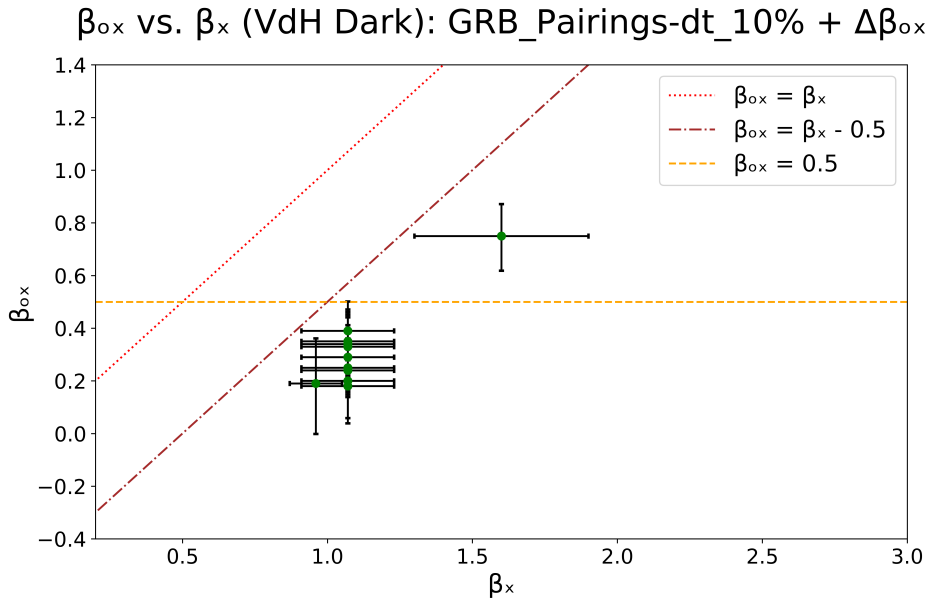


(a)

Figure 18: Graphs of β_{ox} vs. β_x for $\Delta t_{\%} = 10\%$ illustrating (a) all GRB pairings, (b) only those pairings which are dark by the Jakobsson method, and (c) only those pairings which are dark by the Van der Horst method.

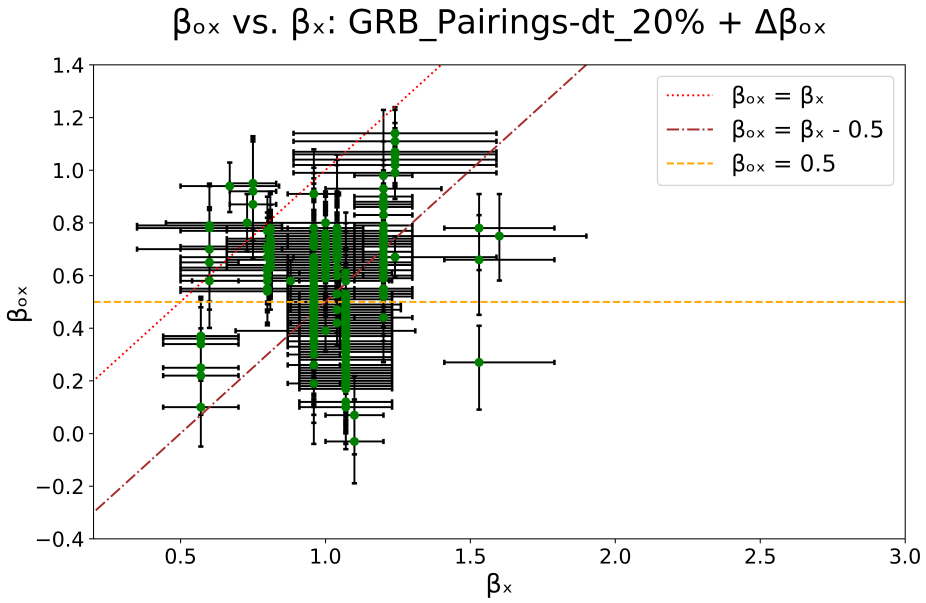


(b)

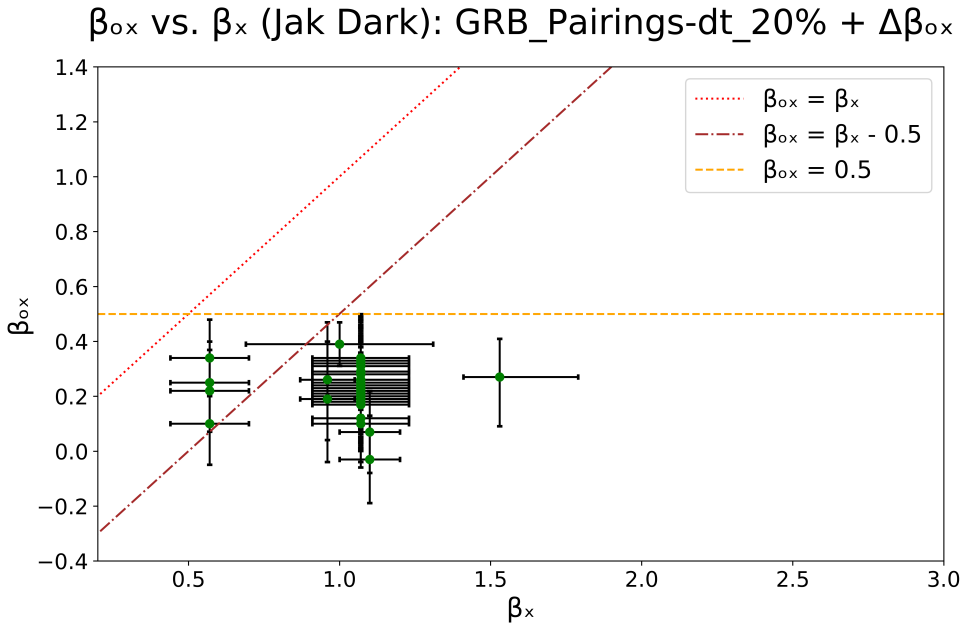


(c)

Figure 18: (Cont.)

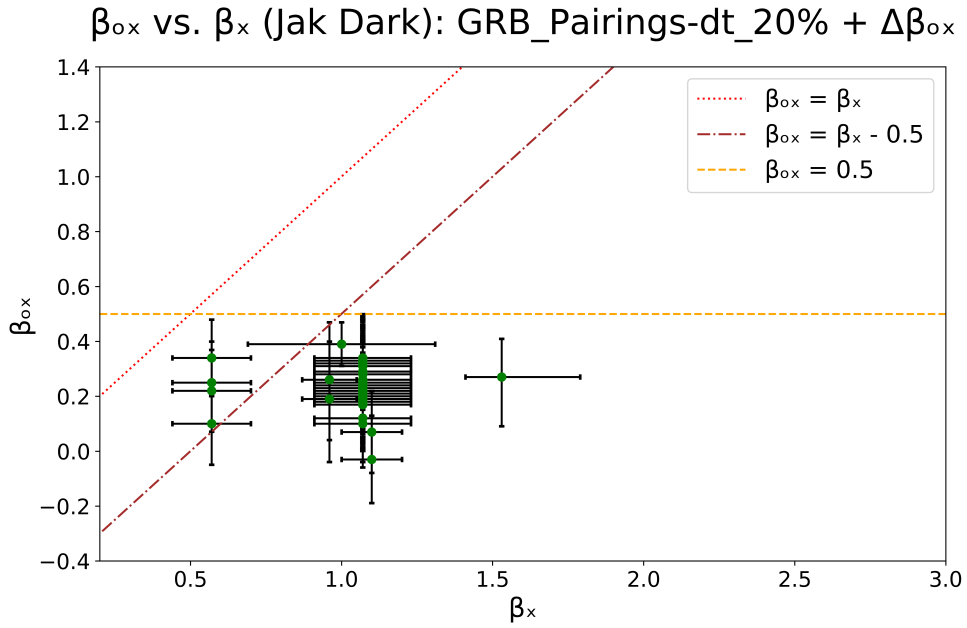


(a)



(b)

Figure 19: Graphs of β_{ox} vs. β_x for $\Delta t_{\%} = 20\%$ illustrating (a) all GRB pairings, (b) only those pairings which are dark by the Jakobsson method, and (c) only those pairings which are dark by the Van der Horst method.



(c)

Figure 19: (Cont.)

4.3 Determination of Dark Bursts

In addition to the graphs depicted in Figures 17-19, the program written in Python generated files matching the graphed subsection of data with fields identical to those of the input files from the program written in C++. Table 5 cross-compiles these files and provides an overview of all 64 identified dark data pairs across the spectrum of $\Delta t_{\%}$.

Table 5
Short GRB X-Ray and Optical Afterglow Catalog with $\Delta\beta_{ox}$

						Dark-20%?		Dark-10%?		Dark-5%?	
GRB	Δt_x [hr]	Δt_o [hr]	$\lvert\delta t\rvert$ [hr]	β_x	β_{ox}	Jak	VdH	Jak	VdH	Jak	VdH
060121	3.06	2.6	0.46	-1.07 ± 0.16	-0.37 ± 0.09	N	Y	N	N	N	N
060121	3.06	2.8	0.26	-1.07 ± 0.16	-0.29 ± 0.08	Y	Y	Y	Y	N	N
060121	3.06	2.8	0.26	-1.07 ± 0.16	-0.35 ± 0.08	N	Y	Y	Y	N	N
060121	3.06	2.8	0.26	-1.07 ± 0.16	-0.34 ± 0.08	Y	Y	Y	Y	N	N
060121	3.06	2.8	0.26	-1.07 ± 0.16	-0.33 ± 0.08	Y	Y	Y	Y	N	N
060121	3.33	2.8	0.53	-1.07 ± 0.16	-0.31 ± 0.08	Y	Y	N	N	N	N
060121	3.33	2.8	0.53	-1.07 ± 0.16	-0.32 ± 0.08	Y	Y	N	N	N	N
060121	3.33	2.8	0.53	-1.07 ± 0.16	-0.29 ± 0.08	Y	Y	N	N	N	N
060121	3.33	2.8	0.53	-1.07 ± 0.16	-0.37 ± 0.09	Y	Y	N	N	N	N
060121	3.33	2.8	0.53	-1.07 ± 0.16	$-0.37^{+0.08}_{-0.07}$	N	Y	N	N	N	N

Table 5

(Continued)

						Dark-20%?		Dark-10%?		Dark-5%?	
GRB	Δt_x [hr]	Δt_o [hr]	$\lvert\delta t\rvert$ [hr]	β_x	β_{ox}	Jak	VdH	Jak	VdH	Jak	VdH
060121	3.33	3.9	0.57	-1.07 ± 0.16	$-0.12^{+0.09}_{-0.08}$	N	Y	N	N	N	N
060121	3.33	3.9	0.57	-1.07 ± 0.16	$-0.12^{+0.09}_{-0.08}$	Y	Y	N	N	N	N
060121	3.33	3.9	0.57	-1.07 ± 0.16	$-0.10^{+0.09}_{-0.08}$	Y	Y	N	N	N	N
060121	3.33	3.9	0.57	-1.07 ± 0.16	-0.18 ± 0.09	Y	Y	N	N	N	N
060121	3.33	3.9	0.57	-1.07 ± 0.16	$-0.17^{+0.09}_{-0.08}$	Y	Y	N	N	N	N
060121	4.44	3.9	0.54	-1.07 ± 0.16	-0.22 ± 0.09	Y	Y	N	N	N	N
060121	4.44	3.9	0.54	-1.07 ± 0.16	-0.24 ± 0.09	Y	Y	N	N	N	N
060121	4.44	3.9	0.54	-1.07 ± 0.17	-0.17 ± 0.09	Y	Y	N	N	N	N
060121	4.44	4.4	0.04	-1.07 ± 0.16	-0.39 ± 0.07	N	Y	N	Y	Y	Y
060121	5.00	4.4	0.06	-1.07 ± 0.16	$-0.40^{+0.07}_{-0.08}$	N	Y	N	N	N	N
060121	4.44	5.4	0.96	-1.07 ± 0.16	$-0.24^{+0.12}_{-0.10}$	Y	Y	N	N	N	N
060121	4.44	5.4	0.96	-1.07 ± 0.16	$-0.26^{+0.12}_{-0.10}$	Y	Y	N	N	N	N
060121	4.44	5.4	0.96	-1.07 ± 0.16	$-0.18^{+0.12}_{-0.10}$	Y	Y	N	N	N	N
060121	4.72	5.4	0.68	-1.07 ± 0.16	$-0.23^{+0.12}_{-0.10}$	Y	Y	N	N	N	N
060121	4.72	5.4	0.68	-1.07 ± 0.16	$-0.22^{+0.12}_{-0.10}$	Y	Y	N	N	N	N
060121	4.72	5.4	0.68	-1.07 ± 0.16	$-0.21^{+0.12}_{-0.10}$	Y	Y	N	N	N	N
060121	5.00	5.4	0.40	-1.07 ± 0.16	$-0.20^{+0.12}_{-0.10}$	Y	Y	Y	Y	N	N
060121	5.00	5.4	0.40	-1.07 ± 0.16	$-0.18^{+0.12}_{-0.10}$	Y	Y	Y	Y	N	N
060121	5.00	5.4	0.40	-1.07 ± 0.16	$-0.29^{+0.11}_{-0.09}$	Y	Y	Y	Y	N	N
060121	6.11	5.4	0.71	-1.07 ± 0.16	$-0.26^{+0.12}_{-0.10}$	Y	Y	N	N	N	N
060121	4.72	5.6	0.88	-1.07 ± 0.16	$-0.21^{+0.07}_{-0.08}$	Y	Y	N	N	N	N
060121	4.72	5.6	0.88	-1.07 ± 0.16	$-0.20^{+0.07}_{-0.08}$	Y	Y	N	N	N	N
060121	4.72	5.6	0.88	-1.07 ± 0.16	$-0.19^{+0.07}_{-0.08}$	Y	Y	N	N	N	N
060121	5.00	5.6	0.60	-1.07 ± 0.16	$-0.19^{+0.07}_{-0.08}$	Y	Y	N	N	N	N
060121	5.00	5.6	0.60	-1.07 ± 0.16	$-0.17^{+0.07}_{-0.08}$	Y	Y	N	N	N	N
060121	5.00	5.6	0.60	-1.07 ± 0.16	$-0.28^{+0.06}_{-0.07}$	Y	Y	N	N	N	N
060121	6.11	5.6	0.51	-1.07 ± 0.16	-0.25 ± 0.07	Y	Y	Y	Y	N	N
060121	6.11	6.5	0.39	-1.07 ± 0.16	$-0.24^{+0.04}_{-0.05}$	Y	Y	Y	Y	N	N
060121	6.11	7.3	1.19	-1.07 ± 0.16	$-0.26^{+0.06}_{-0.07}$	Y	Y	N	N	N	N
060121	6.11	7.3	1.19	-1.07 ± 0.16	-0.29 ± 0.08	Y	Y	N	N	N	N
060313	1.22	1.5	0.28	-0.96 ± 0.09	$-0.26^{+0.13}_{-0.14}$	Y	N	N	N	N	N
060313	1.39	1.5	0.11	-0.96 ± 0.09	$-0.19^{+0.13}_{-0.15}$	N	N	Y	N	N	N
060313	1.58	1.5	0.08	-0.96 ± 0.09	$-0.30^{+0.13}_{-0.15}$	Y	Y	Y	Y	N	N
080503	108.33	97.1	11.23	$-1.53^{+0.26}_{-0.12}$	$-0.66^{+0.09}_{-0.13}$	N	Y	N	N	N	N
080503	108.33	125.0	16.67	$-1.53^{+0.26}_{-0.12}$	$-0.78^{+0.05}_{-0.08}$	N	Y	N	N	N	N
080503	108.33	128.6	20.27	$-1.53^{+0.26}_{-0.12}$	$-0.27^{+0.06}_{-0.10}$	Y	Y	N	N	N	N
100117A	8.61	8.3	0.31	-1.6 ± 0.3	$-0.75^{+0.08}_{-0.9}$	N	Y	N	Y	N	Y
111020A	19.17	17.8	1.37	-1.04 ± 0.16	-0.42 ± 0.01	N	N	Y	N	Y	N
120804A	4.72	5.5	0.78	-1.1 ± 0.1	0.03 ± 0.08	Y	Y	N	N	N	N

Table 5

(Continued)

						Dark-20%?		Dark-10%?		Dark-5%?	
GRB	Δt_x [hr]	Δt_o [hr]	$\lvert\delta t\rvert$ [hr]	β_x	β_{ox}	Jak	VdH	Jak	VdH	Jak	VdH
120804A	6.39	5.5	0.89	-1.1 ± 0.1	-0.07 ± 0.07	Y	Y	N	N	N	N
130603B	1.72	2.1	0.38	-1.2 ± 0.1	$-0.54^{+0.04}_{-0.06}$	N	Y	N	N	N	N
130603B	1.75	2.1	0.35	-1.2 ± 0.1	$-0.53^{+0.04}_{-0.05}$	N	Y	N	N	N	N
130603B	2.72	2.4	0.32	-1.2 ± 0.1	-0.44 ± 0.09	N	Y	N	N	N	N
130603B	2.75	2.4	0.35	-1.2 ± 0.1	-0.52 ± 0.09	N	Y	N	N	N	N
130603B	5.83	7.1	1.27	-1.2 ± 0.1	$-0.55^{+0.04}_{-0.05}$	N	Y	N	N	N	N
130603B	6.11	7.1	0.99	-1.2 ± 0.1	$-0.55^{+0.04}_{-0.05}$	N	Y	N	N	N	N
130912A	0.27	0.27	0.00	-0.57 ± 0.13	-0.10 ± 0.07	Y	N	Y	N	Y	N
130912A	0.31	0.27	0.04	-0.57 ± 0.13	-0.22 ± 0.07	Y	N	N	N	N	N
130912A	0.31	0.38	0.07	-0.57 ± 0.13	-0.25 ± 0.07	Y	N	N	N	N	N
130912A	0.36	0.38	0.02	-0.57 ± 0.13	-0.37 ± 0.07	N	N	Y	N	Y	N
130912A	0.39	0.38	0.01	-0.57 ± 0.13	-0.36 ± 0.07	N	N	Y	N	Y	N
130912A	0.42	0.38	0.04	-0.57 ± 0.13	-0.34 ± 0.06	Y	N	Y	N	N	N
150120A	2.31	2.0	0.31	-1.0 ± 0.31	-0.39	Y	N	N	N	N	N

Table 5: The raw data for GRB pairings and their corresponding darkness status according to the Jakobsson and Van der Horst methods **including the $\Delta\beta_{ox}$ due to temporal separation** for $\Delta t_{\%} = 5\%$, $\Delta t_{\%} = 10\%$, and $\Delta t_{\%} = 20\%$.

5 Discussion

As is shown in Table 5, we identify 64 instances of darkness for all detected short GRBs since the launch of *Swift* in 2004 through March 2015. While some unique GRB IDs yield more dark data points than others (and so have stronger support for being dark than others) as well as varying degrees of darkness across the temporal spectrum of $\Delta t_{\%} = 5\%$, 10%, and 20%, compiling the results from Table 5 and Figures 17-19 allows us to identify 9 short GRBs as being optically-dark out of the 70 short GRBs detected since the launch of *Swift* in 2004 through March 2015: 060121, 060313, 080503, 100117A, 111020A, 120804A, 130603B, 130912A, and 150120A. With these 9 identified, we now individually analyze the properties of all 9 based on case studies carried out by others in the field.

5.1 GRB 060121

Of the 9 GRBs which we highlight, GRB 060121 contains the most dark data points across our chosen temporal range for $\Delta t_{\%}$; specifically, 40 of the 64 instances of darkness correspond to GRB 060121. Moreover, GRB 060121 is the only GRB to have a data pair which is dark at $\Delta t_{\%} = 5\%$ according to both the Jakobsson and Van der Horst methods (Table 5). As such, we are the most confident in defining this burst as optically-dark.

GRB 060121 was initially detected on January 21$^{\text{st}}$, 2006 by the GRB-detecting smallsat *HETE-2* at 22:24:54.5 UTC (Donaghy et al. 2006). Following the initial burst of gamma-rays, optical and X-ray

transients were soon measured by multiple ground observation stations following HETE-2's relay to the GCN burst alert network. In the energy range 85-400 keV, Donaghy et al. (2006) find that this GRB has $T_{90} = 1.60 \pm 0.07$ [s]; in the energy range 30-400 keV, however, they find $T_{90} = 1.97 \pm 0.06$ [s].[34] As such, while GRB 060121 is widely considered by the astrophysics community to be a short GRB, it does lie at the "borderline" of the typical BATSE short GRB definition of $T_{90} < 2$ [s] (Kann et al. 2011).

Perhaps more interesting, however, is the high redshift of GRB 060121. According to de Ugarte Postigo et al. (2006), there is a 35% chance $z = 1.7$, but a 63% chance $z = 4.6$. While the progenitor is considered to be a star-forming disk galaxy (Donaghy et al. 2006), the more likely scenario of $z = 4.6$ makes GRB 060121 one of only a few short bursts with such a large redshift. Moreover, since $z = 4.6$ corresponds to when the universe is only ≈ 1.5 billion years old, this calculation puts a damper on the leading progenitor model for short GRBs because it is unlikely that the universe was old enough to yield a compact binary merger (CBM). At such a young age, the chance of having either a pair of neutron stars (NS-NS) or a neutron star and a black hole (NS-BH), both of which require a supernova to form, are very low; on top of that, these compact bodies would need to circle each other a long time before a CBM could occur. This dilemma seems to imply that the progenitor for GRB 060121 was not a CBM but rather another source, one that perhaps may also shed light on the reasons for the burst's optical darkness.

5.2 GRB 060313

Although we find only three dark data points for GRB 060121, we still consider this GRB to be optically-dark because it generates a pair which is dark at $\Delta t_{\%} = 20\%$ to $\Delta t_{\%} = 10\%$ for both the Jakobsson and Van der Horst methods (Table 5). In addition, another pair indicates it is dark at $\Delta t_{\%} = 20\%$ according to the Jakobsson method, while another indicates that it is dark at $\Delta t_{\%} = 10\%$ according to the Jakobsson method.

GRB 060313 was first detected on March 13$^{\text{th}}$, 2006 by *Swift* at 00:12:06.484 UTC (Roming et al. 2006). Following measurement of the afterglow, T_{90} was determined to be 0.7 ± 0.1 [s] in the 15-350 keV band for this burst (Roming et al. 2006), illustrating that it clearly adheres to the BATSE definition of a short GRB. Furthermore, the redshift for this burst is calculated to be $z < 1.1$ at the 90% confidence level, with the most likely redshift at $z = 0.75$, although it is minimally peaked (Roming et al. 2006).

In terms of identifying its progenitor, however, GRB 060313 presents astrophysicists with a bit of a puzzle. As explained in greater detail by Roming et al. (2006), although the afterglow data indicates that the burst originated from a low-density medium, fluctuations in the UVOT lightcurve and a lack of corresponding fluctuation in the X-ray lightcurve suggests that the circum-stellar medium (CSM) was actually fairly dense, leading to the idea that this GRB originated from within a distant galaxy. However, according to extragalactic databases, there exists no galaxy at the predicted location of GRB 060313, but rather only a cluster of galaxies surrounding the burst's predicted origin. If one were to assume the progenitor of GRB 060313 was located in the less-dense medium surrounding this cluster of galaxies, a massive star progenitor is certainly ruled out. A NS-NS or a NS-BH CBM, on the other hand, could have occurred at this location because of

[34] Changing the energy range for the calculation of T_{90} simply refers to adjusting the bin size for characterizing detected GRB gamma ray emission.

the natal kick associated with the formation of neutron stars.[35] Calculations and modeling affirm such a possibility for various values of $z < 1.1$ (Roming et al. 2006). Still, however, the conflict between UVOT lightcurve data and extragalactic databases make it difficult to confidently identify the progenitor type of this GRB.

5.3 GRB 080503

GRB 080503 yields three dark data points, with all three of the points being dark at $\Delta t_{\%} = 20\%$ according to the Van der Horst method and one of the points being dark at $\Delta t_{\%} = 20\%$ according to the Jakobsson method (Table 5). While this is not as statistically significant as for GRBs 060121 or 060313, journal review of GRB 080503 reveals that the optical portion of the afterglow is extraordinarily faint, never exceeding magnitude +25 in deep observations (Perley et al. 2009). Detected by *Swift* at 12:26:13 on May 3$^{\rm rd}$, 2008, one of the particularly interesting facts about GRB 080503 is that although there is consensus over its classification as a short GRB, NASA's Goddard Space Flight Center's Swift Database finds that $T_{90} = 170$ [s], and Perley et al. (2009) finds that $T_{90} = 232$ [s]. While these values for T_{90} are far greater than the typical cutoff of 2 seconds, GRB 080503 is considered short because there is a characterizing initial short spike around 0.32 ± 0.07 [s] followed by a longer emission in the 15-150 keV range until 232 s (Perley et al. 2009). Additional reasons for classifying GRB 080503 as short are based on other measurements like hardness and fluence which are not discussed here.

While attempts were made to uncover a discrete value for the redshift of GRB 080503, the redshift remains unconstrained except by the g-band photometric detection limit of $z < 4$.[36] Additionally, much like with GRB 060121, GRB 080503 brings along a similar conflict between UVOT lightcurves and extragalactic databases. While calculations reveal no galaxy to be located at the predicted origin of GRB 080503, the fact that the luminosity of the extended emission (after the initial spike) exceeds the initial spike by factors of ≥ 30 shows instead that this is inconsistent with a progenitor located outside its host galaxy (Perley et al. 2009). Rather, an origin in this kind of less-dense medium predicts such extended emission to last only a few seconds instead of > 200 s (for the same luminosity). While Perley et al. (2009) believed this to strengthen other explanations for short GRB progenitors, such as those which predict short GRBs arise from the accretion-induced collapse of supramassive neutron stars (SMNSs; Vietri & Stella 1999), today's astrophysicists conclude on the whole that the most likely progenitor for GRB 080503 was a CBM paired with a natal kick. Still, it is once again difficult to be sure with the conflict between UVOT lightcurve data and extragalactic databases.

5.4 GRB 100117A

Despite GRB 100117A only turning up one dark data pair, this pair is optically-dark for $\Delta t_{\%} = 20\%$, 10%, and 5% according to the Van der Horst method (Table 5). Therefore, although it lies outside of the scope of Jakobsson-dark GRBs, we are confident that the Van der Horst method clearly identifies it as optically-dark.

[35] When a neutron star is born after a massive star collapses in on itself, because the implosion is not always symmetric, neutron stars are often "kicked" in a certain direction following the collapse. This "kick" is called the *natal kick*.

[36] Hydrogen gas, which exists nearly everywhere in the universe, absorbs light at very specific wavelengths. For celestial objects which emit radiation which are close to the Earth, the UV-band is primarily effected by Hydrogen. However, due to redshift caused by the expansion of the universe, celestial objects which are farther away have their visible range effected. Thus, since the g-band (a blue band) is observed for GRB 080503, this puts a maximum limit on how far away it originated.

GRB 100117A was detected by *Swift* at 21:06:19 UTC on January 17$^{\text{th}}$, 2010, with a burst duration of $T_{90} = 0.30 \pm 0.05$ [s] (Fong et al. 2011). With a best-fit spectroscopic redshift of $z = 0.915$, GRB 100117A is one of few bursts which has been unambiguously associated with a specific early-type galaxy, or a galaxy with very little star formation (Fong et al. 2011). This is particularly notable because the chance of a CBM occurring in such an early-type galaxy is rather low. Rather, CBMs are expected to occur in older galaxies because they have a greater chance of producing neutron stars or black holes. The probability of a CBM in an early-type galaxy is not nonzero, however, but this does raise an interesting question as to the circumstances required for this young galaxy to produce the CBM associated with GRB 100117A.

5.5 GRB 111020A

With one dark data pair, we find that GRB 111020A is dark according to the Jakobsson method at $\Delta t_{\%} =$ 10% and 5% (the reason for it not being Jakobsson dark at $\Delta t_{\%} = 20\%$ is due to the increase in error that arises from $\Delta\beta_{ox}$ at such a high temporal separation; Table 5). Fong et al. (2012) also calculates that GRB 111020A is dark, citing similar values for β_x and β_{ox} to what we calculate (Table 5).

GRB 111020A was detected by the Burst Alert Telescope (BAT) on *Swift* at 06:33:49.0 UTC on October 20$^{\text{th}}$, 2011. The gamma-ray emission of this GRB is characterized by a single pulse with $T_{90} = 0.40 \pm 0.09$ [s] in the 15-350 keV band, placing it confidently in the short GRB category. In terms of a progenitor, the calculated circumburst density of GRB 111020A is consistent with what is expected for a NS-NS or NS-BH binary system (Fong et al. 2012). However, although there are three candidates which are close enough to the determined origin to be this GRB's host galaxy, calculations do not provide conclusive evidence that it originated from any of these galaxies. While Fong et al. (2012) interpret this as evidence for placing GRB 111020A in a class of apparently "hostless" short GRBs occuring fairly far from their host galaxies, little evidence has been shown to support this theory since then. In addition, combined with its known optical extinction (darkness), the location of GRB 111020A on the outskirts of its host galaxy implies that it may have originated in a star-forming region or a system which produced the dust itself. However, while such an environment could make sense for a long dark GRB, it is not commonly associated with a CBM. As such, while it would be unlikely that the progenitor of this GRB was something other than a CBM, this discrepancy still casts a slight shadow on determination of the source of GRB 111020A.

5.6 GRB 120804A

GRB 120804A yields two dark data points, both of which are dark at $\Delta t_{\%} = 20\%$ according to both the Jakobsson and Van der Horst methods (Table 5). Berger et al. (2013) also notes that the optical and X-ray flux densities are comparable at $\Delta t \approx 5$ hr, predicting that it is a dark burst according to the Jakobsson method with $\beta_{ox} \approx 0$ (we find $\beta_{ox} = 0.03, -0.07$).

Swift's BAT discovered GRB 120804A at 00:55:47.8 UTC on August 4$^{\text{th}}$, 2012. In the 15-350 keV range, this burst is confidently defined as short with a $T_{90} = 0.81 \pm 0.08$ [s] (Berger et al. 2013). In addition, calculations reveal a highly-likely candidate for this GRB's host galaxy - it is identified as an ultra-luminous infrared galaxy (ULIRG) located at a photometric redshift of $z \approx 1.3$, making GRB 120804A the first short GRB to be associated with a ULIRG. It is believed that this ULIRG host galaxy is most likely part of an interacting/merging system, something which is common to ULIRGs and may also explain GRB 120804A's

optical darkness. However, a merging ULIRG host galaxy does not create ideal conditions for a NS-NS or NS-BH CBM due to the high rate of star formation rather than stars reaching the end of their lifetimes. While CBMs are not explicitly precluded from occuring in ULIRGs, the case of GRB 120804A is undoubtedly unique. This gives rise to the idea that perhaps the near-infrared spectral distribution should be taken into account for classifying dark GRBs. As such, GRB 120804A might be valuable to pursue further in future work regarding dark GRBs.

5.7 GRB 130603B

GRB 130603B generates 5 dark data points, all of which are dark only according to the Van der Horst method at $\Delta t_{\%} = 20\%$ (Table 5). Unique to this GRB is its confirmed linkage with a kilonova, an association which greatly strengthened the kilonova progenitor theory for short GRBs in the astrophysics community (Tanvir et al. 2013). Specifically, the CBM which created the kilonova is considered to be the collision of two neutron stars which resulted in a magnetar with a spin period on the order of 1 [ms] (de Ugarte Postigo et al. 2013).

Identified by BAT on *Swift* at 15:49:14 UTC on June 3$^{\text{rd}}$, 2013, this burst is determined to have a redshift of $z = 0.3565 \pm 0.002$ (de Ugarte Postigo et al. 2013). Because of this burst's undoubted association with a kilonova, scientists were able to determine its cosmological origin more accurately than for many others. It is found that GRB 130603B originated in a tidally disrupted arm in the outskirts of a perturbed spiral galaxy (caused by a past collision with another galaxy), something which was able to be seen in high-resolution imaging using the Hubble Space Telescope (Tanvir et al. 2013, de Ugarte Postigo et al. 2013) and may be linked to its optical darkness. Thus, although different than that of GRB 120804A, this environment plays strongly into the CBM progenitor theory for short GRBs.

5.8 GRB 130912A

GRB 130912A yields 5 dark data points, all of which are dark for some $\Delta t_{\%}$ according to the Van der Horst method and 3 of which are dark at $\Delta t_{\%} = 5\%$ according to the Van der Horst method (Table 5). Detected by the BAT on *Swift* at 08:34:57 UTC on September 12$^{\text{th}}$, 2013, $T_{90} = 0.28 \pm 0.03$ [s] for this burst, confirming its shortness. Although the precise redshift for this burst is unknown, it is assumed to be in the reasonable range $0.1 \leq z \leq 1.4$ (Zhu et al. 2015). Unique to this GRB, however, is it's unusual and extremely long-lasting optical afterglow, spanning $\approx 3.2 \times 10^3$ [s] (Zhu et al. 2015). Yet despite this strange temporal behavior of the optical emission, we note that all 5 of GRB 130912A's dark data points occur early on in the afterglow measurement time, with 0.27 [s] $< \Delta t < 42$ [s]. Thus, we are unconcerned in this optical behavior effecting our darkness calculation. Finally, the circumburst medium of this GRB is found to have a very low density, consistent with the NS-NS or NS-BH CBM progenitor theory for short GRBs (Zhu et al. 2015).

5.9 GRB 150120A

GRB 150120A produces just one dark data point which is dark at $\Delta t_{\%} = 20\%$ according to the Jakobsson method (Table 5). As such, we are not nearly as confident in the classification of this burst as dark, especially considering that the dark pair contains $\sigma_{\pm ox} = 0$. Moreover, there currently exist no formal publications concerning this burst other than GCN circulars. Still, we note that *Swift*'s BAT triggered on GRB 150120A at 02:57:46 UTC on January 20$^{\text{th}}$, 2015. It is calculated to have a $T_{90} \approx 1.20$ [s] and a redshift of $z \approx 0.46$

(Chornock & Fong 2015). More in-depth analysis must be done to reveal any features unique to this GRB that may explain its optical darkness.

6 Conclusion

We have presented a comprehensive analysis and identification of 9 optically-dark short GRBs of the 70 detected since the launch of *Swift* in 2004 through March 2015. In identifying these bursts, we illustrated how we created user-interactive code to computationally automate the calculation of the optical-to-X-ray spectral index, β_{ox}, and the graphing of GRB data on the $\beta_x - \beta_{ox}$ plane. By investigating trends between these 9 bursts' host galaxies, redshifts, UVOT lightcurves, and other relevant factors, we find that the proportion of short GRBs which are dark is $\approx 13\%$, which is far less than the analogous proportion for long GRBs ($\approx 40 - 60\%$). Since optical darkness can be explained by a viscous circum-stellar medium in the host galaxy, our result agrees with the compact binary merger progenitor model for short GRBs because the natal kick associated with neutron star formation makes CBMs less likely to occur in these kind of dense environments. Thus, a smaller relative proportion of dark short GRBs than long GRBs is to be expected. Moreover, review of other journal publications confirms many of our calculations and identifications of optical darkness, as was seen explicitly for GRB 120804A. Despite this affirmation, it is clear that more exhaustive case studies must be done to establish any distinguishing explanation for optical darkness. In addition to adding to the sample those short GRBs which were detected between March 2015 and the present, our analysis also reveals that it may be valuable to incorporate a comparison between the optical spectral distribution to the near-infrared in addition to the X-ray when evaluating optical darkness. It is hoped that the computational tools presented here have paved the way for such future work, as well as greatly eased the process of manipulating large data sets of GRB X-ray and optical data and identifying optically dark GRBs.

7 Acknowledgments

I wish to thank Professor Alexander van der Horst[37] of The George Washington University for his unwavering mentorship, guidance, and support throughout the research process. I further wish to acknowledge Professor Rhonda Dzakpasu[38], Professor Wesley Mathews[39], Michael Rushka[40], and the rest of the Georgetown University Physics Department staff and students for their constant support and commitment to ensuring my success throughout my four undergraduate years. I finally wish to thank my family - especially my parents, sister, and grandparents - for their steadfast encouragement and care, and for always inspiring me to pursue my dreams.

[37] Research Advisor
[38] Major Advisor
[39] Second Reader
[40] Peer Reviewer

8 Appendix

Table 6
Short GRB X-Ray and Optical Afterglow Catalog with $\Delta\beta_{ox} = 0$

						Dark-20%?		Dark-10%?		Dark-5%?	
GRB	Δt_x [hr]	Δt_o [hr]	$\lvert\delta t\rvert$ [hr]	β_x	β_{ox}	Jak	VdH	Jak	VdH	Jak	VdH
060121	3.06	2.6	0.46	-1.07 ± 0.16	-0.37 ± 0.09	Y	Y	N	N	N	N
060121	3.06	2.6	0.46	-1.07 ± 0.16	-0.41 ± 0.09	Y	Y	N	N	N	N
060121	3.06	2.8	0.26	-1.07 ± 0.16	-0.29 ± 0.08	Y	Y	Y	Y	N	N
060121	3.06	2.8	0.26	-1.07 ± 0.16	-0.35 ± 0.08	Y	Y	Y	Y	N	N
060121	3.06	2.8	0.26	-1.07 ± 0.16	-0.34 ± 0.08	Y	Y	Y	Y	N	N
060121	3.06	2.8	0.26	-1.07 ± 0.16	-0.33 ± 0.08	Y	Y	Y	Y	N	N
060121	3.33	2.8	0.53	-1.07 ± 0.16	-0.31 ± 0.08	Y	Y	N	N	N	N
060121	3.33	2.8	0.53	-1.07 ± 0.16	-0.32 ± 0.08	Y	Y	N	N	N	N
060121	3.33	2.8	0.53	-1.07 ± 0.16	-0.29 ± 0.08	Y	Y	N	N	N	N
060121	3.33	2.8	0.53	-1.07 ± 0.16	-0.37 ± 0.09	Y	Y	N	N	N	N
060121	3.33	2.8	0.53	-1.07 ± 0.16	$-0.37^{+0.08}_{-0.07}$	Y	Y	N	N	N	N
060121	3.33	3.9	0.57	-1.07 ± 0.16	$-0.12^{+0.09}_{-0.08}$	Y	Y	N	N	N	N
060121	3.33	3.9	0.57	-1.07 ± 0.16	$-0.12^{+0.09}_{-0.08}$	Y	Y	N	N	N	N
060121	3.33	3.9	0.57	-1.07 ± 0.16	$-0.10^{+0.09}_{-0.08}$	Y	Y	N	N	N	N
060121	3.33	3.9	0.57	-1.07 ± 0.16	-0.18 ± 0.09	Y	Y	N	N	N	N
060121	3.33	3.9	0.57	-1.07 ± 0.16	$-0.17^{+0.09}_{-0.08}$	Y	Y	N	N	N	N
060121	4.44	3.9	0.54	-1.07 ± 0.16	-0.22 ± 0.09	Y	Y	N	N	N	N
060121	4.44	3.9	0.54	-1.07 ± 0.16	-0.24 ± 0.09	Y	Y	N	N	N	N
060121	4.44	3.9	0.54	-1.07 ± 0.17	-0.17 ± 0.09	Y	Y	N	N	N	N
060121	4.44	4.4	0.04	-1.07 ± 0.16	-0.39 ± 0.07	Y	Y	Y	Y	Y	Y
060121	5.00	4.4	0.60	-1.07 ± 0.16	-0.42 ± 0.09	Y	N	N	N	N	N
060121	5.00	4.4	0.60	-1.07 ± 0.16	$-0.40^{+0.07}_{-0.08}$	Y	Y	N	N	N	N
060121	4.44	5.4	0.96	-1.07 ± 0.16	$-0.24^{+0.12}_{-0.10}$	Y	Y	N	N	N	N
060121	4.44	5.4	0.96	-1.07 ± 0.16	$-0.26^{+0.12}_{-0.10}$	Y	Y	N	N	N	N
060121	4.44	5.4	0.96	-1.07 ± 0.16	$-0.18^{+0.12}_{-0.10}$	Y	Y	N	N	N	N
060121	4.72	5.4	0.68	-1.07 ± 0.16	$-0.23^{+0.12}_{-0.10}$	Y	Y	N	N	N	N
060121	4.72	5.4	0.68	-1.07 ± 0.16	$-0.22^{+0.12}_{-0.10}$	Y	Y	N	N	N	N
060121	4.72	5.4	0.68	-1.07 ± 0.16	$-0.21^{+0.12}_{-0.10}$	Y	Y	N	N	N	N
060121	5.00	5.4	0.40	-1.07 ± 0.16	$-0.20^{+0.12}_{-0.10}$	Y	Y	Y	Y	N	N
060121	5.00	5.4	0.40	-1.07 ± 0.16	$-0.18^{+0.12}_{-0.10}$	Y	Y	Y	Y	N	N
060121	5.00	5.4	0.40	-1.07 ± 0.16	$-0.29^{+0.11}_{-0.09}$	Y	Y	Y	Y	N	N
060121	6.11	5.4	0.71	-1.07 ± 0.16	$-0.26^{+0.12}_{-0.10}$	Y	Y	N	N	N	N
060121	4.72	5.6	0.88	-1.07 ± 0.16	$-0.21^{+0.07}_{-0.08}$	Y	Y	N	N	N	N
060121	4.72	5.6	0.88	-1.07 ± 0.16	$-0.20^{+0.07}_{-0.08}$	Y	Y	N	N	N	N
060121	4.72	5.6	0.88	-1.07 ± 0.16	$-0.19^{+0.07}_{-0.08}$	Y	Y	N	N	N	N
060121	5.00	5.6	0.60	-1.07 ± 0.16	$-0.19^{+0.07}_{-0.08}$	Y	Y	N	N	N	N

Table 6

(Continued)

						Dark-20%?		Dark-10%?		Dark-5%?	
GRB	Δt_x [hr]	Δt_o [hr]	$\lvert\delta t\rvert$ [hr]	β_x	β_{ox}	Jak	VdH	Jak	VdH	Jak	VdH
060121	5.00	5.6	0.60	-1.07 ± 0.16	$-0.17^{+0.07}_{-0.08}$	Y	Y	N	N	N	N
060121	5.00	5.6	0.60	-1.07 ± 0.16	$-0.28^{+0.06}_{-0.07}$	Y	Y	N	N	N	N
060121	6.11	5.6	0.51	-1.07 ± 0.16	-0.25 ± 0.07	Y	Y	Y	Y	N	N
060121	6.11	6.5	0.39	-1.07 ± 0.16	$-0.24^{+0.04}_{-0.05}$	Y	Y	Y	Y	N	N
060121	6.11	7.3	1.19	-1.07 ± 0.16	$-0.26^{+0.06}_{-0.07}$	Y	Y	Y	Y	N	N
060121	6.11	7.3	1.19	-1.07 ± 0.16	-0.29 ± 0.08	Y	Y	Y	Y	N	N
060121	24.72	30.0	5.28	-1.07 ± 0.16	-0.47 ± 0.03	Y	Y	Y	Y	N	N
060313	1.39	1.3	0.09	-0.96 ± 0.09	$-0.35^{+0.11}_{-0.12}$	Y	N	Y	N	N	N
060313	1.22	1.5	0.28	-0.96 ± 0.09	$-0.26^{+0.13}_{-0.14}$	Y	Y	N	N	N	N
060313	1.25	1.5	0.25	-0.96 ± 0.09	$-0.33^{+0.13}_{-0.14}$	Y	N	N	N	N	N
060313	1.31	1.5	0.19	-0.96 ± 0.09	$-0.33^{+0.13}_{-0.14}$	Y	N	N	N	N	N
060313	1.33	1.5	0.17	-0.96 ± 0.09	$-0.35^{+0.13}_{-0.14}$	Y	N	N	N	N	N
060313	1.36	1.5	0.14	-0.96 ± 0.09	$-0.35^{+0.14}_{-0.15}$	Y	N	Y	N	N	N
060313	1.39	1.5	0.11	-0.96 ± 0.09	$-0.19^{+0.13}_{-0.15}$	Y	Y	Y	Y	N	N
060313	1.58	1.5	0.08	-0.96 ± 0.09	$-0.30^{+0.13}_{-0.15}$	Y	Y	Y	Y	N	N
060313	1.67	1.5	0.17	-0.96 ± 0.09	$-0.33^{+0.13}_{-0.15}$	Y	N	N	N	N	N
080503	108.33	97.1	11.23	$-1.53^{+0.26}_{-0.12}$	$-0.66^{+0.09}_{-0.13}$	N	Y	N	N	N	N
080503	108.33	125.0	16.67	$-1.53^{+0.26}_{-0.12}$	$-0.78^{+0.05}_{-0.08}$	N	Y	N	N	N	N
080503	108.33	128.6	20.27	$-1.53^{+0.26}_{-0.12}$	$-0.27^{+0.06}_{-0.10}$	Y	Y	N	N	N	N
100117A	8.61	8.3	0.31	-1.6 ± 0.3	$-0.75^{+0.08}_{-0.9}$	N	Y	N	Y	N	Y
111020A	19.17	17.8	1.37	-1.04 ± 0.16	-0.47 ± 0.01	Y	N	Y	N	N	N
120804A	4.72	5.5	0.78	-1.1 ± 0.1	0.03 ± 0.08	Y	Y	N	N	N	N
120804A	6.39	5.5	0.89	-1.1 ± 0.1	-0.07 ± 0.07	Y	Y	N	N	N	N
130603B	1.72	2.1	0.38	-1.2 ± 0.1	$-0.54^{+0.04}_{-0.06}$	N	Y	N	N	N	N
130603B	1.75	2.1	0.35	-1.2 ± 0.1	$-0.53^{+0.04}_{-0.05}$	N	Y	N	N	N	N
130603B	1.81	2.1	0.29	-1.2 ± 0.1	$-0.59^{+0.04}_{-0.05}$	N	Y	N	N	N	N
130603B	2.72	2.4	0.32	-1.2 ± 0.1	-0.44 ± 0.09	N	Y	N	N	N	N
130603B	2.75	2.4	0.35	-1.2 ± 0.1	-0.52 ± 0.09	N	Y	N	N	N	N
130603B	2.78	2.4	0.38	-1.2 ± 0.1	-0.54 ± 0.09	N	Y	N	N	N	N
130603B	5.83	7.1	1.27	-1.2 ± 0.1	$-0.55^{+0.04}_{-0.05}$	N	Y	N	N	N	N
130603B	6.11	7.1	0.99	-1.2 ± 0.1	$-0.55^{+0.04}_{-0.05}$	N	Y	N	N	N	N
130912A	0.27	0.27	0.00	-0.57 ± 0.13	-0.10 ± 0.07	Y	N	Y	N	Y	N
130912A	0.31	0.27	0.04	-0.57 ± 0.13	-0.22 ± 0.07	Y	N	N	N	N	N
130912A	0.31	0.38	0.07	-0.57 ± 0.13	-0.25 ± 0.07	Y	N	N	N	N	N
130912A	0.33	0.38	0.05	-0.57 ± 0.13	-0.37 ± 0.07	Y	N	N	N	N	N
130912A	0.36	0.38	0.02	-0.57 ± 0.13	-0.37 ± 0.07	Y	N	Y	N	Y	N
130912A	0.39	0.38	0.01	-0.57 ± 0.13	-0.36 ± 0.07	Y	N	Y	N	Y	N
130912A	0.42	0.38	0.04	-0.57 ± 0.13	-0.34 ± 0.06	Y	N	Y	N	N	N
150120A	2.31	2.0	0.31	-1.0 ± 0.31	-0.39	Y	N	N	N	N	N

Table 6

(Continued)

						Dark-20%?		Dark-10%?		Dark-5%?	
GRB	Δt_x [hr]	Δt_o [hr]	$\lvert\delta t\rvert$ [hr]	β_x	β_{ox}	Jak	VdH	Jak	VdH	Jak	VdH

Table 6: The raw data for GRB pairings and their corresponding darkness status according to the Jakobsson and Van der Horst methods for $\Delta t_{\%} = 5\%$, $\Delta t_{\%} = 10\%$, and $\Delta t_{\%} = 20\%$ without $\Delta\beta_{ox}$.

Table 7
Optical Telescope Filter Catalog

Telescope	Instrument	Filter	Wavelength [nm]	Frequency [Hz]
Mount John	MOA	R	859	3.49×10^{14}
WIYN	OPTIC	r	625	4.80×10^{14}
Danish Telescope	DFOSC	R	648	4.63×10^{14}
VLT	FORS2	V	557	5.38×10^{14}
VLT	FORS2	I	768	5.38×10^{14}
VLT	FORS2	R	655	4.58×10^{14}
HST	ACS	F814W	833	3.60×10^{14}
VLT	FORS1	V	557	5.38×10^{14}
Magellan/Baade	PANIC	K	2179	1.38×10^{14}
VLT	FORS1	R	655	4.58×10^{14}
VLT	FORS1	I	768	3.90×10^{14}
CAHA 2.2m	CAFOS	I	850	3.53×10^{14}
CAHA 2.2m	CAFOS	R	650	4.61×10^{14}
Magellan/Clay	LDSS3	r	635	4.72×10^{14}
Gemini North	GMOS	r	630	4.76×10^{14}
Gemini North	GMOS	i	780	3.84×10^{14}
Gemini North	GMOS	z	972	3.08×10^{14}
OSN	–	I	889	3.37×10^{14}
NOT	MOSCA	r	680	4.41×10^{14}
NOT	ALFOSC	R	646	4.64×10^{14}
OSN	–	R	660	4.54×10^{14}
Bok	90 Prime	R	700	4.28×10^{14}
Bok	90 Prime	B	440	6.81×10^{14}
WHT	–	K	2221	1.35×10^{14}
WIYN	–	R	640	4.68×10^{14}
APO/ARC	NIC-FPS	K	2147	1.40×10^{14}
Swift	UVOT	V	547	5.48×10^{14}
Swift	UVOT	U	347	8.64×10^{14}
Swift	UVOT	white	400	7.50×10^{14}
SMARTS	ANDICAM	R_c	658	4.56×10^{14}
SMARTS	ANDICAM	I_c	780	3.84×10^{14}
Hale	LFC	r	626	4.79×10^{14}

Table 7

(Continued)

Telescope	Instrument	Filter	Wavelength [nm]	Frequency [Hz]
Blanco	ISPI	J	1250	2.40×10^{14}
VLT	ISAAC	J	1250	2.40×10^{14}
Liverpool	–	r	622.5	4.82×10^{14}
Liverpool	–	i	780	3.84×10^{14}
WHT	–	R	660	4.54×10^{14}
TNG	NICS	J	1279	2.34×10^{14}
Gemini North	GMOS	g	475	6.31×10^{14}
Gemini North	NIRI	K_s	2150	1.39×10^{14}
Gemini North	NIRI	J	1250	2.40×10^{14}
Gemini North	NIRI	H	1650	1.82×10^{14}
ESO/MPG	GROND	r	622	4.82×10^{14}
ESO/MPG	GROND	J	1240	2.42×10^{14}
Keck I	LRIS	R	642	4.67×10^{14}
Keck I	LRIS	g	473	6.34×10^{14}
Keck I	LRIS	B	437	6.86×10^{14}
HST	WFPC2	F606W	591	5.07×10^{14}
Loiano	–	R	625	4.80×10^{14}
Faulkes North	–	R	641	4.68×10^{14}
P200	LFC	R	630	4.76×10^{14}
ESO/MPG	GROND	g	477	6.29×10^{14}
ESO/MPG	GROND	i	763	3.93×10^{14}
ESO/MPG	GROND	z	913	3.28×10^{14}
Gemini South	GMOS	i	780	3.84×10^{14}
TNT	–	R	590	5.08×10^{14}
TNT	–	V	720	4.16×10^{14}
TLS	–	I	900	3.33×10^{14}
TLS	–	R	640	4.68×10^{14}
ESO/MPG	GROND	H	1540	1.95×10^{14}
ESO/MPG	GROND	K	2180	1.38×10^{14}
WIYN	WHIRC	J	1250	2.40×10^{14}
VLT	HAWK I	K	2146	1.40×10^{14}
VLT	HAWK I	J	1258	2.38×10^{14}
Magellan	IMACS	r	665	4.51×10^{14}
Magellan	IMACS	R	680	4.41×10^{14}
Magellan	PANIC	J	1215	2.37×10^{14}
Magellan	FourStar	J	1265	2.37×10^{14}
WHT	ACAM	z	980	3.06×10^{14}
WHT	ACAM	i	770	3.89×10^{14}
Magellan	LDSS3	i	785	3.82×10^{14}
GTC	OSIRIS	r	641	4.68×10^{14}

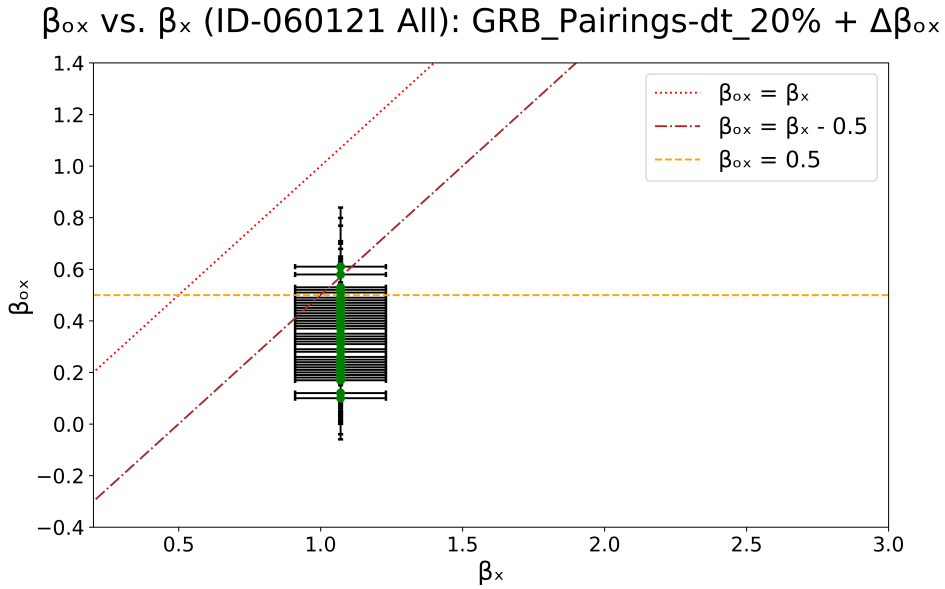
Table 7

(Continued)

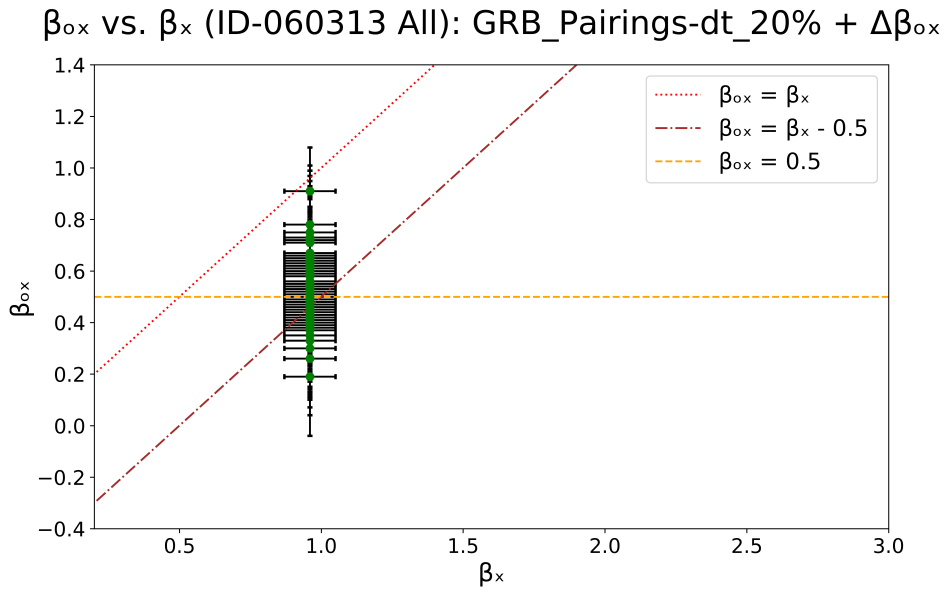
Telescope	Instrument	Filter	Wavelength [nm]	Frequency [Hz]
LCO/duPont	WFCCD	R	600	5.00×10^{14}
TNG	DOLORES	r	620	4.84×10^{14}
WHT	ACAM	z	980	3.06×10^{14}
WHT	ACAM	i	770	3.89×10^{14}
Magellan	LDSS3	i	785	3.82×10^{14}
GTC	OSIRIS	r	641	4.68×10^{14}
LCO/duPont	WFCCD	R	600	5.00×10^{14}
TNG	DOLORES	r	620	4.84×10^{14}
WHT	–	z	1000	3.00×10^{14}
CAHA	DLR MKIII	V	540	5.55×10^{14}
WHT	ACAM	g	765	3.92×10^{14}
Gemini South	GMOS	g	475	6.31×10^{14}
UKIRT	WFCAM	K	2200	1.36×10^{14}
UKIRT	WFCAM	J	1250	2.40×10^{14}
Magellan/Baade	IMACS	r	665	4.51×10^{14}
P60	–	r	624	4.80×10^{14}
WHT	ACAM	g	485	6.18×10^{14}
Johnson	RATIR	r	624	4.80×10^{14}
Johnson	RATIR	i	770	3.89×10^{14}
Johnson	RATIR	Z	878	3.41×10^{14}
Johnson	RATIR	Y	1002	2.99×10^{14}
Johnson	RATIR	J	1250	2.40×10^{14}
Johnson	RATIR	H	1635	1.83×10^{14}
TNG	DOLORES	R	658	4.66×10^{14}
Subaru	IRCS+AO18	K'	2120	1.41×10^{14}
AAO	–	R	654	4.58×10^{14}
Magellan/Baade	Fourstar	J	1265	2.37×10^{14}
DCT	LMI	r	624	4.80×10^{14}
WHT	ACAM	r	640	4.68×10^{14}
Magellan/Baade	IMACS	i	770	3.89×10^{14}
MMT	MMTCam	r	623	4.81×10^{14}
Keck	MOSFIRE	K_s	2147	1.40×10^{14}
Keck	MOSFIRE	J	1253	2.39×10^{14}
Nanshan	–	R	635	4.72×10^{14}
Gemini South	GMOS	r	630	4.76×10^{14}
VLT	HAWK I	H	1620	1.85×10^{14}
Gemini North	GMOS	r	630	4.76×10^{14}

Table 7: A table cataloging the optical telescopes, instruments, filters and their corresponding wavelengths and frequencies for those telescopes which are relevant to this report. Wavelength and corresponding frequency information for optical telescope filters is publicly available online.

8.1 β_{ox} vs. β_x Plots for the 9 Dark GRBs

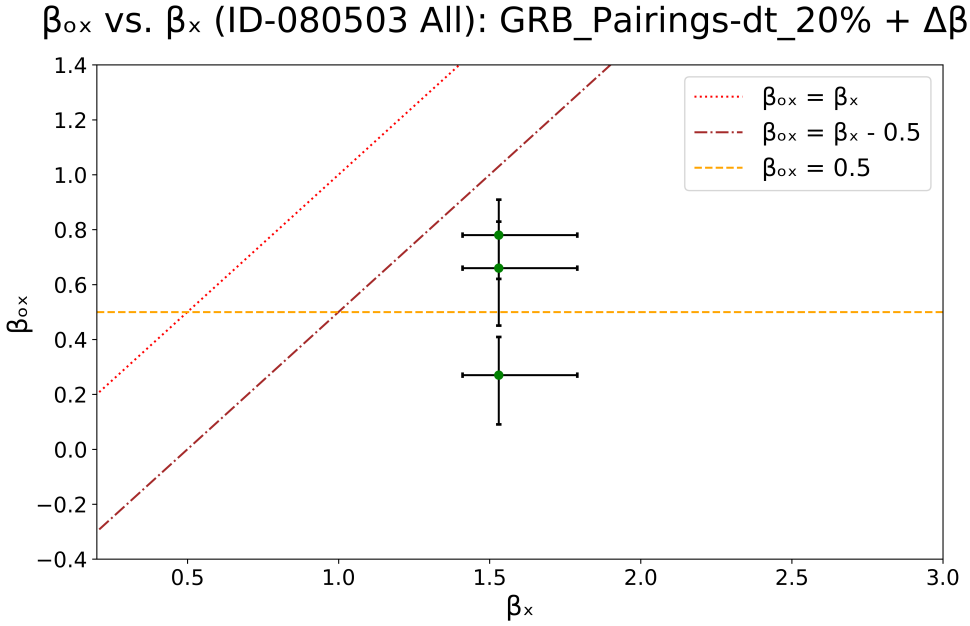


(a)

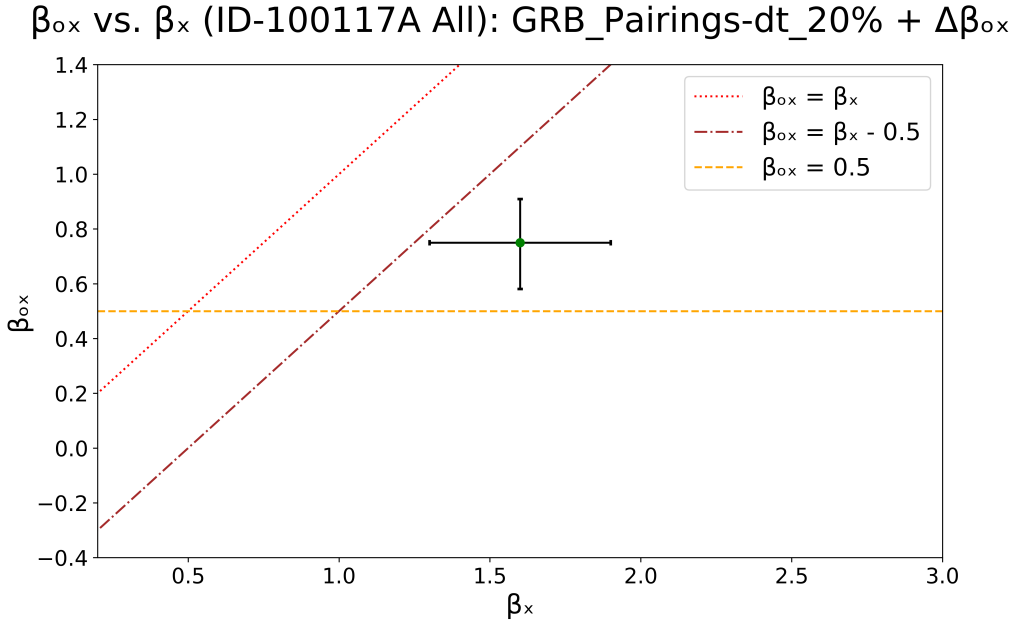


(b)

Figure 20: Graphs of β_{ox} vs. β_x for $\Delta t_{\%} = 20\%$ for all GRB pairings for the 9 identified dark short GRBs: (a) GRB 060121, (b) GRB 060313, (c) GRB 080503, (d) GRB 100117A, (e) GRB 111020A, (f) GRB 120804A, (g) GRB 130603B, (h) GRB 130912A, and (i) GRB 150120A.

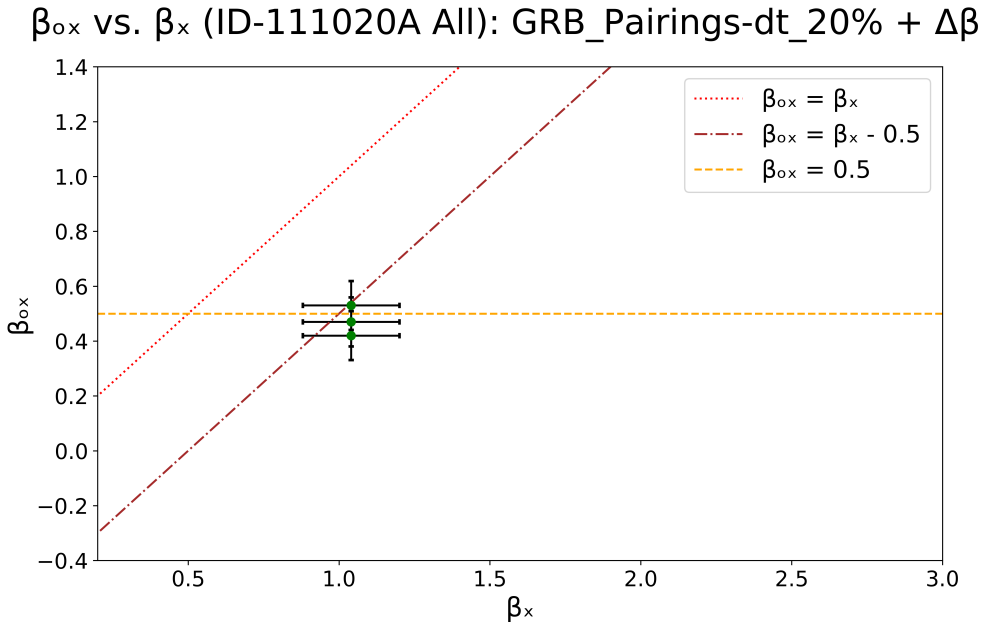


(c)

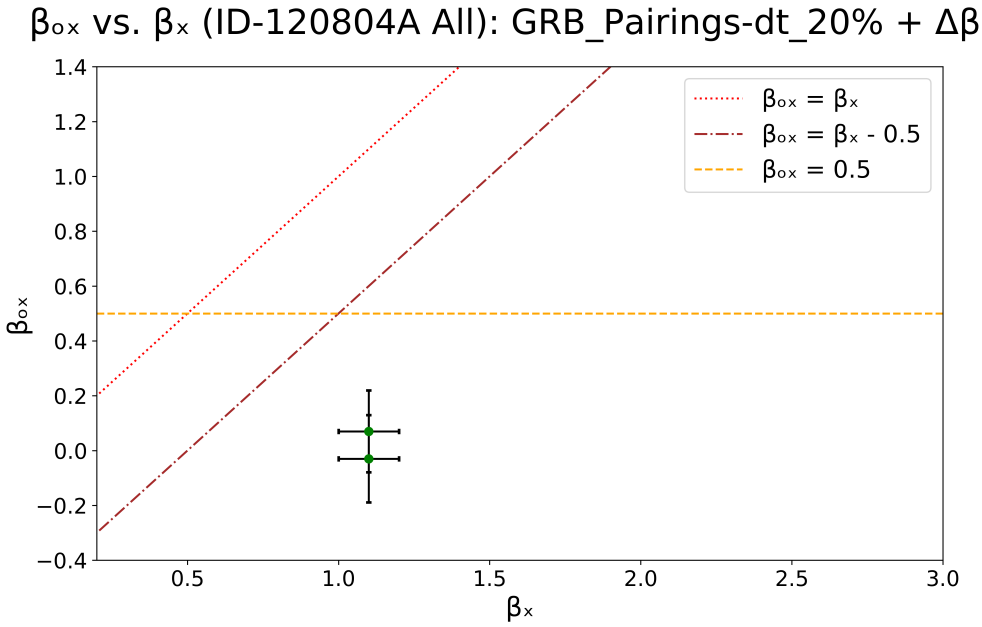


(d)

Figure 20: (Cont.)

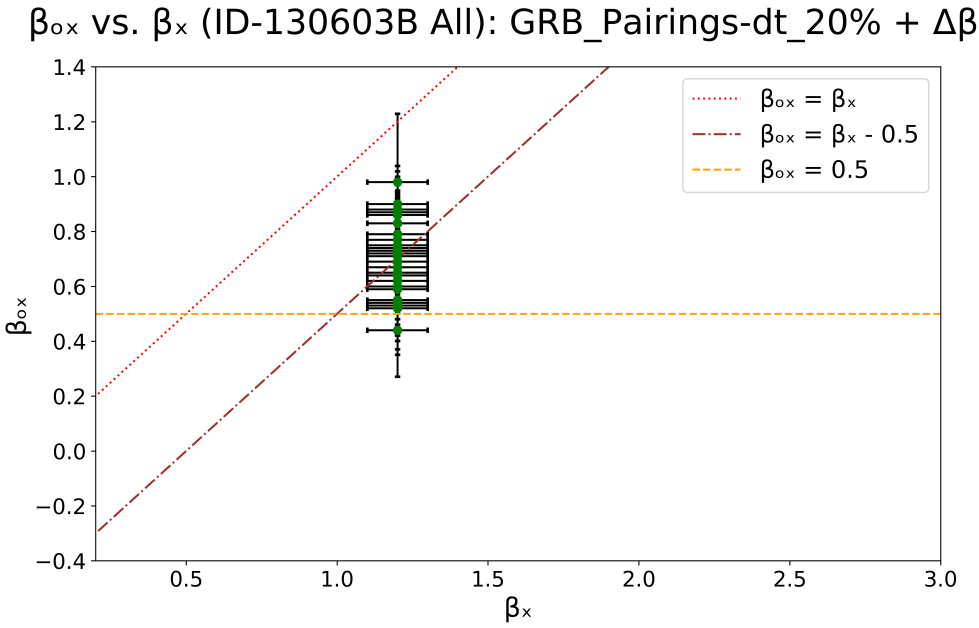


(e)

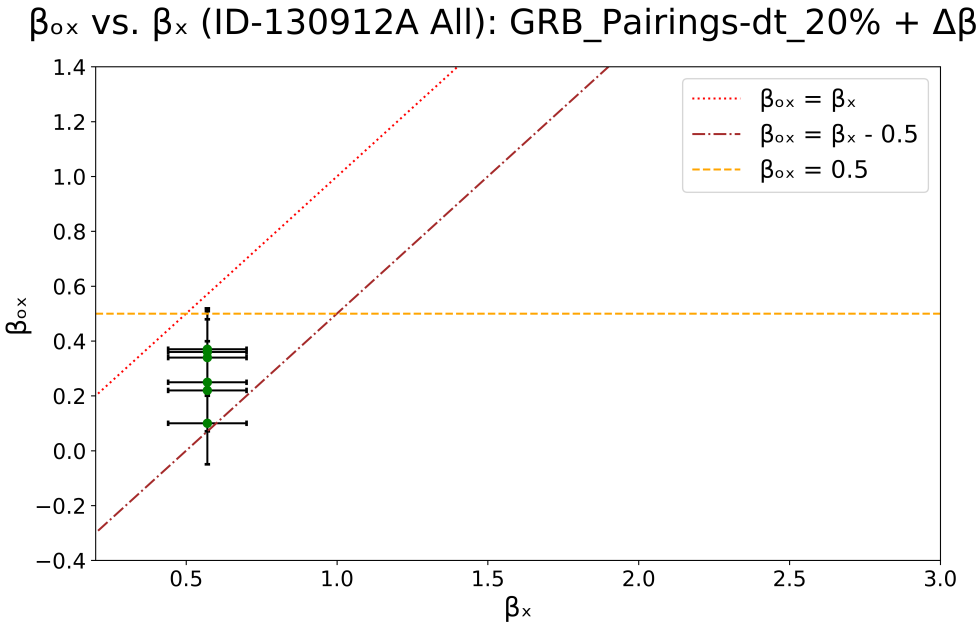


(f)

Figure 20: (Cont.)

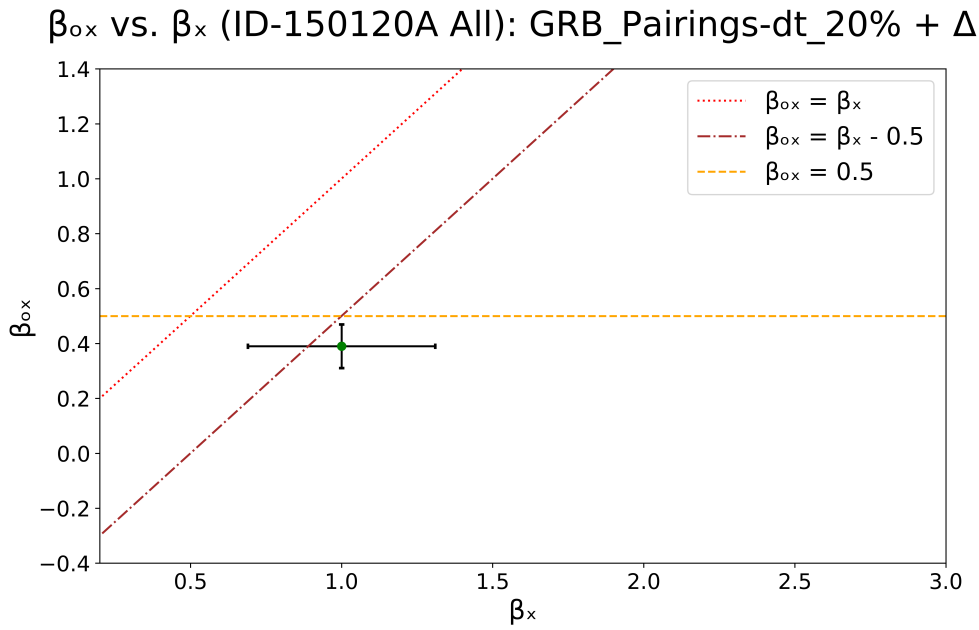


(g)



(h)

Figure 20: (Cont.)



(i)

Figure 20: (Cont.)

8.2 C++ Source Code

```
1
2  /**
3   * Title: "Automating the Calculation of Beta_OX"
4   *
5   * Copyright (C) 2020 David Fitzpatrick
6   *
7   * From: "Analyzing Optically-Dark Short Gamma Ray Bursts"
8   * (1) David Fitzpatrick, (2) Professor Alexander van der Horst, Ph.D.
9   *
10  * 1. Georgetown University, Department of Physics,
11  *    37 and O Streets NW, Washington D.C. 20057
12  * 2. The George Washington University, Department of Physics,
13  *    725 21 Street NW, Washington D.C. 20052
14  *
15  *  I hereby grant to Georgetown University and its agents the non-exclusive, worldwide
16  *  right to reproduce, distribute, display and transmit my thesis in such tangible and
17  *  electronic formats as may be in existence now or developed in the future. I retain all
18  *  ownership rights to the copyright of the thesis including the right to use it in whole
19  *  or in part in future works. I agree to allow the Georgetown University Department of
20  *  Physics to serve as the institutional repository of my thesis and to make it available
21  *  to the Georgetown University community through its website. I certify that the version
22  *  that I have submitted is the same version that was approved by my senior research
23  *  advisor.
24  *
25  * Description: In an effort to automate the calculation of the optical to X-Ray spectral
```

```
 * index (Beta_OX) of well-documented Gamma Ray Bursts (GRBs), this program loads in
 * multiple files containing disparate GRB characteristics (in order: X-ray flux data,
 * Beta_X data, optical flux data, and optical telescope filter data; fields correspond to
 * Tables in Fong et al. 2015) and pairs burst measurements based on ID number and
 * user-defined temporal separation between optical and X-Ray measurements.  The fully-
 * populated GRBs, with parameters which include a calculated value for Beta_OX, are then
 * written to .csv files for further analysis.
 *
 **/

//declare necessary preprocessor directives
#include <iostream>
#include <vector>
#include <string>
#include <iomanip>
#include <cmath>
#include <fstream>
#include <algorithm>
#include <ctime>

//initialize and declare necessary global constants

//set allowed temporal separation [hr]
double DT_PERCENT_DIF;
//set wavelength for X-Rays in nm
//this is based off of an energy of 1 keV and the fact that
// \lambda = hv
const double FREQUENCY_XRAY = 2.415e+17;

//initialize global variable for number of successful pairs after optical
//data is loaded
double optical_pairs = 0;

//initialize variable for total number of possible pairings; that is, if
//there was 100% accuracy for pairing, this would be how many pairs
double total_possible_pairings = 0;

using namespace std;

/*****************************************************************
                      BEGIN CLASS DEFINITIONS
*****************************************************************/

class GRB {

    public:
      // A constructor that creates a GRB beginning with
      // the GRB ID, X-Ray temporal separation, X-Ray
      // exposure time, and X-Ray uncertainty
      GRB(string, double, double, double, double);

      // Prints attributes of particular GRB neatly.  Items should
      // print in the order listed for private data members below
      void report();
```

```
80
81      // Returns the GRB ID of the GRB
82      string getGRB_ID();
83
84      // Accessor and Observers for each field of the X-Ray data.
85      double get_dt_XRay();
86      double get_ExpT_XRay();
87      double get_F_x();
88      double get_sigma_x();
89      string get_References_XRay();
90      void set_Beta_X(double);
91      double get_Beta_X();
92      void set_Beta_X_upper_sigma(double);
93      double get_Beta_X_upper_sigma();
94      void set_Beta_X_lower_sigma(double);
95      double get_Beta_X_lower_sigma();
96
97      // Accessor and Observers for each field of the Optical data
98      void set_dt_Opt(double);
99      double get_dt_Opt();
100     void set_telescope(string);
101     string get_telescope();
102     void set_instrument(string);
103     string get_instrument();
104     void set_filter(string);
105     string get_filter();
106     void set_Exp_Opt(double);
107     double get_Exp_Opt();
108     void set_F_o(double);
109     double get_F_o();
110     void set_sigma_o(double);
111     double get_sigma_o();
112     void set_References_Opt(string);
113     string get_References_Opt();
114     void set_frequency_XRay(double);
115     double get_frequency_XRay();
116     void set_frequency_Opt(double);
117     double get_frequency_Opt();
118     void set_Beta_OX(double);
119     double get_Beta_OX();
120     void set_sigma_OX_upper(double);
121     double get_sigma_OX_upper();
122     void set_sigma_OX_lower(double);
123     double get_sigma_OX_lower();
124
125   private:
126       //17 attributes of the GRB class
127       string GRB_ID;
128       double dt_XRay;
129       double ExpT_XRay;
130       double F_x;
131       double sigma_x;
132       double Beta_X;
133       double Beta_X_upper_sigma;
```

```
        double Beta_X_lower_sigma;
        string References_XRay;
        double dt_Opt;
        string telescope;
        string instrument;
        string filter;
        double ExpT_Opt;
        double F_o;
        double sigma_o;
        string References_Opt;
        double frequency_XRay;
        double frequency_Opt;
        double Beta_OX;
        double sigma_OX_upper;
        double sigma_OX_lower;

};

GRB::GRB(string id, double dt_X, double ExpT_X, double Fx, double sig_x)
{
      //create constructor by the following assignment statements
      GRB_ID = id;
      dt_XRay = dt_X;
      ExpT_XRay = F_x;
      F_x = Fx;
      sigma_x = sig_x;

      //initialize factors loaded after construction to 0 or NULL
      Beta_X = 31415926535;
      Beta_X_lower_sigma = 31415926535;
      Beta_X_upper_sigma = 31415926535;
      dt_Opt = 0;
      telescope = "NULL";
      instrument = "NULL";
      filter = "NULL";
      ExpT_Opt = 0;
      F_o = 0;
      sigma_o = 0;
      References_Opt = "NULL";
      References_XRay = "NULL";
      frequency_XRay = FREQUENCY_XRAY;
      Beta_OX = 0;
      sigma_OX_lower = 0;
      sigma_OX_upper = 0;
      //initialize optical frequency to -1
      //used to check for full population for the Trial print and write function
      frequency_Opt = -1;

}

void GRB::report()
{
    //set the numeric output formatting
    cout << fixed << showpoint << setprecision( 2 );
```

```
188     //print out Subject's attributes from the first-loaded data file
189     cout << GRB_ID << " " << dt_XRay << " " << ExpT_XRay << " " << F_x << " "
190          << sigma_x << " " << Beta_X << " " << Beta_X_upper_sigma << " "
191          << Beta_X_lower_sigma << " " << dt_Opt << " " << telescope << " " << instrument
192          << " " << filter << " " << ExpT_Opt << " " << F_o << " " << sigma_o
193          << " " << " " << frequency_XRay << " " << " " << frequency_Opt << " " << Beta_OX
194          << " " << sigma_OX_upper << " " <<  sigma_OX_lower  <<endl;
195 }
196
197 string GRB::getGRB_ID()
198 {
199     //return GRB ID of GRB
200     return GRB_ID;
201 }
202
203 double GRB::get_dt_XRay()
204 {
205     //return X-Ray dt of GRB
206     return dt_XRay;
207 }
208 double GRB::get_ExpT_XRay()
209 {
210     //return X-Ray Exposure time
211     return ExpT_XRay;
212 }
213 double GRB::get_F_x()
214 {
215     //return the X-Ray flux density
216     return F_x;
217 }
218 double GRB::get_sigma_x()
219 {
220     //return the X-Ray sigma value
221     return sigma_x;
222 }
223 string GRB::get_References_XRay()
224 {
225     //return X-Ray references
226     return References_XRay;
227 }
228 void GRB::set_Beta_X(double b)
229 {
230     //set X-Ray spectral flux density
231     Beta_X = b;
232 }
233 double GRB::get_Beta_X()
234 {
235     //get X-Ray spectral flux density
236     return Beta_X;
237 }
238 void GRB::set_Beta_X_upper_sigma(double u)
239 {
240     //set upper bound of X-Ray spectral flux density
241     Beta_X_upper_sigma = u;
```

```
}
double GRB::get_Beta_X_upper_sigma()
{
    //return upper bound of X-Ray spectral flux density
    return Beta_X_upper_sigma;
}
void GRB::set_Beta_X_lower_sigma(double l)
{
    //set lower bound of X-Ray spectral flux density
    Beta_X_lower_sigma = l;
}
double GRB::get_Beta_X_lower_sigma()
{
    //return lower bound of X-Ray spectral flux density
    return Beta_X_lower_sigma;
}
void GRB::set_dt_Opt(double dt)
{
    //set optical dt of GRB
    dt_Opt= dt;
}

double GRB::get_dt_Opt()
{
    //return optical dt of GRB
    return dt_Opt;
}

void GRB::set_telescope(string tel)
{
    //set optical telescope name
    telescope = tel;
}

string GRB::get_telescope()
{
    //return optical telescope name
    return telescope;
}

void GRB::set_instrument(string i)
{
    //set optical instrument name
    instrument = i;
}

string GRB::get_instrument()
{
    //return optical instrument name
    return instrument;
}

void GRB::set_filter(string f)
{
```

```
296     //set filter name
297     filter = f;
298 }
299 
300 string GRB::get_filter()
301 {
302     //get filter name
303     return filter;
304 }
305 
306 void GRB::set_Exp_Opt(double e)
307 {
308     //set optical exposure time
309     ExpT_Opt = e;
310 }
311 
312 double GRB::get_Exp_Opt()
313 {
314     //return optical exposure time
315     return ExpT_Opt;
316 }
317 
318 void GRB::set_F_o(double f)
319 {
320     //set optical flux density
321     F_o = f;
322 }
323 
324 double GRB::get_F_o()
325 {
326     //get optical flux density
327     return F_o;
328 }
329 
330 void GRB::set_sigma_o(double s)
331 {
332     //set optical flux density standard deviation
333     sigma_o = s;
334 }
335 
336 double GRB::get_sigma_o()
337 {
338     //get optical flux density standard deviation
339     return sigma_o;
340 
341 }
342 
343 void GRB::set_References_Opt(string r)
344 {
345     //set optical references
346     References_Opt = r;
347 }
348 
349 string GRB::get_References_Opt()
```

```
{
    //get optical references
    return References_Opt;
}

void GRB::set_frequency_XRay(double f)
{
    //set X-Ray frequency
    frequency_XRay = f;
}

double GRB::get_frequency_XRay()
{
    //get X-Ray frequency
    return frequency_XRay;
}

void GRB::set_frequency_Opt(double wa)
{
    //set optical frequency
    frequency_Opt = wa;
}

double GRB::get_frequency_Opt()
{
    //get optical frequency
    return frequency_Opt;
}

void GRB::set_Beta_OX(double b)
{
    //set Beta_OX
    Beta_OX = b;
}

double GRB::get_Beta_OX()
{
    //get Beta_OX
    return Beta_OX;
}

void GRB::set_sigma_OX_upper(double s)
{
    //set upper bound on Beta_OX uncertainty
    sigma_OX_upper = s;
}

double GRB::get_sigma_OX_upper()
{
    //get upper bound on Beta_OX uncertainty
    return sigma_OX_upper;
}

void GRB::set_sigma_OX_lower(double s)
```

```
{
    //set lower bound on Beta_OX uncertainty
    sigma_OX_lower = s;
}

double GRB::get_sigma_OX_lower()
{
    //get lower bound on Beta_OX uncertainty
    return sigma_OX_lower;
}

// A class used for determining total number of possible outcomes
// between X-Ray and optical data
class Possibility {

    public:

      // A constructor consisting of the GRB ID number and its
      // corresponding multiplicity
      Possibility( string, int );

      //Accessors and observers for the class
      string get_ID();
      void set_ID(string);
      int get_multiplicity();
      void set_multiplicity(int);

    private:

      //variable for GRB ID
      string ID;
      //variable for multiplicity
      int multiplicity;
};

Possibility::Possibility( string id, int mult )
{
      //create constructor by the following assignment statements
      ID = id;
      multiplicity = mult;
}

string Possibility::get_ID()
{
    //return ID
    return ID;
}

void Possibility::set_ID( string id )
{
    //set ID
    ID = id;
}

```

```
458 int Possibility::get_multiplicity()
459 {
460     //return multiplicity
461     return multiplicity;
462 }
463
464 void Possibility::set_multiplicity( int m )
465 {
466     //set multiplicity
467     multiplicity = m;
468 }
469
470 class Trial {
471
472     public:
473
474       // Simple constructor
475       Trial();
476
477       // Loads X-Ray data file into the vector of GRBs
478       int loadX_RayData(string);
479
480       // Loads Beta_X data file. Uses the GRB ID to locate the
481       // GRB ID in the vector, and then sets the GRB object to
482       // also contain the correct Beta_X values
483       int loadBeta_X(string);
484
485       // Loads optical data file. Uses the GRB ID to locate the
486       // GRB ID in the vector, and then sets the GRB object to
487       // also contain the correct optical data values
488       int loadOpticalData(string);
489
490       // Loads wavelength data file and pairs each GRB based on
491       // telescope and filter information appropriate wavelength
492       int loadWavelengthData(string);
493
494       // A function used to calculate the optical to X-Ray spectral index
495       // as well as the upper and lower bounds on the uncertainty
496       void calculate_Beta_OX();
497
498       // prints out the GRB vector by calling the report method
499       // of each GRB. Used for debugging.
500       void report();
501
502       //writes paired GRB data to file
503       void write_paired_data();
504
505     private:
506
507       // A private function used to return the location
508       // in the  vector of GRBs of a GRB with a
509       // particular GRB ID used in the loadOpticalData
510       // to locate where a particular GRB is
511       int findGRB(string);
```

```
512
513       // A private function which is used to pair X-Ray data and
514       // optical data to a particular GRB based on a small temporal
515       // separation in measurement time
516       int matchGRB( string, double, int);
517
518       // A private function which is used to determine if for a given
519       // GRB ID in the optical data set, there exists at least one GRB
520       // with identical ID read in by the X-Ray Data set
521       int check_ID( string );
522
523       // A private function which is used to remove entities that haven't
524       // been paired with Beta_X data from vector XRay_entries
525       int clean_XRay_entries();
526
527       // A private function which is used to pair GRBs with their
528       // appropriate optical wavelength based on telescope, instrument,
529       // and filter information
530       int matchFrequency( string, string, string, double, double );
531
532       // A private function which is used to determine the total number of
533       // possible pairings between optical and X-Ray data by, in essence,
534       // conducting matrix multiplication between vectors XRay_entries and
535       // optical_entries
536       int find_total_possible_pairings();
537
538       //define a vector of GRBs
539       vector<GRB> GRBs;
540
541       //define a vector of GRBs filled with optical data
542       vector<GRB> GRBs_with_Opt;
543
544       //define a vector which has number of elements corresponding to the total number
545       //of different GRB IDs in the X-Ray file and elements corresponding to the
546       //total number of entries per individual GRB ID
547       //Ex: GRBs_with_Opt.size() = N; GRBs_With_Opt = {1,5,...,n}
548       //This would correspond to the X-Ray file having N different GRB IDs,
549       //and if the first two and last GRBs had IDs 050509B, 050709, and XXXXXX, then
550       //GRB 050509B would have 1 entry, GRB 050709 would have 5 entries, and GRB XXXXXX
551       //would have N entries
552       vector<Possibility> XRay_entries;
553
554       //define a vector identical to XRay_entries but for the optical file
555       vector<Possibility> optical_entries;
556
557       //define a vector of GRBs filled with IDs that exist in optical data but not in X-Ray
558       // data
559       vector<string> IDs_in_Opt_not_X;
560 };
561
562 Trial::Trial()
563 {
564     //constructor is empty because it does not need any attributes
565 }
```

```
566
567
568 int Trial::loadX_RayData( string filename )
569 {
570     //define variables for GRB attributes
571
572     //define variable for GRB ID Number
573     string ID;
574     //define variable for X-Ray dt in seconds
575     double dtX;
576     //define variable for X-Ray exposure time
577     double ExpX;
578     //define variable for optical flux density
579     double Fx;
580     //define variable for std dev of optical flux density
581     double sigmaX;
582
583     //initialize variable for number of GRBs loaded
584     int counter = 0;
585     //initialize string variable for old ID used in GRB ID multiplicity determination
586     string old_ID = "NULL";
587     //initialize counter used for determining number of data points for each unique ID
588     int entries_per_ID = 0;
589
590     //define ifstream variable for file
591     ifstream file;
592
593     //open file specified by user
594     file.open( filename.c_str() );
595
596     //test to see if file opens successfully
597     while ( !file )
598     {
599         //notify user of error and ask to input file name again
600         cout << "Could not open file: " << filename << "." << endl;
601         cout << "Please re-enter filename (or control-C to exit): ";
602         cin >> filename; //assign user input to variable fileName
603         file.open(filename.c_str()); //try again to open file specified by user
604     }
605
606     //display features
607      cout << endl << right << setw(65) << "********************** X-Ray GRB Data ********"
608           << "***************" << endl << endl;
609      cout << right << setw(10) << "GRB ID" << right << setw(10) << "dt_X [s]"
610           << right << setw(20) << "Exposure Time [s]" << right << setw(10)
611           << "F_x [uJy]" << right << setw(15) << "sigma_X [uJy]" << endl << endl;
612     //read in data from file and assign them to corresponding variables
613     //read until file no longer has any more data
614     while ( file >> ID >> dtX >> ExpX >> Fx >> sigmaX)
615     {
616        //create a GRB g and construct it using variables read from file
617        GRB grb( ID, dtX, ExpX, Fx, sigmaX);
618
619        //initialize new_ID to be the GRB ID read in from file
```

```
620         string new_ID = ID;
621
622         //check if this is the first entry read from optical file
623         if ( counter == 0 )
624         {
625             //initialize old_ID to entry read from optical file if
626             //this is the first entry from optical file
627             old_ID = ID;
628         }
629
630         //check if new ID has been reached
631         if ( new_ID.compare(old_ID) == 0 )
632         {
633             //increment counter for number of entries per unique ID because
634             //a multiplicity of the same idea has been found OR it is the first trial
635             entries_per_ID++;
636         }
637         else
638         {
639             //construct Possibility object with corresponding data
640             Possibility new_Possibility( old_ID, entries_per_ID );
641
642             //add in new element to vector containing number of entires per unique
643             //GRB ID read from optical file
644             XRay_entries.push_back(new_Possibility);
645
646             //reset counter used for determining number of data points for each unique ID
647             //counter is reset to 1 because in order to reach this else clause, new_ID !=
648             //old_ID, meaning there is already a new entry
649             entries_per_ID = 1;
650         }
651
652         old_ID = ID;
653         //increment counter
654         counter++;
655
656        //put the GRB object in the vector of GRBs
657        GRBs.push_back( grb );
658
659        //display loaded features
660        cout << right << setw(10) << ID << right << setw(10) << dtX  << right << setw(20)
661             << ExpX << right << setw(10) <<  Fx << right << setw(15) << sigmaX <<endl;
662    }
663
664    //add last pair to vector of multiplicities
665    //construct Possibility object with corresponding data
666    Possibility new_Possibility( old_ID, entries_per_ID );
667
668    //add in new element to vector containing number of entires per unique
669    //GRB ID read from optical file
670    XRay_entries.push_back(new_Possibility);
671
672    cout << endl << "Number of GRBs loaded: " << counter;
673
```

```
674     //close file
675     file.close();
676 
677    return counter;
678 }
679 
680 int Trial::loadBeta_X(string filename)
681 {
682     //define variable for GRB ID
683     string ID;
684     //define variable for Beta_X
685     double Beta_X;
686     //define variable for Beta_X upper uncertainty
687     double Beta_X_upper_sigma;
688     //define variable for Beta_X lower uncertainty
689     double Beta_X_lower_sigma;
690 
691     //define variable for total loaded Beta X elements
692     int total_loaded = 0;
693     //initialize variable for number of successful pairs after Beta_X
694     double Beta_X_pairs = 0;
695     //initialize variable for rate of successful pairings
696     double pairing_rate = 0;
697 
698     //define ifstream variable for file
699     ifstream file;
700 
701     //open file specified by user
702     file.open( filename.c_str() );
703 
704     //test to see if file opens successfully
705     while ( !file )
706     {
707         //notify user of error and ask to input file name again
708         cout << "Could not open file: " << filename << "." << endl;
709         cout << "Please re-enter filename (or control-C to exit): ";
710         cin >> filename; //assign user input to variable fileName
711         file.open(filename.c_str()); //try again to open file specified by user
712     }
713 
714     //display features
715      cout << endl << right << setw(60) << "**********************Beta_X Data*************"
716           << "*************"
717           << endl << endl;
718      cout << right << setw(10) << "GRB ID" << right << setw(10) << "Beta_X"
719           << right << setw(20) << "Upper Uncertainty" << right << setw(20)
720           << "Lower Uncertainty"<< endl << endl;
721     //read in data from file and assign them to corresponding variables
722     //read until file no longer has any more data
723 
724     while ( file >> ID >> Beta_X >> Beta_X_upper_sigma >> Beta_X_lower_sigma )
725     {
726         //display loaded features
727         cout << right << setw(10) << ID << right << setw(10) << Beta_X
```

```
            << right << setw(20) << Beta_X_upper_sigma << right << setw(20)
            << Beta_X_lower_sigma << endl;

        //initialize success boolean to false
        bool success = false;

        //increment counter for successful load
        total_loaded++;

        // run through vector of GRBs
        for ( int a = 0; a < GRBs.size(); a++ )
        {
            //match GRB IDs
            if ( GRBs[a].getGRB_ID() == ID )
            {
                //assign attributes to appropriate GRB
                GRBs[a].set_Beta_X(Beta_X);
                GRBs[a].set_Beta_X_upper_sigma(Beta_X_upper_sigma);
                GRBs[a].set_Beta_X_lower_sigma(Beta_X_lower_sigma);

                //signify match
                success = true;
                //increment success counter for successful pairing
                Beta_X_pairs++;
            }
        }

        //check if no match was found
        if ( success == false )
        {
            //notify user that no match was able to be found
            cout << endl << "Unable to match GRB ID " << ID << " with Beta_X " << Beta_X
                 << endl << endl;
        }
    }

    //calculate percent of loaded GRBs that are paired
    pairing_rate = (Beta_X_pairs /  GRBs.size())*100;

    //notify user of loading statistics
    cout << endl << right << setw(58) << "Number of loaded GRBs from Beta_X file: "
         << right << setw(2) << total_loaded;
    cout << endl << right << setw(57) << "Number of successful pairings with X-Ray Data: "
         << right << setw(3) << Beta_X_pairs;
    cout << endl << right << setw(55) << fixed << showpoint << setprecision(1)
         << "Pairing Rate: " << right << setw(4) << pairing_rate << right << setw(1)
         << "%" << endl;

    cout << endl << right << setw(60) << "...Cleaning up X-Ray Entries..." << endl;
    clean_XRay_entries();

    //close file
    file.close();

```

```
782     return total_loaded;
783 }
784 
785 int Trial::loadOpticalData( string filename )
786 {
787     //define variables for GRB attributes
788 
789     //define variable for GRB ID Number
790     string ID;
791     //define variable for optical dt in hours
792     double dtO_hours;
793     //define variable for optical dt in seconds
794     double dtO_seconds;
795     //define variable for optical telescope name
796     string tel;
797     //define variable for instrument name
798     string inst;
799     //define variable for optical filter name
800     string fil;
801     //define variable for optical exposure time in seconds
802     double ExpO;
803     //define variable for optical flux density in uJy
804     double Fo;
805     //define variable for optical flux density std dev in uJy
806     double sigmaO;
807 
808     //initialize variable for number of subjects loaded
809     double total_loaded = 0;
810     //initialize variable for rate of successful pairings
811     double pairing_rate = 0;
812     //initialize counter for no match found
813     int no_match_found_counter = 0;
814 
815     //initialize string variable for old ID used in GRB ID multiplicity determination
816     string old_ID = "NULL";
817     //initialize counter used for determining number of data points for each unique ID
818     int entries_per_ID = 0;
819     //initialize variable for number of GRBs that are not in
820     //BOTH the X-Ray and optical files
821     int disjoint = 0;
822     //initialize variable for a check on the number of GRBs that are not in
823     //BOTH the X-Ray and optical files
824     int disjoint_confirm = 0;
825 
826     //define ifstream variable for file
827     ifstream file;
828 
829     //open file specified by user
830     file.open( filename.c_str() );
831 
832     //test to see if file opens successfully
833     while ( !file )
834     {
835         //notify user of error and ask to input file name again
```

```
836         cout << "Could not open file: " << filename << "." << endl;
837         cout << "Please re-enter filename (or control-C to exit): ";
838         cin >> filename; //assign user input to variable fileName
839         file.open(filename.c_str()); //try again to open file specified by user
840     }
841     //display features
842      cout << endl << right << setw(110) << "*****************************************"
843           <<" Optical GRB Data *****************************************" << endl
844           << endl;
845      cout << right << setw(10) << "GRB ID" << right << setw(15) << "dt_O [s]"
846           << right << setw(15) << "Telescope" << right << setw(15)
847           << "Instrument" << right << setw(10) << "Filter" << right << setw(20)
848           << "Exposure Time [s]" << right << setw(10) << "F_o [uJy]" << right
849           << setw(15) << "sigma_O [uJy]" << endl << endl;
850
851     //read in data from file and assign them to corresponding variables
852     //read until file no longer has any more data
853     while (file >> ID >> dtO_hours >> tel >> inst >> fil >> ExpO >> Fo >> sigmaO)
854     {
855         //transform optical dt measurement from hours into seconds
856         dtO_seconds = 3600 * dtO_hours;
857
858         //display loaded features
859         cout << right << setw(10) << ID << right << setw(15) << dtO_seconds
860              << right << setw(15) << tel << right << setw(15)
861              << inst << right << setw(10) << fil << right << setw(20)
862              << ExpO << right << setw(10) << Fo << right
863              << setw(15) << sigmaO << endl;
864
865         //initialize variable for location in GRB vector
866         int location = 0;
867         //initialize counter for checking if any pairings were made at all
868         int check = 0;
869
870         //initialize new_ID to be the GRB ID read in from file
871         string new_ID = ID;
872
873         //check if this is the first entry read from optical file
874         if ( total_loaded == 0 )
875         {
876             //initialize old_ID to entry read from optical file if
877             //this is the first entry from optical file
878             old_ID = ID;
879         }
880
881         //check if new ID has been reached
882         if ( new_ID == old_ID )
883         {
884             //increment counter for number of entries per unique ID because
885             //a multiplicity of the same idea has been found OR it is the first trial
886             entries_per_ID++;
887         }
888         else
889         {
```

```
            //construct Possibility object with corresponding data
            Possibility new_Possibility( old_ID, entries_per_ID );

            //add in new element to vector containing number of entires per unique
            //GRB ID read from optical file
            optical_entries.push_back(new_Possibility);

            //disjoint += check_ID(old_ID);

            //reset counter used for determining number of data points for each unique ID
            //counter is reset to 1 because in order to reach this else clause, new_ID !=
            //old_ID, meaning there is already a new entry
            entries_per_ID = 1;
        }

        old_ID = ID;

        //increment counter for total loaded Optical elements
        total_loaded++;

        while ( location < GRBs.size() && location != -1)
        {
            location = matchGRB(ID, dtO_seconds, location);
            check++;

            //cout << "\nBeginning checks\n";
            //cout << "Location is: " << location << endl;

            //check if pairing is made
            if( location != -1 )
            {
                //construct a GRB object that will be added
                //into vector of GRBs with optical data
                GRB copy_grb( GRBs[location].getGRB_ID(), GRBs[location].get_dt_XRay(),
                              GRBs[location].get_ExpT_XRay(), GRBs[location].get_F_x(),
                              GRBs[location].get_sigma_x());

                //set corresponding GRB appropriate Beta_X parameters
                copy_grb.set_Beta_X(GRBs[location].get_Beta_X());
                copy_grb.set_Beta_X_lower_sigma(GRBs[location].get_Beta_X_lower_sigma());
                copy_grb.set_Beta_X_upper_sigma(GRBs[location].get_Beta_X_upper_sigma());

                //set corresponding GRB appropriate optical parameters
                copy_grb.set_dt_Opt(dtO_seconds);
                copy_grb.set_telescope(tel);
                copy_grb.set_instrument(inst);
                copy_grb.set_filter(fil);
                copy_grb.set_Exp_Opt(ExpO);
                copy_grb.set_F_o(Fo);
                copy_grb.set_sigma_o(sigmaO);

                //add newly-created GRB with optical data to
                //vector of GRBs with optical data
                GRBs_with_Opt.push_back(copy_grb);
```

```
944
945                 //increment location
946                 location ++;
947                 //increment counter of successful pairing
948                 optical_pairs++;
949             }
950
951         }
952         //check if no pairing was made
953         if ( check == 1)
954         {
955 //             //notify user that no match was able to be found
956 //             cout << "Unable to match GRB " << ID << " with optical dt " << dtO_hours
957 //                  << " [hr] = " << dtO_seconds << " [s]." << endl << endl;
958
959            //increment counter for no match made
960            no_match_found_counter++;
961         }
962     }
963
964     //add last pair to vector of multiplicities
965     //construct Possibility object with corresponding data
966     Possibility new_Possibility( old_ID, entries_per_ID );
967
968     //add in new element to vector containing number of entires per unique
969     //GRB ID read from optical file
970     optical_entries.push_back(new_Possibility);
971
972     total_possible_pairings = find_total_possible_pairings();
973
974     //calculate percent of loaded GRBs that are paired
975     pairing_rate = (optical_pairs /  total_possible_pairings )*100;
976
977     //notify user of loading statistics
978     cout << endl << right << setw(57) << "Number of loaded GRBs from optical data file: "
979          << right << setw(3) << int(total_loaded);
980     cout << endl << right << setw(57) << fixed << showpoint << setprecision(0)
981          << "Number of possible pairs: " << right << setw(3)
982          << int(total_possible_pairings);
983     cout << endl << right << setw(57) << fixed << showpoint << setprecision(0)
984          << "Number of successful pairs: " << right << setw(3) << int(optical_pairs);
985     cout << endl << right << setw(55) << fixed << showpoint << setprecision(1)
986          << "Pairing Rate: " << right << setw(5) << pairing_rate << "%";
987
988     //close file
989     file.close();
990
991     return total_loaded;
992 }
993
994 int Trial::loadWavelengthData(string filename)
995 {
996     //define variables for GRB attributes
997
```

```
    //define variable for telescope name
    string telName;
    //define variable for instrument name
    string instrumentName;
    //define variable for filter name
    string filterName;
    //define variable for wavelength in [nm]
    double wavelength;
    //define variable for frequency in [Hz]
    double frequency;

    //initialize variable for number of subjects loaded
    int loaded = 0;
    //initialize variable for number of successful pairings loaded
    int success_counter = 0;
     //initialize variable for the counter to check pairing
    int check_counter = 0;

    //define ifstream variable for file
    ifstream file;

    //open file specified by user
    file.open( filename.c_str() );

     //test to see if file opens successfully
    while ( !file )
    {
        //notify user of error and ask to input file name again
        cout << "Could not open file: " << filename << "." << endl;
        cout << "Please re-enter filename (or control-C to exit): ";
        cin >> filename; //assign user input to variable fileName
        file.open(filename.c_str()); //try again to open file specified by user
    }

    //display features
     cout << endl << right << setw(80) << "****************************** Wavelength Data"
          <<" ******************************"
          << endl << endl;
     cout << right << setw(20) << "Telescope" << right << setw(15) << "Instrument"
          << right << setw(10) << "Filter" << right << setw(15)
          << "Wavelength" << right << setw(20) << "Frequency" << endl << endl;

    //read in data from file and assign them to corresponding variables
    //read until file no longer has any more data
    while ( file >> telName >> instrumentName >> filterName >> wavelength >> frequency )
    {
        //display loaded features
        cout << right << setw(20) << telName << right << setw(15) << instrumentName
          << right << setw(10) << filterName << right << setw(15)
          << wavelength << right << setw(20) << frequency << endl;

        //call function to match GRB objects with appropriate wavelength data
        success_counter += matchFrequency( telName, instrumentName, filterName, wavelength,
                                           frequency );
```

```
1052
1053            //increment counter for successful load
1054            loaded++;
1055
1056            //notify user that a match was able to be found
1057            //cout << "Able to match telescope " << telName << " with instrument " <<
1058            //instrumentName <<  " and with filter " << filterName << "." << endl;
1059        }
1060
1061        cout << endl;
1062        //display those GRBs that couldn't get paired
1063        for ( int a = 0; a < GRBs_with_Opt.size(); a++ )
1064        {
1065            //check to see if GRB has been paired with Beta_X, optical, but not wavelength data
1066            if (  GRBs_with_Opt[a].get_frequency_Opt() == -1 )
1067            {
1068                cout << "\nGRB " << GRBs_with_Opt[a].getGRB_ID() << " with optical dt "
1069                     << GRBs_with_Opt[a].get_dt_Opt() << ", telescope "
1070                     << GRBs_with_Opt[a].get_telescope() << ", instrument "
1071                     << GRBs_with_Opt[a].get_instrument() <<  ", and with filter "
1072                     << GRBs_with_Opt[a].get_filter() <<" unpaired." << endl;
1073                check_counter++;
1074            }
1075        }
1076
1077        //display loaded statistics
1078        cout << endl << endl << right << setw(57) << "Number of Wavelength Sets Loaded: "
1079             << right << setw(3) << loaded << endl;
1080        cout << right << setw(57) << "Number of Unsuccessfully Paired: " << right << setw(3)
1081             << check_counter << endl;
1082        cout << right << setw(57) << "Number of Successfully Paired: " << right << setw(3)
1083             << success_counter << endl;
1084        cout << right << setw(55) << "Overall Success Rate: " << right << setw(4) << fixed
1085             << showpoint << setprecision(1)
1086             << 100 * (success_counter / total_possible_pairings ) << right << setw(1) << "%"
1087             << endl;
1088
1089        //close file
1090        file.close();
1091
1092        return loaded;
1093    }
1094
1095    void Trial::calculate_Beta_OX()
1096    {
1097        //initialize variable for counter of successful Beta_OX calculations
1098        int success_counter = 0;
1099        //initialize counter for nan calculations of Beta_OX
1100        int nan = 0;
1101
1102        //initialize variables necessary for calculation
1103        double F_x = 0;
1104        double F_o = 0;
1105        double sigma_x = 0;
```

```
    double sigma_o = 0;
    double frequency_X = FREQUENCY_XRAY;
    double frequency_O = 0;
    double Beta_OX = 0;
    double sigma_OX_upper = 0;
    double sigma_OX_lower = 0;

    //display features
     cout << endl << right << setw(160) << "*******************************************"
          <<"****************************** Beta_OX Data ****************************"
          <<"************************************************"
          << endl << endl;
     cout << right << setw(10) << "GRB ID" << right << setw(15)
          << "F_x [uJy]" << right << setw(15) << "sigma_X [uJy]" << right << setw(15)
          << "F_o [uJy]" << right << setw(15) << "sigma_o [uJy]" << right << setw(25)
          << "Freq_X" << right  << setw(25) << "Freq_O" << right << setw(10)
          << "Beta_OX" << right << setw(20) << "Upper sigma_OX" << right
          << setw(20) << "Lower sigma_OX" << endl;

    // run through vector of GRBs
    for ( int a = 0; a < GRBs_with_Opt.size(); a++ )
    {
        //check to see if GRB has been fully paired
        if (GRBs_with_Opt[a].get_frequency_Opt() != -1)
        {

            // Extract values from GRB objects in vector of GRBs
            F_x = GRBs_with_Opt[a].get_F_x();
            F_o = GRBs_with_Opt[a].get_F_o();
            frequency_O = GRBs_with_Opt[a].get_frequency_Opt();
            sigma_x = GRBs_with_Opt[a].get_sigma_x();
            sigma_o = GRBs_with_Opt[a].get_sigma_o();

            //calculate Beta_OX
            Beta_OX = log(F_x / F_o) / log(frequency_X / frequency_O );

            //check to see if calculation returns nan
            if (Beta_OX != Beta_OX )
            {
                Beta_OX = 0;
                nan++;
            }

            //calculate upper bound on uncertainty for Beta_OX
            sigma_OX_upper = log( (1 + (sigma_x / F_x)) / (1 - (sigma_o / F_o))) /
            log( frequency_X / frequency_O );
            //calculate lower bound on uncertainty for Beta_OX
            sigma_OX_lower = abs( log( (1 - (sigma_x / F_x)) / (1 + (sigma_o / F_o))) /
                                  log( frequency_X / frequency_O ));

            //check to see if calculation returns nan
            if (sigma_OX_upper != sigma_OX_upper)
            {
                sigma_OX_upper = 0;
```

```
            }
            //check to see if calculation returns nan
            if (sigma_OX_lower != sigma_OX_lower )
            {
                sigma_OX_lower = 0;
            }
            //set calculated value into GRB Beta_OX attribute
            GRBs_with_Opt[a].set_Beta_OX(Beta_OX);
            GRBs_with_Opt[a].set_sigma_OX_upper(sigma_OX_upper);
            GRBs_with_Opt[a].set_sigma_OX_lower(sigma_OX_lower);

            cout << right << setw(10) << GRBs_with_Opt[a].getGRB_ID() << right << setw(15)
                  << GRBs_with_Opt[a].get_F_x() << right << setw(15)
                  << GRBs_with_Opt[a].get_sigma_x() << right << setw(15)
                  << GRBs_with_Opt[a].get_F_o() << right << setw(15)
                  << GRBs_with_Opt[a].get_sigma_o() << right << setw(25)
                  << GRBs_with_Opt[a].get_frequency_XRay() << right  << setw(25)
                  << GRBs_with_Opt[a].get_frequency_Opt() << right << setw(10)
                  << GRBs_with_Opt[a].get_Beta_OX() << right << setw(20)
                  << GRBs_with_Opt[a].get_sigma_OX_upper() << right << setw(20)
                  << GRBs_with_Opt[a].get_sigma_OX_lower() << endl;

            //increment success counter for successful calculation
            success_counter++;
        }
    }

    //display loaded statistics
    cout << endl << endl << right << setw(57) << "Number of Successful Beta_OX"
         <<"Calculations: " << right << setw(3) << success_counter << endl;
    cout << right << setw(55) << "Overall Success Rate: " << right << setw(4) << fixed
         << showpoint << setprecision(1) << 100 * (success_counter /
         total_possible_pairings ) << right << setw(1) << "%" << endl;
}

void Trial::report()
{
    //run through vector of GRB
    for ( int a = 0; a < GRBs.size(); a++)
    {
        GRBs[a].report();
    }
}

void Trial::write_paired_data()
{
    //cast temporal percent difference to string
    int dt = static_cast<int>(DT_PERCENT_DIF);
    string percent_dif = to_string(dt);
    //define variable for name of files
    string filename_comprehensive = "./Written_Files/Comprehensive_Paired_Data_Table_"
    + percent_dif + "%.csv";
    string filename_terse = "./Written_Files/GRB_Pairings-dt_" + percent_dif + "%.csv";

```

```
    //define ofstream variables for the files to be written
    ofstream myFile_comprehensive;
    ofstream myFile_terse;

    //open file for comprehensive data file
    myFile_comprehensive.open(filename_comprehensive);

    //print out headers for file
    myFile_comprehensive << "GRB ID,X-Ray dt [hr],X-Ray Exposure Time [s],"
           << "F_x [uJy],Sigma_x [uJy],Beta_X,"
           << "Beta_X Upper Sigma,Beta_X Lower Sigma,"
           << "Optical dt [hr],Telescope,Instrument,Filter,"
           << "Optical Exposure Time [s],F_o [uJy], Sigma_o [uJy],"
           << "Wavelength_X [nm],Frequency_X [Hz],Wavelength_o [nm],"
           << "Frequency_o [Hz],Beta_OX,Upper Bound of Sigma_OX,"
           << "Lower Bound of Sigma_OX,\n";

    //run through vector of populated GRBs
    for ( int a = 0; a < GRBs_with_Opt.size(); a++)
    {
        //double check to see if GRB is fully populated
        if ( GRBs_with_Opt[a].get_frequency_Opt() != -1)
        {
        myFile_comprehensive << fixed << setprecision(2) << GRBs_with_Opt[a].getGRB_ID()
                             << "," << fixed << setprecision(2)
                             <<  GRBs_with_Opt[a].get_dt_XRay()/3600 << "," << fixed
                             << setprecision(2) << GRBs_with_Opt[a].get_ExpT_XRay() << ","
                             << fixed << setprecision(2) << GRBs_with_Opt[a].get_F_x()
                             << "," << fixed << setprecision(2)
                             << GRBs_with_Opt[a].get_sigma_x() << "," << fixed
                             << setprecision(2) << GRBs_with_Opt[a].get_Beta_X()
                             << "," << fixed << setprecision(2)
                             << GRBs_with_Opt[a].get_Beta_X_upper_sigma() << "," << fixed
                             << setprecision(2) << GRBs_with_Opt[a].get_Beta_X_lower_sigma()
                             << "," << fixed << setprecision(2)
                             << GRBs_with_Opt[a].get_dt_Opt()/3600 << "," << fixed
                             << setprecision(2) << GRBs_with_Opt[a].get_telescope()
                             << "," << fixed << setprecision(2)
                             << GRBs_with_Opt[a].get_instrument() << "," << fixed
                             << setprecision(2) << GRBs_with_Opt[a].get_filter()
                             << "," << fixed << setprecision(2)
                             << GRBs_with_Opt[a].get_Exp_Opt() << "," << fixed
                             << setprecision(2) << GRBs_with_Opt[a].get_F_o()
                             << "," << fixed << setprecision(2)
                             << GRBs_with_Opt[a].get_sigma_o() << "," << fixed
                             << GRBs_with_Opt[a].get_frequency_XRay() << "," << fixed
                             << setprecision(2)
                             << GRBs_with_Opt[a].get_frequency_Opt() << "," << fixed
                             << setprecision(2) << GRBs_with_Opt[a].get_Beta_OX() << ","
                             << fixed << setprecision(2)
                             << GRBs_with_Opt[a].get_sigma_OX_upper() << "," << fixed
                             << setprecision(2) << GRBs_with_Opt[a].get_sigma_OX_lower()
                             << "\n";
        }
```

```
    }
    //close file
    myFile_comprehensive.close();

    //open file for comprehensive data file
    myFile_terse.open(filename_terse);

    //print out headers for file
    myFile_terse << "GRB ID,X-Ray dt [hr],"
           << "Optical dt [hr],"
           << "|dt_x - dt_o| [hr],"
           << "Beta_X,"
           << "Beta_X Upper Sigma,Beta_X Lower Sigma,"
           << "Beta_OX,Upper Bound of Sigma_OX,"
           << "Lower Bound of Sigma_OX,\n";

    for ( int a = 0; a < GRBs_with_Opt.size(); a++)
    {
        //check to see if GRB is fully populated
        if ( GRBs_with_Opt[a].get_frequency_Opt() != -1)
        {
        myFile_terse << GRBs_with_Opt[a].getGRB_ID() << "," << fixed << setprecision(2)
                     << GRBs_with_Opt[a].get_dt_XRay()/3600
               << "," << fixed << setprecision(2) << GRBs_with_Opt[a].get_dt_Opt()/3600
               << "," << fixed << setprecision(2)
               << abs(GRBs_with_Opt[a].get_dt_Opt() - GRBs_with_Opt[a].get_dt_XRay())/3600
               << "," << fixed << setprecision(2) << GRBs_with_Opt[a].get_Beta_X()
               << "," << fixed << setprecision(2)
               << GRBs_with_Opt[a].get_Beta_X_upper_sigma() << "," << fixed
               << setprecision(2) << GRBs_with_Opt[a].get_Beta_X_lower_sigma()
               << "," << fixed << setprecision(2) << GRBs_with_Opt[a].get_Beta_OX() << ","
               << fixed << setprecision(2) << GRBs_with_Opt[a].get_sigma_OX_upper()
               << "," << fixed << setprecision(2) << GRBs_with_Opt[a].get_sigma_OX_lower()
               << "\n";
        }
    }
    //close file
    myFile_terse.close();

}

int Trial::matchFrequency( string tel, string inst, string filt, double wavelength,
                           double frequency )
{
    //initialize counter for successful pairing
    int thatsapair = 0;
    //initialize boolean for successful pairing
    bool success = false;

    //run through vector of GRB
    for ( int a = 0; a < GRBs_with_Opt.size(); a++)
    {
        //test to see if GRB ID matches passed ID
        //also test to see if GRB has not yet been populated with wavelength parameters
```

```
        if ( GRBs_with_Opt[a].get_telescope() == tel && GRBs_with_Opt[a].get_instrument()
            == inst && GRBs_with_Opt[a].get_filter() == filt &&
             GRBs_with_Opt[a].get_frequency_Opt() == -1)
        {
            //pair GRB with frequency data
            GRBs_with_Opt[a].set_frequency_Opt(frequency);

            //increment success counter
            thatsapair++;
            //assign boolean to true to signify successful pairing
            success = true;
        }
    }

    //return number of successful pairs
    return thatsapair;

}

int Trial::check_ID( string ID )
{
    //define boolean for match
    bool corresponding_ID_found = false;
    //initialize counter for number of disjoint GRBs
    int disjoint_counter = 0;

    cout << "\n\nIn check_ID!" << endl << endl;

    for ( int kumquat = 0; kumquat < XRay_entries.size(); kumquat++ )
    {
        //check if passed optical IDs are identical to those in GRBs that have
        //already been paired with Beta_X data
        if ( ID.compare( XRay_entries[kumquat].get_ID() ) == 0 )
        {
            //set boolean to true if match found
            corresponding_ID_found = true;
        }
    }
    //check if no match was found
    if ( corresponding_ID_found == false )
    {
        //add ID that is in optical data set but not X-Ray to
        //vector containing all such IDs
        IDs_in_Opt_not_X.push_back(ID);
        //increment counter for lack of match
        disjoint_counter++;
        cout << "\n\nID in Optical but not in X-Ray :(" << endl << endl;
    }

    return disjoint_counter;

}

int Trial::clean_XRay_entries()
```

```
{
    //initialize counter for number we expect to keep
    int should_not_keep = 0;

    cout << endl << right << setw(58) << "Original Number of Unique X-Ray GRBs: "
         << right << setw(2) << XRay_entries.size();

    //traverse through vector XRay_entries
    for (int q = 0; q < XRay_entries.size(); q++ )
    {
        //initialize boolean signifying that XRay_entries[q] should be
        //kept to false
        bool keeper = false;

        //traverse through vector GRBs that contain only some with BetaX data
        for (int n = 0; n < GRBs.size(); n++ )
        {
            //check if the ID in XRay_entries and GRBs are identical and that
            //ID has been paired with BetaX data
            if ( XRay_entries[q].get_ID().compare(GRBs[n].getGRB_ID()) == 0 &&
                 GRBs[n].get_Beta_X() != 31415926535 )
            {
                //assign boolean for keeper to true
                keeper = true;
            }
        }
        //check if no match was made
        if ( keeper == false )
        {
            //increment counter for number we should not keep
            should_not_keep++;

            //initialize iterator to beginning of XRay_entries vector
            auto it = XRay_entries.begin();
            //traverse through XRay_entries with iterator
            while (it != XRay_entries.end())
            {
                // remove entity that had no pairing
                if ((*it).get_ID() == XRay_entries[q].get_ID() )
                {
                    // erase() invalidates the iterator, use returned iterator
                    it = XRay_entries.erase(it);

                    //check if index is at 0
                    if ( q > 0 )
                    {
                        //shift index backwards one due to erased entity
                        q -= 1;
                    }
                    else
                    {
                        //increment iterator
                        //without this the iterator would remain stuck on
                        //index=0 and continuously delete entities
```

```
                        ++it;
                    }
                }
                else
                {
                    //increment iterator
                    ++it;
                }
            }
        }
    }

    cout << endl << right << setw(58) << "Final Number of Unique X-Ray GRBs: " << right
         << setw(2) << XRay_entries.size() << endl;
    cout << right << setw(58) << "Number Removed for Lack of Beta_X Pairing: " << right
         << setw(2) << should_not_keep << endl;
}

int Trial::find_total_possible_pairings()
{
    //initialize counters for disjointed IDs
    int in_Opt_not_XRay = 0;
    int in_XRay_not_Opt = 0;

    //initialize variables for old and new vector sizes
    int old_XRay_entries_size = XRay_entries.size();
    int new_XRay_entries_size = 0;
    int old_optical_entries_size = optical_entries.size();
    int new_optical_entries_size = 0;

    //initialize variable for total number of possibilities
    int total_possibilities = 0;

    //traverse through vector XRay_entries
    for (int q = 0; q < XRay_entries.size(); q++ )
    {
        //initialize boolean signifying that ID is in XRay_entries
        //and also in optical_entries
        bool keeper = false;

        //traverse through optical_entries vector
        for (int n = 0; n < optical_entries.size(); n++ )
        {
            //check if the ID in XRay_entries and optical_entries are identical
            if ( XRay_entries[q].get_ID().compare(optical_entries[n].get_ID()) == 0 )
            {
                //assign boolean for keeper to true
                keeper = true;
            }
        }
        //check if no match was made
        if ( keeper == false )
        {
            //increment counter for number of IDs that exist in
```

```
            //XRay_entries but not in optical_entries
            in_XRay_not_Opt++;

            //initialize iterator to beginning of XRay_entries vector
            auto it = XRay_entries.begin();
            //traverse through XRay_entries with iterator
            while (it != XRay_entries.end())
            {
                // remove entity that had no pairing
                if ((*it).get_ID() == XRay_entries[q].get_ID() )
                {
                    // erase() invalidates the iterator, use returned iterator
                    it = XRay_entries.erase(it);

                    //check if index is at 0
                    if ( q > 0 )
                    {
                        //shift index backwards one due to erased entity
                        q -= 1;
                    }
                    else
                    {
                        //increment iterator
                        //without this the iterator would remain stuck on
                        //index=0 and continuously delete entities
                        ++it;
                    }
                }
                else
                {
                    //increment iteratorDecember 15
                    ++it;
                }
            }
        }
    }
    //appropriately change variable for size of XRay_entries
    new_XRay_entries_size = XRay_entries.size();

    //traverse through vector optical_entries
    for (int l = 0; l < optical_entries.size(); l++ )
    {
        //initialize boolean signifying that ID is in XRay_entries
        //and also in optical_entries
        bool keepme = false;

        //traverse through optical_entries vector
        for (int u = 0; u < XRay_entries.size(); u++ )
        {
            //check if the ID in XRay_entries and optical_entries are identical
            if ( optical_entries[l].get_ID().compare(XRay_entries[u].get_ID()) == 0 )
            {
                //assign boolean for keeper to true
                keepme = true;
```

```
            }
        }
        //check if no match was made
        if ( keepme == false )
        {
            //increment counter for number of IDs that exist in
            //XRay_entries but not in optical_entries
            in_Opt_not_XRay++;

            //initialize iterator to beginning of XRay_entries vector
            auto it_2 = optical_entries.begin();
            //traverse through XRay_entries with iterator
            while (it_2 != optical_entries.end())
            {
                // remove entity that had no pairing
                if ((*it_2).get_ID() == optical_entries[l].get_ID() )
                {
                    // erase() invalidates the iterator, use returned iterator
                    it_2 = optical_entries.erase(it_2);

                    //check if index is at 0
                    if ( l > 0 )
                    {
                        //shift index backwards one due to erased entity
                        l -= 1;
                    }
                    else
                    {
                        //increment iterator
                        //without this the iterator would remain stuck on
                        //index=0 and continuously delete entities
                        ++it_2;
                    }
                }
                else
                {
                    //increment iterator
                    ++it_2;
                }

            }
        }
    }

    //appropriately change variable for new size of optical_entries
    new_optical_entries_size = optical_entries.size();

    cout << endl << endl << right << setw(60) << "Final X-Ray Entries (left) and Optical"
         << "Entries ( right): " << endl;
    cout << endl << right << setw(10) << "GRB ID" << right << setw(15) << "Multiplicity"
         << right << setw(25) << "GRB ID" << right << setw(15) << "Multiplicity" << endl
         << endl;

    for ( int y = 0; y < optical_entries.size(); y++ )
```

```
1592     {
1593        cout << right << setw(10) << XRay_entries[y].get_ID() << right << setw(15)
1594             << XRay_entries[y].get_multiplicity() << right << setw(25)
1595             << optical_entries[y].get_ID() << right << setw(15)
1596             << optical_entries[y].get_multiplicity() << endl;
1597     }
1598 
1599     cout << endl;
1600 
1601     //display appropriate statistics
1602     cout << endl << right << setw(58) << "Original Number of Unique X-Ray GRBs with Beta_X"
1603          <<" Data: " << right << setw(2) << old_XRay_entries_size << endl;
1604     cout << right << setw(58) << "Number of Disjoint GRBs Removed from X-Rays: " << right
1605          << setw(2) << in_XRay_not_Opt << endl;
1606     cout << right << setw(58) << "Final Number of Unique X-Ray GRBs: " << right << setw(2)
1607          << new_XRay_entries_size << endl;
1608     cout << endl << right << setw(58) << "Original Number of Unique Optical Entries: "
1609          << right << setw(2) << old_optical_entries_size << endl;
1610     cout << right << setw(58) << "Number of Disjoint GRBs Removed from Optical: " << right
1611          << setw(2) << in_Opt_not_XRay << endl;
1612     cout << right << setw(58) << "Final Number of Unique Optical GRBs: " << right
1613          << setw(2) << new_optical_entries_size << endl;
1614 
1615     //calculate the total number of possibilities that
1616     //could be obtained from perfect matching
1617     for ( int g = 0; g < optical_entries.size(); g++ )
1618     {
1619         //multiply number of options in XRay_entries by
1620         //number of options in optical_entries per unique
1621         //GRB ID to determine total number of possibilities
1622         total_possibilities += XRay_entries[g].get_multiplicity() *
1623         optical_entries[g].get_multiplicity();
1624     }
1625     //return number of erased
1626     return total_possibilities;
1627 }
1628 
1629 int Trial::matchGRB( string ID, double dtO_s, int location )
1630 {
1631     //initialize bool variable to determine if match occurs
1632     bool paired = false;
1633 
1634     //run through vector of GRB
1635     for ( int a = location; a < GRBs.size(); a++)
1636     {
1637         //test to see if GRB ID matches passed ID
1638         //test to see if GRB has been populated with Beta_X data
1639         if ( GRBs[a].getGRB_ID() == ID && GRBs[a].get_Beta_X() != 31415926535 )
1640         {
1641             //test to see if temporal separation is small enough and GRB hasn't already
1642             //been populated with optical data
1643             if ( ( 100 * abs( GRBs[a].get_dt_XRay() - dtO_s ) / dtO_s ) < DT_PERCENT_DIF )
1644             {
1645                 //return location of this GRB
```

```
                return a;
                //assign match to true to signify successful match
                paired = true;
            }
        }
    }
    //test to see if no match occurred
    if ( paired == false )
    {
        //return -1 if no match occurred
        return -1;
    }
}

int Trial::findGRB(string ID)
{
    //initialize bool variable to determine if match occurs
    bool match = false;

    //run through vector of GRBs
    for ( int a = 0; a < GRBs.size(); a++)
    {
        //test to see if Subject ID matches passed ID
        if ( GRBs[a].getGRB_ID() == ID )
        {
            //return location of this GRB
            return a;
            //assign match to true to signify successful match
            match = true;
        }
    }
    //test to see if no match occurred
    if ( match == false )
    {
        //return -1 if no match occurred
        return -1;
    }
}

/*****************************************************************
                            BEGIN MAIN
*****************************************************************/

int main()
{
    //define variable for string type of desired temporal % difference
    string percent_dif;
    //define variable for file name of X-Ray data file
    string XRayDataFile_name;
    //define variable for file name of Beta_X data file
    string Beta_X_File_name;
    //define variable for file name of optical data file
    string OpticalData_name;
     //define variable for file name of wavelength data file
```

```
    string WavelengthData_name;
    //create a trial object
    Trial t1;


     //ask user for name of file they wish to load for X-Ray data
    cout << "Please enter the name of the X-Ray data file: ";
    //assign user input to variable for file name of subject data
    cin >> XRayDataFile_name;

    //call function for loading X-Ray data
    //pass name of file for X-Ray data
    t1.loadX_RayData(XRayDataFile_name);

    //formatting
    cout << endl << endl;

    //ask user for name of file they wish to load for Beta_X data
    cout << "Please enter the name of the Beta_X data file: ";
    //assign user input to variable for file name of subject data
    cin >> Beta_X_File_name;

    //call function for loading X-Ray data
    //pass name of file for X-Ray data
    t1.loadBeta_X(Beta_X_File_name);

    //formatting
    cout << endl << endl;

    //ask user for desired temporal percent difference
    cout << "Please enter the desired temporal percent difference [%]: ";
    //assign user input to constant for temporal percent difference
    cin >> DT_PERCENT_DIF;

    //ask user for name of file they wish to load for side effects
    cout << "Please enter the name of the optical data file: ";
    //assign user input to variable for file name for side effects
    cin >> OpticalData_name;

    //call function for loading optical data
    //pass name of file for optical data
    t1.loadOpticalData(OpticalData_name);

    //formatting
    cout << endl << endl;

    //ask user for name of file they wish to load for side effects
    cout << "Please enter the name of the wavelength data file: ";
    //assign user input to variable for file name for side effects
    cin >> WavelengthData_name;

    //call function for loading wavelength data
    //pass name of file for wavelength data
    t1.loadWavelengthData(WavelengthData_name);
```

```
1754
1755
1756        cout << endl; //for formatting purposes
1757        //set numeric output formatting
1758        cout << fixed << showpoint << setprecision(2);
1759
1760        cout << endl << endl << "Press any key to calculate Beta_OX." << endl << endl;
1761        cin.ignore();
1762        cin.get();
1763
1764        //calculate X-Ray to optical spectral indices
1765        t1.calculate_Beta_OX();
1766
1767        //write paired data to .csv file
1768        t1.write_paired_data();
1769
1770        return 0;
1771    }
```

Listing 1: C++ code for Automating the Calculation of β_{ox}.

8.3 Python Source Code

```
1
2   """
3    Title: "Automating the Graphing of Beta_OX vs. Beta_X"
4
5    Copyright (C) 2020 David Fitzpatrick
6
7    From: "Analyzing Optically-Dark Short Gamma Ray Bursts"
8    (1) David Fitzpatrick, (2) Professor Alexander van der Horst, Ph.D.
9
10   1. Georgetown University, Department of Physics, 37 and O Streets NW, Washington D.C. 20057
11   2. The George Washington University, Department of Physics, 725 21 Street NW, Washington
12      D.C. 20052
13
14    I hereby grant to Georgetown University and its agents the non-exclusive, worldwide right
15    to reproduce, distribute, display and transmit my thesis in such tangible and electronic
16    formats as may be in existence now or developed in the future. I retain all ownership
17    rights to the copyright of the thesis including the right to use it in whole or in part
18    in future works. I agree to allow the Georgetown University Department of Physics to
19    serve as the institutional repository of my thesis and to make it available to the
20    Georgetown University community through its website. I certify that the version that I
21    have submitted is the same version that was approved by my senior research advisor.
22
23   Description: Following the calculation of Beta_OX from the C++ program titled
24   "Automating the Calculation of Beta_OX", this program reads in a file containing
25   GRB IDs with paired X-ray and optical data pairs and ensuing parameters (including
26   Beta_OX) and creates graphs of Beta_OX vs. Beta_X in order to identify optically
27   dark GRBs by either the Jakobsson method (Jakobsson et al. 2004) or the Van der
28   Horst method (Van der Horst et al. 2009).  The program does this by providing the
29   user with a menu of options from which to choose.
30
31  """
```

```
32
33 import numpy as np
34 import os
35 import math
36 from matplotlib import pyplot as plt
37 from matplotlib import rc
38 import matplotlib
39 # for error bar caps
40 matplotlib.rcParams.update({'errorbar.capsize': 2})
41 from scipy import integrate
42 # for plotting commands
43 from pylab import *
44 # for writing to files
45 import csv
46
47 # create a class of GRBs
48 class GRB:
49     # define constructor for GRB objects
50     def __init__(self, ID, dtX, dtO, del_t, BetaX, upper_sigmaX, lower_sigmaX, BetaOX,
51                  upper_sigmaOX, lower_sigmaOX, D_Jakobsson, D_vanderHorst):
52         # define attribute for GRB ID
53         self.ID = ID
54         # define attribute for X-Ray temporal extent
55         self.dtX = dtX
56         # define attribute for optical temporal extent
57         self.dtO = dtO
58         # define attribute for X-Ray and optical temporal separation
59         self.del_t = del_t
60         # define attribute for X-Ray spectral index
61         self.BetaX = BetaX
62         # define attribute for upper bound of standard deviation of optical-to-X-Ray
63         # spectral index
64         self.upper_sigmaX = upper_sigmaX
65         # define attribute for lower bound of standard deviation of optical-to-X-Ray
66         # spectral index
67         self.lower_sigmaX = lower_sigmaX
68         # define attribute for optical-to-X-Ray spectral index
69         self.BetaOX = BetaOX
70         # define attribute for upper bound of standard deviation of optical-to-X-Ray
71         # spectral index
72         self.upper_sigmaOX = upper_sigmaOX
73         # define attribute for lower bound of standard deviation of optical-to-X-Ray
74         # spectral index
75         self.lower_sigmaOX = lower_sigmaOX
76         # define attribute for darkness distance according to Jakobsson criteria
77         self.D_Jakobsson = D_Jakobsson
78         # define attribute for darkness distance according to van der Horst criteria
79         self.D_vanderHorst = D_vanderHorst
80
81     def set_D_Jakobsson(self, D):
82         # set attribute for darkness distance according to Jakobsson criteria
83         self.D_Jakobsson = D
84     def set_D_vanderHorst(self,D):
85         # set attribute for darkness distance according to van der Horst criteria
```

```
        self.D_vanderHorst = D

# A function which loads in a file containing Terse Beta_OX data for GRBs and
# assigns them to corresponding attributes of a GRB object, all of which are then
# loaded into a list from which a graphing function can pull desired data
def load_file(filename):
    # import library for dealing with files and file paths
    from pathlib import Path

    # define variable for path of stored paired data files
    filepath = Path("C:/Users/gored/OneDrive/Documents/Georgetown/Senior Year/"
                    "GDub Research/GRB Research/Written_Files/")

    # check to see if file can be opened in reading mode successfully
    while True:
        try:
            # pair filename with file path
            file_to_open = filepath / filename
            # open file
            f = open(file_to_open)
            break
        # check if there is an error opening file
        except FileNotFoundError:
            # notify user of unsuccessful file opening and ask for re-input of desired name
            # re-assign user input to variable for file name
            filename = input("Unable to open desired file.  "
                             "Please re-enter the desired file name: ")

    # initialize list to store lines from file
    content_list = []

    # isolate each line in file
    for line in f:
        # remove newline character from end of file line
        stripped_line = line.rstrip('\n')
        # append line from file to list containing lines of files
        content_list.append(stripped_line)

    # initialize counter for number of lines read from file
    number_of_lines = 0
    # initialize counter for number of GRBs in GRB_list
    GRB_counter = 0
    # initialize list for GRB objects
    GRB_list = []

    # run through list containing lines read from file
    for x in content_list:
        # check that we are not at the first line containing column titles
        if number_of_lines > 0:
            # print line from file in content_list
            # print(x)
            # split csv and assign to corresponding variables
            ID, dtX, dtO, dt, BetaX, SigmaX_u, SigmaX_l, BetaOX, SigmaOX_u, \
            SigmaOX_l = x.split(',')
```

```
            # create GRB object from file contents
            grb = GRB(ID, dtX, dt0, dt, BetaX, SigmaX_u, SigmaX_l, BetaOX, SigmaOX_u,
                      SigmaOX_l, "", "")
            # append GRB object to list of GRB objects
            GRB_list.append(grb)
        # increment counter for total number of lines in file
        number_of_lines += 1

    # run through list containing GRB objects
    for l in GRB_list:
        # print out GRB object attributes for each GRB object in list of GRB objects
        print(l.ID, l.dtX, l.dt0, l.del_t, l.BetaX, l.upper_sigmaX, l.lower_sigmaX,
              l.BetaOX, l.upper_sigmaOX, l.lower_sigmaOX)
        # increment counter for total number of GRB objects in list of GRB objects
        GRB_counter += 1

    print()
    # notify user of total number of lines read from file
    # subtract one due to initial column heading row of .csv file
    print("Total number of lines (or number of GRBs) read from file: ", (number_of_lines-1))
    # notify user of total number of GRBs in list of GRBs
    print("Total number of GRBs in list: ", (GRB_counter))
    print()

    # close file
    f.close()

    # return list of GRB objects
    return GRB_list

# a function to create graphs of Beta_OX vs. Beta_X
# @param: list of GRB objects, name of read-in file
def graph(GRB_list, parsed_filename, graph_title, y_or_n_delB, image_name):

    # remove file extension from filename
    # parsed_filename = os.path.splitext(filename)[0]

    # initialize lists used to graph
    Beta_X_list = []
    Beta_OX_list = []
    Beta_X_upper_list = []
    Beta_X_lower_list = []
    Beta_OX_upper_list = []
    Beta_OX_lower_list = []

    # initialize check counter
    count = 0

    # run through list containing GRB objects
    for z in GRB_list:
        if ( float(z.BetaOX) != 0 ):
            # append spectral index parameters to appropriate lists
            # convert from read-in string to float for data manipulation
            Beta_X_list.append(-1 * float(z.BetaX))
```

```
            Beta_X_upper_list.append(float(z.upper_sigmaX))
            Beta_X_lower_list.append(float(z.lower_sigmaX))
            Beta_OX_list.append(-1 * float(z.BetaOX))
            Beta_OX_upper_list.append(float(z.upper_sigmaOX) + delta_beta_ox_t)
            Beta_OX_lower_list.append(float(z.lower_sigmaOX) + delta_beta_ox_t)
        #increment counter
        count+=1
    #print("Check counter: ", (count))

    # create figure
    fig = plt.figure()
    fig.set_size_inches(11,18)
    # formulate axes
    ax1 = fig.add_axes((0.1, 0.4, 0.8, 0.5))

    # set title
    if(y_or_n_delB == 'Y'):
        title = ax1.set_title(graph_title + parsed_filename +
                              " + \u0394\u03B2\u2092\u2093", fontsize=26)
    else:
        title = ax1.set_title(graph_title + parsed_filename, fontsize=26)

    title.set_position([0.5, 1.05])
    # set x-axis
    ax1.set_xlabel('\u03B2\u2093', fontsize=20)
    ax1.set_ylabel('\u03B2\u2092\u2093', fontsize=20)
    plt.xticks(fontsize=16)
    plt.yticks(fontsize=16)


    # create scatter plot of data
    #ax1.scatter(Beta_X_list, Beta_OX_list, c='b')
    plt.errorbar(Beta_X_list, Beta_OX_list, xerr=np.array([Beta_X_lower_list,
                Beta_X_upper_list]), yerr=np.array([Beta_OX_lower_list,
                Beta_OX_upper_list]), fmt='go', ecolor='k', capthick=2)

    # graph Beta_OX = Beta_X
    x = np.linspace(0,10,1000)
    plt.plot(x, x, linestyle=':', color='red', label="\u03B2\u2092\u2093 = \u03B2\u2093")
    # graph Beta_OX = Beta_X - 0.5
    plt.plot(x,x-0.5,linestyle='-.', color='brown', label="\u03B2\u2092\u2093 = "
                                                          "\u03B2\u2093 - 0.5")
    # graph Beta_OX = 0.5
    plt.axhline(y=0.5, linestyle='--', color='orange', label="\u03B2\u2092\u2093 = 0.5")

    # set bounds for x-axis
    plt.xlim(0.2, 3)
    # set bounds for y-axis
    plt.ylim(-0.4, 1.4)
    # make legend
    plt.legend(fontsize=18)

    # show plot
    plt.show()
```

```
248
249     # save file
250     if(y_or_n_delB == 'Y'):
251         # create string for graph's filename
252         graph_filename = image_name+parsed_filename+"_w_delBeta.png"
253     else:
254         # create string for graph's filename
255         graph_filename = image_name + parsed_filename + ".png"
256
257     # save figure as PNG
258     fig.savefig(graph_filename, dpi=600)
259
260     return parsed_filename
261
262 # a function to determine if a GRB is optically dark
263 # @param: list of GRB objects, filename without file extension, Y or N to graph data,
264 # Y or N to del beta, image name, user defined ID
265 def determine_dark_Jakobsson(GRB_list, parsed_filename, yes_or_no_graph, yes_or_no_delB,
266                              image_name, user_defined_ID):
267     # initialize counter for number of dark GRBs
268     dark_counter = 0
269     # initialize list of dark GRBs
270     dark_GRBs_list_Jakobsson = []
271
272     # run through list of GRB objects
273     for y in GRB_list:
274         # calculate Jakobsson distance
275         D_Jakobsson = 0.5 + float(y.BetaOX) - float(y.upper_sigmaOX) - delta_beta_ox_t
276         #check if Jakobsson distance is positive (yielding dark)
277         if ( D_Jakobsson > 0 and ( float(y.BetaOX) != 0 ) ):
278             # set D_Jakobsson for dark GRB
279             y.set_D_Jakobsson(D_Jakobsson)
280             # append GRB object to list of Jakobsson dark GRBs
281             dark_GRBs_list_Jakobsson.append(y)
282
283     # check that there exists a dark GRB according to the Jakobbson method
284     if (len(dark_GRBs_list_Jakobsson) != 0):
285         if ( yes_or_no_graph == "Y"):
286             # graph dark bursts only
287             graph(dark_GRBs_list_Jakobsson, parsed_filename,
288                   "\u03B2\u2092\u2093 vs. \u03B2\u2093 (" + user_defined_ID
289                   + "Jak Dark): ", yes_or_no_delB, image_name)
290
291             # notify user of printing of optically dark GRBs
292             print()
293             print("List of Optically-Dark GRBs:")
294             print()
295             print(" ID Number   \u0394t\u2093 [hr] \u0394t\u2092 [hr] \u0394t [hr]"
296                   "    \u03B2\u2093    \u03C3\u2093_Up   "
297                   "\u03C3\u2093_Low ""    \u03B2\u2092\u2093   \u03C3\u2092\u2093_Up"
298                   "  \u03C3\u2092\u2093_Low ")
299             # print out optically dark bursts
300             for l in dark_GRBs_list_Jakobsson:
301                 # print out GRB object attributes for each dark GRB object in list of dark
```

```
                # GRB objects
                print(l.ID, "    " + l.dtX, "    " + l.dtO,"    " + l.del_t,"   " +
                      l.BetaX,"   " + l.upper_sigmaX,"   " + l.lower_sigmaX,"   " +
                      l.BetaOX,"   " + l.upper_sigmaOX,"   " +l.lower_sigmaOX)
                dark_counter += 1

            if (yes_or_no_delB == 'Y'):
                # write data of optically dark bursts to file
                with open(image_name + parsed_filename + "_w_delBeta.csv", 'w',
                          newline='') as file:
                    # open file object
                    writer = csv.writer(file)
                    # write column headers to file
                    writer.writerow(["ID Number", "dt_x [hr]", "dt_o [hr]", "dt [hr]",
                                     "Beta_x", "sigma_x_Up", "sigma_x_Low", "Beta_ox",
                                     "sigma_ox_Up", "sigma_ox_Low"])
                    # run through list of dark GRBs according to the van der Horst method
                    for q in dark_GRBs_list_Jakobsson:
                        # write GRB object attributes for each dark GRB object to file
                        writer.writerow([q.ID, q.dtX, q.dtO, q.del_t, q.BetaX,
                                         q.upper_sigmaX, q.lower_sigmaX, q.BetaOX,
                                         q.upper_sigmaOX, q.lower_sigmaOX])
            else:
                # write data of optically dark bursts to file
                with open(image_name + parsed_filename + ".csv", 'w', newline='') as file:
                    # open file object
                    writer = csv.writer(file)
                    # write column headers to file
                    writer.writerow(["ID Number", "dt_x [hr]", "dt_o [hr]", "dt [hr]",
                                     "Beta_x", "sigma_x_Up", "sigma_x_Low", "Beta_ox",
                                     "sigma_ox_Up", "sigma_ox_Low"])
                    # run through list of dark GRBs according to the van der Horst method
                    for q in dark_GRBs_list_Jakobsson:
                        # write GRB object attributes for each dark GRB object to file
                        writer.writerow([q.ID, q.dtX, q.dtO, q.del_t, q.BetaX,
                                         q.upper_sigmaX, q.lower_sigmaX, q.BetaOX,
                                         q.upper_sigmaOX, q.lower_sigmaOX])


            print()
            print("Number of Jakobsson Dark GRBs from " + parsed_filename + ": ",
                  (dark_counter))
            print()

    # return list of optically dark bursts
    return dark_GRBs_list_Jakobsson


# a function to determine if a GRB is optically dark
# @param: list of GRB objects, filename without file extension, Y or N to graph data,
# Y or N to del beta, image name, user defined ID
def determine_dark_vanderHorst(GRB_list, parsed_filename, yes_or_no_graph, yes_or_no_delB,
                               image_name, user_defined_ID):
    # initialize counter for number of dark GRBs
```

```
    dark_counter = 0
    # initialize list of dark GRBs
    dark_GRBs_list_vanderHorst = []

    # run through list of GRB objects
    for y in GRB_list:
        # calculate van der Horst distance
        D_vanderHorst= ( -1*float(y.BetaX) - ( float(y.lower_sigmaX) )  + float(y.BetaOX)
                        - ( float(y.upper_sigmaOX) + delta_beta_ox_t ) - 0.5 )/math.sqrt(2)

        # check if van der Horst distance is positive (yielding dark)
        if ( (-1*float(y.BetaX) - 0.5 > -1*float(y.BetaOX) + ( float(y.upper_sigmaOX) +
                                                               delta_beta_ox_t ) ) and
                (-1*float(y.BetaOX) + 0.5 < -1*float(y.BetaX) - float(y.lower_sigmaX) )
                and ( float(y.BetaOX) != 0 ) ):
            # set D_vanderHorst for dark GRB
            y.set_D_vanderHorst(D_vanderHorst)
            # append GRB object to list of van der Horst dark GRBs
            dark_GRBs_list_vanderHorst.append(y)

    # check that there exists a dark GRB according to the Van der Horst method
    if (len(dark_GRBs_list_vanderHorst) != 0):

        if (yes_or_no_graph == "Y"):
            # graph dark bursts only
            graph(dark_GRBs_list_vanderHorst, parsed_filename, "\u03B2\u2092\u2093 vs. "
            "\u03B2\u2093 (" + user_defined_ID + "VdH Dark): ", yes_or_no_delB, image_name)

            # notify user of printing of optically dark GRBs
            print()
            print("List of Optically-Dark GRBs:")
            print()
            print(" ID Number   \u0394t\u2093 [hr] \u0394t\u2092 [hr] \u0394t [hr]    "
                  "\u03B2\u2093    \u03C3\u2093_Up   ""\u03C3\u2093_Low ""    \u03B2\u2092"
                  "\u2093   \u03C3\u2092\u2093_Up  \u03C3\u2092\u2093_Low ")
            # print out optically dark bursts
            for l in dark_GRBs_list_vanderHorst:
                # print out GRB object attributes for each dark GRB object
                # in list of dark GRB objects
                print(l.ID, "    " + l.dtX, "    " + l.dtO,"    " + l.del_t,"   " +
                      l.BetaX,"   " + l.upper_sigmaX, "   " + l.lower_sigmaX,"   " +
                      l.BetaOX,"   " + l.upper_sigmaOX,"   " +l.lower_sigmaOX)
                dark_counter += 1

            if(yes_or_no_delB == 'Y'):
                # write data of optically dark bursts to file
                with open(image_name + parsed_filename + "_w_delBeta.csv", 'w',
                          newline='') as file:
                    # open file object
                    writer = csv.writer(file)
                    # write column headers to file
                    writer.writerow(["ID Number", "dt_x [hr]", "dt_o [hr]", "dt [hr]",
                                     "Beta_x", "sigma_x_Up", "sigma_x_Low", "Beta_ox",
                                     "sigma_ox_Up", "sigma_ox_Low"])
```

```
                    # run through list of dark GRBs according to the van der Horst method
                    for q in dark_GRBs_list_vanderHorst:
                        # write GRB object attributes for each dark GRB object to file
                        writer.writerow([q.ID, q.dtX, q.dtO, q.del_t, q.BetaX,
                                         q.upper_sigmaX, q.lower_sigmaX, q.BetaOX,
                                         q.upper_sigmaOX, q.lower_sigmaOX])
            else:
                # write data of optically dark bursts to file
                with open(image_name + parsed_filename + ".csv", 'w', newline='') as file:
                    # open file object
                    writer = csv.writer(file)
                    # write column headers to file
                    writer.writerow(["ID Number", "dt_x [hr]", "dt_o [hr]", "dt [hr]",
                                     "Beta_x", "sigma_x_Up", "sigma_x_Low", "Beta_ox",
                                     "sigma_ox_Up",
                                     "sigma_ox_Low"])
                    # run through list of dark GRBs according to the van der Horst method
                    for q in dark_GRBs_list_vanderHorst:
                        # write GRB object attributes for each dark GRB object to file
                        writer.writerow([q.ID, q.dtX, q.dtO, q.del_t, q.BetaX,
                                         q.upper_sigmaX, q.lower_sigmaX, q.BetaOX,
                                         q.upper_sigmaOX, q.lower_sigmaOX])

            print()
            print("Number of Van der Horst Dark GRBs from " + parsed_filename + ": ",
                  (dark_counter))
            print()

    # return list of optically dark bursts
    return dark_GRBs_list_vanderHorst


# a function to determine optically-darkest GRB per unique GRB ID using the Jakobsson method
# @param: list of dark GRB objects by Jakobsson method, filename without file extension,
# Y or N to graph data, Y or N to del beta, image name, user defined ID
def determine_darkest_Jakobsson(dark_GRBs_list_Jakobsson, parsed_filename, yes_or_no_graph,
                                yes_or_no_delB, image_name, user_defined_ID):
    # initialize counter
    counter = 0
    # initialize dark counter
    dark_counter = 0
    # initialize list of darkest GRBs
    darkest_GRBs_Jakobsson = []

    # check that there exists a dark GRB according to the Jakobbson method
    if ( len(dark_GRBs_list_Jakobsson) != 0 ):

        # run through dark_GRBs_list
        for z in dark_GRBs_list_Jakobsson:

            # initialize new_ID to be the GRB ID read from list
            new_ID = z.ID
            # initialize new_darkest to be the D read from list
            new_darkest_D = z.D_Jakobsson
```

```
            # initialize GRB object corresponding to darkest pairing for its unique GRB ID
            new_darkest_GRB = z

            # check if this is the first data point in dark_GRBs_list
            if ( counter == 0):
                # assign old_ID to first ID in list
                old_ID = z.ID
                # assign old_darkest_D to first D_Jakobsson in list
                old_darkest_D = z.D_Jakobsson
                # assign old_darkest_GRB to first GRB object in list
                old_darkest_GRB = z
            # check if new ID has been reached
            if ( new_ID == old_ID ):
                # check if this pairing for the same GRB ID has larger D
                # (which would make it "darker")
                if ( new_darkest_D > old_darkest_D ):
                    # re-assign darkest D_Jakobsson
                    old_darkest_D = new_darkest_D
                    # re-assign darkest GRB
                    old_darkest_GRB = new_darkest_GRB
            else:
                darkest_GRBs_Jakobsson.append(old_darkest_GRB)
                # re-assign old_darkest to D of different GRB ID just read in from list
                old_darkest_D = z.D_Jakobsson
                # re-assign darkest GRB to that corresponding to new
                # GRB ID just read in from list
                old_darkest_GRB = z

            # re-assign old_ID to GRB ID of GRB just read in from list
            old_ID = z.ID

            # increment dark counter
            counter+=1

        # add last darkest GRB to list of darkest GRBs
        darkest_GRBs_Jakobsson.append(old_darkest_GRB)

        # check that there exists a dark GRB according to the Jakobbson method
        if (len(darkest_GRBs_Jakobsson) != 0):

            if (yes_or_no_graph == "Y"):
                # graph dark bursts only
                graph(darkest_GRBs_Jakobsson, parsed_filename, "\u03B2\u2092\u2093 vs."
                                                               " \u03B2\u2093 ("
                      + user_defined_ID + "Jak Darkest): ", yes_or_no_delB, image_name)

                # notify user of printing of optically darkest GRBs
                print()
                print("List of Optically-Darkest GRBs by Jakobsson Criteria:")
                print()
                print(
                    " ID Number   \u0394t\u2093 [hr] \u0394t\u2092 [hr] \u0394t [hr]    "
                    "\u03B2\u2093    \u03C3\u2093_Up   ""\u03C3\u2093_Low ""    \u03B2"
                    "\u2092\u2093   \u03C3\u2092\u2093_Up  \u03C3\u2092\u2093_Low ")
```

```
518
519                 # run through list of darkest GRBs
520                 for l in darkest_GRBs_Jakobsson:
521                     # print out GRB object attributes for each dark GRB
522                     # object in list of darkest GRB objects
523                     print(l.ID, "    " + l.dtX, "    " + l.dtO, "    " + l.del_t, "   " +
524                           l.BetaX, "   " + l.upper_sigmaX, "   " + l.lower_sigmaX, "   " +
525                           l.BetaOX, "   " + l.upper_sigmaOX, "   " + l.lower_sigmaOX)
526                     dark_counter += 1
527
528                 print()
529                 print("Number of Darkest Jakobsson Dark GRBs from " + parsed_filename +
530                       ": ", (dark_counter))
531                 print()
532
533                 if (yes_or_no_delB == 'Y'):
534                     # write data of optically dark bursts to file
535                     with open(image_name + parsed_filename + "_w_delBeta.csv", 'w',
536                               newline='') as file:
537                         # open file object
538                         writer = csv.writer(file)
539                         # write column headers to file
540                         writer.writerow(["ID Number", "dt_x [hr]", "dt_o [hr]", "dt [hr]",
541                                          "Beta_x", "sigma_x_Up", "sigma_x_Low", "Beta_ox",
542                                          "sigma_ox_Up", "sigma_ox_Low"])
543                         # run through list of dark GRBs according to
544                         # the Jakobbson method
545                         for q in darkest_GRBs_Jakobsson:
546                             # write GRB object attributes for each dark GRB object to file
547                             writer.writerow(
548                                 [q.ID, q.dtX, q.dtO, q.del_t, q.BetaX, q.upper_sigmaX,
549                                  q.lower_sigmaX, q.BetaOX,q.upper_sigmaOX, q.lower_sigmaOX])
550                 else:
551                     # write data of optically dark bursts to file
552                     with open(image_name + parsed_filename + ".csv", 'w',
553                               newline='') as file:
554                         # open file object
555                         writer = csv.writer(file)
556                         # write column headers to file
557                         writer.writerow(["ID Number", "dt_x [hr]", "dt_o [hr]", "dt [hr]",
558                                          "Beta_x", "sigma_x_Up", "sigma_x_Low", "Beta_ox",
559                                          "sigma_ox_Up", "sigma_ox_Low"])
560                         # run through list of dark GRBs according to the Jakobbson method
561                         for q in darkest_GRBs_Jakobsson:
562                             # write GRB object attributes for each dark GRB object to file
563                             writer.writerow(
564                                 [q.ID, q.dtX, q.dtO, q.del_t, q.BetaX, q.upper_sigmaX,
565                                  q.lower_sigmaX, q.BetaOX,q.upper_sigmaOX, q.lower_sigmaOX])
566
567     # return list of optically dark bursts
568     return darkest_GRBs_Jakobsson
569
570 # a function to determine optically-darkest GRB per unique GRB ID
571 # using the Van der Horst method
```

```
# @param: list of VdH dark GRB objects, filename without file extension,
# Y or N to graph data, Y or N to del beta, image name, user defined ID
def determine_darkest_vanderHorst(dark_GRBs_list_vanderHorst, parsed_filename,
                                  yes_or_no_graph, yes_or_no_delB,
                                  image_name, user_defined_ID):
    # initialize counter
    counter = 0
    # initialize dark counter
    dark_counter = 0
    # initialize list of darkest GRBs
    darkest_GRBs_vanderHorst = []

    # check that there exists a dark GRB according to the Van der Horst method
    if ( len(dark_GRBs_list_vanderHorst) != 0 ):

        # run through dark_GRBs_list
        for z in dark_GRBs_list_vanderHorst:

            # initialize new_ID to be the GRB ID read from list
            new_ID = z.ID
            # initialize new_darkest to be the D read from list
            new_darkest_D = z.D_vanderHorst
            # initialize GRB object corresponding to darkest pairing for its unique GRB ID
            new_darkest_GRB = z

            # check if this is the first data point in dark_GRBs_list
            if (counter == 0):
                # assign old_ID to first ID in list
                old_ID = z.ID
                # assign old_darkest_D to first D_vanderHorst in list
                old_darkest_D = z.D_vanderHorst
                # assign old_darkest_GRB to first GRB object in list
                old_darkest_GRB = z
            # check if new ID has been reached
            if (new_ID == old_ID):
                # check if this pairing for the same GRB ID has larger D
                # (which would make it "darker")
                if (new_darkest_D > old_darkest_D):
                    # re-assign darkest D_vanderHorst
                    old_darkest_D = new_darkest_D
                    # re-assign darkest GRB
                    old_darkest_GRB = new_darkest_GRB
            else:
                darkest_GRBs_vanderHorst.append(old_darkest_GRB)
                # re-assign old_darkest to D of different GRB ID just read in from list
                old_darkest_D = z.D_vanderHorst
                # re-assign darkest GRB to that corresponding to new GRB ID
                # just read in from list
                old_darkest_GRB = z

            # re-assign old_ID to GRB ID of GRB just read in from list
            old_ID = z.ID

            # increment dark counter
```

```
            counter += 1

        # add last darkest GRB to list of darkest GRBs
        darkest_GRBs_vanderHorst.append(old_darkest_GRB)

        # check that there exists a dark GRB according to the Van der Horst method
        if (len(darkest_GRBs_vanderHorst) != 0):

            if (yes_or_no_graph == "Y"):
                # graph dark bursts only
                graph(darkest_GRBs_vanderHorst, parsed_filename, "\u03B2\u2092\u2093 vs."
                                                                 " \u03B2\u2093 ("
                      + user_defined_ID + "VdH Darkest): ", yes_or_no_delB, image_name)

                # notify user of printing of optically darkest GRBs
                print()
                print("List of Optically-Darkest GRBs by Van der Horst Criteria:")
                print()
                print(
                    " ID Number   \u0394t\u2093 [hr] \u0394t\u2092 [hr] \u0394t [hr]    "
                    "\u03B2\u2093    \u03C3\u2093_Up   ""\u03C3\u2093_Low ""    \u03B2"
                    "\u2092\u2093   \u03C3\u2092\u2093_Up  \u03C3\u2092\u2093_Low ")

                # run through list of darkest GRBs
                for l in darkest_GRBs_vanderHorst:
                    # print out GRB object attributes for each dark GRB object in
                    # list of darkest GRB objects
                    print(l.ID, "    " + l.dtX, "    " + l.dtO, "    " + l.del_t, "   " +
                          l.BetaX, "   " + l.upper_sigmaX, "   " + l.lower_sigmaX, "   " +
                          l.BetaOX, "   " + l.upper_sigmaOX, "   " + l.lower_sigmaOX)
                    dark_counter += 1

                print()
                print("Number of Darkest Van der Horst Dark GRBs from " + parsed_filename +
                      ": ", (dark_counter))
                print()

                if (yes_or_no_delB == 'Y'):
                    # write data of optically dark bursts to file
                    with open(image_name + parsed_filename + "_w_delBeta.csv", 'w',
                              newline='') as file:
                        # open file object
                        writer = csv.writer(file)
                        # write column headers to file
                        writer.writerow(["ID Number", "dt_x [hr]", "dt_o [hr]", "dt [hr]",
                                         "Beta_x", "sigma_x_Up", "sigma_x_Low", "Beta_ox",
                                         "sigma_ox_Up", "sigma_ox_Low"])
                        # run through list of dark GRBs
                        # according to the Van der Horst method
                        for q in darkest_GRBs_vanderHorst:
                            # write GRB object attributes for each dark GRB object to file
                            writer.writerow(
                                [q.ID, q.dtX, q.dtO, q.del_t, q.BetaX, q.upper_sigmaX,
                                 q.lower_sigmaX, q.BetaOX,
```

```
                                q.upper_sigmaOX, q.lower_sigmaOX])
            else:
                # write data of optically dark bursts to file
                with open(image_name + parsed_filename + ".csv", 'w',
                          newline='') as file:
                    # open file object
                    writer = csv.writer(file)
                    # write column headers to file
                    writer.writerow(["ID Number", "dt_x [hr]", "dt_o [hr]", "dt [hr]",
                                     "Beta_x", "sigma_x_Up", "sigma_x_Low", "Beta_ox",
                                     "sigma_ox_Up",
                                     "sigma_ox_Low"])
                    # run through list of dark GRBs
                    # according to the Van der Horst method
                    for q in darkest_GRBs_vanderHorst:
                        # write GRB object attributes for each dark GRB object to file
                        writer.writerow(
                            [q.ID, q.dtX, q.dtO, q.del_t, q.BetaX, q.upper_sigmaX,
                             q.lower_sigmaX, q.BetaOX,
                             q.upper_sigmaOX, q.lower_sigmaOX])

        # return list of optically dark bursts
    return darkest_GRBs_vanderHorst

# function to isolate all data points of a particular, user-defined GRB ID
# @params: list of all loaded GRB IDs, parsed filename, user-defined GRB ID
# @returns: list of GRB objects corresponding to user-defined GRB ID
def isolate_ID(GRB_list, parsed_filename, user_defined_ID):

    # create list for GRB objects with user's ID of interest
    user_defined_ID_list = []
    # initialize counter for number of GRBs with user's ID of interest
    interest_counter = 0

    # run through list of GRB objects
    for a in GRB_list:
        # check if ID from full list of GRB objects matches the user's ID of interest
        if ( user_defined_ID == a.ID ):

            # append GRB to list of GRB objects with user's ID of interest
            user_defined_ID_list.append(a)

    # notify user of printing of optically dark GRBs
    print()
    print("List of GRBs with GRB ID " + user_defined_ID + ":")
    print()
    print(" ID Number   \u0394t\u2093 [hr] \u0394t\u2092 [hr] \u0394t [hr]    \u03B2\u2093"
          "    \u03C3\u2093_Up   ""\u03C3\u2093_Low      \u03B2\u2092\u2093   \u03C3\u2092"
          "\u2093_Up  \u03C3\u2092\u2093_Low ")
    # print out optically dark bursts
    for l in user_defined_ID_list:
        # print out GRB object attributes for each GRB object in list of
        # GRB objects with user's ID of interest
        print(l.ID, "    " + l.dtX, "    " + l.dtO, "    " + l.del_t, "   " + l.BetaX,
```

```
734                 "    " + l.upper_sigmaX, "    " + l.lower_sigmaX, "    " + l.BetaOX, "    " +
735                 l.upper_sigmaOX, "    " + l.lower_sigmaOX)
736         interest_counter += 1
737 
738     # check if no data point with user's ID of interest exists
739     if ( interest_counter == 0 ):
740         # notify user that no data point exists for ID of interest
741         print()
742         print("There is no data point with GRB ID " + user_defined_ID + " in file " +
743               parsed_filename + ".csv.  " + "Please try again.")
744         print()
745     else:
746         # notify user of number of GRB objects with their ID of interest
747         print()
748         print("Number of GRBs with ID " + user_defined_ID + ": ", (interest_counter))
749         print()
750 
751     # return list of optically dark bursts
752     return user_defined_ID_list
753 
754 # function to print out user's choices
755 # @params: Variable designating which menu list to print
756 # @returns: user's argument
757 def user_choice(fork_in_the_road):
758 
759     # check if user is choosing from main menu
760     if (fork_in_the_road == "main"):
761         # print out choices for user
762         print()
763         print("Please choose from one of the below options.")
764         print()
765         print()
766         print("1: Graph all data from loaded file.")
767         print("2: Graph those GRB pairings that are optically-dark according to the "
768               "Jakobbson method.")
769         print("3: Graph those GRB pairings that are optically-dark according to the "
770               "Van der Horst method.")
771         print("4: Graph only the GRB pairings that are the darkest for their unique "
772               "GRB ID according to the "
773               "Jakobbson method.")
774         print("5: Graph only the GRB pairings that are the darkest for their unique "
775               "GRB ID according to the Van der "
776               "Horst method.")
777         print("6: Graph all data points for a particular GRB ID.")
778         print("Q: Quit.")
779         print()
780         print("Selection: ", end='')
781     # check if user is choosing from submenu
782     elif(fork_in_the_road == "unique"):
783         # print out choices for user
784         print()
785         print("Please choose from one of the subsection choices below for your GRB ID"
786               " of interest.")
787         print()
```

```
788         print()
789         print("A: Graph all data points with the GRB ID of interest.")
790         print("B: Graph those data points with the GRB ID of interest that are optically-"
791               "dark according to the Jakobbson method.")
792         print("C: Graph those data points with the GRB ID of interest that are optically-"
793               "dark according to the Van der Horst method.")
794         print("D: Graph only the data point that is the darkest for the GRB ID of interest"
795               " according to the Jakobbson method.")
796         print("E: Graph only the data point that is the darkest for the GRB ID of interest"
797               " according to the Van der Horst method.")
798         print("R: Return to the main menu.")
799         print()
800         print("Selection: ", end='')
801
802     # assign variable for user input
803     argument = input().upper()
804
805     # return's user's choice
806     return argument
807
808 def main():
809     # ask user for name of desired file to open
810     # assign user input to variable for name of file
811     print()
812     file_name = input("Please enter the name of the Paired Data file you wish to load: ")
813     print()
814     # define global variable for delta beta_ox due to temporal separation
815     global delta_beta_ox_t
816     delta_beta_ox_t = 0
817
818     # remove file extension from filename
819     parsed_filename = os.path.splitext(file_name)[0]
820
821     # call function for loading in file
822     GRB_list = load_file(file_name)
823
824     print()
825     # ask user if they wish to include delta beta due to temporal separation
826     del_Beta_Y_N = input("Include \u0394\u03B2 due to temporal separation"
827                          " ('Y' or 'N')?  ").upper()
828     print()
829
830     # make sure user enters valid input
831     while True:
832         if(del_Beta_Y_N == "Y" or del_Beta_Y_N == "N"):
833             break
834         else:
835             del_Beta_Y_N = input("Invalid input.  Please enter either 'Y' or 'N': ").upper()
836             print()
837
838     if(del_Beta_Y_N == 'Y'):
839         print()
840         # ask user for their desired delta beta due to temporal separation
841         delta_t_beta = float(input("Please input the loaded file's temporal separation"
```

```
                                    " in % (i.e. '5'): "))
        print()
        delta_beta_ox_t = math.log(1 + (delta_t_beta / 100),10)
        print()
        print(delta_beta_ox_t)
        print()

    # loop menu until user wishes to quit
    while True:
        # assign variable for user's choice
        # call function to display user's options
        argument = user_choice("main")

        # check if user wishes to graph all data from the loaded file
        if( argument == '1'):
            # call function for graphing Beta_OX parameters
            parsed_filename = graph(GRB_list, parsed_filename, "\u03B2\u2092\u2093 vs. "
                                "\u03B2\u2093: ", del_Beta_Y_N, "Beta_OX_Graph_ALL-")
        # check if user wishes to graph those GRB pairings that are optically-dark
        # according to the Jakobbson method
        elif( argument == '2'):
            # call function to determine if burst is optically dark using Jakobsson method
            # assign Y or N to graph to Yes
            check_J_dark = len(determine_dark_Jakobsson(GRB_list, parsed_filename, "Y",
                                                        del_Beta_Y_N, "Jakobsson_Dark-",""))
            # check if there are no dark GRBs according to the Jakobbson method
            # for the user defined ID
            if (check_J_dark == 0):
                # notify user
                print()
                print("There are no dark GRBs according to the Jakobbson method for GRB "
                      + user_defined_ID + ".")
        # check if user wishes to graph those GRB pairings that are optically-dark
        # according to the Van der Horst method
        elif (argument == '3'):
            # call function to determine if burst is optically dark
            # using Van der Horst method
            # assign Y or N to graph to Yes
            check_vdH_dark = len(determine_dark_vanderHorst(GRB_list, parsed_filename,
                                                            "Y", del_Beta_Y_N,
                                                            "VanderHorst_Dark-", ""))
            # check if there are no dark GRBs according to the
            # Van der Horst method for the user defined ID
            if (check_vdH_dark == 0):
                # notify user
                print()
                print("There are no dark GRBs according to the Van der Horst"
                      " method for GRB " + user_defined_ID + ".")
        # Check if user wishes to graph only the GRB pairings that are the darkest
        # for their unique GRB ID according
        # to the Jakobbson method
        elif (argument == '4'):
            # call function to determine if burst is optically dark using Jakobsson method
            # assign Y or N to graph to No
```

```
            dark_GRBs_list_Jakobsson = determine_dark_Jakobsson(GRB_list, parsed_filename,
                                                                "N", del_Beta_Y_N,
                                                                "Jakobsson_Dark-", "")
            # call function to determine darkest burst per
            # unique GRB ID by Jakobsson method
            check_J_dark = len(determine_darkest_Jakobsson(dark_GRBs_list_Jakobsson,
                           parsed_filename, "Y", del_Beta_Y_N, "Jakobsson_Darkest-", ""))
            # check if there are no dark GRBs according to
            # the Jakobbson method for the user defined ID
            if (check_J_dark == 0):
                # notify user
                print()
                print("There are no dark GRBs according to the Jakobbson method for GRB "
                      + user_defined_ID + ".")
        # Check if user wishes to graph only the GRB pairings that are the darkest for
        # their unique GRB ID according to the Van der Horst method
        elif (argument == '5'):
            # call function to determine if burst is optically dark
            # using Van der Horst method
            # assign Y or N to graph to No
            dark_GRBs_list_vanderHorst = determine_dark_vanderHorst(GRB_list,
                           parsed_filename, "N", del_Beta_Y_N, "VanderHorst_Dark-", "")
            # call function to determine darkest burst per unique GRB ID by the
            # Van der Horst method
            check_vdH_dark = len(determine_darkest_vanderHorst(dark_GRBs_list_vanderHorst,
                        parsed_filename, "Y", del_Beta_Y_N, "vanderHorst_Darkest-", ""))
            # check if there are no dark GRBs according to the
            # Van der Horst method for the user defined ID
            if (check_vdH_dark == 0):
                # notify user
                print()
                print("There are no dark GRBs according to the "
                      "Van der Horst method for GRB "+ user_defined_ID + ".")
        # check if user wishes to graph all data points for a particular GRB ID
        elif (argument == '6'):

            # ask user for their GRB ID of interest
            print()
            print("Please enter the GRB ID of interest (i.e. '050204'): ", end='')
            # assign user input of GRB ID of interest to variable
            user_defined_ID = "ID-" + input()

            # pass variable containing user's GRB of interest to function to create a
            # list of all data points with that unique ID and assign to list
            user_defined_ID_list = isolate_ID(GRB_list, parsed_filename, user_defined_ID)

            # check if no GRB object matches user's ID of interest
            if ( len(user_defined_ID_list) != 0 ):

                # loop menu until user wishes to quit
                while True:
                    # assign variable for user's choice
                    # call function to display user's options
                    sub_choice = user_choice("unique")
```

```
                    # check if user wishes to graph all data points
                    # with the GRB ID of interest
                    if ( sub_choice == 'A'):
                        # graph all data points with user's ID of interest
                        graph(user_defined_ID_list, parsed_filename,
                              "\u03B2\u2092\u2093 vs. \u03B2\u2093 (" + user_defined_ID +
                              " All): ", del_Beta_Y_N,
                              "ALL_" + user_defined_ID + "-")
                    # check if user wishes to graph those data points with the GRB ID of
                    # interest that are optically-dark according to the Jakobbson method
                    elif ( sub_choice == 'B'):
                        # call function to determine if burst is optically dark using
                        # the Jakobsson method
                        # assign Y or N to graph to Yes
                        check_J_dark = len(determine_dark_Jakobsson(user_defined_ID_list,
                                parsed_filename, "Y", del_Beta_Y_N, "Jakobsson_Dark_" +
                                user_defined_ID + "-",  user_defined_ID + " "))
                        # check if there are no dark GRBs according to
                        # the Jakobbson method for the user defined ID
                        if (check_J_dark == 0):
                            # notify user
                            print()
                            print("There are no dark GRBs according to the Jakobbson "
                                  "method for GRB " + user_defined_ID + ".")
                    # check if user wishes to graph those data points with the GRB ID of
                    # interest that are optically-dark according to the Van der Horst method
                    elif (sub_choice == 'C'):
                        # call function to determine if burst is optically dark
                        # using the Van der Horst method
                        # assign Y or N to graph to Yes
                        check_vdH_dark = len(determine_dark_vanderHorst(
                            user_defined_ID_list, parsed_filename, "Y",del_Beta_Y_N,
                            "VanderHorst_Dark_" + user_defined_ID + "-", user_defined_ID
                            + " "))
                        # check if there are no dark GRBs according to the
                        # Van der Horst method for the user defined ID
                        if (check_vdH_dark == 0):
                            # notify user
                            print()
                            print("There are no dark GRBs according to the "
                                  "Van der Horst method for GRB " + user_defined_ID + ".")
                    # check if user wishes to graph only the data point that
                    # is the darkest for the GRB ID of interest
                    # according to the Jakobbson method
                    elif (sub_choice == 'D'):
                        # call function to determine if burst is
                        # optically dark using Jakobsson method
                        # assign Y or N to graph to No
                        dark_GRBs_list_J_user_defined = determine_dark_Jakobsson(
                            user_defined_ID_list, parsed_filename, "N", del_Beta_Y_N,
                            "null", "")

                        # call function to determine darkest burst per
                        # unique GRB ID by Jakobsson criteria
```

```
                        check_J_dark = len(determine_darkest_Jakobsson(
                            dark_GRBs_list_J_user_defined, parsed_filename, "Y",
                            del_Beta_Y_N, "Jakobsson_Darkest_" + user_defined_ID + "-",
                            user_defined_ID + " ") )
                        # check if there are no dark GRBs according to the
                        # Jakobbson method for the user defined ID
                        if (check_J_dark == 0):
                            # notify user
                            print()
                            print("There are no dark GRBs according to the Jakobbson"
                                  " method for GRB " + user_defined_ID + ".")
                    # check if user wishes to graph only the data point that is the darkest
                    # for the GRB ID of interest according to the Jakobbson method
                    elif (sub_choice == 'E'):
                        # call function to determine if burst is
                        # optically dark using Van der Horst method
                        # assign Y or N to graph to No
                        dark_GRBs_list_v_user_defined = determine_dark_vanderHorst(
                            user_defined_ID_list, parsed_filename, "N", del_Beta_Y_N,
                            "null", "")
                        # call function to determine darkest burst per
                        # unique GRB ID by Van der Horst method
                        check_vdH_dark = len(determine_darkest_vanderHorst(
                            dark_GRBs_list_v_user_defined, parsed_filename, "Y",
                            del_Beta_Y_N, "VanderHorst_Darkest_" + user_defined_ID + "-",
                            user_defined_ID + " "))
                        # check if there are no dark GRBs according to the
                        # Van der Horst method for the user defined ID
                        if ( check_vdH_dark == 0):
                            # notify user
                            print()
                            print("There are no dark GRBs according to the "
                                  "Van der Horst method for GRB " + user_defined_ID + ".")
                    # check if user wishes to return to the main menu
                    elif( sub_choice == 'R'):
                        break
                    # check if user does not type one of the available options
                    else:
                        print()
                        print("Your response did not match one of the available options."
                              "  Please try again.")
                        print()
        elif ( argument == 'Q'):
            break
        # check if user does not type one of the available options
        else:
            print()
            print("Your response did not match one of the available options."
                  "  Please try again.")
            print()

# determine execution mode and run main function
if __name__ == '__main__':
```

```
1057    main()
```

Listing 2: Python code for Automating the Graphing of β_{ox} vs. β_x.

9 References

Abbott, B.P., Abbott, R., & Afrough, M. et al. 2017, Physical Review Letters, 119, 161101

Atteia, J.L., Heussaff, J.P., & Dezalay, J.P. et al. 2017, The Astrophysical Journal, 837, 119-131

Berger, E., Zauderer, B.A., & Levan, A. et al. 2013, The Astrophysical Journal, 765, 121-132

Chaisson, E., McMillan, S., & Rice, E. 2017, *Astronomy Today*, Pearson: 9$^{\text{th}}$ Edition

Chonrock, R., & Fong, W. 2015, GCN Circular

Costa, E., Frontera, F., & Heise, J. et al. 1997, Nature, 387, 783-785

de Ugarte Postigo, A., Castro-Tirado, A.J., & Guziy, S. et al. 2006, The Astrophysical Journal, 648, L83-L88

de Ugarte Postigo, A., Thoene, C.C., & Rowlinson, A. et al. 2013, Astronomy & Astrophysics, 563, A62

Donaghy, T.Q., Lamb, D.Q., & Sakamoto, T. et al. 2006, The Astrophysical Journal, 60

Fishman, G.J., & Meegan, C.A. 1995, The Annual Review of Astronomy and Astrophysics, 33, 415-458

Fong, W., Berger, E., & Chornock, R. et al. 2011, The Astrophysical Journal, 730, 26-34

Fong, W., Berger, E., & Margutti, R. et al. 2012, The Astrophysical Journal, 756, 189-201

Fong, W., Berger, E., & Margutti, R. et al. 2015, The Astrophysical Journal, 815, 102

Frail, D.A., Kulkarni, S.R., & Djorgovski, S.G. et al. 2001, The Astrophysical Journal, 562, L55

Fynbo, J.U., Jensen, B.L., & Gorosabel, J. et al. 2001, Astronomy and Astrophysics, 369, 373-379

Galama, T.J., Vreeswijk, P.M., & van Paradijs, J. 1998, Nature, 670-672

Gehrels, N.,Ramirez-Ruiz, E., & Fox, D.B. et al. 2009, Annual Review of Astronomy and Astrophysics, 47, 567-617

Greiner, J., Krühler, T., & Klose, S. 2010, Astronomy & Astrophysics, 526,

Gomboc, A. 2011, Contemporary Physics, 53, 1-20

Groot, P.J., Galama, T.J., & Vreeswijk, P.M. et al. 1998, The Astrophysical Journal, 502, L123-L127

Harrison, F.A., Bloom, J.S., & Frail, D.A. 1999, The Astrophysical Journal, 523, L121-L124

Hjorth, J., Sollerman, J., & Palle, M. 2003, Nature, 423, 847-850

Hjorth, J., Björnsson, G., & Andersen, M. et al. 1999, Science, 283, 2073-2075

Iben, Icko Jr., 2013, *Stellar Evolution Physics, Vol. 2: Advanced Evolution of Single Stars*, Cambridge University Press

Jakobsson, P., Hjorth, J., & Fynbo, J. P. U. et al. 2004, The Astrophysical Journal, 617, L21-L24

Kann, D.A., Klose, S., & Zhang, B. et al. 2011, The Astrophysical Journal, 734, 96

Klebesadel, R.W., Strong, I.B., & Olson, R.A. 1973, The Astrophysical Journal, 182, L85-L88

Kouveliotou, C., Meegan, C., & Fishman, G. et al. 1993, The Astrophysical Journal, 413, L101-L104

Li, Z.Y., & Chevalier, R.A. 1999, The Astrophysical Journal, 526, 716-726

Metzger, B.D., Martinez-Pinedo, G., & Darbha, S. et al. 2010, MNRAS, 1-15

Metzger, M.R., Djorgovski, S.G., & Kulkarni et al. 1997, Nature, 387, 878-880

Mészáros, P., & Rees 1992, The Astrophysical Journal, 397, 570

Mészáros, P., Asano, K., & Veres, P. 2014, Journal of Physics: Conference Series, 485, 012001

Nakar, E. 2007, Physics Reports, 442, 166-236

Paciesas, W.S., Meegan, C.A., & Pendleton, G.N. et al. 1996, The Fourth BATSE Gamma Ray Burst Catalog

Perley, D.A., Metzger, B.D., & Granot, J. et al. 2009, The Astrophysical Journal, 696, 1871-1885

Piran, T. 1999, Physics Reports, 314, 575-667

Piro, L. 2004, in ASP Conf. Ser. 312, Gamma-Ray Bursts in the Afterglow Era, ed. M. Feroci, F. Frontera, N. Masetti, & L. Piro (San Francisco: ASP), 149

Rees, M.J., & Mészáros, P. 1992, Monthly Notices of the Royal Astronomical Society, 258, 41-43

Roming, P., Schady, P., & Fox, D. 2006, The Astrophysical Journal, 652, 1416-1422

Roming, P., Berk, D.V., & Palshin, V. et al. 2006, The Astrophysical Journal, 651, 985-993

Sari, R., Piran, T., & Narayan, R. 1998, The Astrophysical Journal, 497, L17-L20

Tanvir, N.R., Levan, A.J., & Fruchter, A.S. et al. 2013, Nature 547-549

van der Horst, A. J., Kouveliotou, C., & Gehrels, N. et al. 2009, The Astrophysical Journal, 699, 1088

van Paradijs, J., Groot, P.J., & Galama, T. 1997, Nature, 386, 686-689

Vietri, M. & Stella, L. 1999, The Astrophysical Journal, 527, L43-L46

Wijers, R., Rees, M.J., & Mészáros, P. 1997, Monthly Notices of the Royal Astronomical Society, 288, L51-L56

Woosley 1993, The Astrophysical Journal, 405, 273-277

Woosley, S.E., & Bloom, J.S., 2006, The Annual Review of Astronomy and Astrophysics, 44, 507-556

Zhu, B., Zhang, F.W., & Zhang, S. et al. 2015, Astronomy & Astrophysics, 576, A71