xxxxx

Abstract

Contents

List of Figures **iii**

List of Tables **v**

1 Introduction **1**

2 The Physics of Accretion **3**

2.1 The Eddington Limit 3

2.2 High-Mass and Low-Mass X-Ray Binaries 3

2.2.1 Black Hole LMXBs 3

2.2.2 Neutron Star LMXBs 3

2.3 The Shakura-Sunyaev Disk Model 3

2.3.1 Disk Instabilities 3

2.4 GRS 1915+105 and IGR J17091-3624 3

2.4.1 A History of Models of GRS 1915-like Variability 3

2.5 Type II Burst Sources 9

3 Tools & Methods **11**

3.1 Instrumentation 11

3.1.1 The *Rossi X-Ray Timing Experiment* 11

3.1.2 The *Neil Gehrels Swift Observatory* 14

3.1.3 The *X-Ray Multi-Mirror Mission* 15

3.1.4 *Chandra* 16

3.1.5 *Suzaku* 17

3.1.6 *NuSTAR* 17

3.1.7 *INTEGRAL* 17

3.2 Methods & Techniques 18

3.2.1 Lightcurve Morphology 18

3.2.2 Timing Analysis 24

3.2.3 Energy Spectral Analysis 24

4 Variability in IGR J17091-3624: Classification **25**

4.1 Introduction 25

4.2 Data and Data Analysis 25
4.2.1 *RXTE* 25
4.2.2 *Swift* 28
4.2.3 *INTEGRAL* 28
4.2.4 *XMM-Newton* 29
4.2.5 *Chandra* 29
4.2.6 *Suzaku* 30
4.3 Results 31
4.3.1 Outburst Evolution 31
4.3.2 *RXTE* 32
4.3.3 Swift 56
4.3.4 INTEGRAL 56
4.3.5 Chandra 58
4.3.6 XMM-Newton 59
4.3.7 *Suzaku* 59
4.4 Discussion 61
4.4.1 Variability Classes: IGR J17091 vs. GRS 1915 63
4.4.2 General Comparison with GRS 1915+105 69
4.4.3 Comparison with the Rapid Burster 70
4.4.4 Comparison with Altamirano *et al.* 2011 70
4.4.5 New Constraints on Accretion Rate, Mass & Distance 71
4.4.6 Implications for Models of 'Heartbeat' Variability 72
4.5 Conclusions 73

5 The Evolution of X-ray Bursts in the "Bursting Pulsar" GRO J1744–28 75
5.1 Introduction 75
5.2 Data and Data Analysis 76
5.2.1 *RXTE* 77
5.2.2 *Swift* 80
5.2.3 *INTEGRAL* 80
5.2.4 *Chandra* 80
5.2.5 *XMM-Newton* 81
5.2.6 *Suzaku* 81
5.2.7 *NuSTAR* 81
5.3 Results 82
5.3.1 Outburst Evolution 82
5.3.2 Categorizing Bursts 84
5.3.3 Normal Bursts 86
5.3.4 Miniburst s 98
5.3.5 Mesobursts 102
5.3.6 Structured "Bursts" 106
5.4 Discussion 110

5.4.1 Evolution of Outburst and Bursting Behaviour 110
5.4.2 Parameter Correlations 112
5.4.3 Comparison with Previous Studies 112
5.4.4 Comparison with other objects 114
5.4.5 Comparison with Models of Type II Bursts 116
5.5 Conclusions 120

6 The Bursting Pulsar GRO J1744-28: the Slowest Transitional Pulsar? 123
6.1 Introduction 123
6.2 GRO J1744-28: The Bursting Pulsar 125
6.3 Comparison 126
6.4 Discussion 128

7 Conclusions 133

A Model-Independent Classification of each Observation of IGR J17091-3624 163

B List of *RXTE* Observations of the Bursting Pulsar 166

C Normal Burst Histograms 168
C.1 Parameter Correlations in Normal Bursts 168

D Parameter Correlations in Normal Bursts 178

E PANTHEON suite 179
E.1 FITS Genie 179
E.2 Plot Demon 189
E.3 Spec Angel 226
E.4 Back Hydra 240
E.5 PAN Lib 243
E.6 XTE-PAN Lib 281
E.7 Mode Get 288

List of Figures

2.1 A schematic diagram illustrating the the process described by Neilsen et al. (2011) to describe the ρ variability class in GRS 1915+105. 1) The X-ray emission from the system originates from both the accretion disc truncated at an inner radius r_{in} (grey) and a cloud of non-thermal electrons (white ellipse). At some time t, an overdensity in the accretion disc (formed by the Lightman-Eardley Instability) propagates inwards towards r_{in}. 2) As the inner disc heats up, r_{in} begins to slowly increase due to an increase in photon pressure. This destabilises the disc. 3) At some critical density, the disc becomes too unstable and collapses inwards, greatly decreasing r_{in} and raising the inner disc temperature. 4) The sudden increase in emission exceeds the local Eddington limit at r_{in}, ejecting matter from the inner accretion disc in the form of extreme winds. 5) Having been excited by matter in the winds passing through it, the non-thermal electron cloud emits a hard Brehmsstrahlung 'pulse'. 8

3.1 A diagram explaining the process of folding a periodic lightcurve with a known period. **1:** a simulated lightcurve with errors. **2:** Divide the lightcurve into sections by cutting it at every time coordinate Np, where p is the known period and N is any integer. Each data point may now be given a phase coordinate ϕ in addition to its time coordinate t, where $\phi = (t/p) - N$ for N such that $0 \leq \phi < 1$. **3:** The lightcurve segments can be realigned in phase-space, such that points with the same value of ϕ sit at the same x-coordinate. **4:** All points within given bins in ϕ-space are averaged to create a lightcurve corresponding to the averaged oscillations of the original lightcurve. The folding has revealed a peak at $\phi = 0.75$ which was not apparent in the unfolded data. 20

3.2 From top-left: (i) An untouched data-set. (ii) The dataset with all $y < T_L$ removed. (iii) The dataset with all contiguous nonzero regions with $\max(y) < T_H$ removed. (iv) The peak x-values x_m. (v) The restored dataset with the peak x-values x_m highlighted. (vi) The boundaries between adjacent peaks. 22

3.3 The Heidke Skill score of a Class IV observation of IGR J17091-3624 for a selection of different values P_L and P_H. 23

4.1 *RXTE* (Panel a), *Swift/XRT* (Panel b), *Swift/BAT* (Panel b) and *INTEGRAL*/IBIS (Panel d) lightcurves of IGR J17091-3624 during its 2011-2013 outburst. Arrows mark times at which *XMM-Newton* (blue), *Chandra* (red) or *Suzaku* (magenta) observed IGR J17091-3624. The cyan line represents MJD 55963, the approximate time IGR J17091-3624 transitions from the soft to the hard state (Drave et al., 2012). *RXTE/PCA* (Jahoda et al., 1996) data are for the 2–16 keV energy band and taken from (Altamirano et al., 2011b), *Swift/BAT* (Barthelmy, 2000) data are for 15–50 keV, *Swift/XRT* (Burrows et al., 2003) data are for 0.3–10 keV and *INTEGRAL/ISGRI* (Ubertini et al., 2003) data are for 20-40 keV. Note that the data from *Swift/XRT* (Panel B) are shown with a logarithmic *y*-axis to better show the late time progression of the outburst. Data points are coloured according to the observing mode used. The *Swift/XRT* data from times later than MJD 56422 are shown to a different scale to better represent the post-outburst evolution of the source. All data are presented in 1 day bins, except for data from *Swift/BAT* which is presented in 4 day bins. See also Figure 4.2, in which data from *RXTE/PCA* is presented on a smaller scale. The Crab count rates used to normalise these data were 2300 cts s^{-1} PCU^{-1}, 747.5 cts s^{-1}, 0.214 cts s^{-1} and 183.5 cts s^{-1} for *RXTE*, *Swift/XRT*, *Swift/BAT* and *INTEGRAL/ISGRI* respectively. *RXTE* data have not been corrected for the 25' offset to avoid contamination from GX 349+2, and for all instruments we implicitly assume that IGR J17091 presents a Crab-like spectrum. 26

4.2 Global 2–3.5 keV Lightcurves of IGR J17091-3524 during the 2011-2013 outburst, with each point corresponding to the mean Crab-normalised count rate of a single *RXTE* observation of the object (in turn corresponding to between 0.4 and 3.6 ks of data). In each lightcurve, every observation identified as belonging to a particular class (indicated on the plot) is highlighted. These are presented along with a characteristic lightcurve (inset) from an observation belonging to the relevant class. Each lightcurve is 250 s in length, and has a *y*-scale from 0 to 250cts s^{-1} PCU^{-1}. Data taken from Altamirano et al. 2011b. 34

4.3 A global colour-colour diagram (a), "soft" hardness-intensity diagram (b) and "hard" hardness-intensity diagram (c) of the 2011-2013 outburst of IGR J17091, using the colours H_{A1} and H_{A2} defined previously. Observations belonging to different classes have been highlighted in different colours. Data taken from Altamirano et al. 2011b. 35

4.4 Plots of the Class I observation 96420-01-01-00, orbit 0. *Top-left*: 1000 s lightcurve binned on 2 seconds to show lightcurve evolution. *Top-right*: Fourier Power Density Spectrum. *Bottom*: 500 s lightcurve binned on 2 seconds. 37

4.5 Plots of the Class II observation 96420-01-11-00, orbit 0. *Top-left*: 1000 s lightcurve binned on 2 seconds to show lightcurve evolution. *Top-right*: Fourier Power Density Spectrum. *Bottom*: Lightcurve binned on 2 seconds. 38

4.6 Plots of the Class III observation 96420-01-04-01, orbit 0. *Top-left*: 1000 s lightcurve binned on 2 seconds to show lightcurve evolution. *Top-right*: Fourier Power Density Spectrum. *Bottom*: Lightcurve binned on 2 seconds. Note that, to emphasise the behaviour of the lightcurve in this class, we have magnified the 500 s lightcurve y-scale by a factor of 2 compared with the lightcurves presented for other classes. 39

4.7 The Lomb-Scargle periodogram of observation 96420-19-01, orbit 0, with significance levels of 1, 2 and 3σ plotted. The peak at 0.31 Hz was used to define a QPO frequency when folding the data from this observation. 40

4.8 The hardness-intensity diagram (HID_1) of the Class III observation 96420-01-04-01, orbit 0. The data have been folded over a period of 79.61 s, corresponding to the peak frequency in the Lomb-Scargle spectrum of this observation. Inset is the folded lightcurve of the same data. 41

4.9 Plots of the Class IV observation 96420-01-05-00, orbit 0. *Top-left*: 1000 s lightcurve binned on 2 seconds to show lightcurve evolution. *Top-right*: Fourier Power Density Spectrum. *Bottom*: Lightcurve binned on 0.5 seconds. 42

4.10 *Top*: The hardness-intensity diagram (HID_1) of the Class IV observation 96420-01-05-00, orbit 0 showing an anticlockwise loop. The data have been folded over a variable period found with the algorithm described in Section 3.2.1. Inset is the folded lightcurve of the same data. *Bottom Left*: The hardness-intensity diagram of Class IV observations 96420-01-24-02 orbit 0, an example of a clockwise loop. *Bottom Right*: The hardness-intensity diagram of Class IV observation 96420-01-06-00 orbit 0, in which we were unable to ascertain the presence of a loop. 44

4.11 Plots of the Class V observation 96420-01-06-03, orbit 0. *Top-left*: 750 s lightcurve binned on 2 seconds to show lightcurve evolution. *Top-right*: Fourier Power Density Spectrum. *Bottom*: Lightcurve binned on 0.5 seconds. 45

4.12 A portion of the lightcurve of observation 96420-01-06-03, orbit 0, showing Type V_1 flares (highlighted in cyan) and Type V_2 flares (highlighted in red). 46

4.13 Every flare in all observations identified as Class V, plotted in a two-dimensional histogram of flare peak count rate against flare duration to show the two-population nature of these events. 47

4.14 *Top*: The hardness-intensity diagram (HID_1) of a type V_1 flaring region in Class V observation 96420-01-07-00, orbit 0 showing a clockwise loop. The data have been folded over a variable period found with the algorithm described in Section 3.2.1. Inset is the folded lightcurve of the same data. *Bottom Left*: The hardness-intensity diagram of Class V observation 96420-01-25-05 orbit 0, an example of an anticlockwise loop. *Bottom Right*: The hardness-intensity diagram of Class V observation 96420-01-25-06 orbit 0, in which we were unable to ascertain the presence of a loop. 48

4.15 Plots of the Class VI observation 96420-01-09-00, orbit 0. *Top-left*: 750 s lightcurve binned on 2 seconds to show lightcurve evolution. *Top-right*: Fourier Power Density Spectrum. *Bottom*: Lightcurve binned on 1 second. . 49

4.16 *Top*: The hardness-intensity diagram (HID_1) of the Class VI observation 96420-01-30-03, orbit 0 showing a clockwise loop. The data have been folded over a variable period found with the algorithm described in Section 3.2.1. Inset is the folded lightcurve of the same data. *Bottom Left*: The hardness-intensity diagram of Class VI observation 96420-01-30-04 orbit 0, an example of an anticlockwise loop. *Bottom Right*: The hardness-intensity diagram of Class VI observation 96420-01-09-03 orbit 0, in which we were unable to ascertain the presence of a loop. 51

4.17 Plots of the Class VII observation 96420-01-18-05, orbit 0. *Top-left*: 750 s lightcurve binned on 2 seconds to show lightcurve evolution. *Top-right*: Fourier Power Density Spectrum. *Bottom*: Lightcurve binned on 0.5 seconds. 52

4.18 A sliding window Lomb-Scargle spectrogram of Class VII observation 96420-01-18-05, showing power density spectra from an overlapping 32 s window moved 1 s at a time. The peak frequency of this low frequency QPO itself appears to oscillate with a frequency of $\sim$ 5mHz. 53

4.19 Plots of the Class VIII observation 96420-01-19-03, orbit 0. *Top-left*: 300 s lightcurve binned on 2 seconds to show lightcurve evolution. *Top-right*: Fourier Power Density Spectrum. *Bottom*: Lightcurve binned on 0.5 seconds. Inset is a zoom of the 25 s portion of the lightcurve highlighted in cyan, to show the second-scale structure in the lightcurve. 54

4.20 Plots of the Class IX observation 96420-01-35-02, orbit 1. *Top-left*: 1200 s lightcurve binned on 2 seconds to show lightcurve evolution. *Top-right*: Fourier Power Density Spectrum. *Bottom*: Lightcurve binned on 2 seconds. 55

4.21 *INTEGRAL*/ISGRI 150–300 keV significance map of a 2° region centred on the position of IGR J17091-3624, showing the first significant detection of this source above 150 keV. The detection significance is 7.6 σ. 57

4.22 1 ks segments of lightcurves taken from *Chandra* observations 12505, 12405 and 12406, showing Class I, Class VII and Class IX variability respectively. The lightcurve presented for observation 12505 is for the energy range 0.06-10 keV, while the other two lightcurves are for the energy range 0.5-10 keV. All three lightcurves are binned to 0.5 s. 58

4.23 Lightcurves of *XMM-Newton* observations 0677980201 and 0700381301, showing Class IV variability and the hard state respectively. Both lightcurves binned to 2 s. Data for observation 0677980201 is taken from *EPIC-MOS2* and data for observation 0700381301 is taken from *EPIC-pn*. 60

4.24 $\nu P(\nu)$-normalised co-added power density spectra of *XMM-Newton* observation 0700381301 and *Suzaku* observation 407037010. Both observations were taken simultaneously on September 29 2012 (MJD 56199). We sample observation 0700381301 up to a frequency of 10 Hz, while the 2 s time resolution of observation 407037010 results in a Nyquist frequency of 0.25 Hz. 61

4.25 A lightcurve of observation 96420-01-06-02, orbit 0, showing a transition in behaviour between Class IV (in cyan, see Section 4.3.2) and Class V (in red, see Section 4.3.2). 62

4.26 Every flare in all observations identified as Class III or Class IV, plotted in a two-dimensional histogram of flare peak count rate against flare duration to show the two-population nature of these events. Flares belonging to Class IV occupy the distribution at higher peak rate and lower duration, whereas flares belonging to Class III occupy the distribution at lower peak rate and higher duration. 65

4.27 Lightcurve from Class III observation 96420-01-10-01 of IGR J17091-3624, with pairs of primary and secondary count rate spikes highlighted in cyan and red respectively. The yellow region highlights a primary count rate spike that did not produce a secondary. 66

4.28 Mass of the compact object in IGR J17091-3624 plotted against its distance, for values of peak Eddington fractions of F_{Edd} =0.05, 0.1, 0.2 and 0.33. . . 72

5.1 Comparisons of the three outbursts of the Bursting Pulsar reported on in this paper. Times corresponding to pointed observations with *Chandra*, *NuSTAR*, *Suzaku*, *Swift* and *XMM-Newton* are marked. 82

5.2 2–49 keV lightcurves for the four classes of bursting behaviour identified in this paper: **a)** Normal Burst, **b)** Miniburst, **c)** Mesoburst, **d)** Structured Bursts. Note that Panel **d** is plotted with a different time scaling to the other panels so as to better show the behaviour of Structured Bursting. On all figures the median count rate, which we use as a proxy for the persistent emission, is plotted in cyan. Lightcurves **a**-**c** are binned to 0.125 s, while lightcurve **d** is binned to 1 s. 85

5.3 **Upper Panel:** A histogram of the peak 1 s binned peak count rates of the joint population of all Normal and Mesobursts seen by *RXTE*. The dashed line indicates the position of the threshold above which we consider a Type II-like burst to be a Normal Burst. The resultant split of the population into Normal and Mesobursts is indicated by blue and red shading respectively. **Upper Panel:** A histogram of the peak 1 s binned peak count rates of the joint population of all Normal and Minibursts seen by *RXTE*, divided by the persistent emission count rate at that time. The dashed line indicates the position of the threshold below which we consider a burst to be a Miniburst. The resultant split of the population into Normal and Minibursts is indicated by blue and green shading respectively. Note that the x-axis of both plots is logarithmic, and so number density is not preserved. 87

5.4 Central panel shows the global 2–16 keV *RXTE*/PCA lightcurve of the 1995–1996 outburst of the Bursting Pulsar, highlighting periods of time during which Mesobursts (blue) Structured Bursts (yellow) or Normal and Mini bursts (red) are observed. A single Mesoburst was also observed on MJD 50253, during the period of the outburst highlighted in yellow (see Figure 5.17). Other panels show example lightcurves which contain the aforementioned types of bursting behaviour. See section 5.3.2 for a detailed treatment of burst classification. Fluxes reported in units of Crab. 88

5.5 Central panel shows the global 2–16 keV *RXTE*/PCA lightcurve of the 1997–1999 outburst of the Bursting Pulsar, highlighting periods of time during which Mesobursts (blue) Structured Bursts (yellow) or Normal and Mini bursts (red) are observed. Other panels show example lightcurves which contain the aforementioned types of bursting behaviour. 89

5.6 **Top:** The distribution of recurrence times between consecutive Normal Bursts seen in pointed *Chandra*, *XMM-Newton*, *NuSTAR* and *Suzaku* observations of Outburst 3 of the Bursting Pulsar. Distributions of bursts observed by different instruments are stacked on top of each other and colour coded. **Bottom:** The distribution of recurrence times between consecutive Normal Bursts seen in *RXTE* observations of Outbursts 1 & 2 of the Bursting Pulsar. 91

5.7 **Top:** a plot of every Normal Burst, centred by the time of its peak, overlaid on top of each other to show the existence of a common pulse profile. **Bottom:** a plot of every Normal Burst in which count rates have been normalised by the persistent emission count rate during the observation from which each burst was observed. As the bursts are on average closer to the average pulse profile in this metric, this suggests that the intensity of a burst is roughly dependent on the persistent emission rate. Some persistent emission-normalised count rates may be artificially low due to dead-time effects. 93

5.8 *RXTE* lightcurves of Normal Bursts with (top) and without (bottom) 'plateau' features, showing the burst structure in each case. The median count rate, which we use as a proxy for the persistent emission, is plotted in cyan to highlight the presence of the count rate 'dip' after each burst. 94

5.9 A schematic explaining the origin of the 12 Normal Burst parameters used in this study, as well as showing the functional forms of both the skewed Gaussian fit to a burst and the 'dipper function' (Equation 5.2) fit to a dip. Note that we do not fit a function to the plateau, and we calculate its fluence by summing the persistent rate-subtracted counts. Diagram is for explanation only and the burst pictured is neither based on real data nor to scale. 95

5.10 Covariance Matrix with a scatter plot of each of the 66 pairings of the 12 Normal Burst parameters listed in section 5.3.3. Amplitudes and fluences have been normalised by dividing by the persistent count rate k. Pairings which show a correlation using the Spearman Rank metric with a significance $\geq 5\,\sigma$ are highlighted in red. 97

5.11 A 1 s-binned hardness-intensity diagram of a Normal Burst from *RXTE*/PCA observation 10401-01-08-00, with an inset 2–60 keV lightcurve. Significant colour evolution can be seen during the burst, taking the form of a loop. 98

5.12 A representative *RXTE* lightcurve of a Miniburst from OBSID 20077-01-03-00 in Outburst 2. 99

5.13 **Top:** a plot of every Miniburst, centred by the time of its peak, overlaid on top of each other. **Bottom:** a plot of every Miniburst in which count rates have been normalised by the persistent emission count rate during the observation from which each burst was observed. 100

5.14 A portion of observation 10401-01-16-00, featuring a Normal Burst ($\sim$ 100 s) and a Miniburst ($\sim$ 500 s). The top panel shows the total 2–10 keV lightcurve. The middle panel shows lightcurves from two different energy bands; the count rates from the soft energy band have been multiplied by 5.4 so they can more easily be compared with the hard energy band. The bottom panel shows the evolution over time of the ratio between the rates in the two bands. As can be seen in panels 2 and 3, the Miniburst has a significantly higher fractional amplitude in the 4–10 keV energy band than in the 2–4 keV band. 103

5.15 A lightcurve from *RXTE* observation 20078-01-17-00 from Outburst 2, showing an apparent 'plateau' feature after a Mesoburst. 104

5.16 **Top:** a plot of every Mesoburst, centred by the time of its peak, overlaid on top of each other. **Bottom:** a plot of every Mesoburst in which count rates have been normalised by the persistent emission count rate during the observation from which each burst was observed. 105

5.17 A lightcurve from *RXTE*/PCA observation 10401-01-57-03, showing a Mesoburst occuring during a period of Structured Bursting. 107

5.18 A 1 s-binned hardness-intensity diagram from *RXTE* observation 20078-01-23-00, showing that hardness tends to correlate with intensity during Structured Bursting. Data are binned to 8 s, and background has been estimated by subtracting mean count rates in the relevant energy bands from *RXTE* OBSID 30075-01-26-00. . . . 108

5.19 A selection of *RXTE* lightcurves from Structured Bursting observations of the Bursting Pulsar. **Top:** a lightcurve from Outburst 1 showing flaring on timescales of $\sim$ 10 s. **Middle:** a lightcurve from Outburst 1 showing the same flaring behaviour with an additional slower modulation over $\sim$ 50 s. **Bottom:** a lightcurve from Outburst 2 showing a regular sequence of flat-bottomed dips and multi-peaked flaring. These show the wide variety of variability patterns that we classify as 'Structured Bursting'. 109

5.20 A series of lightcurves from *RXTE*/PCA observations of Outburst 2, showing a gradual evolution from Mesobursts to Structured Bursting over a period of $\sim$ 30 days. Each inset lightcurve is plotted with the same y-scaling, and each corresponds to 2 ks of data. 111

5.21 *RXTE* lightcurves of representative Long (top) and Short (bottom) bursts from the Rapid Burster. These bursts were identified and classified by Bagnoli et al. (2015). 115

5.22 A scatter plot showing the relationship between burst fluence and 'missing' dip fluence for Normal Bursts (black) and Minibursts (Red), with the best fit power law plotted in solid blue. A power law fit to just the Normal Bursts (blue dashed line) also approaches the Minibursts. Note that the Normal Bursts plotted in grey were not used to calculate this latter fit, as the effects of instrumental dead-time cause high burst fluences to be under-reported. Upper limits on Miniburst dip fluences are shown with arrows. 118

6.1 **Top:** 2–15 keV *XMM* lightcurve from the TMSP PSR J1023+0038. **Middle:** 2–60 keV *RXTE* lightcurves from the BP during its 1996 and 1997 outbursts, showing similar variability patterns to those seen in PSR J1023+0038. **Bottom:** 2–15 keV *XMM* lightcurve from the TMSP IGR J18245-2452. *XMM* lightcurves are shown from 2–15 keV so that they can be more directly compared with *RXTE*. 127

6.2 Histograms of the 1 s binned count rates from all *RXTE* observations of Structured Bursting in the 1996 (left) and 1997(right) outbursts of the BP. For the 1996 outburst, we fit the distribution with three Gaussians, while for the 1997 outburst we fit the distribution with 2 Gaussians. The individual Gaussians are plotted in solid lines, while the combined total is plotted in a dashed line. 128

6.3 A 7–60/2–7 keV hardness-intensity diagram for *RXTE* observation 10401-01-59-00; the lightcurve of this observation is shown in the inset. To correct for the high background of the region, we subtract the median count rate of *RXTE* observation 30075-01-24-00 from each band; at this time, GRO J1744-28 was in quiescence. We find a strong correlation between hardness and count rate, with a Spearman Rank Correlation Coefficient of 0.93. Data for the hardness-intensity diagram are binned to 10 s, while data for the lightcurve are binned to 5 s. 129

6.4 A plot of a number of objects ranging in scale from LMXBs and High-Mass X-ray Binaries (HMXBs) to Cataclysmic Variables (CVs) and Young Stellar Objects (YSOs) (blue diamonds). In each case, the object is plotted at the luminosity which defines its transition between propeller-mode accretion and free accretion. Campana et al., 2017 suggest that any object above the line of best fit accretes freely, whereas all objects below are in the propellor regime. The BP (red circle) is consistent with approaching this line during periods of Structured Bursting. Errorbars on the BP represent the range of the reported magnetic fields as well as a range of stellar radii between 10–20 km. The range in luminosity for the BP is calculated using 1.5-25 keV *RXTE*/PCA flux, assuming a distance of between 4–8 kpc (e.g. Kouveliotou et al., 1996; Gosling et al., 2007; Sanna et al., 2017c) and a bolometric correction factor of 1–3. Data on the other objects taken from Campana et al., 2017. L is the bolometric luminosity of the object in ergs s^{-1}, P is the period in s, R is the radius in cm and μ is the magnetic moment in $Gauss\,cm^3$. 130

C.1 A histogram showing the distribution of burst fluence ϕ_B amongst our sample of Normal Bursts. 169

C.2 A histogram showing the distribution of burst amplitude a_B amongst our sample of Normal Bursts. 169

C.3 A histogram showing the distribution of burst width σ_B amongst our sample of Normal Bursts. 170

C.4 A histogram showing the distribution of burst skewness c amongst our sample of Normal Bursts. 170

C.5 A histogram showing the distribution of dip fluence ϕ_d amongst our sample of Normal Bursts. 171

C.6 A histogram showing the distribution of dip amplitude a_d amongst our sample of Normal Bursts. 171

C.7 A histogram showing the distribution of dip fall-time d amongst our sample of Normal Bursts. 172

C.8 A histogram showing the distribution of dip recovery timescale λ amongst our sample of Normal Bursts. 172

C.9 A histogram showing the distribution of plateau fluence ϕ_p amongst our sample of Normal Bursts. 173

C.10 A histogram showing the distribution of plateau amplitude a_p amongst our sample of Normal Bursts. 173

C.11 A histogram showing the distribution of persistent-emission-normalised burst fluence ϕ_B/k amongst our sample of Normal Bursts. 174

C.12 A histogram showing the distribution of persistent-emission-normalised burst amplitude a_B/k amongst our sample of Normal Bursts. 174

C.13 A histogram showing the distribution of persistent-emission-normalised dip fluence ϕ_d/k amongst our sample of Normal Bursts. 175

C.14 A histogram showing the distribution of persistent-emission-normalised dip amplitude a_d/k amongst our sample of Normal Bursts. 175
C.15 A histogram showing the distribution of persistent-emission-normalised plateau fluence ϕ_p/k amongst our sample of Normal Bursts. 176
C.16 A histogram showing the distribution of persistent-emission-normalised plateau amplitude a_p/k amongst our sample of Normal Bursts. 176
C.17 Covariance Matrix with a scatter plot of each of the 66 pairings of the 12 Normal Burst parameters listed in section 5.3.3. Pairings which show a correlation using the Spearman Rank metric with a significance $\geq 5\,\sigma$ are highlighted in red. 177

List of Tables

4.1 Chandra observations log covering the three observations considered in this paper. *I* refers to Imaging mode, *C* refers to CC33_Graded mode and *T* refers to Timed Exposure Faint mode. HETG refers to the High Energy Transmission Grating. 29

4.2 Lower and upper quartile count rates, fractional RMS and median colour averaged across all observations belonging to each class. Count rates and fractional RMS are taken from the full energy range of *RXTE/PCA*, and fractional RMS values are 2–60 keV taken from lightcurves binned to 0.5 s. Count rates are normalised for the number of PCUs active during each observation. All values are quoted as 1σ ranges. 36

4.3 A tally of the number of times we assigned each of our nine Variability Classes to an *RXTE* orbit. We have also calculated the amount of observation time corresponding to each class, and thus inferred the fraction of the time that IGR J17091 spent in each class. Note: the values in the Total Time column assume that each orbit only corresponds to a single variability Class. 36

4.4 Results from the IBIS/ISGRI analysis of the 2011–2013 Outburst of IGR J17091. The 20–40 keV flux is given in units of mCrab and (10^{-11} ergs s^{-1} cm^{-2}). Conversion between counts and mCrab was obtained using an observation of the Crab taken during Revolution 1597 between MJD 57305.334 and 57305.894 and the conversion factors of Bird et al. (2016) and Bazzano et al. (2006). 56

4.5 The nine variability classes of IGR J17091-3624, showing the name of the closest corresponding variability class in GRS 1915+105. The names of GRS 1915+105 classes are taken from Belloni et al. (2000), where more detailed descriptions can be found. Eight additional classes of GRS 1915+105 have been described; we do not find analogies to these classes in IGR J17091-3624. 63

4.6 The six OBSIDs explicitly classified in Altamirano et al. (2011b). We also present the GRS 1915 class with which we implicitly label each OBSID in this paper. 70

5.1 Information on the three *Chandra* observations of the Bursting Pulsar during Outburst 3. All other observations of the Bursting Pulsar in the Chandra archive were obtained at times that the source was in quiescence. 81
5.2 Information on the three *NuSTAR* observations of the Bursting Pulsar during Outburst 3. 82
5.3 Statistics on the population of bursts we use for this study, as well as the duration and integrated *RXTE*/PCA exposure time of each mode of bursting. All numbers are the sum of values for Outbursts 1 and 2. As Normal and Miniburstsh happen during the same period of time in each outburst, the exposure time and mode duration for these classes of bursting are equal. . . 88
5.4 A table showing the mean and standard deviation of 10 Normal Burst parameters of *RXTE*-sampled bursts. In each case, we give the values for populations from only Outburst 1, from only Outburst 2 and from the combined population from both outbursts. Histograms for each parameter can be found in Appendix C. 96
5.5 A table showing the mean and standard deviation of 7 parameters of *RXTE*-sampled Miniburstsh from Outburst 1, Outburst 2 and both outbursts combined. Fluence is given in cts PCU^{-1}, peak rate is given in cts s^{-1} PCU^{-1} and rise, fall and total time are given in s. k is the persistent emission rate during the observation in which a given burst was detected. 101
5.6 A table showing the mean and standard deviation of 7 burst parameters of *RXTE*-sampled Mesobursts from Outbursts 1 & 2. k is the persistent emission rate during the observation in which a given burst was detected. 104
5.7 A table showing how our burst classes map to those described in Giles et al. (1996). Giles et al. do not consider the times during the outburst when Structured Bursts appear, and we consider G_3 bursts described by Giles et al. to be consistent with flicker noise. 113

A.1 Here is listed the Observation IDs for every *RXTE* observation that was used in this analysis, along with the variability class which has been assigned to it. *Orb.* is the orbit ID (starting at 0) of each observation segment, *Exp.* is the exposure time in seconds and **X** is the prefix 96420-01. This table is continued overleaf in Table A.2. 164
A.2 A continuation of Table A.1. *Orb.* is the orbit ID (starting at 0) of each observation segment, *Exp.* is the exposure time in seconds and **X** is the prefix 96420-01. 165

B.1 A list of all *RXTE* observations of the Bursting Pulsar used in this study. Exposure is given in seconds, and date is given in days from MJD 50000. The prefixes **A**, **B**, **C**, **D** and **E** correspond to OBSIDs beginning with 10401-01, 20077-01, 20078-01, 20401-01 and 30075-01 respectively. 167

Dedication

To the memory of my brother Christopher, whose name will forever appear alongside my own.

Acknowledgements

This work was made possible by financial support from Science and Technology Facility Council (STFC) and the Royal Astronomical Society (RAS).

I would like to express sincere gratitude to my supervisor Dr. Diego Altamirano. Without his experience, incredible patience and willingness to push me to improve myself, this work would not have been possible.

I would like to thank Professor Tomaso Belloni, Professor Ranjeev Misra and Dr Andrea Sanna for hosting me at their respective institutes at various times in my studies. I also thank Professor Rudy Wijnands, Dr. Mayukh Pahari, Dr. Nathalie Degenaar and Professor Kazutaka Yamaoka for their contributions to our extensive collaborations.

I would like to thank my family for their unwavering support during this at-times arduous task. I also thank my friends and colleagues at the university of Southampton, in particular Mr. David Williams and Dr. Aarran Shaw, who have all in some way helped me to survive to the end of this process; whether they realise it or not.

Chapter 1

Introduction

Chapter 2

The Physics of Accretion

2.1 The Eddington Limit

2.2 High-Mass and Low-Mass X-Ray Binaries

2.2.1 Black Hole LMXBs

2.2.2 Neutron Star LMXBs

2.3 The Shakura-Sunyaev Disk Model

2.3.1 Disk Instabilities

L-E xxxxx Using the assumptions present in the thin disk models of Shakura and Sunyaev, 1973 and Novikov and Thorne, 1973, Lightman and Eardley, 1974 show that an annulus within a thin, radiatively dominated disk has an effective negative radial diffusion coefficient. As such, any initially smooth disk under these conditions tends to separate into dense annuli.

2.4 GRS 1915+105 and IGR J17091-3624

xxxxx

2.4.1 A History of Models of GRS 1915-like Variability

Over the years, a number of models and physical scenarios have been suggested to explain the complex variability seen in GRS 1915-like systems. Successful models must also be able to explain why this type of variability is not seen in a wider array of sources.

One of the most best-studied classes of GRS 1915-like variability is Class ρ, the 'heartbeat' class. This variability class is also present in IGR J17091 (e.g. Altamirano et al., 2011b), and has been the focus of many of the models proposed to explain GRS 1915-like variability. It has been shown that hard X-ray photons lag soft X-ray photons in this class (e.g. Janiuk and Czerny, 2005; Massaro et al., 2010). Other classes in GRS 1915 which show quasi-periodic flaring behaviour also exhibit this phase lag.

Previous authors have established models to explain both the hard photon lag as well as the 'heartbeat'-like flaring itself, generally based on the instability in a radiation-dominated disk first reported by Lightman and Eardley, 1974 (see Section 2.3.1).

Belloni et al., 1997a first proposed an empirical model for flaring in GRS 1915. They suggested that this behaviour is due to a rapid emptying of a portion of the inner accretion disk, followed by a slower refilling of this region over a viscous timescale. These authors divided data from a given observation into equal-sized 2-Dimensional bins in count rate-colour space. A spectral model was then fit to each of these bins independently to perform 'pseudo'-phase-resolved spectroscopy (compare with the method outlined in Section 3.2.3). They showed that the quiescent time between flaring events correlates with the maximum inner disk radius during the flare; i.e., a correlation between the amount of the disk which is emptied and the time needed to refill it. They go on to suggest that their model is able to explain all flaring-type events seen in GRS 1915.

The scenario proposed by Belloni et al., 1997a was mathematically formalised by Nayakshin et al., 2000, who found that it was not consistent with a 'slim' accretion disk (Abramowicz et al., 1988) or with a disk in which viscosity α is constant with respect to radius. As such, their model consists of a cold accretion disk with a modified viscosity law, a non-thermal electron corona and a transient jet of discrete plasma emissions which are ejected when the bolometric luminosity approaches the Eddington Limit. Using this model, Nayakshin et al., 2000 found that some formulations of $\alpha(r)$ result in the disk oscillating between two quasi-stable branches in viscosity-temperature space, over timescales consistent with those seen in the flaring of GRS 1915; they found that this occurs for accretion rates greater than 26% of the Eddington limit. They also found that by varying the functional form of $\alpha(r)$, their model gives rise to a number of lightcurve morphologies which generally match what is seen in data from GRS 1915. Janiuk et al., 2000 built on this model further by including the effect of the transient jet in cooling the disk; an effect not considered in the model by Nayakshin et al., 2000. In this formulation, Janiuk et al., 2000 found that GRS 1915-like variability should occur at luminosities as low as 16% of Eddington.

Belloni et al., 2000 found that variability in GRS 1915 can be described by transitions between three phenomenological states, which differ in luminosity and hardness ratio. This phenomenological scenario is at odds with the model of Nayakshin et al., 2000, which only results in two quasi-stable states.

Nobili, 2003 tried to account for the hard X-Ray lag by supposing that a significant proportion of the X-Ray variability from the accretion disk comes from a single hotspot. They suggest that the lag corresponds to a light travel time, after which a portion of this emission is Comptonised by the jet. In this case, the geometric location of this hotspot determines the magnitude of this lag, and whether it is positive or negative. This scenario goes some way to explaining why GRS 1915 is special, as it requires the presence of a jet during a soft-like state.

Tagger et al., 2004 propose a magnetic explanation for the ejection of the inner accretion disk required by Nayakshin et al., 2000 and Janiuk et al., 2000. They suggest a limit cycle in which a poloidal magnetic field is advected towards the inner disk during the refilling of this region. This field is then destroyed in reconnection events, releasing energy which results in the expulsion of matter from the inner disk. They suggest that the three quasi-stable states proposed by Belloni et al., 2000 can be explained as states in the inner accretion disk with different values of plasma β.

Janiuk and Czerny, 2005 attempt to explain the hard lag in the heartbeats of GRS 1915 more simply, by proposing that it is caused by the non-thermal corona smoothly adjusting to changes in luminosity from the disk. They base the variability of the disk on the model of Nayakshin et al., 2000, and show that the presence of a non-thermal corona which reacts to this variability naturally reproduces the lag behaviour seen in Class ρ in GRS 1915.

Merloni and Nayakshin, 2006 also propose a magnetic explanation for the reformulation of $\alpha(r)$ required by the model of Nayakshin et al., 2000. Assuming that the viscosity in the accretion disk is dominated by turbulence due to magnetorotational instability, they find that allowing for a magnetically dominated corona naturally allows for the forms of $\alpha(r)$ required by Nayakshin et al., 2000.

Zheng et al., 2011 suggest that, when the effects of a magnetic field are included, the accretion rate threshold for GRS 1915-like variability should be $\sim 50\%$ of Eddington; significantly higher than the 16% or 26% reported by Janiuk et al., 2000 or Nayakshin et al., 2000. They go on to suggest that this sort of variability is only seen in GRS 1915 due to this source having the highest accretion rate of all permanently soft-state sources. As such, this scenario still relies on a high accretion rate to trigger GRS 1915-like variability.

Xue et al., 2011 derive a mathematical model of the evolution of a slim accretion disk around a Kerr black hole. They hypothesize that the spin of the black hole, not the accretion rate, may be the driving factor behind GRS 1915-variability. However, they find that that the morphology of X-ray lightcurves from such a disk only has a weak dependence on the spin of the black hole, ruling this out as a possible explanation.

Neilsen et al., 2011 performed phase-resolved spectroscopy of the ρ class in GRS 1915. They find a hard 'spike' after each flare, which they associate with the hard lag in this class previously noted by e.g. Janiuk and Czerny, 2005. They propose a scenario in which high-velocity winds formed by the ejection of matter from the inner disk interact directly

with the corona after a light travel time. The corona then re-releases this energy as a hard bremsstrahlung pulse, causing the hard count rate spike seen in phase-resolved spectra. This scenario is outlined in Figure 2.1. The authors expand this scenario in Neilsen et al., 2012 to suggest that this mechanism can explain all classes in GRS 1915 which display ρ-like flaring. However, this scenario still relies on the model of Nayakshin et al., 2000 to generate the instability in the disk, and it implies that hard photons should always lag soft photons in heartbeat-like variability classes.

In their fitting, Neilsen et al., 2011 consider three spectral models:

1. An absorbed disk black body with a high energy cutoff, of which some fraction has been Compton upscattered
2. An absorbed disk black body with a high energy cutoff, plus a Compton component with a seed photon spectrum tied to the emission from the disk
3. An absorbed disk black body plus a Compton component with a seed photon spectrum tied to the emission from the disk and a bremsstrahlung component

They find that the first of these models (Model 1) is the best fit to the data.

Mineo et al., 2012 also performed psuedo-phase-resolved spectroscopy of the ρ class in GRS 1915, using a number of different spectral models to Neilsen et al., 2011 but a significantly lower phase resolution. In this work, the authors consider six models:

1. A multi-temperature disk black body plus a corona containing both thermal and non-thermal electrons (as forumlated by Poutanen and Svensson, 1996)
2. A multi-temperature disk black body plus a multi-temperature disk black body plus a power law
3. A multi-temperature disk black body plus an independent Compton component
4. A multi-temperature disk black body plus a power law plus reflection from the outer disk
5. A model of Comptonization due to the bulk-motion of matter in the disk
6. A multi-temperature disk black body plus a power law plus a standard black body

With the exception of Models *i* and *vi*, the authors find that none of these models are able to satisfactorily fit the data in each of their phase bins independently. As there is no reasonable physical explanation behind Model *vi*, the authors only consider Model *i*. Their results suggest a large reduction of the corona luminosity during each heartbeat flare, which they interpret as the corona condensing onto the disk. They also find that their results are consistent with GRS 1915 having a slim disk, but inconsistent with the hard lag being caused by photon upscattering in the corona.

Massa et al., 2013 found that the magnitude of the lag between hard and soft photons in the ρ-class of GRS 1915 is not constant. They found that the lag varies between $\sim$ 3–10 s, and correlates strongly with count rate. The magnitude of the lag, therefore, is too large to be simply due to a light travel time to the corona from the disk. The authors suggest that their results are instead consistent the thermal adjustment of the inner disk itself as part of the instability limit cycle invoked to explain the flares.

Massaro et al., 2014a constructed a set of differential equations to mathematically explain the behaviour of the oscillator underlying ρ-like flaring in GRS 1915. They find that a change between variability classes likely corresponds to a a change in global accretion rate,

Figure 2.1: A schematic diagram illustrating the the process described by Neilsen et al. (2011) to describe the ρ variability class in GRS 1915+105. 1) The X-ray emission from the system originates from both the accretion disc truncated at an inner radius r_{in} (grey) and a cloud of non-thermal electrons (white ellipse). At some time t, an overdensity in the accretion disc (formed by the Lightman-Eardley Instability) propagates inwards towards r_{in}. 2) As the inner disc heats up, r_{in} begins to slowly increase due to an increase in photon pressure. This destabilises the disc. 3) At some critical density, the disc becomes too unstable and collapses inwards, greatly decreasing r_{in} and raising the inner disc temperature. 4) The sudden increase in emission exceeds the local Eddington limit at r_{in}, ejecting matter from the inner accretion disc in the form of extreme winds. 5) Having been excited by matter in the winds passing through it, the non-thermal electron cloud emits a hard Brehmsstrahlung 'pulse'.

but that the accretion rate within the ρ class is constant. This model reproduces the count rate-lag correlation reported by Massa et al., 2013, as well as a previously reported correlation between flare recurrence time and count rate (Massaro et al., 2010).

Mir et al., 2016 instead propose a model of variability in the outer disk propagating inwards to the hotter inner disk. They propose a model that explains both the hard lag of the fundamental frequency associated with the heartbeat flares, but also the hard lag of the first harmonic. In contrast to the findings of Massaro et al., 2014a, their scenario requires a sinusoidal variation in the global accretion rate as a function of time.

More recently, Zoghbi et al., 2016 found that the reflection spectrum from GRS 1915 does not match what would be expected from the inner disk behaviour assumed by e.g. Nayakshin et al., 2000. They again perform phase-resolved spectroscopy and fit a number of complex spectral models, finding that their data is best-described by the emergence of a bulge in the inner disk which propagates outwards during each flare.

2.5 Type II Burst Sources

xxxxx

Chapter 3

Tools & Methods

3.1 Instrumentation

As the atmosphere of the Earth is opaque to X-rays and gamma-rays, studying high-energy astrophysical phenomena requires the use of space-based observatories. A number of satellites dedicated to the study of X-rays have been launched over the years, starting with *UHURU* in 1970 (Giacconi et al., 1971) and, most recently, *Hitomi* in 2016 (Takahashi et al., 2016). We use data from a number of these missions in the research reported in this thesis; in particular we use data from the NASA satellites *RXTE*, *Swift* and *Chandra*, the European satellites *XMM-Newton* and *INTEGRAL*, and the Japanese satellite *Suzaku*. This section introduces the instruments used in my studies, as well as the tools used to extract their data for further analysis.

3.1.1 The *Rossi X-Ray Timing Experiment*

The *Rossi X-Ray Timing Experiment*, more commonly known as *RXTE*, was a NASA-operated satellite launched from Cape Canaveral on December 30, 1995 (Bradt et al., 1993). *RXTE* was primarily an X-ray observatory, constructed specifically to study X-ray QPOs seen in X-ray Binaries (Bradt et al., 1990). The observatory operated until January 5, 2012, when it was decommissioned.

RXTE contained three instruments. The main instruments consisted of two X-ray telescopes: the Proportional Counter Array (PCA, Jahoda et al., 1996) and the High Energy X-Ray Timing Experiment (HEXTE Gruber et al., 1996). The satellite also carried an X-ray All-Sky Monitor (ASM, Levine et al., 1996).

PCA consisted of 5 Proportional Counting Units (PCUs) which were sensitive between $\sim$ 2–60 keV. The instrument had an excellent time resolution approaching 1 μs, and an energy resolution of $\sim$ 18% at 6 keV. X-rays were guided onto the detectors by a collimator,

resulting in an instrument field of view with a full-width half-maximum of 1°. There was a $6500\,\mathrm{cm}^2$ collecting area, and no angular resolution.

The HEXTE instrument provided complimentary coverage at higher energies, being sensitive between $\sim$ 15–250 keV. This instrument consisted 8 detectors, with a total collecting area of $1600\,\mathrm{cm}^2$, and had a similar field of view to that of PCA. The time resolution is $8\,\mu$s, and the energy resolution is 15% at 60 keV.

Finally, ASM was a soft X-ray all sky-monitor which covered 80% of the sky every 90 minutes. It was sensitive in the range 2–10 keV, with a total collecting area of $90\,\mathrm{cm}^2$ and a spatial resolution of $3' \times 15'$. Due to its near continual coverage of the sky, ASM was excellent for long-term monitoring of transients in the soft X-ray sky.

Data Formatting

Much of the work in this thesis is based largely on data from PCA, which is freely available through the HEASARC archive maintained by NASA's Goddard Space Flight Centre[1]. In PCA, as well as in other X-ray instruments, this data takes one of two forms:

- **Event-Mode Data:** A list of photon arrival times. Depending on the instrument and observing mode, each of these times will have an associated channel, information about where in the detector the photon hit and a flag indicating the pattern that the photon made on the detector.
- **Binned Data:** A list of evenly space time bins with the number of photons which arrived during each. Depending on the instrument and observing mode, this may be accompanied by some information on the channel distribution of photons arriving in each bin.

The channel a photon falls into is denoted by its energy, although the channel-to-energy conversion for a particular instrument changes over time as the instrument degrades or settings are altered. The channel-to-energy conversions for PCA can be found at `https://heasarc.gsfc.nasa.gov/docs/xte/e-c_table.html`.

Both event-mode and binned-mode data are stored in a Flexible Image Transit System (`.fits`) format. This is a hierarchical data format consisting of a number of 'Header Data Units' (HDUs), each of which contains data in some format and a header with details of the format. In addition to either an event list or a table of binned data, these files also contain a list of Good Time Intervals (GTIs) during which the satellite was functioning normally, as well as an amount of housekeeping information such as the start and end times of the observation.

For PCA observations of faint objects, event mode data with full energy information (referred to as `goodxenon`-mode data) is generally available. However when brighter

[1] `https://heasarc.gsfc.nasa.gov/cgi-bin/W3Browse/w3browse.pl`

objects were observed, telemetry constraints sometimes prevented this full information from being transmitted to Earth. In this case, a number of data products are available; `Standard1` data (binned data with 0.125 s time resolution but no energy information), `Standard2` data (binned data with 16 s time resolution, divided into 128 bins by channel) and a number of other data products with various time and energy resolutions. While `Standard2` data is useful for studying spectral variability over long timescales, it is not useful for studying the second-to-minute scale variability reported on in this thesis. We used `goodxenon` data when available, as this allowed us to use the maximum possible time and energy resolutions, and various other datamodes including `Standard1` when `goodxenon` mode data was not available.

Data Extraction & Background Correction

To perform science with PCA or other instruments, we must extract science products (such as lightcurves, power spectra and energy spectra) from the raw data. Tools to create lightcurves and power spectra from PCA data are available as part of `FTOOLS` [2], a free NASA-maintained suite of software for manipulating `.fits` formatted data. These scripts make use of CALDB: a freely available database of calibration files provided by NASA for a number of active and historical X-ray telescopes (). I also wrote my own software `PANTHEON` (Python ANalytical Tools for High-energy Event-data manipulatiON, Court, 2017, presented in Appendix E) to extract a number of additional products, such as power spectra and spectrograms.

In astronomy, the general way to subtract background from data is by selecting an empty piece of sky from the same observation as the source of interest, and then subtract one from the other. However as PCA had no imaging capability, this is not possible to do with PCA data. Instead, the *RXTE* Guest Observatory Facility provides background models, which estimate the background of an observation based on the known X-ray background near the pointing direction and how the environment of the spacecraft changes over its orbit. Two background models are available, for faint[3] ($< 40\,\mathrm{cts\,s^{-1}\,PCU^{-1}}$) and bright[4] ($> 40\,\mathrm{cts\,s^{-1}\,PCU^{-1}}$) sources; these can be used in conjunction with the `pcabackest` tool in `FTOOLS` to automatically subtract the estimated background from binned PCA data.

As the PCA background models do not subtract the contributions from other sources in the field of view, we also use a different technique to subtract background from observations of GRO J1744-28 (which is in a very crowded region of the sky near the Galactic centre). To try and account for these other sources, we instead choose an observation of the region of GRO J1744-28 taken while this source was in quiescence; we assume that all photons in this observation must be from the particle background, the cosmic background or another

[2] `https://heasarc.gsfc.nasa.gov/ftools/`

[3] `http://heasarc.gsfc.nasa.gov/FTP/xte/calib_data/pca_bkgd/Faint/pca_bkgd_cmfaintl7_eMv20051128.mdl`

[4] `http://heasarc.gsfc.nasa.gov/FTP/xte/calib_data/pca_bkgd/Sky_VLE/pca_bkgd_cmbrightvle_eMv20051128.mdl`

source in the field of view. Although this method does subtract some of the background contributed from other sources in the field, it must be treated with caution as these other sources are likely also variable.

To compare photometry data from PCA with data from other instruments, we normalise our data by the flux from the Crab nebula. The Crab is a commonly used reference source in astronomy due to its apparent brightness and low variability across a wide portion of the electromagnetic spectrum. To Crab-normalise PCA data from a given observation, we take the PCA observation of the Crab which is closest in time to the observation of interest and in the same gain epoch. This follows the method employed in Altamirano et al. (2008b).

Long-term lightcurves from ASM are available on the ASM Light Curves Overview web page (`http://xte.mit.edu/asmlc/ASM.html`) maintained by MIT.

3.1.2 The *Neil Gehrels Swift Observatory*

The *Neil Gehrels Swift Observatory*, formerly and more commonly known as *Swift*, is a NASA-operated satellite launched from Cape Canaveral on November 20, 2004 (Gehrels, 2004). *Swift* was specifically designed to study Gamma Ray Bursts (GRBs), and is notable for its fast slew speed.

Swift carries three instruments: the X-Ray Telescope (XRT, Burrows et al., 2003), the wide field-of-view hard X-ray Burst Alert Telescope (BAT, Krimm et al., 2013) and an UltraViolet/ Optical Telescope (UVOT, Roming et al., 2004). XRT is the primary instrument on *Swift*: it is a focusing telescope with an effective energy range of 0.2–10 keV. Unlike PCA, XRT has imaging capabilities, with a field of view with a radius of 23.6' and an angular resolution of 18". The telescope has a minimum time resolution of 1.8 ms and a minimum energy resolution of $\sim 5\%$ at 6 keV. XRT is operated in one of a number of 'operating modes' during each observation, depending on the requirements of the observer. The two main observing modes are:

1. Proportional Counting (PC) Mode: a full 2-dimensional image every 2.5 s.
2. Windowed Timing (WT) Mode: a 1-dimensional image every 2.8 ms.

Both PC and WT modes also contain full energy information.

The main purpose of the wide area BAT telescope is to identify gamma ray bursts as soon as possible after they begin, so that *Swift* can then slew to them for follow-up observation with XRT. Due to its large field of view (1.4 sr) and effective energy range of 15–150 keV, BAT also provides us with long-term hard X-ray lightcurves of many bright sources in the X-ray sky. It has a detecting area of 5200 cm^2 and, when operating in survey mode, a time resolution of 5 minutes.

The final instrument, UVOT, is intended to take simultaneous optical and ultraviolet observations of sources observed with XRT. It observes in the wavelength range between

170-650 nm, and has 7 filters.

Data Extraction

XRT and ASM data on non-GRB transients are available via online portals maintained by the University of Leicester[5] and the Goddard Space Flight Centre[6] respectively. The University of Leicester portal automatically extracts data lightcurves, spectra, images and source positions from raw XRT data of a given target, using the `xrtpipeline` provided in `FTOOLS`. The Goddard Space Flight Centre provides ready-made 15–50 keV lightcurves of 1019[7] X-ray transients, with cadences of either 1 per day or 1 per *Swift* orbit.

3.1.3 The *X-Ray Multi-Mirror Mission*

The *X-Ray-Multi Mirror Mission* (*XMM-Newton*, Jansen et al., 2001) is an ESA-operated satellite which was launched from Kourou, French Guiana on December 10, 1999, and is still operating almost 20 years later. Like *RXTE* and *Swift*, *XMM-Newton* also carries a number of separate instruments: namely the European Photon Imaging Camera (EPIC, Bignami et al., 1990), the Reflection Grating Spectrometer (RGS, den Herder et al., 1994) and an Optical Monitor (OM, Mason et al., 1996). In the research presented in this thesis, I only make use of data from EPIC.

EPIC consists of three CCD cameras which work independently: two metal-oxide semiconductor CCD cameras (EPIC-MOS1 and EPIC-MOS2) and a single pn CCD camera at the focus of the telescope (EPIC-pn). All cameras observe in the energy range 0.15–15 keV, with a Field of View of 30', an angular resolution of 6" and a maximum energy resolution of $\sim 5\%$. These can be operated in full frame, partial window or timing mode, each of which has a greater time resolution but narrower field of view than the last. The maximum time resolution achievable by EPIC is $7\,\mu$s which EPIC-pn is operated in burst mode; a special pn-only variant of timing mode.

Data Extraction & Processing

XMM-Newton data is extracted and processed using the `SAS` software (Ibarra et al., 2009) provided by ESA[8]. These make use of the continuously updated Current Calibration Files (CCF) provided by ESA.

The process of extracting basic data products from the EPIC instruments can be reduced to a number of steps:

[5] `http://www.swift.ac.uk/user_objects/`
[6] `https://swift.gsfc.nasa.gov/results/transients/`
[7] Count as of May 2018.
[8] `https://www.cosmos.esa.int/web/xmm-newton/sas`

- Use the `SAS` command `cifbuild` to create a Calibration Index File (CIF), containing pointers to the information in the CCF needed to reduce the chosen dataset.
- Use the `SAS` command `odfingest` to create a summary file, containing data corrected by the CCF and by the EPIC housekeeping files.
- Construct a photon event list from EPIC-MOS1 and EPIC-MOS2 using the `SAS` command `emproc`, or from EPIC-pn using the command `epproc`.

The event lists that result from this process can then be filtered using `evselect`, which allows the user to sort photons by arrival time, spatial co-ordinate and energy, among other parameters. These filtered event lists can then be used to create science data products, such as lightcurves and energy spectra.

3.1.4 *Chandra*

The *Chandra X-Ray Observatory* (*Chandra*, Weisskopf, 1999) is a NASA-operated satellite which was launched from Cape Canaveral on July 23, 1999 aboard Space Shuttle *Columbia*. The mission is considered to be one of NASA's 'Great Observatories', along with the Hubble Space Telescope (HST, e.g. Holtzman et al., 1995), the *Compton Gamma Ray Observatory* (*CGRO*, Gehrels et al., 1994) and the *Spitzer Space Telescope* (*Spitzer*, Fanson et al., 1998), which collectively observed the sky between infrared and gamma-ray wavelengths. *Chandra* was designed to study the X-ray sky between ~0.1–10 keV, and contains two instruments: the High Resolution Camera (HRC, Kenter et al., 2000) and the Advanced CCD Imaging Spectrometer (ACIS, Nousek et al., 1987). The spacecraft also carries High and Low Energy Transmission Gratings (HETGS and LETGS respectively, Markert et al., 1994; Brinkman et al., 1986), which can be used in conjunction with the aforementioned detectors to produce high-resolution energy spectra.

HRC contains two detectors: the HRC Imager (HRC-I) and the HRC Spectrogram (HRC-S). HRC-I has the largest field of view of any instrument aboard Chandra (30×30'), but no time resolution and only poor spectral resolution. HRC-S is a long-thin detector strip which is intended to be used as the readout for the LETG. This detector can also be used in Fast Timing mode, in which it has no energy resolution but a timing resolution of 16 μs.

ACIS is intended for use either as an imaging camera or as a detector for the output of the HETG. It has a primary field of view of 16.9×16.9', and operates at a maximum time resolution of 2.85 ms.

Data Extraction & Processing

Like *XMM-Newton*, *Chandra* data is analysed using a purpose-built suite of tools. The software for *Chandra* analysis is named `Ciao` (Fruscione et al., 2006), and is freely

provided by Harvard University[9]. `Ciao` filters and bins data based on any of the four possible parameters stored for a photon event (time, energy and two spatial co-ordinates), and facilitates the production of lightcurves, images and energy spectra.

3.1.5 *Suzaku*

Suzaku (Mitsuda et al., 2007) was a JAXA-operated satellite which operated from its launch from the Uchinoura Space Center on July 10, 2005 until being decomissioned on September 2, 2015. The mission was intended for X-ray spectroscopy; however the satellite's primary instrument, the X-Ray Spectrometer (XRS, Kelley et al., 1999), lost all of its liquid helium coolant within the first month of operation, rendering it effectively unusable. The remaining instruments aboard *Suzaku*, namely the X-Ray Imaging Spectrometers (XIS, Koyama et al., 2007) and the Hard X-Ray Detector (HXD, Takahashi et al., 2007) were unaffected by the malfunction and continued to operate normally throughout the spacecraft's lifetime.

XIS

HXD

Reduction

3.1.6 *NuSTAR*

The *Nuclear Spectroscopic Telescope Array* (*NuSTAR*, Harrison et al., 2013) is a NASA-operated satellite launched from the *Stargazer* aircraft off the coast of the Marshall Islands on June 13, 2012. The satellite carries two co-pointing X-ray telescopes, which are matched with Focal Plane Modules referred to as FPMA and FPMB. These detectors are sensetive and calibrated in the range 3–78 keV, and each has an effective are which maximises at $\sim 450\,\mathrm{cm}^2$ at $\sim 10\,\mathrm{keV}$. The telescopes have a field of view of $12.2 \times 12.2'$, and a full-width half-maximum angular resolution of $\gtrsim 18''$. Events are detected by *NuSTAR* with an arrival time error of $\sim \pm 3\,\mathrm{ms}$, while the energy resolution at 50 keV is around 0.4 keV.

For the research presented in this thesis, we reduce data from *NuSTAR* using the `nupipeline` script from the freely available *NuSTAR* Data Analysis Software (NuSTARDAS[10]). This script automatically runs all of the relevant tasks required to reduce *NuSTAR* data, including flagging of events, flagging of bad pixels and correcting for detector gain. It also calculates sky co-ordinates and energy for each event.

[9] `http://cxc.harvard.edu/ciao/`
[10] `https://heasarc.gsfc.nasa.gov/docs/nustar/analysis/nustar_swguide.pdf`

3.1.7 *INTEGRAL*

The *INTernational Gamma-Ray Astrophysics Laboratory* (*INTEGRAL*, Winkler, 1996) is an ESA-operated satellite which was launched from Kazakhstan's Baikonur Cosmodrome on October 17, 2002. The primary purpose of the mission is the spectroscopy of astrophysical sources in the hard X-ray and soft gamma-ray bands, between $\sim$ 4–10,000 keV.

INTEGRAL carries two main scientific instruments: the SPectrometer on *INTEGRAL* (SPI, Vedrenne et al., 2003) and the Imager on-Board *INTEGRAL* (IBIS, Winkler et al., 2003). The spacecraft also carries an X-ray monitor (JEM-X, Schnopper et al., 1996) and an optical camera (OMC, Gimenez and Mas-Hesse, 1998).

SPI is a high-resolution gamma-ray spectrometer, with an energy resolution of $\sim$2.2 keV at 1.33 MeV and an energy range of 18 keV to 8 MeV. It has a field of view of $> 14°$, an angular resolution of 2.5°, a time resolution of 0.129 ms and a collecting area of 500 cm^2.

IBIS has a larger energy range than SPI, from 15 keV to 10 MeV, and a larger collecting area at 3100 cm^2. The timing accuracy is 61 μs, but the energy resolution peaks at only 8% at $\sim$ 100 keV. As IBIS is an imager, it has a good angular resolution of $\sim$ 12' and a field of view of $> 8 \times 8$". IBIS contains two detector plains stacked on top of each other; the top layer (ISGRI, Lebrun et al., 1996) is designed to detect low-energy gamma rays, while the lower layer (PICsIT, Labanti et al., 1996) is designed to detect the higher-energy gamma rays which pass through ISGRI undetected.

Data products from all four instruments are available via the *INTEGRAL* Heavens portal (Lubiński, 2009) maintained by the *INTEGRAL* Science Data Centre [11]. This portal provides images, lightcurves and spectra of data taken from archived *INTEGRAL* observations.

3.2 Methods & Techniques

To extract meaningful physics from the data provided by the space-based observatories described above, we use a number of mathematical and analytical techniques. We analyse three main properties of the data:

1. **Lightcurve Morphology:** Measuring the luminosity of an object, particularly in relation to its Eddington luminosity, and describing how this luminosity changes over time.

2. **Timing Analysis:** Using Fourier and Lomb-Scargle spectroscopy to identify periodic and quasi-periodic features in the data, and how these change with energy and time.

3. **Energy Spectral Analysis:**

[11] `https://www.isdc.unige.ch/heavens/`

Some of the techniques we used to explore these properties are detailed in this section.

3.2.1 Lightcurve Morphology

The morphology of a lightcurve, how the brightness of a source varies over time, can tell us about the physical process at work behind this variability. For example, the rise and fall-times of an X-ray burst can be matched with characteristic physical timescales of an accreting system to better understand the physical nature of the process behind the burst. xxxxx

Lightcurve Folding

In systems with periodic or quasi-periodic behaviour, it is important to understand the morphology of a single cycle of the behaviour. If the system is bright enough that there are good statistics on timescales shorter than that of the oscillation, this is trivial; however it is often the case that the behaviour of a single oscillation in faint systems is difficult to discern. Instead, we can take the average of many cycles, greatly increasing the signal-to-noise ratio of our data. The process of obtaining this average cycle is known as 'folding' data.

To fold a periodic dataset with a known period p, the time t associated with each datapoint must be converted into a phase ϕ (for $0 \leq \phi < 1$) such that datapoints at the same stage of different oscillations have the same ϕ. This can be done using the formula:

$$\phi(t) = \Phi(t) \mod 1 = (t - t_0)/p \mod 1 \tag{3.1}$$

Where $\Phi(t)$ is the fractional number of cycles which have elapsed between times t_0 and t for an arbitrary start time t_0. This is equivalent to $\phi(t) = (t - t_0)/p - N_t$, where N_t is the integer number of complete cycles which have occured between times t_0 and t.

This procedure can also be thought of as cutting a lightcurve into a number of pieces each of length p, and finding fractionally how far along each point is along the piece in which it falls. Once this information has been found for all datapoints, the data can be rebinned in ϕ-space to 'stack' every cycle on top of one another. We illustrate this process visually in Figure 3.1. This method is useful for finding mean oscillation profiles when p is very close to a constant, such as finding the mean pulse profile of a pulsar over a small number of rotations. However in many cases, such as in quasi-periodic oscillations (QPOs), p is not a constant. More complex methods must then be used to find the mean pulse profile.

Flare-Finding Algorithm

To fold a quasi-periodic oscillation, such as the 'heartbeat' flares seen in GRS 1915+105 and IGR J17091-3624, it is first important to find the t-values which characterise each flare.

Figure 3.1: A diagram explaining the process of folding a periodic lightcurve with a known period. **1:** a simulated lightcurve with errors. **2:** Divide the lightcurve into sections by cutting it at every time coordinate Np, where p is the known period and N is any integer. Each data point may now be given a phase coordinate ϕ in addition to its time coordinate t, where $\phi = (t/p) - N$ for N such that $0 \leq \phi < 1$. **3:** The lightcurve segments can be realigned in phase-space, such that points with the same value of ϕ sit at the same x-coordinate. **4:** All points within given bins in ϕ-space are averaged to create a lightcurve corresponding to the averaged oscillations of the original lightcurve. The folding has revealed a peak at $\phi = 0.75$ which was not apparent in the unfolded data.

To this end, I have created an algorithm to locate individual flares in a dataset containing non-periodic high-amplitude flares. The algorithm is performed as such (see also Figure 3.2):

1. Choose some threshold values T_L and T_H. Set the y-value of all datapoints with $y < T_L$ to zero.
2. Retrieve the x-co-ordinate of the highest value remaining in the dataset. Call this value x_m and store it in a list.
3. Set the value of point at x_m to zero.
4. Scan forwards from x_m. If the selected point has a nonzero value, set it to zero and move to the next point. If the selected point has a zero value, move to step (v).
5. Scan backwards from x_m. If the selected point has a nonzero value, set it to zero and move to the previous point. If the selected point has a zero value, move to step (vi).
6. Retrieve the y-co-ordinate of the highest value remaining in the dataset. Call this y_m.
7. If $y_m > T_H$, repeat steps (ii)–(vi). If $y_m < T_H$, proceed to step (viii).
8. Restore the original dataset.
9. Retrieve the list of x_m values found in step (ii). Sort them in order of size.
10. For each pair of adjacent x_m values, find the x-coordinate of the datapoint between them with the lowest y-value. Call these values x_c.
11. This list of x_c can now be used to demarcate the border between peaks.

The values T_L and T_H can also be procedurally generated for a given piece of data:

1. Select a small section of the dataset or a similar dataset (containing ~ 20 peaks by eye) and note the location x_e of all peaks found by eye.
2. Let P_L and P_H be two arbitrary values in the range $[0, 100]$.
3. Let T_L (T_H) be the P_Lth (P_Hth) percentile of the y-values of the subsection of dataset.
4. Run the flare-finding algorithm up to step (ix). Save the list of x_m.
5. Split the dataset into bins on the x-axis such as the bin width $b \ll p$, where p is the rough x-axis separation between peaks.
6. For each bin, note if you found any value in x_m falls in the bin and note if any value of x_e falls in the bin.
7. Using each bin as a trial, compute the Heidke Skill Score (Heidke, 1926) of the algorithm with the method of finding peaks by eye:

$$HSS = \frac{2(AD - BC)}{(A + B)(B + D) + (A + C)(C + D)} \tag{3.2}$$

Figure 3.2: From top-left: (i) An untouched data-set. (ii) The dataset with all $y < T_L$ removed. (iii) The dataset with all contiguous nonzero regions with $\max(y) < T_H$ removed. (iv) The peak x-values x_m. (v) The restored dataset with the peak x-values x_m highlighted. (vi) The boundaries between adjacent peaks.

Figure 3.3: The Heidke Skill score of a Class IV observation of IGR J17091-3624 for a selection of different values P_L and P_H.

Where A is the number of bins that contain both x_e and x_m, B (C) is the number of bins that contain only x_m (x_e) and D is the number of bins which contain neither (Kok, 2000).

8. Repeat steps (iii)–(vii) for all values of $P_H > P_L$ for P_L and P_H in $[1, 100]$. Use a sensible value for the resolution of P_L and P_H. Save the HSS for each pair of values

9. Locate the maximum value of HSS, and note the P_L and P_H values used to generate it. Use these values to generate your final T_L and T_H values.

We show an example of Heidke skill score grid for this algorithm, applied to a Class IV observation, in Figure 3.3.

Using this algorithm on a lightcurve from an astrophysical source returns a list of flare peak times x_m, and a list of trough times x_c.

Variable Period Lightcurve Folding

With the values x_m and x_c found using the algorithm described above, it is possible to recast Equation 3.1 to fold data over a high-amplitude but quasi-periodic oscillation. I detail my method below:

1. Take the ascending list of t-coordinates x_m. Assign the first element a value $\Phi = 0$.

2. Assign each other point in x_m an integer value $\Phi(t)$, such that the Φ value of the ith value of x_m is defined as:

$$\Phi(x_m^i) = \Phi(x_m^{i-1}) + 1, i \geq 2 \tag{3.3}$$

3. If the troughs between bursts are well-defined, proceed to step 4. Otherwise, skip to step 6.

4. If the t-coordinate of the first datapoint in x_c is less than the t-coordinate of the first datapoint in x_m, assign $\Phi(x_c^1) = -0.5$. Otherwise, assign $\Phi(x_c^1) = -0.5$.

5. Assign each other point in x_m a value $\Phi(x)$, such that the Φ value of the ith value of x_c is defined as:

$$\Phi(x_c^i) = \Phi(x_c^{i-1}) + 1, i \geq 2 \tag{3.4}$$

6. Create a general function defining Φ for all t by fitting the t and Φ values of x_m (and x_c, if used) with a monotonically increasing univariate cubic spline[12] $S(t)$.

7. Define the phase $\phi(t)$ of an arbitrary time t as $\phi(t) = S(t) \mod 1$.

With a phase defined for all points in time, the data can be manipulated as if it had been folded in the usual way. If the trough times in addition to the peak times are used to construct the spline, then the folded data is more accurate: however, by definition the rising part of each flare will occupy phases 0.5–1.0, while the falling part will occupy 0.0–0.5, so any assymetry in the rise and fall times of the average flare is lost.

3.2.2 Timing Analysis

Another way of looking at the variability of an astrophysical source is by looking in the frequency domain. Well-established mathematical techniques, in particular Fourier spectroscopy, are able to deconvolve a time series into series of sine waves. The amplitudes of these sine waves indicate how much variability in the system takes place at a given frequency.

Fourier Spectroscopy

Fourier Spectroscopy (Fourier, 1822) is the most common way to perform frequency analysis on a time series. The Fourier transform $\hat{f}(\nu)$ of a time series $f(t)$ is defined as:

$$\hat{f}(\nu) = \int_{\infty}^{\infty} f(t) e^{-2\pi i t \nu} dt \tag{3.5}$$

[12]Computationaly realised as `PchipInterpolator` in the `Scipy` package for Python (Jones et al., 2001).

Where ν is the frequency to be probed and $i \equiv \sqrt{-1}$. The magnitudes of the complex values $\hat{f}(\nu)$ describe the amplitude of the sine wave deconvolution at frequency ν, while the arguments describe the relative phase of each of these sine waves. As such, $|\hat{f}(\nu)|$

Lomb-Scargle Periodograms

3.2.3 Energy Spectral Analysis

Phase-Resolved Spectroscopy

Chapter 4

Variability in IGR J17091-3624: Classification

4.1 Introduction

4.2 Data and Data Analysis

In this paper, we report data from *RXTE*, *INTEGRAL*, *Swift*, *Chandra*, *XMM-Newton* and *Suzaku* covering the 2011-2013 outburst of IGR J17091. Unless stated otherwise, all errors are quoted at the 1σ level.

In Figure 4.1 we present long-term lightcurves from *RXTE*, *INTEGRAL* and *Swift* with one datapoint per observation, as well as indicating when during the outburst *Chandra*, *XMM-Newton* and *Suzaku* observations were made.

4.2.1 *RXTE*

For our variability study, we focus on the data from the Proportional Counter Array (*PCA*, Jahoda et al., 1996) aboard the Rossi X-Ray Riming Experiment (*RXTE*, Bradt et al., 1993). We analysed all 2011 *PCA* observations of IGR J17091, corresponding to OBSIDs 96065-03, 96103-01 and 96420-01. The observations taken for proposals 96065-03 and 96103-01 were contaminated by the nearby X-ray source GX 349+2 (Altamirano et al., 2011b; Rodriguez et al., 2011b). As such we only use observations performed for proposal 96420-01, corresponding to a total of 243 orbits from 215 separate observations. These were offset by 25' such that GX 349+2 was not in the 1° *PCA* field of view. *RXTE* was decommissioned during a period of Sun constraint centred on MJD 55907, and hence the last observation of IGR J17091 was taken on MJD 55879.

Feb
Mar
Apr
May
Jun
Jul
Aug
Sep
Oct
Nov
Dec
Jan 2012
Feb
Mar
Apr
May
Jun
Jul
Aug
Sep
Oct
Nov
Dec
Jan 2013
Feb
a)
RXTE/PCA
Flux (mCrab)
b)
Swift/XRT
Flux (mCrab)
Windowed Timing
Photon Counting
c)
Swift/BAT
Flux (mCrab)
d)
INTEGRAL/ISGRI
Flux (mCrab)
55600
55700
55800
55900
56000
56100
56200
56300
Time (MJD)

We extracted data from the native `FITS` format using our own software[1]. To perform medium- to high-frequency ($\gtrsim$ 1 Hz) timing analysis, we merged files formatted in PCA's 'Good Xenon' data mode and extracted their data at the maximum time resolution ($\sim 9.5 \times 10^{-7}$ s) without accounting for the background. We divided these data into 128 s segments as this allowed us to reach frequencies below $\sim$ 0.015 Hz, partly sampling the high amplitude quasi-periodic flaring behaviour seen in many classes. Using the Fast Fourier Transform (FFT), we produced the power spectrum of each segment separately. We then averaged these spectra to create a one co-added Power Density Spectrum (PDS) for each observation.

For low-frequency ($\leq$ 1 Hz) timing and correlated spectral/timing analysis, we rebinned the data to 0.5 s and normalised count rates by the number of proportional counters (PCUs) active in each observation. Our choice of 1 Hz allows us to analyse high amplitude 'flaring' behaviour (seen at frequencies $\lesssim$ 0.5 Hz) separately from the lower-amplitude behaviour seen at $\gtrsim$ 5 Hz.

We split the data into three energy bands: A (*PCA* channels 0–14, $\sim$ 2–6 keV), B (*PCA* channels 15–35, $\sim$ 6–16 keV) and C (*PCA* channels 36–255, $\sim$ 16–60 keV). We chose these energy bands to be consistent with the energy bands used by the model-independent classification of variability classes of GRS 1915 in Belloni et al. (2000). For each of the energy-filtered lightcurves produced we estimated background using `pcabackest` from the `FTOOLS` package (Blackburn, 1995) with the *PCA* faint source background model[2]. In all observations, we found that counts in the C band were consistent with background. We then created Lightcurves L_A and L_B from background-subtracted photons counted in the A and B bands respectively. We used these lightcurves to define the full-band lightcurve ($L_T = L_A + L_B$) and the soft colour ($C_1 = L_B/L_A$) of each observation. To complement the Fourier spectra, we also constructed Generalised Lomb-Scargle Periodograms of L_T from each dataset, a modified version of the standard Lomb-Scargle periodogram (Lomb, 1976; Scargle, 1982) that takes into account errors in the dataset (Irwin et al., 1989). Using the Lomb-Scargle periodogram instead of the Fourier periodogram here allows us to sample the low-frequency behaviour of lightcurves with data gaps. This is important, for example, in lightcurves which show two populations of flares, as it allows each population to be studied independently by cropping the other from the lightcurve.

We also used data from Altamirano et al. 2011b to sample the long-term colour evolution of IGR J17091. We use 2 hardness ratios defined by Altamirano et al.: H_{A1} and H_{A2}, corresponding to the ratios of the 2–3.5 keV band against the 3.5–6 keV band and the 6–9.7 keV band against the 9.7–16 keV band respectively.

When possible, if low-frequency peaks were present in the Lomb-Scargle spectrum of an observation, we used the position of the highest peak to define a value for a period. This

[1] `https://github.com/jmcourt/pantheon`

[2] `http://heasarc.gsfc.nasa.gov/FTP/xte/calib_data/pca_bkgd/Faint/pca_bkgd_cmfaintl7_eMv20051128.mdl`

period was then used to rebin data by phase (or 'fold' the data) to search for reccurent hysteretic patterns in the hardness-Intensity diagram (hereafter HID_1, a plot of L_T against C_1). We found that quasi-periodic oscillations in our observations tended to show significant frequency shifts on timescales shorter than the length of the observations. As such, we employed the variable-period folding algorithm outlined in Section 3.2.1 where appropriate. For cases in which this algorithm was not appropriate, we considered small sections of each lightcurve, with a length equivalent to small number of periods, before performing folding.

Additionally, in observations which showed a pattern of high-amplitude X-ray flaring in L_T, we used our own algorithm to find individual flares (this algorithm is described in Section 3.2.1) and collect statistics on the amplitude, duration and profile of these events.

A list of all observations used in this study can be found in Appendix A.

4.2.2 *Swift*

In this paper, we consider data from the Burst Alert Telescope (*BAT*, Barthelmy 2000), and the X-ray Telescope (*XRT*, Burrows et al. 2003) aboard Swift Gamma-ray Burst Mission (Gehrels, 2004). IGR J17091-3624 was observed with *XRT* for a total of 172 pointed XRT observations between MJDs 55575 and 56600, corresponding to Target IDs 31921, 34543, 30967, 30973, 31920, 35096, 67137, 81917, 522245, 677582 and 677981. These observations were interrupted during sun constraints centred on MJDs 55907 and 56272. We created a long-term 0.3–10 keV *Swift/XRT* light curve, with one bin per pointed observation, using the online light-curve generator provided by the UK Swift Science Data Centre (UKSSDC; Evans et al., 2007). We have also created a long-term 15–50 keV lightcurve using the publicly available *Swift/BAT* daily-averaged light curve[3]. These are shown in Figure 4.1 Panels (b) and (c) respectively.

4.2.3 *INTEGRAL*

The INTErnational Gamma Ray Astrophisical Laboratory (*INTEGRAL*, Winkler et al., 2003) is a medium-sized ESA mission launched in 2002. Unique hard X-ray (15–1000 keV with the ISGRI detector plane) sensitivity and wide field of view make *INTEGRAL* ideally suited to surveying the hard X-ray sky (see Bird et al. (2016) and Krivonos et al. (2015) for recent surveys).

INTEGRAL observations are divided into short (~2 ks) pointings called Science Windows (ScWs). We analyse all available observations of IGR J17091 with *INTEGRAL*/IBIS (Ubertini et al., 2003) between MJD 55575–55625 where the source is less than 12 degrees from the centre of the field of view and where there is more the 1 ks of good ISGRI time per ScW. This corresponds to the spectrally hardest period of the 2011-2013 outburst. The

[3] `http://swift.gsfc.nasa.gov/results/transients/weak/IGRJ17091-3624/`

filtering of observations results in a total of 188 Science Windows which were processed using the Offline Science Analysis (OSA) software version 10.2 following standard data reduction procedures[4] in four energy bands (20–40, 40–100, 100–150, 150–300 keV). These bands were selected as they are standard energy bands used in the surveys of Bird et al. (2016) and Bazzano et al. (2006) and allow comparison to these previous works. Images were created at the ScW level as well as a single mosaic of all Science Windows in each energy band.

4.2.4 *XMM-Newton*

In this paper we only consider data from the European Photon Imaging Camera (*EPIC* Bignami et al., 1990) aboard *XMM-Newton* (Jansen et al., 2001). *EPIC* consists of one telescope with a pn-type CCD (*EPIC-pn*, Strüder et al., 2001) and two telescopes with MOS CCDs (*EPIC-MOS1* and *-MOS2*, Turner et al., 2001).

XMM/Newton observed IGR J17091 thrice during the period from 2011–2013 (represented by the blue arrows in Figure 4.1). One of these (OBSID 0721200101) was made on 12 September 2013; we do not consider this observation further as IGR J17091 had returned to quiescence by this time (Altamirano et al., 2013). The remaining two observations, corresponding to OBSIDs 0677980201 and 0700381301 respectively, were taken on March 27 2011 (MJD 55647) and September 29 2012 (MJD 56199).

During observation 0677980201, *EPIC-pn* was operating in burst mode and *EPIC-MOS* was operating in timing mode. Given the low efficiency of burst mode, we only consider data from *EPIC-MOS* for this observation. During observation 0700381301, *EPIC-pn* was operating in timing mode, and thus we use data from *EPIC-pn* for this observation.

We used the *XMM-Newton* Science Analysis Software version 15.0.0 (`SAS`, see Ibarra et al., 2009) to extract calibrated event lists from *EPIC* in both observations. We used these to construct lightcurves to study the X-ray variability, following standard analysis threads[5].

4.2.5 *Chandra*

In this paper we consider data from the Advanced CCD Imaging Camera (*ACIS*, Nousek et al., 1987) and the High Resolution Camera (*HRC*, Murray et al., 1987) aboard *Chandra* (Weisskopf, 1999). *Chandra* made 7 observations of IGR J17091 during the period 2011–2013. Four of these observations were taken after IGR J17091 returned to quiescence, and we do not consider these further in this paper. The Chandra observations log is reported in Table 5.1.

We analysed these data using `CIAO` version 4.8 (Fruscione et al., 2006), following the standard analysis threads. In order to apply the most recent calibration files (CALDB 4.7.0,

[4] http://www.isdc.unige.ch/integral/analysis
[5] `http://www.cosmos.esa.int/web/xmm-newton/sas-threads`

OBSID	Instrument	Grating	Exposure (ks)	Mode	MJD
12505	*HRC-I*	NONE	1.13	*I*	55626
12405	*ACIS-S*	HETG	31.21	*C*	55774
12406	*ACIS-S*	HETG	27.29	*T*	55840

Table 4.1: Chandra observations log covering the three observations considered in this paper. *I* refers to Imaging mode, *C* refers to CC33_Graded mode and *T* refers to Timed Exposure Faint mode. HETG refers to the High Energy Transmission Grating.

Graessle et al., 2006), we reprocessed the data from the three observations using the `chandra_repro` script[6], and used this to produce data products following standard procedures.

The first Chandra observation(OBSID 12505) of this source was made shortly after it went into outburst in February 2011. It was a 1 ks observation performed to refine the position of the X-Ray source, using the High-Resolution Camera in Imaging mode (HRC-I). We created the 0.06–10 kev light curve accounting for the Dead Time Factor (DTF) to correct the exposure time and count rate, due to the deviation of the detector from the standard detection efficiency, using the `dmextract` tool in the `CIAO` software.

Two additional observations (OBSIDs 12405 and 12406) were performed within 214 days of this first observation, using the High Energy Transmition Grating Spectrometer (HETGS) on board *Chandra*. The incident X-Ray flux was dispersed onto *ACIS* using a narrow strip array configuration (ACIS-S). Continuous Clocking and Time Exposure modes were use in each observation respectively (see King et al., 2012 for further details). We exclude any events below 0.4 keV, since the grating efficiency is essentially zero below this energy. In the case of the OBSID 12405 observations we also excluded the Flight Grade 66 events in the event file, as they were not appropriately graded. We extracted the 0.5-10 kev HEGTS light curves, excluding the zeroth-order flux, adopting standard procedures.

4.2.6 *Suzaku*

In this paper, we only consider data from the X-ray Imaging Spectrometer (*XIS*, Koyama et al., 2007) aboard *Suzaku* (Mitsuda et al., 2007). *Suzaku* observed IGR J17091 twice during the period 2011–2013; a 42.1 ks observation on October 2–3, 2012 (MJD 56202–56203, ObsID: 407037010) and an 81.9 ks observation on February 19–21, 2013 (MJD 56342–56344, ObsID: 407037020). *XIS* consists of four X-ray CCDs (*XIS* 0, 1, 2 and 3), and all them except for XIS 2 were operating in the 1/4 window mode which has a minimum time resolution of 2 seconds.

We analysed the *Suzaku* data using *HEASOFT* 6.19 in the following standard procedures after reprocessing the data with `aepipeline` and the latest calibration database (version 20160607). We extracted *XIS* light curves in the 0.7–10 keV range, and subtracted

[6]See e.g. `http://cxc.harvard.edu/ciao/ahelp/chandra_repro.html`

background individually for XIS 0, 1 and 3 and then summed these to obtain the total background. We created Power density spectra (PDS) using `powspec` in the `XRONOS` package.

4.3 Results

4.3.1 Outburst Evolution

The 2011-2013 outburst of IGR J17091-3624 was first detected with *Swift/XRT* on MJD 55595 (3 Feb 2011) (Krimm and Kennea, 2011), and was observed by *RXTE* , *Swift/XRT*, *Swift/BAT* and *INTEGRAL/ISGRI* (see Figure 4.1, Panels a, b, c, d respectively). There were also pointed observations by *XMM-Newton*, *Chandra* and *Suzaku* during this time (denoted by coloured arrows in Figure 4.1).

RXTE data were taken within the first week of the outburst, but they were heavily contiminated by the nearby source GX 349+2 (Rodriguez et al., 2011b). Thus these data are not considered here.

The onset of the outburst can be seen in the *Swift/BAT* lightcurve (Figure 4.1 Panel c). In a 22 day period between MJDs 55584 and 55608, the 15–50 keV intensity from IGR J17091 rose from $\sim$ 9 mCrab to a peak of $\sim$ 110 mCrab. This onset rise in intensity can also be seen in 0.3–10 keV *Swift/XRT* data and 20–40 keV *INTEGRAL/ISGRI* data.

After peak intensity, the 15–50 keV flux (*Swift/BAT*) began to steadily decrease, until returning to a level of $\sim$20 mCrab by MJD 55633. A similar decrease in flux can be seen in the data obtained by *INTEGRAL* at this time (Figure 4.1 Panel (d). However, there was no corresponding fall in the flux at lower energies; both the long-term 2–16 keV *RXTE* data and *Swift/XRT* data (Panels a and b respectively) show relatively constant fluxes of 45 mCrab between MJDs 55608 and 55633.

The significant decrease in high-energy flux during this time corresponds to IGR J17091 transitioning from a hard state to a soft intermediate state (Pahari et al., 2014). This transition coincides with a radio flare reported by Rodriguez et al. (2011a) which was observed by the Australian Telescope Compact Array (*ATCA*).

Altamirano et al., 2011c first reported a 10 mHz QPO in *RXTE* data on MJD 55634 , evolving into 'Heartbeat-like' flaring by MJD 55639 (Altamirano et al., 2011a). Between MJDs 55634 and 55879, the global *RXTE* lightcurve shows large fluctuations in intensity on timescales of days to weeks, ranging from a minimum of $\sim$ 1 mCrab on MJD 55731 to a maximum of $\sim$ 66 mCrab on MJD 55756. The *Swift/XRT* lightcurve shows fluctuations that mirror those seen by *RXTE* during this period, but the amplitude of the fluctuations is significantly reduced.

Swift/XRT was unable to observe again until MJD 55952. Between this date and MJD

55989, *Swift/XRT* observed a gradual decrease in intensity corresponding to a return to the low/hard state (Drave et al., 2012).

Between MJD 55989 and the end of the outburst on MJD 56445, we see secondary peaks in the *Swift/XRT*, *Swift/BAT* and *INTEGRAL/ISGRI* lightcurves that evolve over timescales of $\lesssim$ 100 days. Similar humps have been seen before in lightcurves from other objects, for example the black hole candidate XTE J1650-500 (Tomsick et al., 2003) and the neutron stars SAX J1808.4-3658 (Wijnands et al., 2001) and SAX J1750.8-2900 (Allen et al., 2015). These humps are referred to as 're-flares' (also as 'rebrightenings', 'echo-outbursts', 'mini-outbursts' or a 'flaring tail', e.g. Patruno et al., 2016). We see a total of 3 apparent re-flares in the *Swift/BAT* data, centred approximately at MJDs 56100, 56220 and 56375.

The observation with *XMM-Newton/EPIC-pn* on MJD 56547 (12 September 2013) recorded a rate of 0.019 cts s^{-1}. An observation with *EPIC-pn* in 2007, while IGR J17091 was in quiescence (Wijnands et al., 2012), detected a similar count rate of 0.020 cts s^{-1}. Therefore we define MJD 56547 as the upper limit on the endpoint of the 2011-2013 outburst. As such the outburst, as defined here, lasted for $\lesssim$952 days.

After the end of the 2011-2013 outburst, IGR J17091 remained in quiescence until the start of a new outburst around MJD 57444 (26 February 2016, Miller et al., 2016).

4.3.2 *RXTE*

Using the data products described in Section 4.2, we assigned a model-independent variability class to each of the 243 *RXTE/PCA* orbits. To avoid bias, this was done without reference to the classes defined by Belloni et al. (2000) to describe the behaviour of GRS 1915.

Classes were initially assigned based on analysis of lightcurve profiles, count rate, mean fractional RMS (Vaughan et al., 2003), Fourier and Lomb-scargle power spectra and hardness-intensity diagrams. For observations with significant quasi-periodic variability at a frequency lower than $\sim$ 1 Hz, we also attempted to fold lightcurves to analyse count rate and colour as a function of phase. When flares were present in the lightcurve, we used our algorithm (described in Section 3.2.1) to sample the distribution of parameters such as peak flare count rate, flare rise time and flare fall time. All parameters were normalised per active PCU, and fractional RMS values were taken from 2–60 keV lightcurves binned to 0.5 s. We identify nine distinct classes, labelled I to IX; we describe these in the following sections.

Although the criteria for assigning each class to an observation was different, a number of criteria were given the most weight. In particular, the detection, q-value and peak frequency of a QPO in the range 2 Hz–10 Hz were used as criteria for all classes, as well as the presence or absence of high-amplitude quasi-periodic flaring with a frequency between 0.01–1 Hz. The folded profile of these flares, as well as the presence of associated harmonics, were also used as classification diagnostics in observations. Additionally, the

presence or absence of low count-rate 'dips' in a lightcurve was used as a criterion for Classes VI, VIII and IX. Detailed criteria for each individual class are given below in Sections 4.3.2 to 4.3.2.

For hardness-intensity diagrams, we describe looping behaviour with the terms 'clockwise' and 'anticlockwise'; in all cases, these terms refer to the direction of a loop plotted in a hardness-intensity diagram with colour on the x-axis and intensity on the y-axis.

In Appendix A, we present a list of all orbits used in the study along with the variability classes we assigned to them.

In Figure 4.2, we show global 2–16 keV lightcurves of IGR J17091 during the 2011-2013 outburst. In each panel, all observations of a given class are highlighted in red. A characteristic lightcurve is also presented for each class. In Figure 4.3 panel (a), we show a plot of average hardness H_{A2} against H_{A1} for each observation, showing the long-term hysteresis of the object in colour-colour space. Again, observations belonging to each variability class are highlighted. In Figure 4.3 panels (b) and (c), we show global hardness-intensity diagrams for H_{A1} and H_{A2} respectively.

In Figure 4.3 Panel (a), we see that IGR J17091-3624 traces a two branched pattern in colour-colour space corresponding to a branch which is soft (~ 0.9) in H_{A1} and variable in H_{A2} and a branch which is soft (~ 0.5) in H_{A2} and variable in H_{A1}. The 'soft' HID shown in Figure 4.3 Panel (b) is dominated by a branch with a wide spread in H_{A1} and intensities between $\sim$ 40–60 mCrab. A second branch exists at lower intensities, and shows an anticorrelation between intensity and H_{A1}. Finally, the 'hard' HID shown in Figure 4.3 Panel (c) shows an obvious anticorrelation between H_{A2} and intensity, but there is also a secondary branch between $H_{A2} \approx 0.7$–0.9 at a constant intensity of $\sim$ 40 mCrab.

For characteristic count rates and colours in each class, we quote the upper and lower quartile values (Kenney, 1939) instead of the mean. This is due to the presence of high-amplitude but short-lived flares in many of the classes we describe. Using the upper and lower quartiles as our measure of average and distribution means that our values will be less susceptible to outlier values of count rate and colour present in these flares. All count rates have been background corrected (see Section 4.2.1).

We have obtained mean values for these count rate quartiles, as well as values for colour C_1 and fractional RMS, by calculating these values individually for each orbit. Histograms were then constructed from these datasets for each class, such that the mean and standard deviation of these values could be measured for each class. These values are presented in Table 4.2.

We describe QPOs in terms of their q-value; a measure of coherence defined by the ratio of peak frequency and full-width half-maximum of each QPO. We collected these values by fitting our power spectra with Lorentzians.

For each class, we present three standard data products; a 500 s lightcurve, a variable-length

Intensity (mCrab)
60
40
20
Class

(a) *Colour-Colour Diagram*

(b) *"Soft" (H_{A1}) Hardness-Intensity Diagram*

Class	LQ Rate (cts s^{-1})	UQ Rate (cts s^{-1})	Frac. RMS	Median C_1
I	84–108	106–132	0.13–0.19	0.4–0.68
II	43–57	59–71	0.15–0.23	0.4–0.68
III	64–84	80–110	0.17-0.23	0.35–0.45
IV	63–81	92–122	0.27–0.37	0.32–0.4
V	49–67	88–134	0.44–0.54	0.28–0.46
VI	64–98	111–155	0.29–0.47	0.33–0.61
VII	65–79	128–140	0.45–0.57	0.32–0.42
VIII	62–88	142–178	0.42–0.52	0.36–0.49
IX	87–111	114–144	0.16–0.24	0.42-0.6

Table 4.2: Lower and upper quartile count rates, fractional RMS and median colour averaged across all observations belonging to each class. Count rates and fractional RMS are taken from the full energy range of *RXTE/PCA*, and fractional RMS values are 2–60 keV taken from lightcurves binned to 0.5 s. Count rates are normalised for the number of PCUs active during each observation. All values are quoted as 1σ ranges.

Class	Orbits	Total Time (s)	Fraction
I	31	69569	14.8%
II	26	50875	10.8%
III	14	26228	5.6%
IV	31	69926	14.9%
V	35	72044	15.3%
VI	29	54171	11.5%
VII	11	19241	4.1%
VIII	16	26553	5.7%
IX	50	81037	17.3%

Table 4.3: A tally of the number of times we assigned each of our nine Variability Classes to an *RXTE* orbit. We have also calculated the amount of observation time corresponding to each class, and thus inferred the fraction of the time that IGR J17091 spent in each class. Note: the values in the Total Time column assume that each orbit only corresponds to a single variability Class.

lightcurve where the length has been selected to best display the variability associated with the class and a Fourier PDS. Unless otherwise stated in the figure caption, the 500 s lightcurve and the Fourier PDS are presented at the same scale for all classes. In Table 4.3 we present a tally of the number of times we assigned each Variability Class to an *RXTE* orbit.

Class I – Figure 4.4

In the 2 s binned lightcurve of a Class I observation, there is no structured second-to-minute scale variability. The Fourier PDS of all observations in this class shows broad band noise between $\sim$ 1–10 Hz, as well as a weak QPO (with a q-value of $\sim$ 5) which peaks at around 5 Hz.

Figure 4.4: Plots of the Class I observation 96420-01-01-00, orbit 0. *Top-left*: 1000 s lightcurve binned on 2 seconds to show lightcurve evolution. *Top-right*: Fourier Power Density Spectrum. *Bottom*: 500 s lightcurve binned on 2 seconds.

Figure 4.5: Plots of the Class II observation 96420-01-11-00, orbit 0. *Top-left*: 1000 s lightcurve binned on 2 seconds to show lightcurve evolution. *Top-right*: Fourier Power Density Spectrum. *Bottom*: Lightcurve binned on 2 seconds.

Class II – Figure 4.5

Class II observations are a factor of ~ 2 fainter in the L_T band than Class I observations. They also occupy a different branch in a plot of hardness H_{A2} against flux (see Figure 4.3c). The PDS shows no significant broad band noise above $\sim$ 1Hz unlike that which is seen in Class I. The ~5Hz QPO seen in Class I is absent in Class II.

Class III – Figure 4.6

Unlike Classes I & II, Class III lightcurves show structured flaring, with a peak-to-peak recurrence time of 42–80 s. Most flares consist of a steady $\sim$ 60 s rise in count rate and then an additional and sudden rise to a peak count rate at $\gtrsim$ 200cts s^{-1} PCU^{-1}which lasts for $\lesssim$0.5 s before returning to continuum level (we have magnified the y-scaling in the lightcurve of Figure 4.6 to emphasise this behaviour). This sudden rise is not present in every flare; in some observations it is absent from every flare feature. No 5Hz QPO is present in the PDS and there is no significant variability in the range between $\sim$ 1–10Hz.

As this class has a well-defined periodicity, we folded data in each observation to improve statistics using the best-fit period obtained from generalised Lomb-Scargle Periodogram

Figure 4.6: Plots of the Class III observation 96420-01-04-01, orbit 0. *Top-left*: 1000 s lightcurve binned on 2 seconds to show lightcurve evolution. *Top-right*: Fourier Power Density Spectrum. *Bottom*: Lightcurve binned on 2 seconds. Note that, to emphasise the behaviour of the lightcurve in this class, we have magnified the 500 s lightcurve y-scale by a factor of 2 compared with the lightcurves presented for other classes.

Figure 4.7: The Lomb-Scargle periodogram of observation 96420-19-01, orbit 0, with significance levels of 1, 2 and 3σ plotted. The peak at 0.31 Hz was used to define a QPO frequency when folding the data from this observation.

Analysis; we show a representative Lomb-Scargle periodogram in Figure 4.7. We find an anticlockwise hysteretic loop in the folded HID_1 of all 15 Class III orbits. In Figure 4.8 we show an example of one of these loops.

Class IV – Figure 4.9

The lightcurves in this class show regular variability with a peak-to-peak recurrence time of 25–39 s. We performed peak analysis (see Section 3.2.1) on observations belonging to this class, finding that each peak has a rise time with lower and upper quartile values of 19.5 and 33.5 s, a fall time with lower and upper quartile values of 4.6 and 13.5 s and a peak count rate of 159–241cts s^{-1} PCU^{-1} . There are no prominent significant QPOs in the Fourier PDS above $\sim$ 1Hz.

We folded individual Class IV lightcurves and found anticlockwise hysteretic loops in the HID_1 of 14 out of 30 Class IV observations. In the top panel of Figure 4.10 we show an example of one of these loops. However, we also find clockwise hysteretic loops in 6 Class IV observations, and in 10 orbits the data did not allow us to ascertain the presence of a loop. We provide an example of both of these in the lower panels of Figure 4.10. We note that the structure of clockwise loops are more complex than anticlockwise loops in Class

Figure 4.8: The hardness-intensity diagram (HID_1) of the Class III observation 96420-01-04-01, orbit 0. The data have been folded over a period of 79.61 s, corresponding to the peak frequency in the Lomb-Scargle spectrum of this observation. Inset is the folded lightcurve of the same data.

Figure 4.9: Plots of the Class IV observation 96420-01-05-00, orbit 0. *Top-left*: 1000 s lightcurve binned on 2 seconds to show lightcurve evolution. *Top-right*: Fourier Power Density Spectrum. *Bottom*: Lightcurve binned on 0.5 seconds.

IV, consisting of several lobes[7] rather than a single loop (Figure 4.10, bottom-left).

Compared with Class III, the oscillations in Class IV occur with a significantly lower period, with a mean peak-to-peak recurrence time of ~ 30 s compared to ~ 60 s in Class III.

In Figure 4.3 we show that Classes III and IV can also be distinguished by average hardness, as Class III tends to have a greater value of H_{A2} than Class IV.

Class V – Figure 4.11

The lightcurves in this class, like in Classes III and IV, show flaring behaviour, with flares separated by a few tens of seconds. At higher frequencies, the PDS shows a prominent QPO centred at $\sim$ 4Hz with as q-value of ~ 3. There is also significant broad band noise between $\sim$ 0.1–1Hz

In Figure 4.12 we show that the flaring in this class is more complex than that seen in Classes III and IV. Class V lightcurves consist of short strongly peaked symmetrical flares (hereafter Type V_1) and a longer more complex type of flare (hereafter Type V_2). The Type V_2 flare consists of a fast rise to a local maximum in count rate, followed by a ~ 10 s period in which this count rate gradually reduces by $\sim 50\%$ and then a much faster peak with a maximum count rate between 1 and 2 times that of the initial peak. In both types of flare, we find that the increase in count rate corresponds with an increase in soft colour. The two-population nature of flares in Class V can also clearly be seen in Figure 4.13, where we show a two-dimensional histogram of flare peak count rate against flare duration.

We folded all individual Class V lightcurves, in each case cropping out regions of V_2 flaring. We find clockwise hysteretic loops in the HID_1 of 30 out of 33 Class V observations, suggesting a lag in the aforementioned relation between count rate and soft colour. In the upper panel Figure 4.14 we present an example of one of these loops. In one observation however, we found an anticlockwise loop in the HID_1 (shown in Figure 4.14 lower-left panel). We were unable to ascertain the presence of loops in the remaining 2 orbits; for the sake of completeness, we show one of these in the lower-right panel of Figure 4.14.

Class VI – Figure 4.15

The lightcurves of observations of this class show large dips in count rate; this can be seen in Figure 4.15 at, for example, $t \approx 125$–150 s . These dips vary widely in duration, from ~ 5 to ~ 50 seconds, and the count rate in both L_A and L_B fall to a level consistent with background. The dips' rise and fall times are fast, both lasting no longer than a second. They do not appear to occur with any regular periodicity.

[7] In HIDs with multiple lobes, the loop direction we assign to the observation corresponds to the direction of the largest lobe.

Figure 4.10: *Top*: The hardness-intensity diagram (HID_1) of the Class IV observation 96420-01-05-00, orbit 0 showing an anticlockwise loop. The data have been folded over a variable period found with the algorithm described in Section 3.2.1. Inset is the folded lightcurve of the same data. *Bottom Left*: The hardness-intensity diagram of Class IV observations 96420-01-24-02 orbit 0, an example of a clockwise loop. *Bottom Right*: The hardness-intensity diagram of Class IV observation 96420-01-06-00 orbit 0, in which we were unable to ascertain the presence of a loop.

Figure 4.11: Plots of the Class V observation 96420-01-06-03, orbit 0. *Top-left*: 750 s lightcurve binned on 2 seconds to show lightcurve evolution. *Top-right*: Fourier Power Density Spectrum. *Bottom*: Lightcurve binned on 0.5 seconds.

Figure 4.12: A portion of the lightcurve of observation 96420-01-06-03, orbit 0, showing Type V_1 flares (highlighted in cyan) and Type V_2 flares (highlighted in red).

Figure 4.13: Every flare in all observations identified as Class V, plotted in a two-dimensional histogram of flare peak count rate against flare duration to show the two-population nature of these events.

Figure 4.14: *Top*: The hardness-intensity diagram (HID_1) of a type V_1 flaring region in Class V observation 96420-01-07-00, orbit 0 showing a clockwise loop. The data have been folded over a variable period found with the algorithm described in Section 3.2.1. Inset is the folded lightcurve of the same data. *Bottom Left*: The hardness-intensity diagram of Class V observation 96420-01-25-05 orbit 0, an example of an anticlockwise loop. *Bottom Right*: The hardness-intensity diagram of Class V observation 96420-01-25-06 orbit 0, in which we were unable to ascertain the presence of a loop.

Figure 4.15: Plots of the Class VI observation 96420-01-09-00, orbit 0. *Top-left*: 750 s lightcurve binned on 2 seconds to show lightcurve evolution. *Top-right*: Fourier Power Density Spectrum. *Bottom*: Lightcurve binned on 1 second.

Aside from the dips, Class VI observations show other structures in their lightcurves. Large fluctuations in count rate, by factors of $\lesssim 3$, occur on timescales of $\sim$ 1–5 s; no periodicity in these oscillations could be found. This behaviour is reflected in the PDS, which shows high-amplitude broad band noise below $\sim$ 0.5Hz with RMS-normalized power (Belloni and Hasinger, 1990) of up to $\sim 1.1\mathrm{Hz}^{-1}$. As can be seen in Figure 4.15, this feature takes the form of a broad shoulder of noise which shows a either weak peak or no clear peak at all. The $\sim$ 5Hz QPO seen in the PDS of other classes is not present in Class VI observations.

We attempted to fold all individual Class VI lightcurves, ignoring the sections of data corresponding to the large count rate dips described above. In general, folding lightcurves belonging to this class is difficult; many orbits showed low-amplitude oscillations which were difficult to fold using our flare-finding algorithm (see Section 3.2.1), while many others only showed oscillatory behaviour for a small number of periods between each pair of dips. As such, we only succesfully folded 23 of the 40 Class VI orbits. Of these, 19 showed clockwise loops in the HID_1 (top panel, Figure 4.16), 3 showed anticlockwise loops (bottom-left panel, Figure 4.16). In the remaining 1 observation, the data did not allow us to ascertain the presence of loops (bottom-right panel, Figure 4.16).

Like in Class VI, we note that the clockwise loops in Class VI appear more complex than clockwise loops. Again, the clockwise loop shown in Figure 4.16 appears to have a 2-lobe structure; this is repeated in all clockwise loops found in this class.

Class VII – Figure 4.17

Class VII shows high-amplitude flaring behaviour with a peak-to-peak recurrence time of 6–12 s. In Figure 4.18 we show a dynamical Lomb-Scargle spectrogram of a Class VII observation, showing that the fast flaring behaviour has a frequency which moves substantially over time. This in turn accounts for the large spread in the value of the flare peak-to-peak recurrence time.

In Figure 4.18 we show that the peak frequency of the QPO also varies in a structured way. We also suggest that the variabilitity of the frequency is itself a QPO with a period of $\sim$ 150.

At higher frequencies, the PDS shows a weak QPOs centred at $\sim$ 8Hz, with a q-values of $\sim$ 2.

We used our flare-finding algorithm (see Section 3.2.1) to perform variable-frequency folding of Class VII orbits. We find clockwise loops in 9 out of 11 Class VII orbits. In the remaining two observations, the oscillations were extremely fast. As a result, the errors in the HID_1 of these too observations were too large to succesfully select peaks, and we are unable to confirm or reject the presence of loops.

Figure 4.16: *Top*: The hardness-intensity diagram (HID_1) of the Class VI observation 96420-01-30-03, orbit 0 showing a clockwise loop. The data have been folded over a variable period found with the algorithm described in Section 3.2.1. Inset is the folded lightcurve of the same data. *Bottom Left*: The hardness-intensity diagram of Class VI observation 96420-01-30-04 orbit 0, an example of an anticlockwise loop. *Bottom Right*: The hardness-intensity diagram of Class VI observation 96420-01-09-03 orbit 0, in which we were unable to ascertain the presence of a loop.

Figure 4.17: Plots of the Class VII observation 96420-01-18-05, orbit 0. *Top-left*: 750 s lightcurve binned on 2 seconds to show lightcurve evolution. *Top-right*: Fourier Power Density Spectrum. *Bottom*: Lightcurve binned on 0.5 seconds.

Figure 4.18: A sliding window Lomb-Scargle spectrogram of Class VII observation 96420-01-18-05, showing power density spectra from an overlapping 32 s window moved 1 s at a time. The peak frequency of this low frequency QPO itself appears to oscillate with a frequency of ~ 5mHz.

Figure 4.19: Plots of the Class VIII observation 96420-01-19-03, orbit 0. *Top-left*: 300 s lightcurve binned on 2 seconds to show lightcurve evolution. *Top-right*: Fourier Power Density Spectrum. *Bottom*: Lightcurve binned on 0.5 seconds. Inset is a zoom of the 25 s portion of the lightcurve highlighted in cyan, to show the second-scale structure in the lightcurve.

Figure 4.20: Plots of the Class IX observation 96420-01-35-02, orbit 1. *Top-left*: 1200 s lightcurve binned on 2 seconds to show lightcurve evolution. *Top-right*: Fourier Power Density Spectrum. *Bottom*: Lightcurve binned on 2 seconds.

Class VIII – Figure 4.19

The lightcurve of this variability class shows the dipping behaviour seen in Class VI, as can be seen in Figure 4.19 at $t \approx 125$–150 s. The dips are less frequent than in Class VI. The behaviour outside of the dips is dominated by highly structured high-amplitude oscillations consisting of flares with a peak to peak separation of 3.4 ± 1.0 s. The PDS shows this behaviour as a very significant (q-value > 20) QPO; two harmonics of this QPO are also visible. The PDS also shows a strong (q-value = 4.7) QPO at ~ 9Hz.

We attempted to fold Class VIII lightcurves, ignoring the portions of data corresponding to dips, using our flare-finding algorithm. The high frequency of the dominant oscillation in Class VIII resulted in large errors in the peak times of individual flares, which translated to large errors in all HID_1s; however, we were able to ascertain the presence in loops in 8 out of 16 orbits. All 8 of these loops are clockwise.

Class IX – Figure 4.20

The 1 s lightcurve of a Class IX observation is superficially similar to the lightcurve of a Class I observation, with little obvious structured variability at timescales larger than 2 s;

however, large count rate dips like those seen in Classes VI and VIII (e.g. the feature at $t \approx 410$ s in the lightcurve of Figure 4.20) are very occasionally observed. These dips may in turn be coupled to short second-scale flares in which count rate briefly increases by a factor of 2–3.

Outside of these dips and flares, the lightcurve of a Class IX observation is indistinguishable from the lightcurve of a Class I or Class II observation. However, in Figure 4.3, we show that Class IX occupies a very different part of the global H_{A2}/H_{A1} colour-colour diagram. Class IX observations show a significantly larger H_{A2} than Class I and II observations, but a significantly lower H_{A1}.

The PDS reveals significant broad band noise peaked at ~0.3 Hz, and the ~ 5Hz QPO seen in other classes is absent. Altamirano and Belloni (2012) discovered high frequency (~ 66Hz) QPOs in observations corresponding to this variability class.

4.3.3 Swift

Observations with *Swift* took place throughout the 2011-2013 outburst of IGR J17091-3624. Between MJDs 55622 and 55880, 17 *Swift/XRT* were at least partly simultaneous with an *RXTE* observation, corresponding to at least one observation of all 9 classes. In each case, the *Swift* and *RXTE* lightcurves were similar. The remainder of the *Swift/XRT* observations during this time were also consistent with belonging to one of our nine classes. Given that the *RXTE* data have higher count rate and time resolution, we do not further discuss the *Swift* observations taken before MJD 55880. A more detailed comparison of *RXTE* and *Swift* data is beyond the scope of this paper.

Between MJD 55952 and 56445, *Swift* observations showed IGR J17091-3624 decreasing in flux. For all observations longer than 500 s, we rebinned the lightcurves to 10 s and calculated the RMS. We find the lower and upper quartiles of the fractional RMS in these measurements to be 18.3% and 21.7% respectively. *INTEGRAL* observations taken as part of a scan programme of the Galactic Plane (Fiocchi et al., 2012) and reported by Drave et al. (2012) suggest that IGR J17091-3624 returned to the hard state between MJDs 55952 and 55989. Therefore these observations sample IGR J17091-3624 the hard state.

4.3.4 INTEGRAL

The results of the *INTEGRAL*/IBIS analysis are presented in Table 4.4. We see clear detections of IGR J17091-3624 in all energy bands during the hardest period (MJD 55575–55625) of the 2011–2013 outburst. Conversion from detected counts to flux was achieved using an *INTEGRAL*/IBIS observation of the Crab taken between MJD 57305.334 and 57305.894. Conversion from Crab units to standard flux units was obtained by conversion factors listed in Bird et al. (2016) and Bazzano et al. (2006).

Energy (keV)	Intensity (cts/s)	Significance σ	Exposure (ks)	Flux (mCrab)	Flux (10^{-10}ergs s^{-1} cm^{-2})
20–40	12.39±0.05	247	115	93.5±0.38	7.08±0.03
40–100	7.06±0.05	157	163	83.5±0.60	7.87±0.06
100–150	1.05±0.03	40	173	66.9±1.91	2.14±0.06
150–300	0.23±0.03	7.6	179	46.6±5.96	2.24±0.29

Table 4.4: Results from the IBIS/ISGRI analysis of the 2011–2013 Outburst of IGR J17091. The 20–40 keV flux is given in units of mCrab and (10^{-11} ergs s^{-1} cm^{-2}). Conversion between counts and mCrab was obtained using an observation of the Crab taken during Revolution 1597 between MJD 57305.334 and 57305.894 and the conversion factors of Bird et al. (2016) and Bazzano et al. (2006).

Comparing these results with those of Bazzano et al. (2006), we see that IGR J17091 is detected for the first time above 150 keV with a detection significance of 7.6 σ, corresponding to a flux of $2.24 \pm 0.29 \times 10^{-10}$ ergs s^{-1} cm^{-2} (Figure 4.21).

4.3.5 Chandra

In Figure 4.22, we present lightcurves from the three *Chandra* observations considered in this paper (see also Table 5.1 for details of these observations).

Observation 12505 was performed within 24 hours of *RXTE* observation 96420-01-02-01, which showed Class I variability. No structured variability is seen in the lightcurve of OBSID 12505 (Figure 4.22, upper panel), which is consistent with Class I. Note that we consider the energy range 0.06-10 keV for this observation but 0.5-10 keV for observations 12405 and 12406.

Observation 12405 was performed within 24 hours of *RXTE* observation 96420-01-23-03, which showed Class V variability. The two observations were not simultaneous; OBSID 12405 began ~ 8.4 ks after OBSID 96420-01-2303 finished. The lightcurve of *Chandra* OBSID 12405 (shown in Figure 4.22, middle panel) shows a mean count rate of 41 cts s^{-1}. The lightcurve shows fast flaring behaviour (with a recurrence time on the order of 10s of seconds) in which the frequency changes widely on timescales of ~ 1000 s. This observation strongly resembles a Class VII lightcurve, but with its characteristic timescales increased by a factor of ~ 4. This leads to the possibility that the low number of Class VII *RXTE* observations we identify is due to a selection effect; we would not have been able to see this observation's long-term Class VII-like behaviour if the observation had been shorter than ~ 2 ks.

Observation 12406 was performed within 24 hours of *RXTE* observation 96420-01-32-06, which showed Class IX variability. The lightcurve presented for *Chandra* OBSID 12406 shows a mean count rate (36 cts s^{-1}), which is consistent with IGR J17091 being harder in this observation than in Observation 12505. This, combined with the lack of variability seen in its lightcurve, suggests that Observation 12505 is consistent with Class IX.

Figure 4.21: *INTEGRAL*/ISGRI 150–300 keV significance map of a 2° region centred on the position of IGR J17091-3624, showing the first significant detection of this source above 150 keV. The detection significance is 7.6 σ.

Figure 4.22: 1 ks segments of lightcurves taken from *Chandra* observations 12505, 12405 and 12406, showing Class I, Class VII and Class IX variability respectively. The lightcurve presented for observation 12505 is for the energy range 0.06-10 keV, while the other two lightcurves are for the energy range 0.5-10 keV. All three lightcurves are binned to 0.5 s.

Figure 4.23: Lightcurves of *XMM-Newton* observations 0677980201 and 0700381301, showing Class IV variability and the hard state respectively. Both lightcurves binned to 2 s. Data for observation 0677980201 is taken from *EPIC-MOS2* and data for observation 0700381301 is taken from *EPIC-pn*.

4.3.6 XMM-Newton

In Figure 4.23 we show lightcurves from two *XMM-Newton* observations. The lightcurve of *XMM-Newton* observation 0677980201, shown in the upper panel of Figure 4.23, shows the regular flares characteristic of Class IV variability. A simultaneous *RXTE* observation (OBSID 96420-01-05-000) also showed Class IV variability.

XMM-Newton observation 070038130, shown in the lower panel of Figure 4.23, was made after the end of *RXTE* observations IGR J17091-3624. As such it cannot be compared with contemporaneous *RXTE* data. The 5 s binned lightcurve shows no apparent variability, but a Fourier PDS of the observation (shown in Figure 4.24) reveals a QPO centred at around ~ 0.15Hz and a broad band noise component at lower frequencies. Drave et al. (2012) reported that IGR J17091 transited to the hard state in February 2012, seven months before this observation was taken. As such, we find that observation 0677980201 samples the hard state in IGR J17091 and is thus beyond the scope of our set of variability classes.

Figure 4.24: $\nu P(\nu)$-normalised co-added power density spectra of *XMM-Newton* observation 0700381301 and *Suzaku* observation 407037010. Both observations were taken simultaneously on September 29 2012 (MJD 56199). We sample observation 0700381301 up to a frequency of 10 Hz, while the 2 s time resolution of observation 407037010 results in a Nyquist frequency of 0.25 Hz.

4.3.7 *Suzaku*

The two *Suzaku* observations of IGR J17091-3624 considered, OBSIDs 407037010 and 407037020, were performed during the 2nd and 3rd re-flares of the hard state phase of the 2011–2013 outburst. OBSID 407037010 was taken simultaneously with *XMM-Newton* observation 0700381301. The XIS 0 count rates are 7.8 cts s^{-1} and 2.5 cts s^{-1} respectively.

Neither lightcurve shows 'heartbeats' or any other type of GRS 1915-like variability. However, we find evidence of a low frequency QPO feature at ~0.15 Hz in the OBSID 407037010; this QPO is also seen in *XMM-Newton* observation 0700381301 (Figure 4.24). The presence of a QPO below 1 Hz and flat-topped power density spectrum confirm that IGR J17091 was in the hard state at this time.

4.4 Discussion

Using observations from *XMM-Newton*, *RXTE* and *Chandra*, we describe the complex variability seen in IGR J17091 as a set of nine variability 'classes', labelled I to IX. These classes are distinguished from each other by values of upper and lower quartile (i.e. 25th and 75th percentile) count rates, mean RMS, the presence of QPOs in Fourier PDS, the shape of flare and dip features in the lightcurve and the presence of loops in the 6–16/2–6 keV hardness-intensity diagram HID_1. See Section 4.3 for a full description of these classes.

Figure 4.25: A lightcurve of observation 96420-01-06-02, orbit 0, showing a transition in behaviour between Class IV (in cyan, see Section 4.3.2) and Class V (in red, see Section 4.3.2).

The classification of some observations is clearer than others. Some orbits were too short to definitively quantify the behaviour of the source, whereas some other orbits contain a transition between two classes. An example lightcurve showing a transition from Class III to Class IV is presented in Figure 4.25.

Our set of classes is analogous to, but not based upon, the set of variability classes defined by Belloni et al. 2000 to describe the behaviour of the similarly complex LMXB GRS 1915. This ensures that our set of classes is not biased by an *a priori* assumption that the two objects are similar. However if we do assume that wide range of variability seen in these two objects are driven by the same physical processes, a direct comparison between the variability classes in the two systems can further our understanding of the physics that drive these exotic objects.

We also use all 2011-2013 IGR J17091-3624 data from *RXTE*, *XMM-Newton*, *Chandra*, *Swift*, *INTEGRAL* and *Suzaku* to analyse the long-term evolution of the 2011–2013 outburst. This in turn corresponds to all available X-ray data taken during this outburst.

Table 4.5: The nine variability classes of IGR J17091-3624, showing the name of the closest corresponding variability class in GRS 1915+105. The names of GRS 1915+105 classes are taken from Belloni et al. (2000), where more detailed descriptions can be found. Eight additional classes of GRS 1915+105 have been described; we do not find analogies to these classes in IGR J17091-3624.

IGR J17091-3624 Class	GRS 1915+105 Class
I	χ
II	ϕ
III	ν
IV	ρ
V	μ
VI	λ
VII	*None*
VIII	*None*
IX	γ

4.4.1 Variability Classes: IGR J17091 vs. GRS 1915

As observations of IGR J17091 and GRS 1915 suffer from different values of interstellar absorption N_H, we cannot directly compare the absolute colours of these two objects. However, we can compare the evolution of colour both over time and as a function of count rate. We therefore use these parameters, along with power spectra and lightcurve morphology, when comparing GRS 1915 with IGR J17091.

For seven of our classes, we were able to assign the closest matching class described by Belloni et al. 2000 for GRS 1915 (see Table 4.5). We are unable to find analogues to our classes VII and VIII in observations of GRS 1915, and we suggest that these classes are unique to IGR J17091.

Below, we evaluate our mapping between GRS 1915 and IGR J17091 classes, and interpret the differences between each matched pair.

Classes I and II – Figures 4.4, 4.5

Classes I and II both show low count rates and little structure in their lightcurves. The two classes in GRS 1915 that also show this lightcurve behaviour are Class χ[8] and Class ϕ. Belloni et al. 2000 differentiate between Classes ϕ and χ based on the hard colour (corresponding to C_2), as Class χ has a significantly higher value for this colour than Class ϕ.

Data from *RXTE* indicates that the transition from the hard state to the soft intermediate state between MJDs 55612 and 55615 (Drave et al., 2012). This was confirmed by a radio spectrum taken on MJD 55623 which was consistent with an observation of discrete ejecta

[8]Note that, in GRS 1915+105, Class χ is further subdivided into four classes based on hard colour (Belloni et al., 2000; Pahari et al., 2013a). As we cannot obtain hard colour for IGR J17091, we treat χ as a single variability class here.

(Rodriguez et al., 2011a). This observation of discrete ejecta at the transition between the hard state and the intermediate state has been reported in other LMXBS (e.g. XTE J1550-564, Rodriguez et al., 2003), and has also been associated with transitions to the χ Class in GRS 1915 (Rodriguez et al., 2008, see also review by Fender, 2006).

Using Fourier PDS, we conclude that Class I is analogous to Class χ in GRS 1915, while Class II is analogous to Class ϕ. In Class χ observations of GRS 1915, broad band noise between $\sim 1 - 10$Hz and a QPO at around 5Hz are seen in the PDS. We find that both of these are present in Class I observations of IGR J17091. On the other hand, we find that Class ϕ observations of GRS 1915 do not show this broad band noise, and show either a weak (q-value $\lesssim 3$) QPO at $\sim$ 5Hz or no QPO at all. We find that the weak QPO and lack of broad band noise are also seen in the PDS of Class II observations.

Classes III and IV – Figures 4.6, 4.9

Classes III and IV both show highly regular flaring activity in their lightcurves, but they differ in terms of timescale and pulse profile. As can be seen in lightcurves in Figure 4.9, flares in Class IV occur every $\sim$ 32 s and are nearly identical to each other in shape. On the other hand, as can be seen in Figure 4.6, flares in Class III occur every $\sim$ 61 s and may or may not end in a much faster sharp peak which is never seen in Class IV. In Figure 4.26 we show a two-dimensional histogram of flare peak count rate against flare duration, showing all flares in all observations classified as Class III or Class IV. In this figure, we can see that flares tend to group in one of two regions in count rate-duration space; a region between $\sim$ 90–110 cts s^{-1} PCU^{-1}and $\sim$ 35–55 s, corresponding to flares seen in Class III, and a region between $\sim$ 150–250 cts s^{-1} PCU^{-1}and $\sim$ 20–55 s, corresponding to flares seen in Class IV. From this plot, we conclude that the flares seen in Class III exist in a different population to the flares seen in Class IV.

The GRS 1915 classes that show behaviour most similar to these are ρ and ν; both produce similar structures in their lightcurve, but Class ν is differentiated from Class ρ by the presence of a secondary count rate peak which occurs $\sim$ 5 s after the primary (Belloni et al., 2000).

The secondary peak is present in most Class III observations and some Class IV observations (Figure 4.27), suggesting that both classes consist of a mix of ρ-like and ν-like observations. However, the poor statistics sometimes make the presence of this secondary peak difficult to detect. As such, we do not use the presence or absence of this peak as a criterion when assigning classes. Instead we choose to separate Classes III and IV based on the larger-scale structure in their lightcurves (see Section 4.3.2). Due to the aforementioned difference in burst populations between the two classes, we suggest that classes III and IV do represent two distinct classes rather than a single class with a period that drifts over time. We suggest that Classes ρ and ν in GRS 1915 could also be re-partitioned in this way.

However, HID$_1$ loops are found to generally execute in an anticlockwise direction in

Figure 4.26: Every flare in all observations identified as Class III or Class IV, plotted in a two-dimensional histogram of flare peak count rate against flare duration to show the two-population nature of these events. Flares belonging to Class IV occupy the distribution at higher peak rate and lower duration, whereas flares belonging to Class III occupy the distribution at lower peak rate and higher duration.

Figure 4.27: Lightcurve from Class III observation 96420-01-10-01 of IGR J17091-3624, with pairs of primary and secondary count rate spikes highlighted in cyan and red respectively. The yellow region highlights a primary count rate spike that did not produce a secondary.

Classes III and IV (previously noted by e.g. Altamirano et al., 2011b); the opposite direction to the clockwise loops in Classes ρ and ν reported by e.g. Belloni et al., 2000 and repeated by us using the same method we apply to data from IGR J17091-3624 (see Section 4.2). This suggests that Classes III and IV could be generated by a different physical mechanism to Classes ρ and ν. Alternatively, Classes III and IV could be generated by the same mechanism as ρ and ν if some other unknown process was able to alter the spectral evolution of flares in these classes.

Class V – Figure 4.11

The lightcurve of a Class V observation appears similar to that of a Class μ observation of GRS 1915, as both are characterised by rapid ρ-like flares which occur less regularly than in Class ρ. In addition to this, flares in Class μ fall into two clear populations, as do the flares in Class V. However, significant differences exist between Class V and Class μ. Class μ observations are characterised by long ($\sim$ 100 s) excursions to plateaus of high count rate, a behaviour which is not seen in any Class V observation thus far.

We note that the HID_1 in Class V observations displays a loop in the clockwise direction; the opposite direction to the looping seen in Classes III and IV but the same direction seen in Class μ.

Regarding the two-population nature of flares seen in this class (see Section 4.3.2), we suggest that V_2 flares may simply be two V_1 flares that occur close together in time, such that the second flare starts during the decay of the first flare. This would result in an apparent two-peaked flare structure, as we see in type V_2 flares. This interpretation also accounts for the bimodal distribution of flare duarations shown in the 2D histogram of Figure 4.13, as this could be caused by the misinterpretation of two-flare V_2 events as a single event. This also accounts for the Gaussian distribution of peak flare intensities seen in Figure 4.13), as the constituents of each V_2 event would be from the same population as V_1 flares.

Class VI – Figure 4.15

Class VI is dominated by long flaring periods which separate periods of low count rate, as can be seen in the lightcurve presented in Figure 4.15. Similar behaviour is seen in the lightcurves of observations of GRS 1915 belonging to Classes λ and ω (Klein-Wolt et al., 2002). However, the long count rate 'dips' are far less regular in Class VI than in Classes λ and ω, and we also note long periods of medium count rate during which neither flares nor dips occur. This variability class is noted by Pahari et al. (2012) who suggest that this class is unique to IGR J17091[9]. However, Pahari et al. (2013b) show that, in a plot of burst decay

[9]Pahari et al. (2012) refers to Class VI as Class C2.

time against burst rise time, Classes VI and λ fall in a straight line, suggesting a similar physical origin for both.

While it is cetainly true that Class VI is not a perfect analogue of either Class λ or Class ω, Class VI only differs noticeably from Class λ during the extended low-variability portions of its lightcurves. As such, we associate Class VI with Class λ.

Class VII – Figure 4.17

We are unable to find an analogue of Class VII in observations of GRS 1915. This class, and its apparent uniqueness, have previously been noted by Pahari et al., 2012[10]. Pahari et al. found that the C_2 hard colour in this class increases during count rate dips and decreases during count rate peaks. Here we reproduced the results of Pahari et al. and found that the anti-correlation between hard-colour and intensity is not physical, but due to the definition of C_2: the count rate in band L_C is approximately constant and consistent with background, and therefore $C_2 = L_C/L_A \propto L_A^{-1}$, which will naturally anticorrelate with intensity.

Although a correlation between QPO frequency and count rate has been noted in the $\sim$ 5Hz QPO seen in GRS 1915 (e.g. Markwardt et al., 1999; Vignarca et al., 2003), this QPO is also seen in Class VII observations at the same time as the $\sim$ 0.1Hz QPO. As such, the flux-frequency relationship in the very low frequency ($\sim$ 0.1Hz) QPO in Class VII is apparently unique amongst the classes of both IGR J17091 and GRS 1915.

Class VIII – Figure 4.19

We are unable to find an analogue of Class VIII in observations of GRS 1915. When it is flaring, the lightcurve waveform is similar to that seen in Class ρ, with rapid regular spikes in count rate. The lightcurve also shows irregular dips in count rate similar to those seen in Class VI and in Class λ in GRS 1915.

However, the amplitude of the flares in Class VIII is much larger, and the frequency much higher, than in Classes VI or λ. The amplitude of the flares in Class VIII can approach $\sim$ 350cts s^{-1} PCU^{-1}, while the flare separation time of 4–5 s makes Class VIII the fastest flaring activity seen in any class of IGR J17091 or GRS 1915. As such, we consider this variability class distinct from both Class VI and Class λ.

Class IX - Figure 4.20

Class IX is defined by long periods of high amplitude but unstructured variability (with a broad peaked noise component in the Fourier spectrum peaked at ~0.3 Hz) punctuated with infrequent irregular short-duration 'spikes' in which the count rate increases by a factor of

[10]Pahari et al. (2012) refers to Class VII as Class C1.

$\sim$ 2–3. A similarity between this Class and Class γ in GRS 1915 has been previously noted by Altamirano and Belloni (2012). However, the irregular spikes seen in some Class IX lightcurves are not reproduced in Class γ lightcurves of GRS 1915.

4.4.2 General Comparison with GRS 1915+105

Overall, variability in IGR J17091 tends to be faster than structurally similar variability in GRS 1915, as can be noted in Classes III and IV compared to Classes ρ and ν (see also Altamirano et al., 2011b). Additionally, IGR J17091 also displays highly structured variability unlike anything yet seen in GRS 1915, with classes VII and VIII in particular showing very fine detail in their lightcurves.

In total we find 2 variability classes which are seen in IGR J17091 but not in GRS 1915, compared with 8 that are seen in GRS 1915 but not in IGR J17091. As relatively little data exists on GRS 1915-like variability in IGR J17091, the presence of classes in GRS 1915 that are not seen in IGR J17091 could simply be an observational effect. It is unknown how long each variability class lasts for and, as such, additional variability classes could have occurred entirely while IGR J17091 was not being observed (however, see Huppenkothen et al., 2017 for a study on GRS1915 based on more than 16 years of data). However, GRS 1915 has displayed variability classes consistently since its discovery in 1992, implying that the two classes seen only in IGR J17091 are either completely absent in GRS 1915 or that they occur with a much lower probability. In either case, this implies physical differences between methods of generating GRS 1915-like variability in the two objects.

As noted in sections 4.4.1 to 4.4.1, variability classes seen in both IGR J17091 and GRS 1915 show differences in the different objects. In particular, we note the presence of irregular flares in Class IX which are not seen in the analogous Class γ. If these classes are indeed generated by the same processes in both objects, the differences between them must represent physical differences between the objects themselves.

It has previously been noted that, while the hardness ratios in IGR J17091 and GRS 1915 during ρ-like classes are different, the fractional hardening between the dip and peak of each flare is consistent with being the same in both objects (Capitanio et al., 2012). This suggests that the same physical process is behind the 'heartbeats' seen in both objects.

We note the presence of hysteretic HID_1 loops in some classes of both objects. Although these loops are always clockwise in GRS 1915, they can be executed in either direction in IGR J17091. Classes in IGR J17091 that show loops all have a preferred loop direction: anticlockwise in Classes III and IV and clockwise in classes V, VI, VII and VIII. In cases where the loop direction was opposite to that expected for a given class, loop detections were generally only marginally significant. In particular, we note that Classes IV and V tend to show loops in opposite directions, despite the similarities between their lightcurves and the ρ, ν and μ classes in GRS 1915. The fact that IGR J17091 can show HID_1 loops in both directions suggests that an increase in soft emission can either precede or lag a

correlated increase in hard emission from IGR J17091. Whether soft emission precedes or lags hard emission is in turn is dependent on the variability class.

There are also non-trivial similarities between variability in the two objects. We note the presence of a $\sim$ 5Hz QPO in many of the classes seen in IGR J17091, and this same 5Hz QPO is seen in lightcurves of GRS 1915. Similarly Altamirano and Belloni (2012) reported the discovery of a 66Hz QPO in IGR J17091; a very similar frequency to the 67Hz QPO observed in GRS 1915 (Morgan et al., 1997). It is not clear why these QPOs would exist at roughly the same frequencies in both objects when other variability in IGR J17091 tend to be faster.

4.4.3 Comparison with the Rapid Burster

In 2015, Bagnoli and in't Zand (2015) discovered the existence of two GRS 1915-like variability classes in the neutron star binary MXB 1730-335, also known as the 'Rapid Burster'. Specifically, Bagnoli and in't Zand (2015) note the presence of variability similar to Classes ρ and θ in GRS 1915.

Class θ-like variability, seen in *RXTE* observation 92026-01-20-02 of the Rapid Burster, is not closely matched by any of the classes we identify for IGR J17091. However, the lightcurves of a Class θ observation feature large dips in count rate similar to those seen in Classes VI and VIII in IGR J17091.

Conversely, Class ρ-like variability is seen in all three objects. Bagnoli and in't Zand (2015) note that the variability of the ρ-like flaring is slower in the Rapid Burster than in either GRS 1915 or IGR J17091. It has previously been suggested that the maximum rate of flaring in LMXBs should be inversely proportional to the mass of the central object (e.g. Belloni et al., 1997b; Frank et al., 2002). In this case, the fact that variability is faster in IGR J17091 than in GRS 1915 could simply be due to a lower black hole mass in the former object (Altamirano et al., 2011b). However if variability in the Rapid Burster is assumed to be physically analogous to variability in these two black hole objects, then we note that a correlation between central object mass and variability timescale no longer holds.

4.4.4 Comparison with Altamirano *et al.* 2011

Altamirano et al. (2011b) identify 5 GRS 1915 variability classes in a subset of observations from the 2011-2013 outburst of IGR J17091: six of these observations are presented in Table 4.6 along with the best-fit GRS 1915 class that we assign it in this paper (see also Table 4.5).

We acknowledge differences between the classifications assigned by this paper and by Altamirano et al. (2011b). We ascribe these differences to the different approaches we have used to construct our classes. In particular while we have constructed an independent set of

Table 4.6: The six OBSIDs explicitly classified in Altamirano et al. (2011b). We also present the GRS 1915 class with which we implicitly label each OBSID in this paper.

OBSID	Altamirano *et al.* Class	Court *et al.* Class (implied)
96420-01-04-03	α	ρ/ν
96420-01-05-00	ν	ρ/ν
96420-01-06-00	ρ	ρ/ν
96420-01-07-01	ρ	μ
96420-01-08-03	β/λ	λ
96420-01-09-06	μ	λ

variability classes for IGR J17091 which we have then compared to the Belloni et al. classes for GRS 1915, Altamirano et al. applied the Belloni et al. classes for GRS 1915 directly to IGR J17091.

In general, the variability classes we find to be present in IGR J17091 are broadly the same as those noted by Altamirano et al. (2011b). We do not associate any class with Class α in GRS 1915, but we find examples of all of the other variability classes posited by Altamirano et al. to exist in IGR J17091.

Altamirano et al., 2011b noted the presence of an anticlockwise loop in the HID of 'heartbeat'-like observations of IGR J17091, opposed to the clockwise loop seen in HID of ρ-class observations of GRS 1915. This is consistent with our finding that hysteretic loops in classes III and IV also tend to execute in an anticlockwise direction. However, we additionally find that hysteretic loops in classes V, VI, VII and VIII tend to execute in a clockwise direction. This is also different from GRS 1915, in which the loop is executed in the same direction in all classes. We also additionally report that clockwise loops tend to be more complex than anticlockwise loops seen in IGR J17091, with many showing a multi-lobed structure not seen in GRS 1915. This apparent inconsistency between the objects strengthens the suggestion in Altamirano et al., 2011b that the heartbeat-like classes in GRS 1915 and IGR J17091 may be generated by physically different mechanisms.

4.4.5 New Constraints on Accretion Rate, Mass & Distance

The constraints that Altamirano et al., 2011b placed on the mass and distance of IGR J17091 assumed that the object emitted at its Eddington luminosity at the peak of the 2011–2013 outburst. They report a peak 2–50 keV flux of 4×10^{-9} ergs s^{-1} cm^{-2} during flares in 'heartbeat'-like lightcurves during this time. The correction factor $C_{Bol,Peak}$ to convert 2–50 keV flux to bolometric flux is not well constrained, but Altamirano et al., 2011b suggest an order-of-magnitude estimate of $\lesssim 3$, corresponding to a peak bolometric flux of $\lesssim 1.2 \times 10^{-8}$ ergs s^{-1} cm^{-2}.

Maccarone, 2003 performed a study of the soft to hard transitions in 10 LMXBs. They found that all but one perform this transition at a luminosity consistent with between 1%

Figure 4.28: Mass of the compact object in IGR J17091-3624 plotted against its distance, for values of peak Eddington fractions of F_{Edd} =0.05, 0.1, 0.2 and 0.33.

and 4% of the Eddington limit. We use *Swift* observation 00031921058 taken on MJD 55965 to create a spectrum of IGR J17091 during the approximate time of its transition from a soft to a hard state (Drave et al., 2012). We fit this spectrum above 2 keV with a power-law, and extrapolate to find a 2–50 keV flux of 8.56×10^{-10} ergs s^{-1} cm^{-2}. Assuming that the transition bolometric correction factor $C_{Bol,Tran}$ is also $\lesssim 3$, this corresponds to a bolometric flux of $\lesssim 2.5 \times 10^{-9}$ ergs s^{-1} cm^{-2}.

By comparing this with the results of Maccarone, 2003 and Altamirano et al., 2011b, we find that IGR J17091-3624 was likely emitting at no more than ~ 5–20% of its Eddington Limit at its peak. This number becomes ~ 6–25% if we instead use $C_{Bol,Tran}$ = 2.4, or ~ 8–33% if $C_{Bol,Tran}$ = 1.8. With this new range of values, we are able to re-derive the compact object mass as the function of the distance (Figure 4.28). We find that for a black hole mass of ~ $10M_\odot$, as suggested by Iyer et al., 2015b, IGR J17091 is within the galaxy at a distance of 6–17 kpc. This is consistent with the estimated distance of ~ 11–17 kpc estimated by Rodriguez et al., 2011a for a compact object mass of $10M_\odot$.

4.4.6 Implications for Models of 'Heartbeat' Variability

We have found that hysteretic HID loops can execute in both directions in IGR J17091 (e.g. Section 4.4.4), as well as found a revised estimate that IGR J17091 accretes at $\lesssim 20\%$ Eddington (Section 4.4.5). Both of these findings have implications for physical models of GRS 1915-like variability in this source.

Firstly, we find that Eddington-limited accretion is neither necessary nor sufficient for GRS 1915-like variability. The discovery of GRS 1915-like variability in the sub-Eddington Rapid Burster (Bagnoli and in't Zand, 2015; Bagnoli et al., 2015) provided the first evidence that Eddington-limited accretion may not be a driving factor in this type of variability. We strengthen this case by finding that IGR J17091-3624 is also likely sub-Eddington. As such, we further rule out any scenario in which Eddington-limited accretion is required for GRS 1915-like variability in black hole LMXBs specifically.

Secondly, by using the direction of hysteretic HID loops, we find that hard photon lag in 'heartbeat'-like classes of IGR J17091 can be either positive or negative. This could mean that we must rule out the causal connection between soft and hard emission being common to all classes.

In either case, we find that scenarios that require high global accretion rates or predict a consistent hard photon lag (e.g. Neilsen et al., 2011; Janiuk and Czerny, 2005), are not able to explain GRS 1915-like variability in IGR J17091 unless they also feature geometric obscuration in a subset of variability classes. We note that simulations by Nayakshin et al., 2000 require an Eddington fraction of $\gtrsim 0.26$ before GRS 1915-like variability, a value which falls in the range ~ 0.05–0.33 that we find for the peak Eddington fraction of IGR J17091.

In addition to being near its Eddington limit GRS 1915 also has the largest orbit of any known LMXB (e.g. McClintock and Remillard, 2006). Sądowski, 2016 have also shown that thin, radiation dominated regions of disks in LMXBs require a large-scale threaded magnetic field to be stable, and the field strength required to stabilise such a disk in GRS 1915 is higher than for any other LMXB they studied. We suggest that one of these parameters is more likely to be the criterion for GRS 1915-like variability. If better constraints can be placed on the disk size and minimum stabilising field strength in IGR J17091, it will become clear whether either of these parameters can be the unifying factor behind LMXBs that display GRS 1915-like variability.

4.5 Conclusions

We have constructed the first model-independent set of variability classes for the entire portion of the 2011–2013 outburst of IGR J17091 that was observed with *RXTE*. We find that the data are well-described by a set of 9 classes; 7 of these appear to have direct

counterparts in GRS 1915, while two are, so far, unique to IGR J17091. We find that variability in IGR J17091 is generally faster than in the corresponding classes of GRS 1915, and that patterns of quasi-periodic flares and dips form the basis of most variability in both objects. Despite this, we find evidence that 'heartbeat'-like variability in both objects may be generated by different physical processes. In particular, while hard photons always lag soft in GRS 1915, we find evidence that hard photons can lag or precede soft photons in IGR J17091 depending on the variability class.

We also report on the long-term evolution of the 2011–2013 outburst of IGR J17091, in particular noting the presence of 3 re-flares during the later part of the outburst. Using an empirical relation between hard-soft transition luminosity and Eddington luminosity (Maccarone, 2003), we estimate that IGR J17091 was likely accreting at no greater than $\sim 33\%$ of its Eddington limit at peak luminosity.

We use these result to conclude that any model of GRS 1915-like variability which requires a near-Eddington global accretion rate is insufficient to explain the variability we see in IGR J17091. As such we suggest that an extreme value of some different parameter, such as disk size or minimum stabilising large-scale magnetic field, may be the unifying factor behind all objects which display GRS 1915-like variability. This would explain why sub-Eddington sources such as IGR J17091 and the Rapid Burster do display GRS 1915-like variability, while other Eddington-limited sources such as GX 17+2 and V404 Cyg do not.

Chapter 5

The Evolution of X-ray Bursts in the "Bursting Pulsar" GRO J1744–28

5.1 Introduction

Low Mass X-ray Binaries (hereafter LMXBs) are extremely dynamic astrophysical systems, which exhibit high-amplitude X-ray variability on timescales of milliseconds to years. In these systems a compact object accretes matter from a stellar companion, either via a stellar wind or via Roche-lobe overflow. The donated matter spirals in towards the compact object, forming an accretion disk of matter which heats up by friction to temperatures of $\gtrsim 1$ keV.

LMXBs are an excellent laboratory in which to explore the behaviour of matter under extreme physical conditions. In addition to extreme temperatures, the inner portion of an accretion disk is a region of extreme gravity, gas pressure and photon pressure. If the primary object in the binary is a neutron star, these systems also contain regions of extreme magnetic fields.

Many LMXBs containing a neutron star are known to exhibit 'bursts'; discrete periods of increased X-ray emission over timescales of seconds. These bursts are generally categorised as either Type I or Type II, depending on the profile of the burst and its spectral evolution (Hoffman et al., 1978; Lewin et al., 1993), and occur during outbursts; extended periods of increased X-ray binary luminosity which last weeks to months. Type I bursts are caused by accreted matter on the surface of the neutron star reaching a critical pressure and temperature which triggers runaway thermonuclear burning (see e.g. Lewin et al., 1993; Strohmayer and Bildsten, 2006). They appear in X-ray lightcurves as a sudden increase in intensity, followed by an exponential decay, over a timescale of a few $\sim$ 10s of seconds.

Type II bursts are believed to be caused by viscous instabilities in the accretion disk (Lewin et al., 1976b). However, the exact details of the mechanism responsible for Type II bursts remain unclear. This type of bursts is more varied in its phenomenological appearance, ranging from near-Gaussian in shape over timescales of ~ 10 s to broad flat-topped

lightcurve features which last for ~ 100 s (e.g. Bagnoli et al., 2015).

Type I X-ray bursts are seen in data from over a hundred neutron star LMXBs, while regular Type II bursts have only been unambiguously identified in two sources: the "Rapid Burster" MXB 1730-335 (Lewin et al., 1976b) and the "Bursting Pulsar" GRO J1744–28 (Kouveliotou et al., 1996). Isolated Type II bursts may have also been observed in at least one additional X-ray Binary (SMC X-1, Angelini et al., 1991), but the identification of these features remains unclear.

The Type II bursting behaviour in the Rapid Burster has been extensively studied (see e.g. Lewin et al., 1976b; Hoffman et al., 1978). Bagnoli et al. (2015) performed a full population study of all Type II bursts observed in this object by the *Rossi X-ray Timing Explorer* (*RXTE*, Bradt et al., 1993). Their results suggest that gating of the accretion by a strong magnetic field plays some role in the creation of Type II bursts. To further probe the physics behind Type II X-ray bursts, in this paper we perform a similar population study on bursts from the Bursting Pulsar.

The Bursting Pulsar (Paciesas et al., 1996) is a system containing a neutron star and a G or K class evolved companion star (e.g. Sturner and Dermer, 1996; Gosling et al., 2007; Masetti et al., 2014). The system lies at a distance of ~ 4–8 kpc in the direction of the Galactic centre (e.g. Kouveliotou et al., 1996; Gosling et al., 2007; Sanna et al., 2017c), and it is the only known pulsar that regularly displays Type II bursts. Unlike in the Rapid Burster, unambiguous Type I bursts have never been observed from the Bursting Pulsar (e.g. Giles et al., 1996, however see also Lamb et al., 1996; Doroshenko et al., 2015). Type II bursts were first identified upon discovery in 1995 by the Burst and Transient Source Experiment (BATSE) aboard the *Compton Gamma Ray Observatory* (*CGRO*, Gehrels et al., 1994). Additional outbursts have occurred irregularly; specifically in 1997 and 2014 (Woods et al., 1999; Kennea et al., 2014). An additional outburst may have occurred in 2017 (Sanna et al., 2017b), but it was significantly less luminous than previous outbursts and the Bursting Pulsar did not transition to the soft state (such events are referred to as 'failed outbursts' or 'failed state-transition outbursts', see e.g. Sturner and Shrader, 2005).

Previous work by Giles et al. (1996) indicated that Type II bursts in the 1995–1996 outburst of the Bursting Pulsar could be separated into a number of distinct populations based on peak flux. This is a notable difference from the Rapid Burster, in which all Type II bursts have peak fluxes approximately equal to or less than object's Eddington Luminosity (Tan et al., 1991). In this paper we expand on the work of Giles et al. (1996) and analyze *RXTE*, *NuSTAR*, *Chandra*, *XMM-Newton*, *Swift* and *INTEGRAL* data to fully quantify the population of Type II bursts in the Bursting Pulsar during all outbursts in which they were observed. We study how the bursting in this object evolves over time throughout each outburst, and we link this behaviour to the long-term evolution of the source. We also perform basic timing, morphology and spectral analysis on bursts, to try and understand the physical processes behind these phenomena.

5.2 Data and Data Analysis

Since discovery, the Bursting Pulsar has undergone three bright outbursts, which began in 1995, 1997 and 2014. We refer to these outbursts as Outbursts 1, 2 and 3. We do not consider the outburst in 2017 in this paper, as no Type II bursts were observed during this time, nor do we analyse data taken while the source was in quiesence. See Daigne et al. (2002), Wijnands and Wang (2002) and Degenaar et al. (2012) for studies of the Bursting Pulsar during quiescence.

We analysed data from all X-ray instruments which observed the Bursting Pulsar during these outbursts. Specifically, we analysed lightcurves, the evolution of hardness ratios as a function of time and of count rate, and performed statistical analysis of properties associated with each individual burst.

5.2.1 *RXTE*

We analysed data from the Proportional Counter Array (PCA, Jahoda et al., 1996) aboard *RXTE* corresponding to the Outbursts 1 & 2 of the Bursting Pulsar. This in turn corresponded to observation IDs starting with 10401-01, 20077-01, 20078-01, 20401-01 and 30075-01, between MJDs 50117 and 51225. This resulted in a total of 743 ks of data over 300 observations, which we have listed in Appendix B. Lightcurve data were extracted from `fits` files using `FTOOLS`[1]. Errors were calculated and quoted at the 1 σ level.

We also use data from the All-Sky Monitor (ASM, Levine et al., 1996) to monitor the long-term evolution of the source. ASM data was taken from MIT's ASM Light Curves Overview website[2].

Long-Term Evolution

To analyse the long-term evolution of the source during its outbursts, we extracted 2–16 keV count rates from the *Standard2* data in each observation. Following Altamirano et al. (2008b), we normalised the intensity estimated in each observation by the intensity of the Crab nebula, using the Crab observation that is the closest in time but within the same PCA gain epoch as the observation in question (see Jahoda et al., 2006).

Burst Identification and Analysis

To perform population studies on the Type II bursts in the Bursting Pulsar, we first extracted lightcurves from the `Standard1` data in each observation, as this data is available for all

[1] `https://heasarc.gsfc.nasa.gov/ftools/ftools_menu.html`
[2] `http://xte.mit.edu/ASM_lc.html`

RXTE observations. We used our own software[3] to search these lightcurves and return a list of individual bursts, using the algorithm described in Appendix A of Court et al. (2017). We manually cleaned spurious detections from our sample. We defined a 'burst' as an event that lasted at least 3 seconds during which the 1 s binned count rate exceeded 3 standard deviations above the persistent emission level and reached a maximum of at least five standard devations above the persistent emission level. We did not subtract background, as all count rate-related parameters we analyse are persistent emission subtracted, automatically removing background contribution.

During the analysis, we discovered a number of different "classes", similar to the multiple classes of burst described by Giles et al. (1996). Our classes varied significantly in terms of overall structure, and as such needed to be treated separately; we show representative lightcurves from our classes in Figure 5.2. These classes were separated from one another by a number of criteria including peak count rate and recurrence time (the time between peaks of consecutive bursts).

The vast majority of detected bursts resembled the Type II seen in the Rapid Burster (referred to as 'Normal Bursts' in Section 5.3) in terms of shape, duration and amplitude. We rebinned the data corresponding to these Normal Bursts to 0.5 s. We sampled the persistent emission before the burst, and defined the start of the burst as the first point at which count rate exceeded 5 standard deviations above the persistent emission before the burst. The end of the burst was defined similarly, but instead sampling the persistent emission after the burst; by doing this, we avoid making the implicit assumption that the persistent emission is equal before and after the burst. We fitted phenomenologically-motivated lightcurve models to each of these bursts (described in detail in Section 5.3.3), and used these fits to extract a number of parameters which characterise the shape and energetics of a burst (such as burst duration, total photon counts associated with a burst and persistent emission count rate).

Due to the very high peak count rates of Normal Bursts, data were substantially affected by dead-time (compare e.g. *GRANAT* data presented in Sazonov et al., 1997). These effects are complex and non-linear, and we did not correct for them here. The dead time corrections required for the count rates seen in other classes of burst are minimal, as they are orders of magnitude fainter (Giles et al., 1996). To test for correlations between parameters in a model-independent way, we used the Spearman's Rank correlation coefficient (as available in `Scipy`, Jones et al., 2001). This metric only tests the hypothesis that an increase in the value of one parameter is likely to correspond to an increase in the value of another parameter, and it is not affected by the shape of the monotonic correlation to be measured. Although dead-time effects lead to artificially low count rates being reported, a higher intensity still corresponds to a higher reported count rate. As such, using this correlation coefficient removed the effects of dead time on our detection of any correlations.

To calculate the distribution of recurrence times between consecutive bursts, we considered

[3] `https://github.com/jmcourt/pantheon`, (Court, 2017)

observations containing multiple bursts. If fewer than 25 s of data gap exists between a pair of bursts, we considered them to be consecutive and added their recurrence time to the distribution. We choose this maximum gap size as this is approximately the timescale over which a Normal Burst occurs.

To perform basic phenomelogical analysis of the spectral behaviour of these bursts, we divided our data into two energy bands when `SB_62us_0_23_500ms` and `SB_62us_24_249_500ms` mode data were available: A (PCA channels 0–23, corresponding to ~ 2–7 keV[4]) and B (channels 24–249, corresponding to ~ 8–60 keV[5]). The evolution of colour (defined as the ratio of the count rates in B and A) throughout a burst could then be studied. Due to the very high count rates during Normal Bursts, we did not correct for background. During fainter types of burst we estimate the background in different energy bands by subtracting count rates from *RXTE* observation 30075-01-26-00 of this region, when the source was inactive. Unlike using the *RXTE* background model, this method subtracts the contributions from other sources in the field. However, as it is unclear whether any of the rest of these sources are variable, the absolute values of colours we quote should be treated with caution. We created hardness-intensity diagrams to search for evidence of hysteretic loops in hardness-intensity space.

Following Bagnoli et al. (2015), we used the total number of persistent emission-subtracted counts as a proxy for fluence for all bursts other than Normal Bursts. As the contribution of the background does not change much during a single observation, this method also automatically subtracts background counts from our results.

Detecting Pulsations

The Bursting Pulsar is situated in a very dense region of the sky close to the Galactic centre, and so several additional objects also fall within the 1° *RXTE*/PCA field of view. Therefore it is important to confirm that the variability we observe in our data does in fact originate from the Bursting Pulsar.

To ascertain that all bursts considered in this study are from the Bursting Pulsar, we analyse the coherent X-ray pulse at the pulsar spin frequency to confirm that the source was active. We first corrected the photon time of arrivals (ToA) of the *RXTE* PCA dataset, to take into account the motion of the Earth-spacecraft system with respect to the Solar System barycentre, by using the `faxbary` tool (DE-405 Solar System ephemeris). To this end we adopted the source position obtained from a *Chandra* observation of the source during a quiescence state (RA=266.137875° DEC=-28.740833°, 1σ confidence radius of 0.8" Wijnands and Wang, 2002). To take into account the delays caused by the binary orbital motion, we then corrected the PCA ToAs through the recursive formula (assuming

[4]In *RXTE* gain epoch 1, corresponding to dates before MJD 50163. This corresponds to ~ 2–9 keV in epoch 2 (MJDs 50163–50188) and ~ 2–10 keV in epoch 3 (MJDs 50188–51259).

[5]In *RXTE* gain epoch 1. This corresponds to ~ 9–60 keV in epoch 2 and ~ 10–60 keV in epoch 3.

eccentricity $e \ll 1$):

$$t + x \sin\left(\frac{2\pi}{P_{orb}}(t - T_{NOD})\right) = t_{arr} \tag{5.1}$$

where t is the photon emission time, t_{arr} is the photon arrival time at the Solar System barycentre, x is the projected semi-major axis of the neutron star orbit in light seconds, P_{orb} is the orbital period and T_{NOD} is the time of passage through the ascending node. We corrected for the binary motion by using the orbital parameters reported by Finger et al. (1996a).

For each PCA observation we investigated the presence of the ~ 2.14 Hz coherent pulsation by performing an epoch-folding search of the data using 16 phase bin and starting with the spin frequency value $\nu = 2.141004$ Hz, corresponding to the spin frequency measured from the 1996 outburst of the source (Finger et al., 1996a), with a frequency step of 10^{-5} Hz for 10001 total steps. We detected X-ray coherent pulsations in all PCA observations performed during Outbursts 1 & 2. We then fitted each best folded profile with a sinusoid of unitary period in order to determine the corresponding sinusoidal amplitude. We selected only pulse profiles for which the ratio between the sinusoidal amplitude and its 1σ error was larger than 3. We corrected fractional amplitude values to take into account the PCA background contribution to the observed X-ray emission, that we estimated using the bright-background model suitable for the Bursting Pulsar count rate (see Jahoda et al., 2006).

Outbursts 1 and 2 occurred during different PCA gain epochs (see Jahoda et al., 2006), and we did not correct for this here. However, the reported 2–16 keV Crab count rate changes by no more than a factor 1.09 in any of the PCUs during the time period corresponding to these outbursts. Therefore the effects of the gain on our results are negligible.

5.2.2 *Swift*

In this study, we made use of data from the X-Ray Telescope (XRT, Burrows et al., 2003) and the Burst Alert Telescope (BAT, Krimm et al., 2013) aboard the Neil Gehrels Swift Observatory (*Swift*, Gehrels, 2004). We extracted a long-term 0.3–10 keV *Swift*/XRT lightcurve of Outburst 3 using the lightcurve generator provided by the UK Swift Science Data Centre (UKSSDC, Evans et al., 2007). We also make use of *Swift*/BAT lightcurves from the Swift/BAT Hard X-ray Transient website[6] (see Krimm et al., 2013).

5.2.3 *INTEGRAL*

We also made use of data from the Imager on Board *INTEGRAL* (Winkler et al., 2003). We extracted 17.3–80 keV IBIS/ISGRI lightcurves of the Rapid Burster during Outburst 3 using

[6] `https://swift.gsfc.nasa.gov/results/transients/`

OBSID	Exposure (ks)	MJD	Reference
16596	10	56719	Younes et al. (2015)
16605	35	56745	Degenaar et al. (2014a)
16606	35	56747	Degenaar et al. (2014a)

Table 5.1: Information on the three *Chandra* observations of the Bursting Pulsar during Outburst 3. All other observations of the Bursting Pulsar in the Chandra archive were obtained at times that the source was in quiescence.

the *INTEGRAL* Heavens portal. This is provided by the *INTEGRAL* Science Data Centre (Lubiński, 2009).

5.2.4 *Chandra*

The Bursting Pulsar was targeted with *Chandra* (Weisskopf, 1999) three times during Outburst 3 (Table 5.1). One of these observations (OBSID 16596) was taken simultaneously with a *NuSTAR* observation (80002017004). In all three observations data were obtained with the High Energy Transmission Grating (HETG), where the incoming light was dispersed onto the ACIS-S (Garmire et al., 2003) array. The ACIS-S was operated in continued clocking (CC) mode to minimize the effects of pile-up. The Chandra/HETG observations were analysed using standard tools available within `ciao` v. 4.5 (Fruscione et al., 2006). We extracted 1 s binned light curves from the `evt2` data using `dmextract`, where we combined the first order positive and negative grating data from both the Medium Energy Grating (MEG; 0.4-5 keV) and the High Energy Grating (HEG; 0.8–8 keV).

5.2.5 *XMM-Newton*

A single pointed *XMM-Newton* observation of the Bursting Pulsar was taken during Outburst 3 on MJD 56722 (OBSID 0729560401), and no observations were taken during Outbursts 1 or 2. We extracted a 0.5–10 keV lightcurve from EPIC-PN at 1 s resolution using `xmmsas` version 15.0.0. During this observation, EPIC-PN was operating in Fast Timing mode. We use EPIC-PN as the statistics are better than in MOS1 or MOS2.

5.2.6 *Suzaku*

Suzaku (Mitsuda et al., 2007) observed the Bursting Pulsar once during Outburst 3 on MJD 56740 (OBSID 908004010), and no observations were taken during Outbursts 1 or 2.

To create a lightcurve, we reprocessed and screened data from the X-ray Imaging Spectrometer (XIS, Koyama et al., 2007)using the `aepipeline` script and the latest calibration database released on June 7, 2016. The attitude correction for the thermal wobbling was made by `aeattcor2` and `xiscoord` (Uchiyama et al., 2008). The source was

OBSID	Exposure (ks)	MJD	Reference
80002017002	29	56703	D'Aì et al. (2016)
80002017004	9	56719	Younes et al. (2015)

Table 5.2: Information on the three *NuSTAR* observations of the Bursting Pulsar during Outburst 3.

extracted within a radius of 250 pixels corresponding to 260" from the image center. The background was extracted near the edge of the XIS chip, and subtracted from the source.

5.2.7 *NuSTAR*

NuSTAR (**?**) consists of two grazing-incident X-ray telescopes. These instruments only observed the Bursting Pulsar twice during its outbursts, both times in Outburst 3 (Table 5.2). We extracted lightcurves from both of these observations using `nupipeline` and `nuproducts`, following standard procedures[7].

5.3 Results

5.3.1 Outburst Evolution

We show the long-term monitoring lightcurves of Outbursts 1,2 and 3 in Figure 5.1, as well as mark the dates of pointed observations with various instruments.

The Bursting Pulsar was discovered already in outburst on December 12 1995 (Fishman et al., 1995); BATSE data suggest that this outburst began several days earlier on December 3 (Paciesas et al., 1996; Bildsten et al., 1997). The main outburst ended around May 10 1996 (Woods et al., 2000). We show the global lightcurve of this outburst in Figure 5.1, Panel 1. As *RXTE* did not observe the object before or during the peak of Outburst, we can only obtain a lower limit of ~ 1.75 Crab for the peak 2–16 keV flux.

There are atleast two major rebrightening events in the tail of Outburst 1, which can be seen clearly in Figure 5.1 centred at MJDs of ~ 50235 and ~ 50280. During these rebrightening events, the 2–16 keV flux peaked at ~ 0.10 and ~ 0.18 Crab respectively.

Outburst 2 began on December 1 1996 and ended around April 7 1997 (Woods et al., 1999). The 2–16 keV flux peaked at 1.02 Crab on MJD 50473; we show the global lightcurve of this outburst in Figure 5.1, Panel 2. Type II bursts are seen in *RXTE*/PCA lightcurves from Outburst 2 between MJDs 50466 and 50544. One rebrightening event occurred during the tail of Outburst 2, centred at an MJD of ~ 50615 with a peak 2–16 keV flux of ~ 54 mCrab. A second possible rebrightening event occurs at MJD 50975, with a peak 2–16 keV flux of 11 mCrab, but the cadence of *RXTE*/PCA observations was too low to unambiguously confirm the existence of a reflare at this time.

[7] See `https://www.cosmos.esa.int/web/xmm-newton/sas-threads`.

Figure 5.1: Comparisons of the three outbursts of the Bursting Pulsar reported on in this paper. Times corresponding to pointed observations with *Chandra*, *NuSTAR*, *Suzaku*, *Swift* and *XMM-Newton* are marked.

Outburst 3 began on January 31, 2014 (Negoro et al., 2014; Kennea et al., 2014) and ended around April 23 (e.g. D'Aì et al., 2015). The daily 0.3–10 keV Swift/XRT rate peaked at 81 cts s^{-1} on MJD 56729, corresponding to 0.4 Crab. We show the global lightcurve of this outburst in Figure 5.1, Panel 3.

During the main part of Outburst 3, *Swift*, *XMM-Newton* and *Suzaku* made one pointed observation each, *Chandra* made four observations, and *NuSTAR* made three observations. The *Chandra* observation on March 3 2014 was made simultaneously with one of the *NuSTAR* observations (see Younes et al., 2015). After the main part of the outburst, the source was not well-monitored, although it remained detectable by *Swift*/BAT, and it is unclear whether any rebrightening events occured. A single *NuSTAR* observation was made during the outburst tail on August 14 2014.

As can be seen in Figure 5.1, the main section of all three outburst follow a common profile, over a timescale of $\sim$ 150 days. A notable difference between outbursts 1 & 2 is the number of rebrightening events; while we find two rebrightening events associated with Outburst 1, we only find one associated with Outburst 2 unless we assume the event at MJD 50975 is associated with the outburst. Additionally, Outburst 2 was at least a factor $\sim$ 1.7 fainter at its peak than Outburst 1 (see also Woods et al., 1999), while Outburst 3 was a factor of $\gtrsim$ 4 fainter at peak than Outburst 1.

Pulsations

We found pulsations in PCA data throughout the entirety of Outbursts 1 & 2, confirming that all the types of Type II bursts we see are from the Bursting Pulsar. Previous studies have shown that pulsations were also present during Outburst 3 (e.g. Sanna et al., 2017c).

Bursting Behaviour

Bursts are seen in *RXTE*/PCA lightcurves from the start of the Outburst 1 (e.g. Kouveliotou et al., 1996). These Type II bursts occur until around MJD 50200, as the source flux falls below ~ 0.1 Crab in the 2–16 keV band.

During the latter part of the first rebrightening after Outburst 1, between MJDs 50238 and 50246, we found Type II-like bursts with amplitudes ~ 2 orders of magnitude smaller than those found during the main outburst event. These gradually increased in frequency throughout this period of time until evolving into a period of highly structured variability which persisted until MJD 50261.

In Outburst 2, we found Type II bursts occuring between MJDs ~ 50466 and 50542. Low-amplitude Type II-like bursts are seen during the latter stages of the main outburst, between MJDs 50562 and 50577. These again evolve into a period of highly structured variability; this persists until MJD 50618, just after the peak of the rebrightening event.

High-amplitude Type II bursts were also seen in Outburst 3 (e.g. Linares et al., 2014). As no soft ($\lesssim 10$ keV) X-ray instrument was monitoring the Bursting Pulsar during the latter part of Outburst 3, it is unknown whether this Outburst showed the lower-amplitude bursting behaviour seen at the end of Outbursts 1 & 2. Low amplitude bursting bhaviour is not seen in the pointed *NuStar* observation which was made during this time.

5.3.2 Categorizing Bursts

We find that bursts in the Bursting Pulsar fall into a number of discrete classes, lightcurves from which we show in Figure 5.2. These classes are as follows:

- Normal Bursts (Figure 5.2, Panel a): Type II bursts. These are the brightests bursts seen from this source, with peak count 1 s binned rates of ~ 10000 cts s^{-1} PCU^{-1}, and recurrence timescales of order ~ 1000 s. These bursts are roughly Gaussian in shape with durations of ~ 10 s, and are followed by a 'dip' in the persistent emission count rate with a duration of order 100 s (see also e.g. Giles et al., 1996).
- Minibursts (Figure 5.2, Panel b): faint bursts with 1 s-binned peak count rates of ~ 2 times the persistent emission count rate. Minibursts are variable, with duration timescales between ~ 5–50 s. These bursts are also sometimes followed by dips similar to those seen after Normal Bursts.

Figure 5.2: 2–49 keV lightcurves for the four classes of bursting behaviour identified in this paper: **a)** Normal Burst, **b)** Miniburst, **c)** Mesoburst, **d)** Structured Bursts. Note that Panel **d** is plotted with a different time scaling to the other panels so as to better show the behaviour of Structured Bursting. On all figures the median count rate, which we use as a proxy for the persistent emission, is plotted in cyan. Lightcurves **a**-**c** are binned to 0.125 s, while lightcurve **d** is binned to 1 s.

- Mesobursts (Figure 5.2, Panel c): Type II-like bursts. These bursts differ from Normal Bursts in that they do not show well-defined subsequent 'dips'. They are also fainter than Normal Bursts, with peak count 1 s binned count rates of $\sim 1000\,\mathrm{cts\,s^{-1}\,PCU^{-1}}$. Their burst profiles show fast rises on timescales of seconds, with slower decays and overall durations of ~ 50 s. The structure of the bursts is very non-Gaussian, appearing as a small forest of peaks in lightcurves.

- Structured Bursts (Figure 5.2, Panel d): the most complex class of bursting behaviour we observe from the Bursting Pulsar, consisting of patterns of flares and dips in the X-ray lightcurve. The amplitudes of individual flares are similar to those of the faintest Mesobursts. The recurrence timescale is of the order of the timescale of an individual flare, meaning that is it difficult to fully separate individual flares of this class.

In the upper panel Figure 5.3 we show a histogram of persistent-emission-subtracted peak count rates for all Normal and Mesobursts observed by *RXTE*. We split these two classes based on the bimodal distribution in peak count rate as well as the lack of dips in Mesobursts.

In the lower panel Figure 5.3, we show the histogram of peak count rates for all Normal and Miniburst observed by *RXTE* as a fraction of the persistent emission at that time. We split these two classes based on the strongly bimodal distribution in fractional amplitude.

We also find 6 bursts with fast (~ 1 s) rises and exponential decays that occur during the lowest flux regions of the outburst ($\lesssim 50$ mCrab). Strohmayer et al. (1997) and Galloway et al. (2008) have previously identified these bursts as being Type I X-ray bursts from another source in the *RXTE* field of view. To show that these unrelated Type I bursts would not be confused with Minibursts, we add examples of the Type I bursts to lightcurves from observations containing Minibursts. We find that the peak count rates in Type I bursts are roughly equal to the amplitude of the noise in the persistent flux in these observations, hence they would not be detected by our algorithms.

We show when in Outbursts 1 & 2 each type of burst was observed in Figures 5.4 and 5.5 respectively. Normal Bursts and Minibursts (red) occur during the same periods of time from around the peak of an outburst until the persistent emission falls beneath ~ 0.1 Crab; assuming an Eddington Limit of ~ 1 Crab (e.g Sazonov et al., 1997), this corresponds to an Eddington ratio of ~ 0.1. After this point, bursting is not observed for a few tens of days. Mesobursts (blue) begin at the end of a rebrightening event in Outburst 1 and during the final days of the main part of the outburst in Outburst 2. Structured Bursts (yellow) occur during the first part of a rebrightening event in both outbursts. Although there was a second rebrightening event after Outburst 1, neither Mesobursts nor Structured Bursts were observed at this time. Based on this separation, as well as differences in structure, we treat each class of burst separately below.

Figure 5.3: Upper Panel: A histogram of the peak 1 s binned peak count rates of the joint population of all Normal and Mesobursts seen by *RXTE*. The dashed line indicates the position of the threshold above which we consider a Type II-like burst to be a Normal Burst. The resultant split of the population into Normal and Mesobursts is indicated by blue and red shading respectively. **Upper Panel:** A histogram of the peak 1 s binned peak count rates of the joint population of all Normal and Minibursts seen by *RXTE*, divided by the persistent emission count rate at that time. The dashed line indicates the position of the threshold below which we consider a burst to be a Miniburst. The resultant split of the population into Normal and Minibursts is indicated by blue and green shading respectively. Note that the *x*-axis of both plots is logarithmic, and so number density is not preserved.

Figure 5.4: Central panel shows the global 2–16 keV *RXTE*/PCA lightcurve of the 1995–1996 outburst of the Bursting Pulsar, highlighting periods of time during which Mesobursts (blue) Structured Bursts (yellow) or Normal and Mini bursts (red) are observed. A single Mesoburst was also observed on MJD 50253, during the period of the outburst highlighted in yellow (see Figure 5.17). Other panels show example lightcurves which contain the aforementioned types of bursting behaviour. See section 5.3.2 for a detailed treatment of burst classification. Fluxes reported in units of Crab.

Figure 5.5: Central panel shows the global 2–16 keV *RXTE*/PCA lightcurve of the 1997–1999 outburst of the Bursting Pulsar, highlighting periods of time during which Mesobursts (blue) Structured Bursts (yellow) or Normal and Mini bursts (red) are observed. Other panels show example lightcurves which contain the aforementioned types of bursting behaviour.

Bursting Mode	Bursts	Total Exposure (ks)	Duration (d)
Normal Bursts	99	192	76
Miniburststs	48	192	76
Mesobursts	43	44	25
Structured Bursts	-	80	54

Table 5.3: Statistics on the population of bursts we use for this study, as well as the duration and integrated *RXTE*/PCA exposure time of each mode of bursting. All numbers are the sum of values for Outbursts 1 and 2. As Normal and Miniburststs happen during the same period of time in each outburst, the exposure time and mode duration for these classes of bursting are equal.

5.3.3 Normal Bursts

We define Normal Bursts as the set of all bursts with a persistent-emission-subtracted peak 1 s binned *RXTE*/PCA-equivalent count rate above 3000 cts s^{-1} PCU^{-1}. Normal Bursts account for 99 out of the 190[8] bursts identified for this study. They are observed during all three outbursts covered in this study. They occurred between MJDs 50117 and 50200 in Outburst 1, and between 50466 and 50542 in Outburst 2; during these intervals, *RXTE* observed the source for a total of 192 ks. See Table 5.3 to compare these with numbers for the other classes of burst identified in this study. They occur during the same time intervals during which Miniburststs are present. In both of these outbursts, the region of Normal and Miniburststs correspond to the time between the peak of the outburst and and the time that the persistent intensity falls below ~ 0.1 Crab.

Recurrence Time

Using Outburst 3 data from *Chandra*, *XMM-Newton*, *NuSTAR*[9] and *Suzaku*, we find minimum and maximum recurrence times of ~ 345 and ~ 5660 s respectively. We show the histogram of recurrence times from Outburst 3 in Figure 5.6, showing which parts of the distribution were observed with which observatory. Compared to data from *Chandra* and *XMM-Newton*, data from *Suzaku* generally suggests shorter recurrence times. This is likely due to *Suzaku* observations consisting of a number of ~ 2 ks windows; as this number is of the same order of magnitude as the recurrence time between bursts, there is a strong selection effect against high recurrence times in the *Suzaku* dataset.

From the *RXTE* data we find minimum and maximum burst recurrence times of ~ 250 and ~ 2510 s during Outburst 1, and minimum and maximum recurrence times of ~ 250 and ~ 2340 s during Outburst 2. As the length of an *RXTE* pointing ($\lesssim 3$ ks) is also of the same order of magnitude as the recurrence time between bursts, selection effects bias us against sampling pairs of bursts with longer recurrence times, and hence this upper value is likely an underestimate. We show the combined histogram of recurrence times from Outbursts 1

[8]This number does not include Structured Bursts as their complex structure makes them difficult to separate.

[9]To avoid double-counting peak pairs, we do not use *NuSTAR* observation 80002017004, which was taken simultaneously with *Chandra* observation 16596.

and 2 in the top panel of Figure 5.6.

To test whether consecutive bursts are independent events, we tested the hypothesis that bursts are randomly distributed in time in a Poisson distribution (Poisson, 1837). Assuming our hypothesis, as well as assuming that the frequency of Normal Bursts does not change during an outburst (e.g. Aptekar et al., 1998), we could concatenate different observations and the resultant distribution of burst times will still be Poissonian. For each of Outbursts 1 & 2, we concatenated all *RXTE* data during the Normal Bursting part of the outburst into a single lightcurve. We split our lightcurves into windows of length w and counted how many bursts were in each, forming a histogram of number of bursts per window. We fit this histogram with a Poisson probability density function, obtaining the value λ which is the mean number of bursts in a time w. λ/w is therefore an expression of the true burst frequency per unit time, and should be independent of our choice of w. We tried values of w between 100 and 10000 s for both outbursts, and found that in all cases λ/w depends strongly on w. Therefore our assumptions cannot both be valid, and we rejected the hypothesis that these bursts are from a Poisson distribution with constant λ. This in turn suggests at least one of the following must be correct:

1. The frequency of bursts was not constant throughout the outburst. Or:
2. The arrival time of a given burst depends on the arrival time of the preceding burst, and therefore bursts are not independent events.

Burst Structure

In the top panel of Figure 5.7 we show a plot of all Normal Bursts observed with *RXTE* overlayed on top of one another. We find that all Normal Bursts follow a similar burst profile with similar rise and decay timescales but varying peak intensities. In the lower panel of Figure 5.7 we show a plot of Normal Bursts overlaid on top of each other after being normalised by the persistent emission count rate in their respective observation. The bursts are even closer to following a single profile in this figure, suggesting a correlation between persistent emission level in an outburst and the individual fluence of its bursts.

The structure of the lightcurve of a Normal Burst can be described in three well-defined parts:

1. The main burst: roughly approximated by a skewed Gaussian.
2. A ‘plateau’: a period of time after the main burst during which count rate remains relatively stable at a level above the persistent rate.
3. A ‘dip’: a period during which the count rate falls below the persistent level, before exponentially decaying back up towards the persistent level (e.g. Younes et al., 2015).

The dip is present after every burst in our *RXTE* sample from Outbursts 1 & 2, whereas the plateau is only seen in 39 out of 99. We show example lightcurves of bursts with and

Figure 5.6: Top: The distribution of recurrence times between consecutive Normal Bursts seen in pointed *Chandra*, *XMM-Newton*, *NuSTAR* and *Suzaku* observations of Outburst 3 of the Bursting Pulsar. Distributions of bursts observed by different instruments are stacked on top of each other and colour coded. **Bottom:** The distribution of recurrence times between consecutive Normal Bursts seen in *RXTE* observations of Outbursts 1 & 2 of the Bursting Pulsar.

Figure 5.7: Top: a plot of every Normal Burst, centred by the time of its peak, overlaid on top of each other to show the existence of a common pulse profile. **Bottom:** a plot of every Normal Burst in which count rates have been normalised by the persistent emission count rate during the observation from which each burst was observed. As the bursts are on average closer to the average pulse profile in this metric, this suggests that the intensity of a burst is roughly dependent on the persistent emission rate. Some persistent emission-normalised count rates may be artificially low due to dead-time effects.

Figure 5.8: *RXTE* lightcurves of Normal Bursts with (top) and without (bottom) 'plateau' features, showing the burst structure in each case. The median count rate, which we use as a proxy for the persistent emission, is plotted in cyan to highlight the presence of the count rate 'dip' after each burst.

without plateaus in Figure 5.8, which also show that the dip is present in both cases.

In order to study Normal Bursts, we fit the burst profiles with phenomenologically-motivated mathematical functions. In Figure 5.9 we show a schematic plot of our model, as well as annotations explaining the identities of the various parameters we use. We fit the main burst with a skewed Gaussian, centred at $t = x_0$ with amplitude a_b, standard deviation σ_B and skewness c, added to the persistent emission rate k. We fit the 'dip' with the continuous piecewise function $f(t)$:

$$f(t) = \begin{cases} k - \dfrac{a_d(t - t_0)}{d - t_0}, & \text{if } t \leq d \\ k - a_d \exp\left(\dfrac{d - t}{\lambda}\right), & \text{otherwise} \end{cases} \tag{5.2}$$

Where t is time, t_0 is the start time of the dip, a_d is the amplitude of the dip, d is the time at the local dip minimum and λ is the dip recovery timescale. This function is based on the finding by Younes et al. (2015) that dip count rates recover exponentially, but has the added advantage that the start of the recovery phase can also be fit as an independent parameter. Using this fit, we can estimate values for burst fluence ϕ_B, burst scale-length σ_B, 'missing'

Figure 5.9: A schematic explaining the origin of the 12 Normal Burst parameters used in this study, as well as showing the functional forms of both the skewed Gaussian fit to a burst and the 'dipper function' (Equation 5.2) fit to a dip. Note that we do not fit a function to the plateau, and we calculate its fluence by summing the persistent rate-subtracted counts. Diagram is for explanation only and the burst pictured is neither based on real data nor to scale.

dip fluence ϕ_D and dip scale-length λ and compare these with other burst parameters. When present, we also calculate the fluence of the plateau ϕ_p by summing the persistent emission-subtracted counts during the region between the end of the burst (as defined in Section 5.2.1) and the start of the dip. For each pair of parameters, we do not consider datapoints when the magnitude of the error on a parameter is greater than the value of the parameter.

We only extract these parameters from Normal Bursts observed by *RXTE* during Outbursts 1 & 2. This ensures that the resultant parameter distributions we extracted are not affected by differences between instruments.

Parameter Distributions

We extracted a total of ten parameters from our fit to each burst: the parameters a_d, d and λ of the fit to the dip, the missing fluence ϕ_D of the dip, the parameters a_b, σ_B and c of the skewed Gaussian fit to the main burst, the main burst fluence ϕ_B, the maximum persistent

	Outburst 1		Outburst 2		Outbursts 1&2	
	Mean	S.D.	Mean	S.D.	Mean	S.D.
ϕ_B	2.74e6	7.8e5	2.25e6	7.6e5	2.43e6	8.0e5
a_B	3.18e5	8.4e4	2.72e5	9.9e4	2.90e5	9.6e4
σ_B	3.39	0.35	3.42	0.59	3.41	0.52
c	2.68	1.9	2.79	2.0	2.75	2.0
ϕ_d	1.74e6	1.3e6	1.17e6	3.6e5	1.38e6	8.7e5
a_d	550	335	536	307	541	318
d	49	46	20	22	31	36
λ	294	176	229	124	254	150
ϕ_p	1.89e5	2.3e5	7577	5707	1.4e5	1.8e5
a_p	1289	1113	767	463	1063	928

Table 5.4: A table showing the mean and standard deviation of 10 Normal Burst parameters of *RXTE*-sampled bursts. In each case, we give the values for populations from only Outburst 1, from only Outburst 2 and from the combined population from both outbursts. Histograms for each parameter can be found in Appendix C.

emission-subtracted rate in the plateau a_p and the plateau fluence ϕ_P.

Using our *RXTE* sample of Normal Bursts, we can construct distributions for all of the burst parameters described in Section 5.3.3 for bursts in Outbursts 1 & 2. We give the mean and standard deviation for each parameter in each outburst in Table 5.4, and histograms for each can be found in Appendix C.

The mean value of most parameters differs by no more than $\sim 50\%$ between outbursts. Notable exceptions are d, ϕ_p and a_p, which are ~ 2.5, ~ 2.5 and ~ 1.7 times greater in Outburst 1 than in Outburst 2 respectively.

Correlations

In total, we extracted 12 parameters for each Normal Burst in our *RXTE* sample: the 10 burst parameters listed in Section 5.3.3, the recurrence time s_t until the next burst and the persistent emission rate k at the time of the burst.

As the amplitude of all 3 components in a burst scale with the persistent level, we rescaled our values of a_b, a_d, ϕ_B, ϕ_D and ϕ_P by a factor $\frac{1}{k}$. We show the covariance matrix with all 66 possible pairings of these normalised parameters in Figure 5.10 (we present the covariance matrix of these parameters before being rescaled in Appendix D). Using the Spearman's Rank Correlation Coefficient, we find the following $\geq 5\,\sigma$ correlations which are highlighted in Figure 5.10:

- Persistent emission k anticorrelates with normalised burst fluence ϕ_B/k ($> 10\,\sigma$) and normalised burst amplitude a_b/k ($> 10\,\sigma$).
- Normalised burst fluence ϕ_B/k correlates with normalised burst amplitude a_B/k ($8.0\,\sigma$).

Figure 5.10: Covariance Matrix with a scatter plot of each of the 66 pairings of the 12 Normal Burst parameters listed in section 5.3.3. Amplitudes and fluences have been normalised by dividing by the persistent count rate k. Pairings which show a correlation using the Spearman Rank metric with a significance $\geq 5\,\sigma$ are highlighted in red.

- Normalised dip fluence ϕ_d/k correlates with dip recovery timescale λ ($6.3\,\sigma$).

- Normalised dip amplitude a_d/k anticorrelates with dip falltime d ($5.7\,\sigma$) and dip recovery timescale λ ($7.1\,\sigma$).

- Normalised plateau fluence ϕ_p/k correlates with normalised plateau amplitude a_p ($6.4\,\sigma$).

As ϕ_B can be approximated to first order as a product of a_B and σ, the correlation between ϕ_B and a_B is expected as they are not independent parameters. Similarly, the correlations between ϕ_d & λ and ϕ_p and a_p are likely due to these pairs of parameters not being independent.

Figure 5.11: A 1 s-binned hardness-intensity diagram of a Normal Burst from *RXTE*/PCA observation 10401-01-08-00, with an inset 2–60 keV lightcurve. Significant colour evolution can be seen during the burst, taking the form of a loop.

Colour Evolution

To explore the spectral behaviour of Normal Bursts, we studied the evolution of the hardness (the ratio between count rate in the energy bands ~ 2–7 and ~ 8–60 keV energy bands) as a function of count rate during the individual bursts. These 'hardness-intensity diagrams' allow us to check for spectral evolution in a model-independent way. We do not correct them for background as the count rates in both bands are very high.

We find evidence of hysteretic loops in hardness-intensity space in some, but not all, of the Normal Bursts in our sample; see Figure 5.11 for an example of such a loop. The existence of such a loop suggests significant spectral evolution throughout the burst. This finding is at odds with results from previous studies based on smaller samples (e.g. Woods et al., 1999) which suggested no spectral evolution during Type II bursts in this source.

5.3.4 Minibursts

We define Minibursts as the set of all bursts with a peak 1 s binned *RXTE*/PCA-equivalent count rate of < 300% of the persistent rate. Minibursts account for 48 out of the 190 bursts identified for this study. They are observed during all 3 Outbursts, and occur during the same times that Normal Bursts are present. Minibursts occurred between MJDs 50117 and 50200 in Outburst 1, and between 50466 and 50542 in Outburst 2; during these intervals, *RXTE* observed the source for a total of 192 ks. These intervals correspond to the times between the peak of each outburst and and the time that the persistent intensity falls below ~ 0.1 Crab.

Recurrence Time

There are only 10 observations with *RXTE* which contain multiple Minibursts. Using these, we find minimum and maximum Miniburst recurrence times of 116 and 1230 s.

We find 17 *RXTE* observations which contain both a Miniburst with a preceding Normal Burst, and find minimum and maximum Normal Burst → Miniburst recurrence times of 461 and 1801 s.

Structure

In Figure 5.12, we show a representative Miniburst, and we show all Minibursts overplotted on each other in Figure 5.13. These bursts are roughly Gaussian in shape with a large variation in peak count rate; as can be seen in Figure 5.13, however, the persistent-normalised peak count rates of Minibursts are all roughly consistent with 2.

Minibursts are all ~ 5 s in duration, and some show signs of a 'dip' feature similar to those seen in Normal Bursts. We find that the timescales of these dips are all $\lesssim 10$ s. We estimate the fluence of these dips as being equal to the integrated counts of the median-subtracted between the end of the burst and a point 10 s later. If the 'missing fluence' is less than half of the standard deviation in count rate multiplied by 5 s, which represents the smallest triangular dip which would be visible in a given dataset, we treat the dip in that outburst as not being detected.

Due to the relatively short duration and low peak count rates of Minibursts, we are unable to reliably discern whether they contain a single peak or multiple peaks. For this reason we also do not fit them mathematically.

Parameters & Correlations

For each Miniburst, we are able to extract the same parameters that we extracted from Mesobursts (see list in Section 5.3.5). The mean and standard deviation of each of these

Figure 5.12: A representative *RXTE* lightcurve of a Miniburst from OBSID 20077-01-03-00 in Outburst 2.

parameters, calculated from *RXTE* data, is presented in Table 5.5 for Outburst 1, Outburst 2 and the combined population of Minibursts from Outbursts 1 & 2. The standard deviations on the fluence and peak rates of Minibursts are very large, suggesting that these parameters are distributed broadly.

Using the Spearman's Rank metric, we find only two correlations above the 5 σ level:

- Fluence is correlated with peak rate (7.3 σ).
- Fluence divided by persistent rate is correlated with peak rate divided by persistent rate (7.1 σ).

Due to the poor statstics associated with Miniburst parameters, it is likely that other

	Outburst 1		Outburst 2		Outbursts 1&2	
	Mean	S.D.	Mean	S.D.	Mean	S.D.
Fluence	6792	5776	4474	3307	5422	4627
Peak Rate	3501	2851	2473	1664	2902	2293
Fluence/k	3.67	1.13	3.58	1.47	3.61	1.34
Peak Rate/k	1.90	0.37	1.76	0.28	1.82	0.32
Rise Time	2.33	0.8	2.03	1.1	2.15	1.0
Fall Time	2.32	0.9	2.35	1.0	2.32	0.9
Tot. Time	4.61	1.0	4.38	01.0	4.47	1.0

Table 5.5: A table showing the mean and standard deviation of 7 parameters of *RXTE*-sampled Minibursts from Outburst 1, Outburst 2 and both outbursts combined. Fluence is given in cts PCU^{-1}, peak rate is given in cts s^{-1} PCU^{-1} and rise, fall and total time are given in s. k is the persistent emission rate during the observation in which a given burst was detected.

Figure 5.13: Top: a plot of every Miniburst, centred by the time of its peak, overlaid on top of each other. **Bottom:** a plot of every Miniburst in which count rates have been normalised by the persistent emission count rate during the observation from which each burst was observed.

parameter pairs are also correlated.

Colour Evolution

Minibursts show the greatest magnitude of evolution in colour of all the classes of burst. In Figure 5.14, we show how the hardness ratio between the 4–10 and 2–4 keV energy bands changes during an observation containing both a Miniburst and a Normal Burst. We find that the hardness ratio increases by $\sim 50\%$ in a Miniburst, significantly more than the change in hardness during Normal or Mesobursts. The statistics in minibursts were too poor to check for the presence of hysteresis.

5.3.5 Mesobursts

We define Mesobursts as the set of all bursts with a persistent-emission-subtracted peak 1 s binned *RXTE*/PCA-equivalent count rate below 3000 cts s^{-1} PCU^{-1} in which the peak of the burst reaches at least 300% of the persistent rate. Mesobursts account for 43 out of the 190 bursts identified for this study. They are observed in *RXTE* data from both Outbursts 1 & 2; in both cases they occur after the main outburst and before or during a rebrightening event. Mesobursts occurred between MJDs 50238 and 50248 in Outburst 1, and between 50562 and 50577 in Outburst 2; during these intervals, *RXTE* observed the source for a total of 44 ks. As no soft X-ray instrument monitored the Bursting Pulsar during the latter stages of Outburst 3, it is unclear whether Mesobursts occurred during this outburst. The one pointed observation of *NuSTAR* made during this time did not detect any Mesobursts.

Recurrence Time

Only 6 *RXTE* observations in Outburst 1, and 4 in Outburst 2, contain multiple Mesobursts. From our limited sample we find minimum and maximum recurrence times of ~ 230 and ~ 1550 s in Outburst 1 and minimum and maximum recurrence times of ~ 310 and ~ 2280 s in Outburst 2.

Structure

The structure of the main part of a Mesoburst is significantly more complex than in Normal Bursts, consisting of a large number of secondary peaks near the main peak of the burst. Mesobursts never show the post-burst ‘dip’ feature that we see in Normal Bursts, but they can show ‘plateaus’. In Figure 5.15 we show an example of a Mesoburst with a plateau similar to those seen after Normal Bursts, suggesting a connection between the two classes.

In Figure 5.16 we show the plot of all Mesobursts observed by *RXTE* overlayed on top of each other before (top panel) and after (bottom panel) being renormalised by persistent

Figure 5.14: A portion of observation 10401-01-16-00, featuring a Normal Burst (~ 100 s) and a Miniburst (~ 500 s). The top panel shows the total 2–10 keV lightcurve. The middle panel shows lightcurves from two different energy bands; the count rates from the soft energy band have been multiplied by 5.4 so they can more easily be compared with the hard energy band. The bottom panel shows the evolution over time of the ratio between the rates in the two bands. As can be seen in panels 2 and 3, the Miniburst has a significantly higher fractional amplitude in the 4–10 keV energy band than in the 2–4 keV band.

Figure 5.15: A lightcurve from *RXTE* observation 20078-01-17-00 from Outburst 2, showing an apparent 'plateau' feature after a Mesoburst.

emission rate. It can be seen that the intensity and structure of these bursts is much more variable than in Normal Bursts (see Figure 5.7). However, each Mesoburst has a fast rise followed by a slow decay, and they occur over similar timescales of $\sim$ 10–30 s.

Parameters & Correlations

Due to the complexity structure of Mesobursts, we do not fit them mathematically as we did for Normal Bursts. Instead we define a number of different parameters for each Mesoburst, listed below:

- Total burst fluence and burst fluence divided by persistent emission.
- Peak 1 s binned rate and peak rate divided by persistent emission.
- Rise time, fall time and total time.

The mean and standard deviation of each of these parameters, calculated from *RXTE* data, is presented in Table 5.6. Due to the relative low number of Mesobursts compared to Normal Bursts, we only present the results from the combined set of bursts in both Outbursts 1 & 2. In general, Mesobursts are longer in duration than Normal Bursts, and have significantly smaller amplitudes and fluences (compare e.g. Table 5.4).

Using the Spearman's Rank metric, we find a number correlations above the 5 σ level:

- Fluence is correlated with peak rate ($> 10\,\sigma$), peak rate divided by persistent rate ($6.7\,\sigma$), fall time ($6.8\,\sigma$), total time ($6.0\,\sigma$).
- Fluence divided by persistent rate is correlated with peak rate divided by persistent rate ($7.3\,\sigma$).

Figure 5.16: Top: a plot of every Mesoburst, centred by the time of its peak, overlaid on top of each other. **Bottom:** a plot of every Mesoburst in which count rates have been normalised by the persistent emission count rate during the observation from which each burst was observed.

	Mean	Standard Deviation
Fluence (cts PCU^{-1})	6067	6707
Peak Rate (cts s^{-1} PCU^{-1})	665.4	658.4
Fluence/k	48.6	32.8
Peak Rate/k	5.32	4.0
Rise Time (s)	6.95	4.9
Fall Time (s)	18.28	10.8
Total Time (s)	25.88	13.3

Table 5.6: A table showing the mean and standard deviation of 7 burst parameters of *RXTE*-sampled Mesobursts from Outbursts 1 & 2. k is the persistent emission rate during the observation in which a given burst was detected.

- Peak rate is also correlated with peak rate divided by persistent rate (7.4 σ), fall time (5.8 σ) and persistent level (6.2 σ).
- Rise time correlates with total time (5.4 σ).
- Fall time correlates with total time (> 10 σ).

Colour Evolution

The hardness ratio of the emission from the source decreases significantly during Mesobursts, with the PCA 8–60/2–7 keV colour decreases from ~ 0.6 between bursts to ~ 0.2 at the peak of a burst. Due to the poor statistics of these features compared with Normal Bursts, we were unable to check for evidence of hardness-intensity hysteresis.

5.3.6 Structured "Bursts"

We define Structured Burst observations as observations in which the recurrence time between bursts is less than, or approximately the same as, the duration of a single burst. Structured Bursts constitute the most complex behaviour we find in our dataset. Unlike the other classes of burst we identify, Structured Bursts are not easily described as discrete phenomena. We find Structured Bursts in 54 observations which are listed in Appendix B.

In both outbursts covered by *RXTE*, Structured Bursts occur in the time between the end of the main outburst and the start of a rebrightening event. In both cases these periods of structured outbursts are preceded by a period populated by Mesobursts. Mesobursts occurred between MJDs 50248 and 50261 in Outburst 1, and between 50577 and 50618 in Outburst 2; during these intervals, *RXTE* observed the source for a total of 81 ks. Notably, as we show in Figure 5.17, one Outburst 1 *RXTE* lightcurve containing Structured Bursting also contains a bright Mesoburst.

In both outbursts, the amplitude of Structured Bursting behaviour decreases as the outburst approaches the peak of the rebrightening event. This amplitude continues to decrease as the

Figure 5.17: A lightcurve from *RXTE*/PCA observation 10401-01-57-03, showing a Mesoburst occuring during a period of Structured Bursting.

Structured Burst behaviour evolves into the low-amplitude noisy behaviour associated with the source's evolution towards the hard state.

Colour Evolution

We produce hardness-intensity diagrams for a number of Structured Burst observations; we show a representative example in Figure 5.18. We find that hardness is strongly correlated with count rate during this class of bursting, but that the magnitude of the change in hardness is no greater than $\sim 15\%$. This is significantly less than the change in hardness that we see during Normal or Minibursts. We also find no evidence of hysteretic hardness-intensity loops from Structured Bursts.

Types of Structured Bursting

In Figure 5.19, we present a selection of lightcurves which show the different types of variability that can be seen during periods of Structured Bursting. These consist of a variety of patterns of peaks and flat-bottomed dips, and both *RXTE*-observed outbursts show several of these different patterns of Structured Bursting. As all types of Structured Bursting have similar amplitudes and occur in the same part of each outburst, we consider

Figure 5.18: A 1 s-binned hardness-intensity diagram from *RXTE* observation 20078-01-23-00, showing that hardness tends to correlate with intensity during Structured Bursting. Data are binned to 8 s, and background has been estimated by subtracting mean count rates in the relevant energy bands from *RXTE* OBSID 30075-01-26-00.

them to be generated by the same physical process. We do not seperate these patterns into separate subclasses in this paper.

5.4 Discussion

We analyse all available X-ray data from the first 3 outbursts of the Bursting Pulsar. The bursting behaviour evolves in a similar way during these outbursts, strongly associating them with the Bursting Pulsar and suggesting an underlying connection between the classes of burst. We also find that both Outbursts 1 & 2 showed 'rebrightening events' similar to those seen in a number of other LMXBs (e.g. Wijnands et al., 2001; Patruno et al., 2016)

We find that the Type II X-ray bursts from these data can be best described as belonging to four phenomenological classes: Normal Bursts, Minibursts, Mesobursts and Structured Bursts. For each of these four classes, we collect a number of statistics to shed light on the physical mechanisms that generate these lightcurve features.

Normal Bursts and Minibursts both represent the "Type II" bursting behaviour which is observed most commonly from this source. Mesobursts occur much later on in the outburst and show fast-rise slow-decay profiles; they are generally much fainter and more structured than Normal Bursts. Finally, Structured Bursts form continuous highly structured regions of bursting over timescales of days. All Normal Bursts and some Minibursts show count rate 'dips' after the main burst, while Mesobursts and Structured Bursts do not. In addition to this, some Normal and Mesobursts show count rate 'plateaus'; regions of roughly stable count rate above the persistent level which last for $\sim$ 10s of seconds. These features are also sometimes seen in Mesobursts, while Minibursts and Structured Bursts never show these structures.

Here we discuss these results in the context of models proposed to explain Type II bursting. We also compare our results with those of previous studies on bursting in both the Bursting Pulsar and the Rapid Burster.

5.4.1 Evolution of Outburst and Bursting Behaviour

In Figure 5.1 we have shown that Outbursts 1 & 2 evolve in a similar way. In both outbursts, the intensity of the Bursting Pulsar reaches a peak of order $\sim$ 1 Crab before decreasing over the next $\sim$ 100 days to a level of a few tens of mCrab. A few 10s of days after reaching this level, the lightcurves of both outbursts show a pronounced 'rebrightening' event, during which the intensity increases to $\sim$ 100 mCrab for $\sim$ 10 days. Outburst 1 shows a second rebrightening event $\sim$ 50 days after the first. It is unclear whether any rebrightening events occurred in Outburst 3 due to a lack of late-time observations with soft X-ray telescopes. X-ray 'rebrightening' events have been seen after the outbursts of a number of other LMXBs with both neutron star and black hole primaries:

Figure 5.19: A selection of *RXTE* lightcurves from Structured Bursting observations of the Bursting Pulsar. **Top:** a lightcurve from Outburst 1 showing flaring on timescales of $\sim$ 10 s. **Middle:** a lightcurve from Outburst 1 showing the same flaring behaviour with an additional slower modulation over $\sim$ 50 s. **Bottom:** a lightcurve from Outburst 2 showing a regular sequence of flat-bottomed dips and multi-peaked flaring. These show the wide variety of variability patterns that we classify as 'Structured Bursting'.

including SAX J1808.4-3658 (Wijnands et al., 2001), XTE J1650-500 (Tomsick et al., 2003) and IGR J17091-3625 (Court et al., 2017).

As we have shown in Figures 5.4 & 5.5, the nature of bursts from the Bursting Pulsar also evolves in a similar way in both Outbursts 1 & 2. Starting from around the peak of each outburst, both Normal and Minibursts are observed. The fluence of these bursts decrease over time as the X-ray intensity of the source decreases, before bursting shuts off entirely when the 2–16 keV flux falls below ~ 0.1 Crab. After a few 10 s of days with no bursts, bursting switches back on in the form of Mesobursts; this occurs during the tail of a rebrightening event in Outburst 1, but in the tail of the main outburst in Outburst 2. Mesobursting continues until the 2–16 keV source flux falls below ~ 0.03 Crab, at which point we observe the onset of Structured Bursting. In both Outbursts, Structured Bursting stops being visible a few 10s of days later during the start of a rebrightening event. Because this evolution is common to both of the outbursts observed by *RXTE*, this strongly indicates that the nature of bursting in the Bursting Pulsar is connected with the evolution of its outbursts. Additionally, with the exceptions of Normal and Minibursts, we show that each class of burst is mostly found in a distinct part of the outburst corresponding to a different level of persistent emission.

In Figure 5.20, we show lightcurves from Outburst 2 taken a few days before and after the transition from Mesobursts to Structured Bursting. We can see that, as the system approaches this transition, Mesobursts become more frequent and decrease in amplitude. Additionally in Figure 5.17 we show a lightcurve which contains both a Mesoburst and Structured Bursting. We find that, instead of a well-defined transition between these bursting classes, there is a more gradual change as Mesobursting evolves into Structured Bursting. This suggests that the same mechanism is likely to be responsible for both of these types of burst.

The transition between Normal Bursts and Mesobursts, however, is not smooth; in both outbursts these two classes of bursting are separated by ~ 10 day gaps in which no bursts of any kind were observed at all. If all our classes of burst are caused by the same or similar processes, any model to explain them will also have to explain these periods with no bursts.

5.4.2 Parameter Correlations

We extracted a number of phenomenological parameters from each Normal Burst, Miniburst and Mesoburst. For Normal Bursts, we extracted a large number of parameters by fitting a phenomenological model described in Section 5.3.3. For Minibursts and Mesobursts we extracted recurrence times and persistent emission-subtracted peak rates; we also calculated burst fluences by integrating the persistent emission-subtracted rate over the duration of the burst. We do not extract similar parameters for Structured Bursts due to their complex nature.

In all three of the classes of burst we consider, we found that fluence and peak rate correlate

Figure 5.20: A series of lightcurves from *RXTE*/PCA observations of Outburst 2, showing a gradual evolution from Mesobursts to Structured Bursting over a period of $\sim$ 30 days. Each inset lightcurve is plotted with the same y-scaling, and each corresponds to 2 ks of data.

strongly with persistent emission. For each type of burst case, the slope of these correlations is consistent with being equal during Outbursts 1 & 2.

We also compared the Normal Bursts in Outburst 1 with the Normal Bursts in Outburst 2. The only significant statistical differences we found between these two populations were in the burst peak rate and the burst fluence; both of these parameters are generally higher for Normal Bursts in Outburst 1. As both of these parameters strongly depend on the persistent emission, both of these differences can be attributed to the fact that Outburst 1 was significantly brighter at peak than Outburst 2.

For Normal Bursts, we found additional correlations. Of particular note, we found that both the fall time and the recovery timescale of a 'dip' is proportional to its amplitude, which has implications for the possible mechanism behind these features. We discuss this further in Section 5.4.5.

These findings strongly suggest that the properties of Normal, Mini and Mesobursts depend on the persistent luminosity of the Bursting Pulsar. Assuming that this persistent luminosity is proportional to $\dot{M}$, this suggests that all classes of bursting are sensitive to the accretion rate of the system. Additionally, with the exceptions of Normal and Minibursts, we find that each class of burst is mostly found in a distinct part of the outburst corresponding to a different level of persistent emission. We suggest that Normal, Meso and Structured Bursts may in fact be manifestations of the same physical instability but at different accretion rates. This is supported by the observation of a Mesoburst during a period of Structured Bursting, which we show in the lightcurve in Figure 5.17. This shows that the conditions for both Meso and Structured Bursting can be met at the same time.

Our Class	Giles et al. Class
Normal Bursts	G_1
Mesobursts	G_1
Minibursts	G_2
Structured Bursts	-
-	G_3

Table 5.7: A table showing how our burst classes map to those described in Giles et al. (1996). Giles et al. do not consider the times during the outburst when Structured Bursts appear, and we consider G_3 bursts described by Giles et al. to be consistent with flicker noise.

5.4.3 Comparison with Previous Studies

In their study of bursts in the Bursting Pulsar, Giles et al. (1996) found evidence for three distinct classes of Type II bursts in the Bursting Pulsar:

- "Bursts" (hereafter G_1 bursts to avoid confusion), the common Type II bursts seen from the source.
- "Minibursts" (hereafter G_2 Bursts), with smaller amplitudes up to ~ 2 times the persistent emission level.
- "Microbursts" (hereafter G_3 Bursts), second-scale bursts with amplitudes of ~ 50–100% of the persistent level.

We find that Giles et al's G_1 category contains the bursts that we identify as Normal Bursts, while our Miniburst category contains the same bursts as Giles' G_2 category. Giles et al. (1996) only consider bursts up to MJD 50204 in their classification, and they could not classify any bursts that we identify as Mesobursts; under their framework, we find that Mesobursts would also be categorised as G_1. We present the full mapping between Giles classes and our classes in a schematic way in Table 5.7.

Giles et al. (1996) note the presence of both dips and plateaus in Normal Bursts. To calculate the fluence of each main burst and its associated dip, Giles et al. integrate the total persistent-emission-subtracted counts in each feature. They calculate that ratio between burst fluence and 'missing' dip fluence (ϕ_B/ϕ_d) is between 0.26 and 0.56 in Outburst 1 before correcting for dead-time effects. Using bursts in which our mathematical fit gave well-constrained ($> 5\,\sigma$) values for both burst and dip fluence, we find that ϕ_B/ϕ_d is between 1.3 and 2.0 in Outburst 1 and between 1.3 and 2.9 in Outburst 2. Our values differ significantly from those reported from Giles et al.; this is likely due to differing definitions of the persistent emission level and the start and end times of each dip, as Giles et al. do not report how they define these features.

Our values for the ratios between burst and dip fluences, as well as those of Giles et al., are significantly affected by dead-time effects. These effects cause the fluence of bursts to be under-reported, as can be inferred from Figure 5.22, but the integrated counts in dips are not significantly affected (Giles et al., 1996). Therefore correcting for dead time can only

increase the value of ϕ_B/ϕ_d, and our results shows that the fluence of a burst is always greater than the fluence 'missing' from a dip.

We find evidence of significant colour evolution during both Normal Bursts and Minibursts, which is strongly indicative of a spectral evolution. This contrasts with previous studies (e.g. Woods et al., 1999) which have found no significant spectral evolution during a Normal Burst in the Bursting pulsar. Further work on the time-resolved spectra of this source will likely allow us to better understand the underlying physics of its behaviour.

Using data from the KONUS experiments aboard the GGS-Wind and Kosmos-2326 satellites, Aptekar et al. (1998) have previously found that the recurrence times between consecutive bursts in Outburst 1 are distributed with a constant mean of ~ 1776 s. This is substantially longer than our value of 1209 s that we find for Outburst 1, but our value is likely an underestimate due to a selection bias caused by the relatively short pointings of *RXTE*.

Using *Chandra* and *XMM-Newton* data, we find a mean recurrence time for Outburst 3 of 1986 s; as pointings with these instruments are significantly longer than the burst recurrence timescale, windowing effects are negligible. As this value is close to the value that Aptekar et al. (1998) find for mean recurrence time, our result is consistent with the burst rate in all three outbursts being approximately the same.

Previous studies with *CGRO*/BATSE have found that the burst rate during the first few days of Outbursts 1 & 2 was significantly higher than during the rest of each outburst (Kouveliotou et al., 1996; Woods et al., 1999). As *RXTE* did not observe either of these times, we are unable to test this result.

5.4.4 Comparison with other objects

In Court et al. (2018a) we discuss the possibility that some of the behaviour in the Bursting Pulsar could be due to fluctuations in the magnetospheric radius of the system close to the corotation radius. This behaviour (e.g. Bogdanov et al., 2015; Ferrigno et al., 2014) is also seen in 'Transitional Millisecond Pulsars' (TMSPs): objects which alternate between appearing as X-ray pulsars and radio pulsars (see e.g. Archibald et al., 2009; Papitto et al., 2013a).

Another natural comparison to the Bursting Pulsar is the Rapid Burster (Lewin et al., 1976a), a neutron star LMXB in the globular cluster Liller I. This object is the only LMXB other than the Bursting Pulsar known to unambiguously exhibit Type II bursting behaviour during outbursts. Rappaport and Joss (1997) have previously proposed that the Bursting Pulsar, the Rapid Burster and other neutron star LMXBs form a continuum of objects with different magnetic field strengths.

We compare our study of bursts in the Bursting Pulsar with studies of Type II bursts in the Rapid Burster, particularly the detailed population study performed by Bagnoli et al.

Figure 5.21: *RXTE* lightcurves of representative Long (top) and Short (bottom) bursts from the Rapid Burster. These bursts were identified and classified by Bagnoli et al. (2015).

(2015). Bagnoli et al. (2015) found that Type II bursting begins during the decay of an outburst in the Rapid Burster. This is the same as what we see in the Bursting Pulsar, where we find Normal Bursting behaviour starts during the outburst decay. Bagnoli et al. (2015) found that all bursting in the Rapid Burster shuts off above an Eddington Fraction of $\gtrsim 0.05$, whereas we find bursting in the Bursting Pulsar shuts off below a 2–16 keV flux of Eddington fraction of ~ 0.1 *Crab*: assuming that the peak persistent luminosity of the Bursting Pulsar was approximately Eddington Limited (e.g. Sazonov et al., 1997), this value corresponds to an Eddington fraction of order ~ 0.1. This suggests that Type II bursting in these two objects happen in very different accretion regimes.

Bagnoli et al. showed that bursting behaviour in the Rapid Burster falls into a number of 'bursting modes', defined by the morphology of individual Type II bursts. In particular, they find that Type II bursts in the Bursting Pulsar fall into two classes (see also Marshall et al., 1979), lightcurves of which we reproduce in Figure 5.21:

- Short near-symmetric Bursts with timescales of $\sim 10s$ of seconds and peak rates near the Eddington Limit.
- Long bursts with a fast rise, a long ~ 100 s plateau at peak rate followed by a fast decay. The level of the plateau is generally at or near the Eddington Limit.

Short bursts are very similar in shape to Normal Bursts in the Bursting Pulsar, but we find

no analogue of long bursts in our study. Bagnoli et al. (2015) suggests that the 'flat-top' profile of long bursts could be due to the effects of near-Eddington accretion, and they show that the intensity at the 'flat top' of these bursts is close to Eddington limit. Previous works have shown that the persistent emission of the Bursting Pulsar is Eddington-limited at peak, and therefore bursts from the Bursting Pulsar are significantly super-Eddington (Sazonov et al., 1997). We suggest, therefore, that Long Bursts cannot occur in systems with a persistent rate approaching the Eddington Limit. This could explain why Long Bursts are not seen during periods of Normal Bursting in the Bursting Pulsar (during which the persistent emission is $\gtrsim 20\%$ of Eddington), but it remains unclear why these features are not seen later in each outburst when the Bursting Pulsar is fainter. Alternatively, all the differences we see between bursts produced by the Rapid Burster and the Bursting Pulsar could be explained if the physical mechanisms behind these bursts are indeed different between the objects.

Bagnoli et al. (2015) also find a number of correlations between burst parameters in the Rapid Burster, which we can compare with our results for the Bursting Pulsar. We find a number of similarities between the two objects:

- The fluence of a burst correlates with its amplitude.
- The duration of a burst does not correlate[10] with the persistent emission.
- The recurrence time between consecutive bursts does not depend on the persistent emission.

There are also a number of differences between the set of correlations between burst parameters in these two systems:

- Burst duration is correlated with burst fluence in the Rapid Burster, but these have not been seen to correlate in the Bursting Pulsar.
- Burst duration, peak rate and burst fluence are all correlated with burst recurrence time in the Rapid Burster. We have not found any of these parameters to correlate with burst recurrence time in the Bursting Pulsar.
- Peak rate and burst fluence correlate with persistent emission in the Bursting Pulsar, but this has not been seen in the Rapid Burster.

As the fluence of a burst in the Rapid Burster does not depend on persistent emission, which can be used as a proxy for $\dot{M}$, this suggests that the process that triggers Type-II bursts in this source is not strongly dependent on the global accretion rate. However the strong correlations between persistent emission and burst peak and fluence we find in the Bursting Pulsar show that the energetics of individual bursts strongly depend global accretion rate at that time.

It has previously been noted that consecutive Normal Bursts in the Bursting pulsar do not

[10]We state two parameters do not correlate if their Spearman Rank score corresponds to a significance $< 3\sigma$.

show a strong correlation between recurrence time and fluence (Taam and Lin, 1984; Lewin et al., 1996, however see Aptekar et al., 1997). This correlation would be expected if the instability took the form of a relaxation oscillator, as it does in the Rapid Burster (Lewin et al., 1976b). However, we also find that the arrival times of Normal Bursts from the Bursting Pulsar are not consistent with a Poisson distribution with constant mean. This implies either that bursts are also not independent events in the Bursting Pulsar, or that the frequency of these bursts is not constant throughout an outburst as reported by Aptekar et al. (1998).

5.4.5 Comparison with Models of Type II Bursts

To our knowledge no models have been proposed which can fully explain Type II bursting behaviour, but several models have been proposed in the context of Type II bursting from the Rapid Burster MXB 1730-33. A number of models invoke viscous instabilities in the inner disk as the source of cyclical bursting (e.g. Taam and Lin, 1984; Hayakawa, 1985), but these fail to explain why the majority of Neutron Star LMXBs do not show this behaviour.

Spruit and Taam (1993) use a different approach. They show that, in some circumstances, the interaction between an accretion disk and a rapidly rotating magnetospheric boundary can naturally set up a cycle of discrete accretion events rather than a continuous flow (see also D'Angelo and Spruit, 2010, 2012; van den Eijnden et al., 2017; Scaringi et al., 2017). Walker (1992) suggests that, for a neutron star with a radius less than its ISCO, a similar cycle of accretion can be set up when considering the effects of a high radiative torque. All of these models suggest that Type II bursts are caused by sporadic accretion events onto the neutron star, which in turn are caused by instabilities that originate in the inner part of the accretion disk. For a more detailed review of these models, see Lewin et al. (1993).

All of the models discussed above are able to reproduce some of the features we see from bursts in the Bursting Pulsar. In particular, the 'dip' we see after Normal Bursts has previously been interpreted as being caused by the inner disk refilling after a sudden accretion event (e.g. Younes et al., 2015). As these dips are also seen after some Minibursts, we could also interpret Minibursts as being caused by a similar cycle. To test this idea, in Figure 5.22 we present a scatter plot of the burst and dip fluences for all Normal Bursts and Minibursts. In both classes of burst, there is a strong correlation between these two parameters. We find that a power law fit to the Normal Bursts in this parameter space also describes the Minibursts. This suggests that the same relationship between burst fluence and missing dip fluence holds for both types of burst, although the two populations are not continuous. This suggests that Minibursts are energetically consistent with being significantly fainter versions of Normal Bursts.

The models of Spruit and Taam (1993) and Walker (1992) also have shortcomings when used to describe the Bursting Pulsar. Walker (1992) state that their model only produces Type II bursts for a very specific set of criteria on the system parameters. One of these

Figure 5.22: A scatter plot showing the relationship between burst fluence and 'missing' dip fluence for Normal Bursts (black) and Minibursts (Red), with the best fit power law plotted in solid blue. A power law fit to just the Normal Bursts (blue dashed line) also approaches the Minibursts. Note that the Normal Bursts plotted in grey were not used to calculate this latter fit, as the effects of instrumental dead-time cause high burst fluences to be under-reported. Upper limits on Miniburst dip fluences are shown with arrows.

criteria is an essentially non-magnetic ($B = 0$) neutron star. This is inconsistent with observations of cyclotron lines from the Bursting Pulsar, which suggest a surface field strength of order 10^{11} G (Doroshenko et al., 2015).

Unlike models based on viscous instability, the model of Spruit and Taam (1993) does not impose a correlation between burst fluence and burst recurrence time. However, it does predict a strong correlation between burst recurrence time and mean accretion rate, which is not consistent with our results for the Bursting Pulsar.

In general, we find that models established to explain bursting in the Rapid Burster are poor at explaining bursting in the Bursting Pulsar. Any model which can produce Type II bursting in both systems fails to explain why other systems do not also show this behaviour. Our results suggest that Type II bursts in the Rapid Burster and the Bursting Pulsar may require two separate models to be explained.

Evidence of Thermonuclear Burning

We also consider the possibility that some of our observations could be explained by thermonuclear burning in the Bursting Pulsar. A thermonuclear origin for the main part of Normal Type II X-ray bursts has been ruled out by previous authors (e.g. Lewin et al., 1996), but it is less clear that associated features could not be explained by this process.

Bildsten and Brown (1997) have previously analysed whether thermonuclear burning in the Bursting Pulsar could be stable. They find that the presence and profile of a thermonuclear burning event on the Bursting Pulsar would be strongly dependent on both the accretion rate $\dot{M}$ and the magnetic field strength B. They predict that, for $B \gtrsim 3 \times 10^{10}$ G, burning events would take the form of a slowly propagating burning front which would result in a low-amplitude X-ray burst with a timescale of several minutes. Measurements of the Bursting Pulsar taken during Outburst 3 suggest a surface field strength of $> 10^{11}$ G, in turn suggesting that the Bursting Pulsar exists in the regime in which this burning behaviour is possible.

The ‘plateau’ events after Normal Bursts are consistent with the slow burning predicted by Bildsten and Brown (1997). This picture is consistent with models for Type II X-ray bursts involving spasmodic accretion events (e.g. Spruit and Taam, 1993; Walker, 1992), as plateaus always occur after a Type II burst has deposited a large amount of ignitable material onto the neutron star surface. However in this picture it would be unclear why many Normal Bursts do not show this plateau feature. Mesobursts can also exhibit plateaus, and are therefore likely to also be products of spasmodic accretion onto the neutron star.

However, the interpretation of Mesobursts as being caused by discrete accretion events is difficult to reconcile with the fact that these features never show dips. Bildsten and Brown (1997) show that, at smaller values of $\dot{M}$, nuclear burning on the Bursting Pulsar could become unstable. Mesobursts are only seen during the latter stages of Outbursts 1 & 2,

when the accretion rate is well below 0.1 Eddington. An interesting alternative possibility is that Mesobursts are a hybrid event, consisting of a flash of unstable thermonuclear X-ray burning followed by a slower quasi-stable burning of residual material in the form of a propagating burning front. Additional studies of the spectral evolution of these phenomena will be required to explore this possibility.

Previous authors have discussed the possibility of a marginally stable burning regime on the surface of neutron stars (not to be confused with the previously mentioned quasi-stable burning). In this regime, which occurs close to the boundary between stable and unstable burning, Heger et al. (2007) showed that an oscillatory mode of burning may occur. They associated this mode of burning with the mHz QPOs which have been observed in a number of neutron star LMXBs (e.g. Revnivtsev et al., 2001; Altamirano et al., 2008c). These QPOs only occur over a narrow range of source luminosities, show a strong decrease in amplitude at higher energies, and they disappear after a Type I burst (e.g. Altamirano et al., 2008c).

Lightcurves of objects undergoing marginally stable burning qualitatively resemble those of Structured Bursting in the Bursting Pulsar, raising the possibility of a thermonuclear explanation for Structured Bursting. However, as we show in Figure 5.4, Structured Bursting during Outburst 1 occurred during a period of time in which the Bursting Pulsar's luminosity changed by ~ 1 order of magnitude. In addition to this, in Figure 5.17 we show an example of a Mesoburst during a period of Structured Bursting. If Mesobursts can be associated with Type I bursts, any marginally stable burning on the surface of the Bursting Pulsar should have stopped after this event. Due to these inconsistencies with observations of marginally stable burning on other sources, it is unlikely that Structured Bursting is a manifestation of marginally stable burning on the Bursting Pulsar.

Linares et al. (2012) observed yet another mode of thermonuclear burning during the 2010 outburst of the LMXB Terzan 5 X-2. They observed a smooth evolution from discrete Type I bursts into a period of quasi-periodic oscillations resembling Structured Bursting. This behaviour resembles the evolution we observe between Mesobursts and Structured Bursting in Outbursts 1 & 2 of the Bursting Pulsar (as shown in Figure 5.20; compare with Figure 1 in Linares et al., 2012). However there are a number of differences between the evolutions seen in both objects. In Terzan 5 X-2 the recurrence timescale of Type I bursts during the evolution is strongly related to the accretion rate of the source at the time, whereas there is no such strong relation between the two in Mesobursts from the Bursting Pulsar. Additionally, the quasi-periodic oscillations in Terzan X-2 evolved smoothly back into Type I bursts later in the outburst, whereas Structured Bursting does not evolve back into Mesobursts in the Bursting Pulsar. As such, it is unclear that Mesobursts and Structured Bursting can be associated with the unusual burning mode seen on Terzan 5 X-2.

5.5 Conclusions

We analyse all X-ray bursts from the Rapid Burster seen by *RXTE*/PCA during its first and second outbursts, as well as bursts seen by other missions during the third outburst of the source. We conclude that these bursts are best described as belonging to four separate classes of burst: Normal Bursts, Mesobursts, Minibursts and Structured Bursts. We find that the bursting behaviour in these four classes evolves in a similar way throughout the first two outbursts of the Bursting Pulsar. We present a new semi-mathematical model to fit to the Normal Bursts in this object. Using this new framework, we will be able better quantify Bursting-Pulsar-like X-ray bursts when they are observed in other objects in the future.

As so-called "Type II" bursts are extremely varied, both between different objects and from the same objects, we find that "Type II burst" is ill-defined as a phenomenological class. Specifically, we find that the differences between bursts in the Rapid Burster and the Bursting Pulsar to be different in burst profile, peak Eddington ratio, and durations. While the fluence of Type II bursts in the Bursting Pulsar depend strongly on the persistent emission at the time, this is not the case in the Rapid Burster. Additionally the waiting time between bursts in the Rapid Burster depend heavily on the fluence of the preceding burst, but we do not find this in the Bursting Pulsar. Therefore, it is reasonable to conclude that the bursting in these two objects is generated by two very different mechanisms or in two very different physical regimes.

Chapter 6

The Bursting Pulsar GRO J1744-28: the Slowest Transitional Pulsar?

6.1 Introduction

Millisecond Pulsars are old radio pulsars with spin periods of order ~ 10 ms (Backer et al., 1982). They have long been believed to be the end product of systems containing a neutron star (NS) in a Low Mass X-ray Binary (LMXB). In these systems, matter from a Roche-lobe overflowing star donates angular momentum to a NS, spinning it up to frequencies of several 100 Hz (Alpar et al., 1982). A number of fast-spinning X-ray pulsars (accreting Millisecond Pulsars, or AMXPs) have been found in LMXBs (e.g. Wijnands and van der Klis, 1998; Altamirano et al., 2008a; Patruno et al., 2017; Sanna et al., 2017a), seemingly confirming this physical picture. At the end of this so-called 'recycling' process, the system should transition from an accretion-powered pulsar to a rotation-powered pulsar. As such, it has long been expected that such a transition could be observed by finding a system which changes its character from an accreting NS at one time to a radio pulsar at some later time. Subsequently a small family of 7 candidate objects have been discovered or proposed: these are referred to as Transitional Millisecond Pulsars (TMSPs).

The first of these objects, **PSR J1023+0038**, was identified by Archibald et al., 2009. Although it appeared as a non-accreting radio pulsar at the time of identification in 2009, previous optical studies showed that this system contained an accretion disk in 2002 (Szkody et al., 2003). As such, the pulsar in this system must have switched from an accreting phase to a radio pulsar phase at some point between 2003 and 2009, strongly suggesting the identification of this system as a TMSP. The pulsar in this system has a spin period of 1.69 ms, and the companion is a star with a mass between $\sim$0.14–0.42 $M_\odot$. Archibald et al., 2009 suggested that the low X-ray luminosity of PSR J1023+0038 in its accreting phase was due to accretion taking place in the 'propeller regime' (Illarionov and Sunyaev, 1975). In this regime, accreting matter is halted by magnetic pressure above the

co-rotation radius of the NS magnetosphere. This matter is then ejected from the system as a wind. Whether a system is in the propeller regime depends on its spin and its magnetic field strength (Lewin et al., 1988). Additionally, below a certain accretion rate, no stable balance between ram pressure and radiation pressure can form and any disk is ejected from the system (e.g. Campana et al., 1998). Archibald et al., 2009 suggested that the current accretion rate in PSR J1023+0038 is only slightly below this critical value, and that any small increase in accretion rate could cause accretion in this system to resume. They suggested the possibility of TMSP systems which flip back and forth between accreting and radio pulsar phases multiple times.

Papitto et al., 2013a identified **IGR J18245-2452** as the first pulsar to switch from a radio pulsar to an AMXP and back to a radio pulsar. This source was first observed as a radio pulsar (Manchester et al., 2005), before being observed several years later by *XMM-Newton* (Eckert et al., 2013) as an AMXP. Several months after the *XMM-Newton* observation, Papitto et al., 2013b found that the source had reactivated as a radio pulsar during X-ray quiescence. The pulsar in this system has a period of 3.93 ms, and the companion star has a mass of $> 0.17\,M_\odot$ (Papitto et al., 2013a). During the 2013 outburst of IGR J18245-2452, Ferrigno et al., 2014 reported the presence of high-amplitude variability in the X-ray lightcurve. They interpreted this as being due to the accretion rate $\dot{M}$ being very close to the critical rate at which the propeller effect begins to dominate the flow geometry. In this regime, small fluctuations in $\dot{M}$ cause so-called 'hiccups', in which matter alternates between being ejected by the propeller effect and being accreted onto the NS poles. Similar X-ray variability has subsequently been found in lightcurves from outbursts during the accreting phase of PSR J1023+0038 (Bogdanov et al., 2015), suggesting that this variability is somehow intrinsic to TMSPs as a class of objects.

1FGL J1227.9-4852 was first identified in the first *Fermi*/LAT source catalogue (Abdo et al., 2010). Hill et al., 2011 found that the γ-ray spectral characteristics of this source are consistent with known millisecond radio pulsars, although no radio pulsations were found. They suggested that this object could be associated with the X-ray source XSS J12270-4859. Before 2009, XSS J12270-4859 showed optical emission lines typical of an accretion disk (Pretorius, 2009). Hill et al., 2011 suggested that XSS J12270-4859 may also be a TMSP, which switched from an accreting phase to a radio pulsar millisecond pulsar phase between 2009 and 2011. Subsequent studies have found pulsations in both the radio (Roy et al., 2015) and γ-ray (Johnson et al., 2015) emissions of this source, confirming the system contains a pulsar and establishing its spin period at 1.69 ms.

XMM J174457-2850.3 is a neutron star X-ray binary. Although no X-ray or radio pulsations have been detected due to the faintness of the source, Degenaar et al., 2014b have found that the X-ray variability properties of this source are similar to those seen in other TMSPs. This object also exhibits extended low-luminosity states during outbursts, which Degenaar et al., 2014b suggest may be symptomatic of TMSPs.

3FGL J1544.6-1125 was also first identified in *Fermi*/LAT data. Bogdanov and Halpern,

2015 associated this object with the X-Ray source 1RXS J154439.4-112820. Due to the presence of γ-rays, as well as the presence of variability in the X-ray lightcurve similar to IGR J18245-2452, they proposed that this object is a TMSP in the accreting state. However, no pulsations from this system have been detected in the X-ray or the radio, so the pulsar period is not known. Bogdanov and Halpern, 2015 found a bimodality in count rate during the period of X-ray variability, suggesting that this behaviour can be explained as quick transitions between three quasi-stable accretion modes known as 'low' , 'high' and 'flaring'. This effect has also been seen in the TMSP IGR J18245-2452 (Ferrigno et al., 2014).

Strader et al., 2016 identified the γ-ray source, **3FGL J0427.9-6704**, as a TMSP. They found that this source also displays X-ray variability similar to what is seen from the other known TMSPs. Finally, Rea et al., 2017 have proposed that the X-ray source **XMM J083850.4-282759** may also be a TMSP. Although this source has not been detected in the gamma or the radio, the authors argued that X-ray variability coupled with X-ray flaring seen from this object is reminiscent of similar behaviour seen in other TMSPs during subluminous disk states.

The phenomenology of currently known TMSPs is varied, and different methods have been used to conclude (or propose) that each individual system belongs to this class. The fact that 6 of the 7 objects show similar patterns of X-ray variability during outburst suggests that this variability can be used as an indication that a system may be a TMSP. In this letter we present evidence that a 7th object, the so-called 'Bursting Pulsar' **GRO J1744-28**, may also be associated with this family of objects.

6.2 GRO J1744-28: The Bursting Pulsar

GRO J1744-28, or the 'Bursting Pulsar' (hereafter BP, Paciesas et al., 1996), is an LMXB containing a 2.4 Hz pulsar and a $\sim 0.07 - 0.2\,\mathrm{M}_{\odot}$ companion star (e.g. Sturner and Dermer, 1996; Finger et al., 1996b; Rappaport and Joss, 1997; Degenaar et al., 2014a); the orbital period of the systems is 11.8 d (Finger et al., 1996a; Sanna et al., 2017c). Estimates for the NS magnetic field range between 2 and 50×10^{10} G (Finger et al., 1996b; Degenaar et al., 2014a; D'Aì et al., 2015; Doroshenko et al., 2015). The BP is one of only two known systems that show 'Type II' X-ray bursts: bright, ~10 s-long non-thermonuclear X-ray flares. The "Rapid Burster" (i.e. MXB 1730-33; hereafter RB), is the other system (e.g. Lewin et al., 1976b; Kouveliotou et al., 1996).

A number of physical models have been proposed to explain Type II bursts, including among others, viscous instabilities and interactions between the disk and the rotating NS magnetosphere (e.g. Taam and Lin, 1984; Spruit and Taam, 1993). None of the proposed models can fully reproduce the observed phenomenology (Lewin et al., 1993), nor explain the significant differences observed between the BP and the RP (e.g. Lewin et al., 1996, Court et al. *in prep*). Most importantly, so far it is not understood what differentiates the RB and the BP from the more than 100 known NS-LMXBs (e.g. Liu et al., 2007) which do not show Type II bursts. De Martino et al., 2013 previously noted similarities between Type II

bursting and X-ray variability in the lightcurve of the TMSP 1FGL J1227.9-4852 (Hill et al., 2011). However, the authors note that the energetics are inconsistent with these phenomena being physically the same.

In Court et al. (*in prep.*), we performed a detailed analysis of all archival X-ray data (including *RXTE*, *Swift*, *Chandra*, *XMM-Newton*, *Suzaku*, *NuStar*, *Fermi* and *INTEGRAL*) and found that the Type II phenomenology in the BP is much richer than previously thought (e.g. Giles et al., 1996): the characteristics of the flaring evolve with time and source luminosity. Near the end of this evolution, we observed periods of highly–sctructured and complex high–amplitude X-ray variability. We refer to this variability as 'Structured Bursting', which is unlike what is seen other LMXBs but very similar to the 'hiccup' accretion observed in TMSPs.

6.3 Comparison

Rappaport and Joss, 1997 have previously suggested that the BP represents a slow X-ray pulsar nearing the end of its accreting phase. As such it is natural to compare this system with TMSPs, which are also believed to be systems approaching this evolutionary stage. In addition to this, Degenaar et al., 2014b have previously noted that the BP shows extended low-luminosity states during outburst, similar to those seen in the TMSP candidate XMM J174457-2850.3.

In Figure 6.1, we show *RXTE* lightcurves of 'Structured Bursting' from the BP alongside lightcurves from periods of 'hiccup' variability observed in the confirmed TMSPs PSR J1023+0038 and IGR J18245–2452. All three sources show similar patterns of X-ray variability: (i) *Plateaus*: periods of approximately constant count rate with high-amplitude flicker noise (all plateaus in a given observation have approximately the same mean rate), (ii) *Dips*: Periods of low count rate ($\lesssim 0.5$ of the rate in plateaus) with significantly less flicker noise, and (iii) *Flares*: Relatively short-lived increases of the count rate to values $\gtrsim 2$ times greater than the rate during plateaus. In TMSPs, these features are interpreted as representing three quasi-stable accretion modes: the 'high', 'low' and 'flaring' modes respectively (e.g. Bogdanov et al., 2015). The most significant difference is that, in general, the variability in the BP occurs on timescales ~ 1 order of magnitude longer than those in TMSPs.

In Figure 6.2 we show histograms of the 1 s-binned count-rate from all *RXTE* observations of Structured Bursting in the 1996 (left) and 1997 (right) outbursts of the BP. As is the case for TMSPs, the histograms can be described with a number of log-Normally distributed populations: 3 populations in the 1996 outburst and 2 in the 1997 outburst. It is unclear why a population would be absent from the 1997 outburst, but some TMSPs have been observed to miss the 'high' mode during hiccup accretion (e.g. IGR J18245-2452, Ferrigno et al., 2014).

Detailed works on the low and high modes observed in the light curves of TMSPs show that

Figure 6.1: Top: 2–15 keV *XMM* lightcurve from the TMSP PSR J1023+0038. **Middle:** 2–60 keV *RXTE* lightcurves from the BP during its 1996 and 1997 outbursts, showing similar variability patterns to those seen in PSR J1023+0038. **Bottom:** 2–15 keV *XMM* lightcurve from the TMSP IGR J18245-2452. *XMM* lightcurves are shown from 2–15 keV so that they can be more directly compared with *RXTE*.

Figure 6.2: Histograms of the 1 s binned count rates from all *RXTE* observations of Structured Bursting in the 1996 (left) and 1997(right) outbursts of the BP. For the 1996 outburst, we fit the distribution with three Gaussians, while for the 1997 outburst we fit the distribution with 2 Gaussians. The individual Gaussians are plotted in solid lines, while the combined total is plotted in a dashed line.

X-ray pulsations are seen during both modes. Pulsations are fractionally weaker in the low state than the high state (for example varing between 4.0 ± 0.2% and 16.8 ± 0.2% in the TMSP IGR J18245-2452, Ferrigno et al., 2014). In the case of the BP, we detect pulsations both during the low and the high modes; much like in TMSPs, the pulsations are weaker in the low mode. For example in *RXTE* OBSID 10401-01-59-00 (in 1996), the pulsations had amplitudes of 3.5 ± 0.2% and 4.9 ± 0.2% respectively, while in OBSID 20078-01-23-00 (in 1997), the pulsations had amplitudes of 4.5 ± 0.1% and 6.0 ± 0.1% respectively. A reduction in pulse fraction in accreting pulsars has been interpreted as a change in accretion geometry due to a sudden decrease in the amount of matter reaching the compact object (e.g. Ibragimov and Poutanen, 2009), and as such this result provides direct evidence that the Structured Bursting in the BP is caused by switches between accretion and propeller-driven outflows.

TMSPs are amongst the only LMXBs which are also significant γ-ray sources (e.g. Hill et al., 2011). The *Fermi* point source 3FGL J1746.3–2851c is spatially coincident with the BP. While the field is too crowded to unambiguously associate 3FGL J1746.3–2851c with the BP, the existence of a γ-ray point source at this location is consistent with the possibility that the BP and TMSPs show the same phenomenology.

The spectral evolution of known TMSPs is varied. In PSR J1023+0038, the low, high and flaring modes all present similar spectra (Bogdanov et al., 2015). However in IGR J18245-2452, Ferrigno et al., 2014 have found a strong correlation between spectral hardness and intensity during hiccups, showing that there is spectral evolution over time in this source. In Figure 6.3 we show the hardness-intensity diagram of the BP during periods of Structured Bursting. We find a significant correlation, similar to what is seen in IGR

Figure 6.3: A 7–60/2–7 keV hardness-intensity diagram for *RXTE* observation 10401-01-59-00; the lightcurve of this observation is shown in the inset. To correct for the high background of the region, we subtract the median count rate of *RXTE* observation 30075-01-24-00 from each band; at this time, GRO J1744-28 was in quiescence. We find a strong correlation between hardness and count rate, with a Spearman Rank Correlation Coefficient of 0.93. Data for the hardness-intensity diagram are binned to 10 s, while data for the lightcurve are binned to 5 s.

J18245-2452 (Ferrigno et al., 2014). This is in contrast with other slow accreting pulsar systems such as Vela X-1, which show an anticorrelation between these parameters during periods of variability (Kreykenbohm et al., 2008).

6.4 Discussion

In this letter we compare the lightcurve, spectral and timing properties of the Bursting Pulsar at the end of its 1996 and 1997 outburst with those observed from Transitional Millisecond Pulsars. The data suggest that the BP may have undergone "hiccup" accretion similar to that seen in TMSPs, during which transferred matter alternates between being accreted onto the poles of the NS and being ejected from the system by the 'propeller' effect (e.g. Ferrigno et al., 2014). This similarity raises the exciting prospect of studying the physics of TMSPs in a completely different regime.

Very recently Campana et al., 2017 proposed a universal relation between magnetic moment, spin frequency, stellar radius and luminosity at the boundary between accretion and the propeller effect. Any object that exists on one side of this boundary should be able to accrete, whereas objects on the other side should be in the propeller phase or not accreting at all. In Figure 6.4 we reproduce Campana et al., 2017's results and include our estimates for the BP during the periods of Structured Bursting. We find that the BP is consistent with lying on or near the boundary between propeller-mode and direct accretion,

Figure 6.4: A plot of a number of objects ranging in scale from LMXBs and High-Mass X-ray Binaries (HMXBs) to Cataclysmic Variables (CVs) and Young Stellar Objects (YSOs) (blue diamonds). In each case, the object is plotted at the luminosity which defines its transition between propeller-mode accretion and free accretion. Campana et al., 2017 suggest that any object above the line of best fit accretes freely, whereas all objects below are in the propellor regime. The BP (red circle) is consistent with approaching this line during periods of Structured Bursting. Errorbars on the BP represent the range of the reported magnetic fields as well as a range of stellar radii between 10–20 km. The range in luminosity for the BP is calculated using 1.5-25 keV *RXTE*/PCA flux, assuming a distance of between 4–8 kpc (e.g. Kouveliotou et al., 1996; Gosling et al., 2007; Sanna et al., 2017c) and a bolometric correction factor of 1–3. Data on the other objects taken from Campana et al., 2017. L is the bolometric luminosity of the object in ergs s^{-1}, P is the period in s, R is the radius in cm and μ is the magnetic moment in $Gauss\,cm^3$.

clustering with High Mass X-ray Binaries (as expected due to the BP's high magnetic field), and supporting the link between "hiccups" and Structured Bursting.

If the "hiccups" in the BP show that the system is transiting to a radio pulsar, then the BP should not lie in the P-$\dot{P}$ 'graveyard' region (e.g. van den Heuvel, 1993). To our knowledge, there is no measurment yet of the NS spin down during the BP's X-ray quiescent state. Under the assumption that the BP becomes a radio pulsar, and that the possible spin down during that period is due to the same mechanism as those of the known radio pulsars, we can position the BP in the P-$\dot{P}$ diagram (the plot of pulsar spin P against spin-down rate $\dot{P}$, not shown) by using the orbital period and estimates of its magnetic field. At $B \sim 2 \times 10^{11}$G, the BP falls well outside of the pulsar graveyard. We note that Pandey-Pommier et al. (2014) and Russell et al. (2017) did not detect a significant radio source at the location of the BP during X-ray outburst. To our knowledge, there is no report of Radio detection/non-detection during X-ray quiescence.

In addition to the BP, several additional sub-10 Hz accreting X-ray pulsars have been discovered (e.g. GX 1+4 and 4U 1626-67, Lewin et al., 1971; Rappaport et al., 1977). The

reason behind the slow spins of these objects is poorly understood, but a number of these systems have been seen to undergo 'torque reversal' events, during which $\dot{P}$ switches sign (e.g. Chakrabarty et al., 1997b,a). In some sources, the magnitude of the spin-down during an event is of the same order magnitude as the preceding period of spin-up, resulting in little or no net spin change. Torque reversal events occur irregularly, but the recurrence timescale varies between objects from weeks to decades (e.g. Bildsten et al., 1997).

Given that the BP has a strongly stripped stellar companion (Bildsten and Brown, 1997), a high magnetic field and shows significant spin-up during outburst (e.g. Finger et al., 1996b; Sanna et al., 2017c), it is difficult to explain its low spin by suggesting the system is young or that the angular momentum transfer is inefficient. Rappaport and Joss, 1997 suggest that the magnetic field and spin could be explained if much of the mass transfer in the system occurred before the primary became a neutron star, but they note that this scenario is inconsistent with the low mass of the donor star.

Torque reversal events in the BP (similar to those seen in other slow accreting pulsars, e.g. Bildsten et al., 1997) could explain why the pulsar has failed to reach a spin rate on par with TMSPs. Although no torque reversal event has been reported from the BP, it is feasible that the recurrence timescale of such an event is longer than the ~ 20 years for which the object has been studied (this is consistent with the recurrence timescales seen in other slow accreting pulsars). The discovery of torque reversal in the BP would strongly link it with the other known slow accreting pulsars.

The BP has a spin rate ~ 2 orders of magnitude less than previously known TMSPs, and a magnetic field ~ 2 orders of magnitude stronger, but it still shows lightcurve, timing and spectral behaviour which are remarkably similar to TMSPs. This raises the exciting prospect of exploring the physics of TMSPs in a previously unexplored physical regime. If the BP itself is a transitional pulsar, it should emit radio pulsations during X-ray quiescence. Future detections of radio pulsations from this object would unambiguously confirm it as a transitional pulsar.

Chapter 7

Conclusions

Bibliography

Abdo, A. A., Ackermann, M., Ajello, M., Allafort, A., Antolini, E., Atwood, W. B., Axelsson, M., Baldini, L., Ballet, J., Barbiellini, G., and et al. (2010). Fermi Large Area Telescope First Source Catalog. *ApJS*, 188:405–436.

Abramowicz, M. A., Czerny, B., Lasota, J. P., and Szuszkiewicz, E. (1988). Slim accretion disks. *ApJ*, 332:646–658.

Allen, J. L., Linares, M., Homan, J., and Chakrabarty, D. (2015). Spectral Softening Between Outburst and Quiescence In The Neutron Star Low-Mass X-Ray Binary SAX J1750.8-2900. *ApJ*, 801:10.

Alpar, M. A., Cheng, A. F., Ruderman, M. A., and Shaham, J. (1982). A new class of radio pulsars. *Nature*, 300:728–730.

Altamirano, D. and Belloni, T. (2012). Discovery of High-frequency Quasi-periodic Oscillations in the Black Hole Candidate IGR J17091-3624. *ApJ*, 747:L4.

Altamirano, D., Belloni, T., Krimm, H., Casella, P., Curran, P., Kennea, J., Kalamkar, M., van der Klis, M., Wijnands, R., Linares, M., Motta, S., Munoz-Darias, T., and Stiele, H. (2011a). IGR J17091-3624 undergoes 'heartbeat' oscillations similar to those of GRS 1915+105. *The Astronomer's Telegram*, 3230.

Altamirano, D., Belloni, T., Linares, M., van der Klis, M., Wijnands, R., Curran, P. A., Kalamkar, M., Stiele, H., Motta, S., Muñoz-Darias, T., Casella, P., and Krimm, H. (2011b). The Faint "Heartbeats" of IGR J17091-3624: An Exceptional Black Hole Candidate. *ApJ*, 742:L17.

Altamirano, D., Casella, P., Patruno, A., Wijnands, R., and van der Klis, M. (2008a). Intermittent Millisecond X-Ray Pulsations from the Neutron Star X-Ray Transient SAX J1748.9-2021 in the Globular Cluster NGC 6440. *ApJ*, 674:L45.

Altamirano, D., Linares, M., van der Klis, M., Wijnands, R., Kalamkar, M., Casella, P., Watts, A., Patruno, A., Armas-Padilla, M., Cavecchi, Y., Degenaar, N., Kaur, R., Yang, Y., and Rea, N. (2011c). Discovery of 10 mHz quasi-periodic oscillations likely from IGR J17091-3624. *The Astronomer's Telegram*, 3225.

Altamirano, D., van der Klis, M., Méndez, M., Jonker, P. G., Klein-Wolt, M., and Lewin, W. H. G. (2008b). X-Ray Time Variability Across the Atoll Source States of 4U 1636-53. *ApJ*, 685:436–450.

Altamirano, D., van der Klis, M., Wijnands, R., and Cumming, A. (2008c). Millihertz Oscillation Frequency Drift Predicts the Occurrence of Type I X-Ray Bursts. *ApJ*, 673:L35.

Altamirano, D., Wijnands, R., and Belloni, T. (2013). The black hole candidate IGR J17091-3624 going to quiescence. *The Astronomer's Telegram*, 5112.

Altamirano, D., Wijnands, R., Belloni, T., and Motta, S. (2012). The black hole candidate IGR J17091-3624 is still active. *The Astronomer's Telegram*, 3913.

Angelini, L., White, N. E., and Stella, L. (1991). The discovery of an X-ray burst and a study of aperiodic variability from SMC X-1. *ApJ*, 371:332–341.

Aptekar, R. L., Butterworth, P. S., Cline, T. L., Frederiks, D. D., Golenetskii, S. V., Il'inskii, V. N., Mazets, E. P., Stilwell, D. E., and Terekhov, M. M. (1998). Hard X-Ray Bursts from GRO J1744-28. I. Observations by the Konus-Wind and Konus-A Experiments. *ApJ*, 493:404–407.

Aptekar, R. L., Butterworth, P. S., Golenetskii, S. V., Il'Inskii, V. N., Cline, T. L., Mazets, E. P., Stillwell, D. E., Terekhov, M. M., and Frederiks, D. D. (1997). Long-term activity of the bursting x-ray pulsar GRO J1744-28 from Konus-Wind and Konus-A observations. *Astronomy Letters*, 23:147–154.

Archibald, A. M., Kaspi, V. M., Bogdanov, S., Hessels, J. W. T., Stairs, I. H., Ransom, S. M., and McLaughlin, M. A. (2010). X-ray Variability and Evidence for Pulsations from the Unique Radio Pulsar/X-ray Binary Transition Object FIRST J102347.6+003841. *ApJ*, 722:88–95.

Archibald, A. M., Stairs, I. H., Ransom, S. M., Kaspi, V. M., Kondratiev, V. I., Lorimer, D. R., McLaughlin, M. A., Boyles, J., Hessels, J. W. T., Lynch, R., van Leeuwen, J., Roberts, M. S. E., Jenet, F., Champion, D. J., Rosen, R., Barlow, B. N., Dunlap, B. H., and Remillard, R. A. (2009). A Radio Pulsar/X-ray Binary Link. *Science*, 324:1411.

Arnaud, K. A. (1996). XSPEC: The First Ten Years. In Jacoby, G. H. and Barnes, J., editors, *Astronomical Data Analysis Software and Systems V*, volume 101 of *Astronomical Society of the Pacific Conference Series*, page 17.

Astropy Collaboration, Robitaille, T. P., Tollerud, E. J., Greenfield, P., Droettboom, M., Bray, E., Aldcroft, T., Davis, M., Ginsburg, A., Price-Whelan, A. M., Kerzendorf, W. E., Conley, A., Crighton, N., Barbary, K., Muna, D., Ferguson, H., Grollier, F., Parikh, M. M., Nair, P. H., Unther, H. M., Deil, C., Woillez, J., Conseil, S., Kramer, R., Turner, J. E. H., Singer, L., Fox, R., Weaver, B. A., Zabalza, V., Edwards, Z. I., Azalee

Bostroem, K., Burke, D. J., Casey, A. R., Crawford, S. M., Dencheva, N., Ely, J., Jenness, T., Labrie, K., Lim, P. L., Pierfederici, F., Pontzen, A., Ptak, A., Refsdal, B., Servillat, M., and Streicher, O. (2013). Astropy: A community Python package for astronomy. *A&A*, 558:A33.

Atwood, W. B., Abdo, A. A., Ackermann, M., Althouse, W., Anderson, B., Axelsson, M., Baldini, L., Ballet, J., Band, D. L., Barbiellini, G., and et al. (2009). The Large Area Telescope on the Fermi Gamma-Ray Space Telescope Mission. *ApJ*, 697:1071–1102.

Backer, D. C., Kulkarni, S. R., Heiles, C., Davis, M. M., and Goss, W. M. (1982). A millisecond pulsar. *Nature*, 300:615–618.

Bagnoli, T. and in't Zand, J. J. M. (2015). Discovery of GRS 1915+105 variability patterns in the Rapid Burster. *MNRAS*, 450:L52–L56.

Bagnoli, T., in't Zand, J. J. M., D'Angelo, C. R., and Galloway, D. K. (2015). A population study of type II bursts in the Rapid Burster. *MNRAS*, 449:268–287.

Bagnoli, T., in't Zand, J. J. M., Galloway, D. K., and Watts, A. L. (2013). Indications for a slow rotator in the Rapid Burster from its thermonuclear bursting behaviour. *MNRAS*, 431:1947–1955.

Barthelmy, S. D. (2000). Burst Alert Telescope (BAT) on the Swift MIDEX mission. In Flanagan, K. A. and Siegmund, O. H., editors, *X-Ray and Gamma-Ray Instrumentation for Astronomy XI*, volume 4140 of *Proc. SPIE*, pages 50–63.

Bazzano, A., Bird, A. J., Kuulkers, E., Sidoli, L., Fiocchi, V. S. M., Natalucci, L., Tarana, A., Ubertini, P., Capitanio, F., Del Santo, M., Sabatini, S., Tavani, M., Bassani, L., Malizia, A., Drave, S. P., and Winkler, C. (2011). Announcement of INTEGRAL Galactic Plane monitoring program and detection of 2 new hard X-ray sources. *The Astronomer's Telegram*, 3361.

Bazzano, A., Stephen, J. B., Fiocchi, M., Bird, A. J., Bassani, L., Dean, A. J., Malizia, A., Ubertini, P., Lebrun, F., Walter, R., and Winkler, C. (2006). INTEGRAL IBIS Census of the Sky Beyond 100 keV. *ApJ*, 649:L9–L12.

Belloni, T. and Hasinger, G. (1990). An atlas of aperiodic variability in HMXB. *A&A*, 230:103–119.

Belloni, T., Klein-Wolt, M., Méndez, M., van der Klis, M., and van Paradijs, J. (2000). A model-independent analysis of the variability of GRS 1915+105. *A&A*, 355:271–290.

Belloni, T., Méndez, M., King, A. R., van der Klis, M., and van Paradijs, J. (1997a). A Unified Model for the Spectral Variability in GRS 1915+105. *ApJ*, 488:L109–L112.

Belloni, T., Méndez, M., King, A. R., van der Klis, M., and van Paradijs, J. (1997b). An Unstable Central Disk in the Superluminal Black Hole X-Ray Binary GRS 1915+105. *ApJ*, 479:L145–L148.

Belloni, T., Psaltis, D., and van der Klis, M. (2002). A Unified Description of the Timing Features of Accreting X-Ray Binaries. *ApJ*, 572:392–406.

Belloni, T. M. and Motta, S. E. (2016). Transient Black Hole Binaries. *ArXiv e-prints*.

Bignami, G. F., Villa, G. E., Boella, G., Bonelli, G., Caraveo, P., Chiappetti, L., Quadrini, M. E., Di Cocco, G., Trifoglio, M., and Ubertini, P. (1990). European Photon Imaging Camera (EPIC) for X-ray astronomy. In Siegmund, O. H. W. and Hudson, H. S., editors, *EUV, X-ray, and Gamma-ray instrumentation for astronomy*, volume 1344 of *Proc. SPIE*, pages 144–153.

Bildsten, L. and Brown, E. F. (1997). Thermonuclear Burning on the Accreting X-Ray Pulsar GRO J1744-28. *ApJ*, 477:897–904.

Bildsten, L., Chakrabarty, D., Chiu, J., Finger, M. H., Koh, D. T., Nelson, R. W., Prince, T. A., Rubin, B. C., Scott, D. M., Stollberg, M., Vaughan, B. A., Wilson, C. A., and Wilson, R. B. (1997). Observations of Accreting Pulsars. *ApJS*, 113:367–408.

Bird, A. J., Bazzano, A., Malizia, A., Fiocchi, M., Sguera, V., Bassani, L., Hill, A. B., Ubertini, P., and Winkler, C. (2016). The IBIS Soft Gamma-Ray Sky after 1000 Integral Orbits. *ApJS*, 223:15.

Blackburn, J. K. (1995). FTOOLS: A FITS Data Processing and Analysis Software Package. In Shaw, R. A., Payne, H. E., and Hayes, J. J. E., editors, *Astronomical Data Analysis Software and Systems IV*, volume 77 of *Astronomical Society of the Pacific Conference Series*, page 367.

Bogdanov, S., Archibald, A. M., Bassa, C., Deller, A. T., Halpern, J. P., Heald, G., Hessels, J. W. T., Janssen, G. H., Lyne, A. G., Moldón, J., Paragi, Z., Patruno, A., Perera, B. B. P., Stappers, B. W., Tendulkar, S. P., D'Angelo, C. R., and Wijnands, R. (2015). Coordinated X-Ray, Ultraviolet, Optical, and Radio Observations of the PSR J1023+0038 System in a Low-mass X-Ray Binary State. *ApJ*, 806:148.

Bogdanov, S. and Halpern, J. P. (2015). Identification of the High-energy Gamma-Ray Source 3FGL J1544.6-1125 as a Transitional Millisecond Pulsar Binary in an Accreting State. *ApJ*, 803:L27.

Boldt, E. (1987). The cosmic X-ray background. In Hewitt, A., Burbidge, G., and Fang, L. Z., editors, *Observational Cosmology*, volume 124 of *IAU Symposium*, pages 611–615.

Bradt, H. V., Rothschild, R. E., and Swank, J. H. (1993). X-ray timing explorer mission. *A&AS*, 97:355–360.

Bradt, H. V., Swank, J. H., and Rothschild, R. E. (1990). The X-ray Timing Explorer. *Advances in Space Research*, 10:297–310.

Brinkman, A. C., van Rooijen, J. J., Bleeker, J. A. M., Dijkstra, J. H., and Heise, J. (1986). Low energy X-ray transmission grating spectrometer for AXAF. In Culhane, J. L., editor, *X-ray instrumentation in astronomy*, volume 597 of *Proc. SPIE*, pages 232–240.

Burrows, D. N., Hill, J. E., Nousek, J. A., Wells, A. A., Short, A. T., Ambrosi, R. M., Chincarini, G., Citterio, O., and Tagliaferri, G. (2003). Swift x-ray telescope (XRT). In Truemper, J. E. and Tananbaum, H. D., editors, *X-Ray and Gamma-Ray Telescopes and Instruments for Astronomy.*, volume 4851 of *Proc. SPIE*, pages 1320–1325.

Campana, S., Colpi, M., Mereghetti, S., Stella, L., and Tavani, M. (1998). The neutron stars of Soft X-ray Transients. *A&A Rev.*, 8:279–316.

Campana, S., Stella, L., Mereghetti, S., and de Martino, D. (2017). A universal relation for the propeller mechanisms in magnetic rotating stars at different scales. *ArXiv e-prints*.

Capitanio, F., Del Santo, M., Bozzo, E., Ferrigno, C., De Cesare, G., and Paizis, A. (2012). The peculiar 2011 outburst of the black hole candidate IGR J17091-3624, a GRS 1915+105-like source? *MNRAS*, 422:3130–3141.

Castro-Tirado, A. J., Brandt, S., and Lund, N. (1992). GRS 1915+105. *IAU Circ.*, 5590.

Chakrabarty, D., Bildsten, L., Finger, M. H., Grunsfeld, J. M., Koh, D. T., Nelson, R. W., Prince, T. A., Vaughan, B. A., and Wilson, R. B. (1997a). On the Correlation of Torque and Luminosity in GX 1+4. *ApJ*, 481:L101–L105.

Chakrabarty, D., Bildsten, L., Grunsfeld, J. M., Koh, D. T., Prince, T. A., Vaughan, B. A., Finger, M. H., Scott, D. M., and Wilson, R. B. (1997b). Torque Reversal and Spin-down of the Accretion-powered Pulsar 4U 1626-67. *ApJ*, 474:414–425.

Chakrabarty, D., Jonker, P. G., and Markwardt, C. B. (2014). Chandra Localization and Detection of a Burst and Pulsations from GRO J1744-28. *The Astronomer's Telegram*, 5895.

Chen, X., Swank, J. H., and Taam, R. E. (1996). The Pattern of Correlated X-ray Timing and Spectral Behavior in GRS 1915+105. In *American Astronomical Society Meeting Abstracts*, volume 28 of *Bulletin of the American Astronomical Society*, page 1315.

Cole, D. M., Vanden Berk, D. E., Severson, S. A., Miller, M. C., Quashnock, J. M., Nichol, R. C., Lamb, D. Q., Hurley, K., Blanco, P., Lidman, C., and Glazebrook, K. (1997). Optical/Near-Infrared Observations of GRO J1744-28. *ApJ*, 480:377–382.

Court, J. (2017). jmcourt/pantheon: Zenodo release.

Court, J., Altamirano, D., and Sanna, A. (2018a). The Bursting Pulsar GRO J1744-28: the Slowest Transitional Pulsar? *ArXiv e-prints*.

Court, J. M. C., Altamirano, D., Albayati, A. C., Overton, T., Sanna, A., Degenaar, N., Yamaoka, K., Belloni, T., and Blaes, O. ((in prep) 2018b). A Population Study of X-Ray bursts in the "Bursting Pulsar" GRO J1744-28. *MNRAS*.

Court, J. M. C., Altamirano, D., Pereyra, M., Boon, C. M., Yamaoka, K., Belloni, T., Wijnands, R., and Pahari, M. (2017). An atlas of exotic variability in IGR J17091-3624: a comparison with GRS 1915+105. *MNRAS*, 468:4748–4771.

Courvoisier, T. J.-L., Walter, R., Beckmann, V., Dean, A. J., Dubath, P., Hudec, R., Kretschmar, P., Mereghetti, S., Montmerle, T., Mowlavi, N., Paltani, S., Preite Martinez, A., Produit, N., Staubert, R., Strong, A. W., Swings, J.-P., Westergaard, N. J., White, N., Winkler, C., and Zdziarski, A. A. (2003). The INTEGRAL Science Data Centre (ISDC). *A&A*, 411:L53–L57.

D'Aì, A., Burderi, L., Di Salvo, T., Iaria, R., Pintore, F., Riggio, A., and Sanna, A. (2016). Discovery of hard phase lags in the pulsed emission of GRO J1744-28. *MNRAS*, 463:L84–L88.

D'Aì, A., Di Salvo, T., Iaria, R., García, J. A., Sanna, A., Pintore, F., Riggio, A., Burderi, L., Bozzo, E., Dauser, T., Matranga, M., Galiano, C. G., and Robba, N. R. (2015). GRO J1744-28: an intermediate B-field pulsar in a low-mass X-ray binary. *MNRAS*, 449:4288–4303.

Daigne, F., Goldoni, P., Ferrando, P., Goldwurm, A., Decourchelle, A., and Warwick, R. S. (2002). XMM-Newton observation of the bursting pulsar GRO J1744-28 in quiescence. *A&A*, 386:531–534.

D'Amico, F., Heindl, W. A., Rothschild, R. E., and Gruber, D. E. (2001). High-Energy X-Ray Timing Experiment Detections of Hard X-Ray Tails in Scorpius X-1. *ApJ*, 547:L147–L150.

D'Angelo, C. R. and Spruit, H. C. (2010). Episodic accretion on to strongly magnetic stars. *MNRAS*, 406:1208–1219.

D'Angelo, C. R. and Spruit, H. C. (2012). Accretion discs trapped near corotation. *MNRAS*, 420:416–429.

De Martino, D., Belloni, T., Falanga, M., Papitto, A., Motta, S., Pellizzoni, A., Evangelista, Y., Piano, G., Masetti, N., Bonnet-Bidaud, J.-M., Mouchet, M., Mukai, K., and Possenti, A. (2013). X-ray follow-ups of XSS J12270-4859: a low-mass X-ray binary with gamma-ray Fermi-LAT association. *A&A*, 550:A89.

Degenaar, N., Miller, J. M., Harrison, F. A., Kennea, J. A., Kouveliotou, C., and Younes, G. (2014a). High-resolution X-Ray Spectroscopy of the Bursting Pulsar GRO J1744-28. *ApJ*, 796:L9.

Degenaar, N., Wijnands, R., Cackett, E. M., Homan, J., in't Zand, J. J. M., Kuulkers, E., Maccarone, T. J., and van der Klis, M. (2012). A four-year XMM-Newton/Chandra monitoring campaign of the Galactic centre: analysing the X-ray transients. *A&A*, 545:A49.

Degenaar, N., Wijnands, R., Reynolds, M. T., Miller, J. M., Altamirano, D., Kennea, J., Gehrels, N., Haggard, D., and Ponti, G. (2014b). The Peculiar Galactic Center Neutron Star X-Ray Binary XMM J174457-2850.3. *ApJ*, 792:109.

den Herder, J.-W., Aarts, H. J., van den Berg, M. L., Bixler, J. V., den Boggende, A. J., Branduardi-Raymont, G., Brinkman, A. C., Decker, T. A., Dubbeldam, L., Hailey, C. J., Jansen, F. A., Kahn, S. M., de Korte, P. A., Mauche, C. W., Montesanti, R. C., Paerels, F. B., Spruijt, H., Thomsen, K., Verhoeve, P., and Zehnder, A. (1994). Reflection grating spectrometer onboard the ESA x-ray multi-mirror (XMM) mission. In Cerutti-Maori, M. G. and Roussel, P., editors, *Space Optics 1994: Earth Observation and Astronomy*, volume 2209 of *Proc. SPIE*, pages 451–462.

Di Salvo, T., Robba, N. R., Iaria, R., Stella, L., Burderi, L., and Israel, G. L. (2001a). Detection of a Hard Tail in the X-Ray Spectrum of the Z Source GX 349+2. *ApJ*, 554:49–55.

Di Salvo, T., Stella, L., Robba, N. R., Burderi, L., and van der Klis, M. (2001b). The discovery of a hard X-ray component in the horizontal branch spectrum of the Z source GX 17+2. In Gimenez, A., Reglero, V., and Winkler, C., editors, *Exploring the Gamma-Ray Universe*, volume 459 of *ESA Special Publication*, pages 341–344.

Done, C., Wardziński, G., and Gierliński, M. (2004). GRS 1915+105: the brightest Galactic black hole. *MNRAS*, 349:393–403.

Doroshenko, R., Santangelo, A., Doroshenko, V., Suleimanov, V., and Piraino, S. (2015). BeppoSAX observations of GRO J1744-28: cyclotron line detection and the softening of the burst spectra. *MNRAS*, 452:2490–2499.

Doxsey, R., Bradt, H., Johnston, M., Gursky, H., Schwartz, D. A., and Schwarz, J. (1978). Position for the rapid burster MXB 1730-335 determined with the scanning modulation collimator on HEAO 1. *ApJ*, 221:L53–L55.

Drave, S. P., Fiocchi, M., Sguera, V., Bazzano, A., Bird, A. J., Sidoli, L., and Kuulker, E. (2012). A possible state change in the black-hole candidate IGR J17091-3624. *The Astronomer's Telegram*, 3916.

Eckert, D., Del Santo, M., Bazzano, A., Watanabe, K., Paizis, A., Bozzo, E., Ferrigno, C., Caballero, I., Sidoli, L., and Kuiper, L. (2013). IGR J18245-2452: a new hard X-ray transient discovered by INTEGRAL. *The Astronomer's Telegram*, 4925.

Evans, P. A., Beardmore, A. P., Page, K. L., Osborne, J. P., O'Brien, P. T., Willingale, R., Starling, R. L. C., Burrows, D. N., Godet, O., Vetere, L., Racusin, J., Goad, M. R., Wiersema, K., Angelini, L., Capalbi, M., Chincarini, G., Gehrels, N., Kennea, J. A., Margutti, R., Morris, D. C., Mountford, C. J., Pagani, C., Perri, M., Romano, P., and Tanvir, N. (2009). Methods and results of an automatic analysis of a complete sample of Swift-XRT observations of GRBs. *MNRAS*, 397:1177–1201.

Evans, P. A., Beardmore, A. P., Page, K. L., Tyler, L. G., Osborne, J. P., Goad, M. R., O'Brien, P. T., Vetere, L., Racusin, J., Morris, D., Burrows, D. N., Capalbi, M., Perri, M., Gehrels, N., and Romano, P. (2007). An online repository of Swift/XRT light curves of γ-ray bursts. *A&A*, 469:379–385.

Fanson, J. L., Fazio, G. G., Houck, J. R., Kelly, T., Rieke, G. H., Tenerelli, D. J., and Whitten, M. (1998). Space Infrared Telescope Facility (SIRTF). In Bely, P. Y. and Breckinridge, J. B., editors, *Space Telescopes and Instruments V*, volume 3356 of *Proc. SPIE*, pages 478–491.

Fender, R. (2006). *Jets from X-ray binaries*, pages 381–419.

Fender, R. and Belloni, T. (2004). GRS 1915+105 and the Disc-Jet Coupling in Accreting Black Hole Systems. *ARA&A*, 42:317–364.

Ferrigno, C., Bozzo, E., Papitto, A., Rea, N., Pavan, L., Campana, S., Wieringa, M., Filipović, M., Falanga, M., and Stella, L. (2014). Hiccup accretion in the swinging pulsar IGR J18245-2452. *A&A*, 567:A77.

Finger, M. H., Koh, D. T., Nelson, R. W., Prince, T. A., Vaughan, B. A., and Wilson, R. B. (1996a). Discovery of hard X-ray pulsations from the transient source GRO J1744 - 28. *Nature*, 381:291–293.

Finger, M. H., Wilson, R. B., Harmon, B. A., Hagedon, K., and Prince, T. A. (1996b). GRO J1744-28. *IAU Circ.*, 6285.

Fiocchi, M. T., Natalucci, L., and GPS Team (2012). The INTEGRAL Galactic Plane Scanning. In *Proceedings of "An INTEGRAL view of the high-energy sky (the first 10 years)" - 9th INTEGRAL Workshop and celebration of the 10th anniversary of the launch (INTEGRAL 2012). 15-19 October 2012. Bibliotheque Nationale de France, Paris, France. Published online at* `http://pos.sissa.it/cgi-bin/reader/conf.cgi?confid=176`*, id.82*, page 82.

Fishman, G. J., Kouveliotou, C., van Paradijs, J., Harmon, B. A., Paciesas, W. S., Briggs, M. S., Kommers, J., and Lewin, W. H. G. (1995). Galactic Center. *IAU Circ.*, 6272.

Fourier, J. (1822). *Théorie analytique de la chaleur*. Chez Firmin Didot, père et fils.

Frank, J., King, A., and Raine, D. J. (2002). *Accretion Power in Astrophysics: Third Edition*.

Fruscione, A., McDowell, J. C., Allen, G. E., Brickhouse, N. S., Burke, D. J., Davis, J. E., Durham, N., Elvis, M., Galle, E. C., Harris, D. E., Huenemoerder, D. P., Houck, J. C., Ishibashi, B., Karovska, M., Nicastro, F., Noble, M. S., Nowak, M. A., Primini, F. A., Siemiginowska, A., Smith, R. K., and Wise, M. (2006). CIAO: Chandra's data analysis system. In *Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series*, volume 6270 of *Proc. SPIE*, page 62701V.

Galloway, D. K. and in't Zand, J. J. M. (2010). Type-II bursts from the new Terzan 5 transient: a GRO J1744-28 analogue? *The Astronomer's Telegram*, 3000.

Galloway, D. K., Muno, M. P., Hartman, J. M., Psaltis, D., and Chakrabarty, D. (2008). Thermonuclear (Type I) X-Ray Bursts Observed by the Rossi X-Ray Timing Explorer. *ApJS*, 179:360–422.

Garmire, G. P., Bautz, M. W., Ford, P. G., Nousek, J. A., and Ricker, Jr., G. R. (2003). Advanced CCD imaging spectrometer (ACIS) instrument on the Chandra X-ray Observatory. In Truemper, J. E. and Tananbaum, H. D., editors, *X-Ray and Gamma-Ray Telescopes and Instruments for Astronomy.*, volume 4851 of *Proc. SPIE*, pages 28–44.

Gehrels, N. (2004). The Swift Gamma-Ray Burst Mission. In Schoenfelder, V., Lichti, G., and Winkler, C., editors, *5th INTEGRAL Workshop on the INTEGRAL Universe*, volume 552 of *ESA Special Publication*, page 777.

Gehrels, N., Chipman, E., and Kniffen, D. (1994). The Compton Gamma Ray Observatory. *ApJS*, 92:351–362.

Ghosh, A. and Chakrabarti, S. K. (2014). Periods of two enigmatic black hole candidates GRS 1915+105 and IGR J17091-3624. *ArXiv e-prints*.

Giacconi, R., Kellogg, E., Gorenstein, P., Gursky, H., and Tananbaum, H. (1971). An X-Ray Scan of the Galactic Plane from UHURU. *ApJ*, 165:L27.

Giles, A. B., Swank, J. H., Jahoda, K., Zhang, W., Strohmayer, T., Stark, M. J., and Morgan, E. H. (1996). The Main Characteristics of GRO J1744-28 Observed by the Proportional Counter Array Experiment on the Rossi X-Ray Timing Explorer. *ApJ*, 469:L25.

Gimenez, A. and Mas-Hesse, J. M. (1998). Optical monitoring camera on board the ESA high-energy mission INTEGRAL. In Doty, F. P. and Hoover, R. B., editors, *Hard X-Ray and Gamma-Ray Detector Physics and Applications*, volume 3446 of *Proc. SPIE*, pages 257–265.

Giorgini, J. D., Yeomans, D. K., Chamberlin, A. B., Chodas, P. W., Jacobson, R. A., Keesey, M. S., Lieske, J. H., Ostro, S. J., Standish, E. M., and Wimberly, R. N. (1997). JPL's On-Line Solar System Ephemeris and Data Service. In Bietenholz, M. F., Bartel, N., Rupen, M. P., Beasley, A. J., Graham, D. A., Altunin, V. I., Venturi, T., Umana, G., and Conway, J. E., editors, *Bulletin of the American Astronomical Society*, volume 29 of *Bulletin of the American Astronomical Society*, page 1099.

Gosling, A. J., Bandyopadhyay, R. M., Miller-Jones, J. C. A., and Farrell, S. A. (2007). GRO J1744-28, search for the counterpart: infrared photometry and spectroscopy. *MNRAS*, 380:1511–1520.

Graessle, D. E., Evans, I. N., Glotfelty, K., He, X. H., Evans, J. D., Rots, A. H., Fabbiano, G., and Brissenden, R. J. (2006). The Chandra X-ray Observatory calibration database

(CalDB): building, planning, and improving. In *Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series*, volume 6270 of *Proc. SPIE*, page 62701X.

Gruber, D. E., Blanco, P. R., Heindl, W. A., Pelling, M. R., Rothschild, R. E., and Hink, P. L. (1996). The high energy X-ray timing experiment on XTE. *A&AS*, 120:641–644.

Hannikainen, D. C., Hjalmarsdotter, L., Rodriguez, J., Vilhu, O., Zdziarski, A. A., and Belloni, T. (2007). Identifying a New Class of Variability in GRS 1915+105. In *ESA Special Publication*, volume 622 of *ESA Special Publication*, page 353.

Harrison, F. A., Craig, W. W., Christensen, F. E., Hailey, C. J., Zhang, W. W., Boggs, S. E., Stern, D., Cook, W. R., Forster, K., Giommi, P., Grefenstette, B. W., Kim, Y., Kitaguchi, T., Koglin, J. E., Madsen, K. K., Mao, P. H., Miyasaka, H., Mori, K., Perri, M., Pivovaroff, M. J., Puccetti, S., Rana, V. R., Westergaard, N. J., Willis, J., Zoglauer, A., An, H., Bachetti, M., Barrière, N. M., Bellm, E. C., Bhalerao, V., Brejnholt, N. F., Fuerst, F., Liebe, C. C., Markwardt, C. B., Nynka, M., Vogel, J. K., Walton, D. J., Wik, D. R., Alexander, D. M., Cominsky, L. R., Hornschemeier, A. E., Hornstrup, A., Kaspi, V. M., Madejski, G. M., Matt, G., Molendi, S., Smith, D. M., Tomsick, J. A., Ajello, M., Ballantyne, D. R., Baloković, M., Barret, D., Bauer, F. E., Blandford, R. D., Brandt, W. N., Brenneman, L. W., Chiang, J., Chakrabarty, D., Chenevez, J., Comastri, A., Dufour, F., Elvis, M., Fabian, A. C., Farrah, D., Fryer, C. L., Gotthelf, E. V., Grindlay, J. E., Helfand, D. J., Krivonos, R., Meier, D. L., Miller, J. M., Natalucci, L., Ogle, P., Ofek, E. O., Ptak, A., Reynolds, S. P., Rigby, J. R., Tagliaferri, G., Thorsett, S. E., Treister, E., and Urry, C. M. (2013). The Nuclear Spectroscopic Telescope Array (NuSTAR) High-energy X-Ray Mission. *ApJ*, 770:103.

Hayakawa, S. (1985). X-rays from accreting neutron stars. *Phys. Rep.*, 121:317–406.

Heger, A., Cumming, A., and Woosley, S. E. (2007). Millihertz Quasi-periodic Oscillations from Marginally Stable Nuclear Burning on an Accreting Neutron Star. *ApJ*, 665:1311–1320.

Heidke, P. (1926). Berechnung des erfolges und der gute der windstarkvorhersagen im sturmwarnungsdienst. *Geogr. Ann.*, 8:301349.

Hill, A. B., Szostek, A., Corbel, S., Camilo, F., Corbet, R. H. D., Dubois, R., Dubus, G., Edwards, P. G., Ferrara, E. C., Kerr, M., Koerding, E., Kozieł, D., and Stawarz, Ł. (2011). The bright unidentifiedγ-ray source 1FGL J1227.9-4852: can it be associated with a low-mass X-ray binary? *MNRAS*, 415:235–243.

Hirose, S., Krolik, J. H., and Blaes, O. (2009). Radiation-Dominated Disks are Thermally Stable. *ApJ*, 691:16–31.

Hoffman, J. A., Marshall, H. L., and Lewin, W. H. G. (1978). Dual character of the rapid burster and a classification of X-ray bursts. *Nature*, 271:630–633.

Holtzman, J. A., Hester, J. J., Casertano, S., Trauger, J. T., Watson, A. M., Ballester, G. E., Burrows, C. J., Clarke, J. T., Crisp, D., Evans, R. W., Gallagher, III, J. S., Griffiths, R. E., Hoessel, J. G., Matthews, L. D., Mould, J. R., Scowen, P. A., Stapelfeldt, K. R., and Westphal, J. A. (1995). The performance and calibration of WFPC2 on the Huble Space Telescope. *PASP*, 107:156–178.

Hunter, J. D. (2007). Matplotlib: A 2d graphics environment. *Computing In Science & Engineering*, 9(3):90–95.

Huppenkothen, D., Heil, L. M., Hogg, D. W., and Mueller, A. (2017). Using machine learning to explore the long-term evolution of GRS 1915+105. *MNRAS*, 466:2364–2377.

Huppenkothen, D., Younes, G., Ingram, A., Kouveliotou, C., Göğüş, E., Bachetti, M., Sánchez-Fernández, C., Chenevez, J., Motta, S., van der Klis, M., Granot, J., Gehrels, N., Kuulkers, E., Tomsick, J. A., and Walton, D. J. (2016). Detection of Very Low-Frequency Quasi-Periodic Oscillations in the 2015 Outburst of V404 Cygni. *ArXiv e-prints*.

Iaria, R., Gambino, A. F., Di Salvo, T., Burderi, L., Matranga, M., Riggio, A., Sanna, A., Scarano, F., and D'Aı, A. (2018). A possible solution of the puzzling variation of the orbital period of MXB 1659-298. *MNRAS*, 473:3490–3499.

Ibarra, A., Calle, I., Gabriel, C., Salgado, J., and Osuna, P. (2009). XMM-Newton Science Analysis Software: How to Bring New Technologies to Long-life Satellite Missions. In Bohlender, D. A., Durand, D., and Dowler, P., editors, *Astronomical Data Analysis Software and Systems XVIII*, volume 411 of *Astronomical Society of the Pacific Conference Series*, page 322.

Ibragimov, A. and Poutanen, J. (2009). Accreting millisecond pulsar SAX J1808.4-3658 during its 2002 outburst: evidence for a receding disc. *MNRAS*, 400:492–508.

Icke, V. (1979). Disk accretion in a soft potential well. *A&A*, 78:21–24.

Illarionov, A. F. and Sunyaev, R. A. (1975). Why the Number of Galactic X-ray Stars Is so Small? *A&A*, 39:185.

Irwin, A. W., Campbell, B., Morbey, C. L., Walker, G. A. H., and Yang, S. (1989). Long-period radial-velocity variations of Arcturus. *PASP*, 101:147–159.

Iyer, N., Nandi, A., and Mandal, S. (2015a). Determination of the Mass of IGR J17091-3624 from "Spectro-temporal" Variations during the Onset Phase of the 2011 Outburst. *ApJ*, 807:108.

Iyer, N., Nandi, A., and Mandal, S. (2015b). Estimating the mass of IGR J17091-3624: statistical challenges and methods. In *Astronomical Society of India Conference Series*, volume 12 of *Astronomical Society of India Conference Series*.

Jahoda, K., Markwardt, C. B., Radeva, Y., Rots, A. H., Stark, M. J., Swank, J. H., Strohmayer, T. E., and Zhang, W. (2006). Calibration of the Rossi X-Ray Timing Explorer Proportional Counter Array. *ApJS*, 163:401–423.

Jahoda, K., Swank, J. H., Giles, A. B., Stark, M. J., Strohmayer, T., Zhang, W., and Morgan, E. H. (1996). In-orbit performance and calibration of the Rossi X-ray Timing Explorer (RXTE) Proportional Counter Array (PCA). In Siegmund, O. H. and Gummin, M. A., editors, *EUV, X-Ray, and Gamma-Ray Instrumentation for Astronomy VII*, volume 2808 of *Proc. SPIE*, pages 59–70.

Janiuk, A. and Czerny, B. (2005). Time-delays between the soft and hard X-ray bands in GRS 1915+105. *MNRAS*, 356:205–216.

Janiuk, A., Czerny, B., and Siemiginowska, A. (2000). Radiation Pressure Instability as a Variability Mechanism in the Microquasar GRS 1915+105. *ApJ*, 542:L33–L36.

Jansen, F., Lumb, D., Altieri, B., Clavel, J., Ehle, M., Erd, C., Gabriel, C., Guainazzi, M., Gondoin, P., Much, R., Munoz, R., Santos, M., Schartel, N., Texier, D., and Vacanti, G. (2001). XMM-Newton observatory. I. The spacecraft and operations. *A&A*, 365:L1–L6.

Johnson, T. J., Ray, P. S., Roy, J., Cheung, C. C., Harding, A. K., Pletsch, H. J., Fort, S., Camilo, F., Deneva, J., Bhattacharyya, B., Stappers, B. W., and Kerr, M. (2015). Discovery of Gamma-Ray Pulsations from the Transitional Redback PSR J1227-4853. *ApJ*, 806:91.

Jones, E., Oliphant, T., Peterson, P., et al. (2001). SciPy: Open source scientific tools for Python.

Kelley, R. L., Audley, M. D., Boyce, K. R., Breon, S. R., Fujimoto, R., Gendreau, K. C., Holt, S. S., Ishisaki, Y., McCammon, D., Mihara, T., Mitsuda, K., Moseley, S. H., Mott, D. B., Porter, F. S., Stahle, C. K., and Szymkowiak, A. E. (1999). ASTRO-E high-resolution x-ray spectrometer. In Siegmund, O. H. and Flanagan, K. A., editors, *EUV, X-Ray, and Gamma-Ray Instrumentation for Astronomy X*, volume 3765 of *Proc. SPIE*, pages 114–127.

Kennea, J. A. and Capitanio, F. (2007). Swift/XRT Observations of IGR J17091-3624 and IGR J17098-3628. *The Astronomer's Telegram*, 1140.

Kennea, J. A., Kouveliotou, C., and Younes, G. (2014). GRO J1744-28: Swift XRT confirmation of outburst. *The Astronomer's Telegram*, 5845.

Kenney, J. (1939). *Mathematics of Statistics*. Number v. 1 in Mathematics of Statistics. D. Van Nostrand Company, Incorporated.

Kenter, A. T., Chappell, J. H., Kraft, R. P., Meehan, G. R., Murray, S. S., Zombeck, M. V., Hole, K. T., Juda, M., Donnelly, R. H., Patnaude, D., Pease, D. O., Wilton, C., Zhao, P., Austin, G. K., Fraser, G. W., Pearson, J. F., Lees, J. E., Brunton, A. N., Barbera, M.,

Collura, A., and Serio, S. (2000). In-flight performance and calibration of the Chandra high-resolution camera imager (HRC-I). In Truemper, J. E. and Aschenbach, B., editors, *X-Ray Optics, Instruments, and Missions III*, volume 4012 of *Proc. SPIE*, pages 467–492.

King, A. L., Miller, J. M., Raymond, J., Fabian, A. C., Reynolds, C. S., Kallman, T. R., Maitra, D., Cackett, E. M., and Rupen, M. P. (2012). An Extreme X-Ray Disk Wind in the Black Hole Candidate IGR J17091-3624. *ApJ*, 746:L20.

King, A. L., Miller, J. M., Raymond, J., Reynolds, M. T., and Morningstar, W. (2015). High-resolution Chandra HETG Spectroscopy of V404 Cygni in Outburst. *ApJ*, 813:L37.

Klein-Wolt, M., Fender, R. P., Pooley, G. G., Belloni, T., MigliariF, S., Morgan, E. H., and van der Klis, M. (2002). Hard X-ray states and radio emission in GRS 1915+105. *MNRAS*, 331:745–764.

Kok, C. (2000). *On the Behaviour of a Few Popular Verification Scores in Yes No Forecasting*. Scientific report. Koninklijk Nederlands Meteorologisch Insutuut.

Kommers, J. M., Fox, D. W., Lewin, W. H. G., Rutledge, R. E., van Paradijs, J., and Kouveliotou, C. (1997). Postburst Quasi-periodic Oscillations from GRO J1744-28 and from the Rapid Burster. *ApJ*, 482:L53–L56.

Kouveliotou, C., van Paradijs, J., Fishman, G. J., Briggs, M. S., Kommers, J., Harmon, B. A., Meegan, C. A., and Lewin, W. H. G. (1996). A new type of transient high-energy source in the direction of the Galactic Centre. *Nature*, 379:799–801.

Koyama, K., Tsunemi, H., Dotani, T., Bautz, M. W., Hayashida, K., Tsuru, T. G., Matsumoto, H., Ogawara, Y., Ricker, G. R., Doty, J., Kissel, S. E., Foster, R., Nakajima, H., Yamaguchi, H., Mori, H., Sakano, M., Hamaguchi, K., Nishiuchi, M., Miyata, E., Torii, K., Namiki, M., Katsuda, S., Matsuura, D., Miyauchi, T., Anabuki, N., Tawa, N., Ozaki, M., Murakami, H., Maeda, Y., Ichikawa, Y., Prigozhin, G. Y., Boughan, E. A., Lamarr, B., Miller, E. D., Burke, B. E., Gregory, J. A., Pillsbury, A., Bamba, A., Hiraga, J. S., Senda, A., Katayama, H., Kitamoto, S., Tsujimoto, M., Kohmura, T., Tsuboi, Y., and Awaki, H. (2007). X-Ray Imaging Spectrometer (XIS) on Board Suzaku. *PASJ*, 59:23–33.

Kraft, R. P., Chappell, J. H., Kenter, A. T., Meehan, G. R., Murray, S. S., Zombeck, M. V., Donnelly, R. H., Drake, J. J., Johnson, C. O., Juda, M., Patnaude, D., Pease, D. O., Ratzlaff, P. W., Wargelin, B. J., Zhao, P., Austin, G. K., Fraser, G. W., Pearson, J. F., Lees, J. E., Brunton, A. N., Barbera, M., Collura, A., and Serio, S. (2000). In-flight performance and calibration of the Chandra high-resolution camera spectroscopic readout (HRC-S). In Truemper, J. E. and Aschenbach, B., editors, *X-Ray Optics, Instruments, and Missions III*, volume 4012 of *Proc. SPIE*, pages 493–517.

Kreykenbohm, I., Wilms, J., Kretschmar, P., Torrejón, J. M., Pottschmidt, K., Hanke, M., Santangelo, A., Ferrigno, C., and Staubert, R. (2008). High variability in Vela X-1: giant flares and off states. *A&A*, 492:511–525.

Krimm, H. A., Barthelmy, S. D., Baumgartner, W., Cummings, J., Fenimore, E., Gehrels, N., Kennea, J. A., Markwardt, C. B., Palmer, D., Sakamoto, T., Skinner, G., Stamatikos, M., Tueller, J., and Ukwatta, T. (2011). Swift/BAT reports renewed activity from IGR J17091-3624. *The Astronomer's Telegram*, 3144.

Krimm, H. A., Holland, S. T., Corbet, R. H. D., Pearlman, A. B., Romano, P., Kennea, J. A., Bloom, J. S., Barthelmy, S. D., Baumgartner, W. H., Cummings, J. R., Gehrels, N., Lien, A. Y., Markwardt, C. B., Palmer, D. M., Sakamoto, T., Stamatikos, M., and Ukwatta, T. N. (2013). The Swift/BAT Hard X-Ray Transient Monitor. *ApJS*, 209:14.

Krimm, H. A. and Kennea, J. A. (2011). Swift/XRT Observations Confirm that IGR J17091-3624 is in Outburst. *The Astronomer's Telegram*, 3148:1.

Krivonos, R., Tsygankov, S., Lutovinov, A., Revnivtsev, M., Churazov, E., and Sunyaev, R. (2015). INTEGRAL 11-year hard X-ray survey above 100 keV. *MNRAS*, 448:3766–3774.

Kuulkers, E., Homan, J., van der Klis, M., Lewin, W. H. G., and Méndez, M. (2002). X-ray bursts at extreme mass accretion rates from GX 17+2. *A&A*, 382:947–973.

Kuulkers, E., Lutovinov, A., Parmar, A., Capitanio, F., Mowlavi, N., and Hermsen, W. (2003). Igr J17091-3624. *The Astronomer's Telegram*, 149.

Kuulkers, E., Shaw, S., Paizis, A., Mowlavi, N., Courvoisier, T., Ebisawa, K., Kretschmar, P., Markwardt, C., Oosterbroek, T., Orr, A., and Wijnands, R. (2005). Announcement of INTEGRAL Galactic Bulge monitoring program and (re)brightening of GRO J1655-40. *The Astronomer's Telegram*, 438.

Kuulkers, E., Shaw, S. E., Paizis, A., Chenevez, J., Brandt, S., Courvoisier, T. J.-L., Domingo, A., Ebisawa, K., Kretschmar, P., Markwardt, C. B., Mowlavi, N., Oosterbroek, T., Orr, A., Rísquez, D., Sanchez-Fernandez, C., and Wijnands, R. (2007). The INTEGRAL Galactic bulge monitoring program: the first 1.5 years. *A&A*, 466:595–618.

Labanti, C., Di Cocco, G., Malaguti, G., Mauri, A., Rossi, E., Schiavone, F., and Traci, A. (1996). PICsIT: the high-energy detection plane of the IBIS instrument onboard INTEGRAL. In Ramsey, B. D. and Parnell, T. A., editors, *Gamma-Ray and Cosmic-Ray Detectors, Techniques, and Missions*, volume 2806 of *Proc. SPIE*, pages 269–279.

Lam, S. K., Pitrou, A., and Seibert, S. (2015). Numba: a llvm-based python jit compiler. In *LLVM '15 Proceedings of the Second Workshop on the LLVM Compiler Infrastructure in HPC*.

Lamb, D. Q., Miller, M. C., and Taam, R. E. (1996). GRO J1744-28: Last Gasps of a Dying Low-mass X-ray Binary. *ArXiv Astrophysics e-prints*.

Leahy, D. A., Darbro, W., Elsner, R. F., Weisskopf, M. C., Kahn, S., Sutherland, P. G., and Grindlay, J. E. (1983). On searches for pulsed emission with application to four globular cluster X-ray sources - NGC 1851, 6441, 6624, and 6712. *ApJ*, 266:160–170.

Lebrun, F., Blondel, C., Fondeur, I., Goldwurm, A., Laurent, P., and Leray, J. P. (1996). ISGRI: a CdTe array imager for INTEGRAL. In Ramsey, B. D. and Parnell, T. A., editors, *Gamma-Ray and Cosmic-Ray Detectors, Techniques, and Missions*, volume 2806 of *Proc. SPIE*, pages 258–268.

Leiter, D. (1983). Electron-positron processes and spectral evolution in black hole accretion disk dynamo models for AGN sources of the cosmic X-ray and gamma ray backgrounds. In Burns, M. L., Harding, A. K., and Ramaty, R., editors, *Positron-Electron Pairs in Astrophysics*, volume 101 of *American Institute of Physics Conference Series*, pages 337–342.

Lemiere, A., Terrier, R., Jouvin, L., Marandon, V., and Khelifi, B. (2015). Study of the VHE diffuse emission in the central 200 pc of our Galaxy with H.E.S.S. In *34th International Cosmic Ray Conference (ICRC2015)*, volume 34 of *International Cosmic Ray Conference*, page 838.

Levine, A. M., Bradt, H., Cui, W., Jernigan, J. G., Morgan, E. H., Remillard, R., Shirey, R. E., and Smith, D. A. (1996). First Results from the All-Sky Monitor on the Rossi X-Ray Timing Explorer. *ApJ*, 469:L33.

Lewin, W. H. G., Clark, G., and Doty, J. (1976a). X-Ray Bursts. *IAU Circ.*, 2922.

Lewin, W. H. G., Doty, J., Clark, G. W., Rappaport, S. A., Bradt, H. V. D., Doxsey, R., Hearn, D. R., Hoffman, J. A., Jernigan, J. G., Li, F. K., Mayer, W., McClintock, J., Primini, F., and Richardson, J. (1976b). The discovery of rapidly repetitive X-ray bursts from a new source in Scorpius. *ApJ*, 207:L95–L99.

Lewin, W. H. G., Ricker, G. R., and McClintock, J. E. (1971). X-Rays from a New Variable Source GX 1+4. *ApJ*, 169:L17.

Lewin, W. H. G., Rutledge, R. E., Kommers, J. M., van Paradijs, J., and Kouveliotou, C. (1996). A Comparison between the Rapid Burster and GRO J1744-28. *ApJ*, 462:L39.

Lewin, W. H. G., van Paradijs, J., and Taam, R. E. (1993). X-Ray Bursts. *Space Sci. Rev.*, 62:223–389.

Lewin, W. H. G., van Paradijs, J., and van der Klis, M. (1988). A review of quasi-periodic oscillations in low-mass X-ray binaries. *Space Sci. Rev.*, 46:273–378.

Lightman, A. P. and Eardley, D. M. (1974). Black Holes in Binary Systems: Instability of Disk Accretion. *ApJ*, 187:L1.

Lin, D., Remillard, R. A., and Homan, J. (2009). Spectral States of XTE J1701 - 462: Link Between Z and Atoll Sources. *ApJ*, 696:1257–1277.

Linares, M., Altamirano, D., Chakrabarty, D., Cumming, A., and Keek, L. (2012). Millihertz Quasi-periodic Oscillations and Thermonuclear Bursts from Terzan 5: A Showcase of Burning Regimes. *ApJ*, 748:82.

Linares, M., Kennea, J., Krimm, H., and Kouveliotou, C. (2014). Swift detects bursting activity from GRO J1744-28. *The Astronomer's Telegram*, 5883.

Liu, Q. Z., van Paradijs, J., and van den Heuvel, E. P. J. (2007). A catalogue of low-mass X-ray binaries in the Galaxy, LMC, and SMC (Fourth edition). *A&A*, 469:807–810.

Lomb, N. R. (1976). Least-squares frequency analysis of unequally spaced data. *Ap&SS*, 39:447–462.

Lubiński, P. (2009). Analysis of extremely low signal-to-noise ratio data from INTEGRAL/PICsIT. *A&A*, 496:557–576.

Maccarone, T. J. (2003). Do X-ray binary spectral state transition luminosities vary? *A&A*, 409:697–706.

Madsen, K. K., Forster, K., Grefenstette, B. W., Harrison, F. A., and Stern, D. (2017). Measurement of the Absolute Crab Flux with NuSTAR. *ApJ*, 841:56.

Makishima, K., Maejima, Y., Mitsuda, K., Bradt, H. V., Remillard, R. A., Tuohy, I. R., Hoshi, R., and Nakagawa, M. (1986). Simultaneous X-ray and optical observations of GX 339-4 in an X-ray high state. *ApJ*, 308:635–643.

Manchester, R. N., Hobbs, G. B., Teoh, A., and Hobbs, M. (2005). The Australia Telescope National Facility Pulsar Catalogue. *AJ*, 129:1993–2006.

Markert, T. H., Canizares, C. R., Dewey, D., McGuirk, M., Pak, C. S., and Schattenburg, M. L. (1994). High-Energy Transmission Grating Spectrometer for the Advanced X-ray Astrophysics Facility (AXAF). In Siegmund, O. H. and Vallerga, J. V., editors, *EUV, X-Ray, and Gamma-Ray Instrumentation for Astronomy V*, volume 2280 of *Proc. SPIE*, pages 168–180.

Markwardt, C. B., Swank, J. H., and Taam, R. E. (1999). Variable-Frequency Quasi-periodic Oscillations from the Galactic Microquasar GRS 1915+105. *ApJ*, 513:L37–L40.

Marshall, H., Grindlay, J., and Weisskopf, M. (1979). Observations of the Rapid Burster (MXB1730-335) Using the EINSTEIN Observatory. In *Bulletin of the American Astronomical Society*, volume 11 of *Bulletin of the American Astronomical Society*, page 788.

Masetti, N., D'Avanzo, P., Blagorodnova, N., and Palazzi, E. (2014). The near-infrared counterpart of GRO J1744-28. *The Astronomer's Telegram*, 5999.

Mason, K. O., Cropper, M. S., Hunt, R., Horner, S. D., Priedhorsky, W. C., Ho, C., Cordova, F. A., Jamar, C. A., and Antonello, E. (1996). XMM Optical/UV Monitor Telescope. In Siegmund, O. H. and Gummin, M. A., editors, *EUV, X-Ray, and Gamma-Ray Instrumentation for Astronomy VII*, volume 2808 of *Proc. SPIE*, pages 438–447.

Massa, F., Massaro, E., Mineo, T., D'Aì, A., Feroci, M., Casella, P., and Belloni, T. (2013). The complex behaviour of the microquasar GRS 1915+105 in the ρ class observed with BeppoSAX. III. The hard X-ray delay and limit cycle mapping. *A&A*, 556:A84.

Massaro, E., Ardito, A., Ricciardi, P., Massa, F., Mineo, T., and D'Aì, A. (2014a). Non-linear oscillator models for the X-ray bursting of the microquasar GRS 1915+105. *Ap&SS*, 352:699–714.

Massaro, E., Ardito, A., Ricciardi, P., Massa, F., Mineo, T., and D'Aì, A. (2014b). Non-linear oscillator models for the X-ray bursting of the microquasar GRS 1915+105. *Ap&SS*, 352:699–714.

Massaro, E., Ventura, G., Massa, F., Feroci, M., Mineo, T., Cusumano, G., Casella, P., and Belloni, T. (2010). The complex behaviour of the microquasar GRS 1915+105 in the ρ class observed with BeppoSAX. I. Timing analysis. *A&A*, 513:A21.

McClintock, J. E. and Rappaport, S. A. (1985). Low-mass X-ray binaries. In Lamb, D. Q. and Patterson, J., editors, *Cataclysmic Variables and Low-Mass X-ray Binaries*, volume 113 of *Astrophysics and Space Science Library*, pages 61–77.

McClintock, J. E. and Remillard, R. A. (2006). *Black hole binaries*, pages 157–213.

Merloni, A. and Nayakshin, S. (2006). On the limit-cycle instability in magnetized accretion discs. *MNRAS*, 372:728–734.

Miller, J. M., Reynolds, M., Kennea, J., King, A. L., and Tomsick, J. (2016). Renewed Activity in the Galactic Black Hole IGR J17091-3624. *The Astronomer's Telegram*, 8742.

Miller, M. C., Lamb, F. K., and Psaltis, D. (1996). Sonic-Point Model of Kilohertz QPOs in LMXBs. In *American Astronomical Society Meeting Abstracts*, volume 28 of *Bulletin of the American Astronomical Society*, page 1329.

Mineo, T., Massaro, E., D'Ai, A., Massa, F., Feroci, M., Ventura, G., Casella, P., Ferrigno, C., and Belloni, T. (2012). The complex behaviour of the microquasar GRS 1915+105 in the ρ class observed with BeppoSAX. II. Time-resolved spectral analysis. *A&A*, 537:A18.

Mir, M. H., Misra, R., Pahari, M., Iqbal, N., and Ahmad, N. (2016). A model for the energy-dependent time-lag and rms of the heartbeat oscillations in GRS 1915+105. *MNRAS*, 457:2999–3005.

Misner, C. W., Thorne, K. S., and Wheeler, J. A. (1973). *Gravitation*.

Mitsuda, K., Bautz, M., Inoue, H., Kelley, R. L., Koyama, K., Kunieda, H., Makishima, K., Ogawara, Y., Petre, R., Takahashi, T., Tsunemi, H., White, N. E., Anabuki, N., Angelini, L., Arnaud, K., Awaki, H., Bamba, A., Boyce, K., Brown, G. V., Chan, K.-W., Cottam, J., Dotani, T., Doty, J., Ebisawa, K., Ezoe, Y., Fabian, A. C., Figueroa, E., Fujimoto, R., Fukazawa, Y., Furusho, T., Furuzawa, A., Gendreau, K., Griffiths, R. E., Haba, Y.,

Hamaguchi, K., Harrus, I., Hasinger, G., Hatsukade, I., Hayashida, K., Henry, P. J., Hiraga, J. S., Holt, S. S., Hornschemeier, A., Hughes, J. P., Hwang, U., Ishida, M., Ishisaki, Y., Isobe, N., Itoh, M., Iyomoto, N., Kahn, S. M., Kamae, T., Katagiri, H., Kataoka, J., Katayama, H., Kawai, N., Kilbourne, C., Kinugasa, K., Kissel, S., Kitamoto, S., Kohama, M., Kohmura, T., Kokubun, M., Kotani, T., Kotoku, J., Kubota, A., Madejski, G. M., Maeda, Y., Makino, F., Markowitz, A., Matsumoto, C., Matsumoto, H., Matsuoka, M., Matsushita, K., McCammon, D., Mihara, T., Misaki, K., Miyata, E., Mizuno, T., Mori, K., Mori, H., Morii, M., Moseley, H., Mukai, K., Murakami, H., Murakami, T., Mushotzky, R., Nagase, F., Namiki, M., Negoro, H., Nakazawa, K., Nousek, J. A., Okajima, T., Ogasaka, Y., Ohashi, T., Oshima, T., Ota, N., Ozaki, M., Ozawa, H., Parmar, A. N., Pence, W. D., Porter, F. S., Reeves, J. N., Ricker, G. R., Sakurai, I., Sanders, W. T., Senda, A., Serlemitsos, P., Shibata, R., Soong, Y., Smith, R., Suzuki, M., Szymkowiak, A. E., Takahashi, H., Tamagawa, T., Tamura, K., Tamura, T., Tanaka, Y., Tashiro, M., Tawara, Y., Terada, Y., Terashima, Y., Tomida, H., Torii, K., Tsuboi, Y., Tsujimoto, M., Tsuru, T. G., Turner, M. J. L. ., Ueda, Y., Ueno, S., Ueno, M., Uno, S., Urata, Y., Watanabe, S., Yamamoto, N., Yamaoka, K., Yamasaki, N. Y., Yamashita, K., Yamauchi, M., Yamauchi, S., Yaqoob, T., Yonetoku, D., and Yoshida, A. (2007). The X-Ray Observatory Suzaku. *PASJ*, 59:1–7.

Morgan, E. H., Remillard, R. A., and Greiner, J. (1997). RXTE Observations of QPOs in the Black Hole Candidate GRS 1915+105. *ApJ*, 482:993–1010.

Murray, S. S., Chappell, J. H., Elvis, M. S., Forman, W. R., and Grindlay, J. E. (1987). The AXAF high resolution camera (HRC) and its use for observations of distant clusters of galaxies. *Astrophysical Letters and Communications*, 26:113–125.

Nayakshin, S., Rappaport, S., and Melia, F. (2000). Time-dependent Disk Models for the Microquasar GRS 1915+105. *ApJ*, 535:798–814.

Negoro, H., Sugizaki, M., Krimm, H. A., Kennea, J. A., Ueno, S., Tomida, H., Nakahira, S., Kimura, M., Ishikawa, M., Nakagawa, Y. E., Mihara, T., Serino, M., Morii, M., Sugimoto, J., Takagi, T., Yoshikawa, A., Matsuoka, M., Kawai, N., Usui, R., Ishikawa, K., Yoshii, T., Tachibana, Y., Yoshida, A., Sakamoto, T., Nakano, Y., Kawakubo, Y., Ohtsuki, H., Tsunemi, H., Sasaki, M., Uchida, D., Nakajima, M., Sakakibara, H., Fukushima, K., Onodera, T., Suzuki, K., Ueda, Y., Shidatsu, M., Kawamuro, T., Hori, T., Tsuboi, Y., Higa, M., Kawagoe, A., Yamauchi, M., Yoshidome, K., Ogawa, Y., Yamada, H., Morooka, Y., and Yamaoka, K. (2014). MAXI/GSC and Swift/BAT detection of enhanced hard X-ray emission from the Galactic center region, renewed activity of GRO J1744-28 ? *The Astronomer's Telegram*, 5790.

Negri, L. H. and Vestri, C. (2017). lucashn/peakutils: v1.1.0.

Neilsen, J., Remillard, R. A., and Lee, J. C. (2011). The Physics of the "Heartbeat" State of GRS 1915+105. *ApJ*, 737:69.

Neilsen, J., Remillard, R. A., and Lee, J. C. (2012). Radiation Pressure and Mass Ejection in ρ-like States of GRS 1915+105. *ApJ*, 750:71.

Nobili, L. (2003). A Disk-Jet Interaction Model for the X-Ray Variability in Microquasars. *ApJ*, 582:954–958.

Nousek, J. A., Garmire, G. P., Ricker, G. R., Collins, S. A., and Reigler, G. R. (1987). The AXAF CCD Imaging Spectrometer Experiment (ACIS). *Astrophysical Letters and Communications*, 26:35–41.

Novikov, I. D. and Thorne, K. S. (1973). Astrophysics of black holes. In Dewitt, C. and Dewitt, B. S., editors, *Black Holes (Les Astres Occlus)*, pages 343–450.

Paciesas, W. S., Harmon, B. A., Fishman, G. J., Zhang, S. N., and Robinson, C. R. (1996). Galactic Center. *IAU Circ.*, 6284.

Paczynski, B. (1979). Evolution of binary X-ray systems. In Baity, W. A. and Peterson, L. E., editors, *X-ray Astronomy*, pages 251–255.

Pahari, M., Bhattacharyya, S., Yadav, J. S., and Pandey, S. K. (2012). Evidence of two unique variability classes from IGR J17091-3624. *MNRAS*, 422:L87.

Pahari, M., Neilsen, J., Yadav, J. S., Misra, R., and Uttley, P. (2013a). Comparison of Time/Phase Lags in the Hard State and Plateau State of GRS 1915+105. *ApJ*, 778:136.

Pahari, M. and Pal, S. (2009). Discovery of one new class in the light curve of GRS 1915+105. *ArXiv e-prints*.

Pahari, M., Yadav, J. S., and Bhattacharyya, S. (2014). X-Ray Spectral State Evolution in IGR J17091-3624 and Comparison of its Heartbeat Oscillation Properties with those of GRS 1915+105. *ApJ*, 783:141.

Pahari, M., Yadav, J. S., Rodriguez, J., Misra, R., Bhattacharyya, S., and Pandey, S. K. (2013b). Properties of Unique Hard X-Ray Dips Observed from GRS 1915+105 and IGR J17091-3624 and Their Implications. *ApJ*, 778:46.

Pandey-Pommier, M., Masetti, N., and Durouchoux, P. (2014). GRO J1744-28: search for radio counterpart with the GMRT. *The Astronomer's Telegram*, 5904.

Papitto, A. (2017). The discovery of transitional (and optical) millisecond pulsars. La Gomera Accretion Week 2017.

Papitto, A., Ferrigno, C., Bozzo, E., Rea, N., Pavan, L., Burderi, L., Burgay, M., Campana, S., di Salvo, T., Falanga, M., Filipović, M. D., Freire, P. C. C., Hessels, J. W. T., Possenti, A., Ransom, S. M., Riggio, A., Romano, P., Sarkissian, J. M., Stairs, I. H., Stella, L., Torres, D. F., Wieringa, M. H., and Wong, G. F. (2013a). Swings between rotation and accretion power in a binary millisecond pulsar. *Nature*, 501:517–520.

Papitto, A., Hessels, J. W. T., Burgay, M., Ransom, S., Rea, N., Possenti, A., Stairs, I., Ferrigno, C., and Bozz, E. (2013b). The transient low-mass X-ray binary IGR J18245-2452 is again active as a radio pulsar. *The Astronomer's Telegram*, 5069.

Patruno, A., Haskell, B., and Andersson, N. (2017). The Spin Distribution of Fast-spinning Neutron Stars in Low-mass X-Ray Binaries: Evidence for Two Subpopulations. *ApJ*, 850:106.

Patruno, A., Maitra, D., Curran, P. A., D'Angelo, C., Fridriksson, J. K., Russell, D. M., Middleton, M., and Wijnands, R. (2016). The Reflares and Outburst Evolution in the Accreting Millisecond Pulsar SAX J1808.4-3658: A Disk Truncated Near Co-Rotation? *ApJ*, 817:100.

Patruno, A., Watts, A., Klein Wolt, M., Wijnands, R., and van der Klis, M. (2009). 1 Hz Flaring in SAX J1808.4-3658: Flow Instabilities near the Propeller Stage. *ApJ*, 707:1296–1309.

Patruno, A. and Watts, A. L. (2012). Accreting Millisecond X-Ray Pulsars. *ArXiv e-prints*.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Pence, W. D. (1992). FTOOLS - A New Package of Programs to Manipulate and Process FITS Format Files. In *American Astronomical Society Meeting Abstracts #180*, volume 24 of *Bulletin of the American Astronomical Society*, page 750.

Poisson, S. (1837).

Poutanen, J. and Svensson, R. (1996). The Two-Phase Pair Corona Model for Active Galactic Nuclei and X-Ray Binaries: How to Obtain Exact Solutions. *ApJ*, 470:249.

Pretorius, M. L. (2009). Time-resolved optical observations of five cataclysmic variables detected by INTEGRAL. *MNRAS*, 395:386–393.

Pringle, J. E. (1981). Accretion discs in astrophysics. *ARA&A*, 19:137–162.

Psaltis, D. (2008). Probes and Tests of Strong-Field Gravity with Observations in the Electromagnetic Spectrum. *Living Reviews in Relativity*, 11:9.

Rao, A. and Vadawale, S. V. (2012). Why is IGR J17091-3624 So Faint? Constraints on Distance, Mass, and Spin from "Phase-resolved" Spectroscopy of the "Heartbeat" Oscillations. *ApJ*, 757:L12.

Rappaport, S. and Joss, P. C. (1997). The Nature and Evolutionary History of GRO J1744-28. *ApJ*, 486:435–444.

Rappaport, S., Markert, T., Li, F. K., Clark, G. W., Jernigan, J. G., and McClintock, J. E. (1977). Discovery of a 7.68 second X-ray periodicity in 3U 1626-67. *ApJ*, 217:L29–L33.

Rea, N., Zelati, F. C., Esposito, P., D'Avanzo, P., de Martino, D., Israel, G. L., Torres, D. F., Campana, S., Belloni, T. M., Papitto, A., Masetti, N., Carrasco, L., Possenti, A., Wieringa, M., Wilhelmi, E. D. O., Li, J., Bozzo, E., Ferrigno, C., Linares, M., Tauris, T. M., Hernanz, M., Ribas, I., Monelli, M., Borghese, A., Baglio, M. C., and Casares, J. (2017). Multiband study of RX J0838-2827 and XMM J083850.4-282759: a new asynchronous magnetic cataclysmic variable and a candidate transitional millisecond pulsar. *MNRAS*, 471:2902–2916.

Rebusco, P., Moskalik, P., Kluźniak, W., and Abramowicz, M. A. (2012). Period doubling and non-linear resonance in the black hole candidate IGR J17091-3624? *A&A*, 540:L4.

Reid, M. J., McClintock, J. E., Steiner, J. F., Steeghs, D., Remillard, R. A., Dhawan, V., and Narayan, R. (2014). A Parallax Distance to the Microquasar GRS 1915+105 and a Revised Estimate of its Black Hole Mass. *ApJ*, 796:2.

Remillard, R. A., McClintock, J. E., Sobczak, G. J., Bailyn, C. D., Orosz, J. A., Morgan, E. H., and Levine, A. M. (1999a). X-Ray Nova XTE J1550-564: Discovery of a Quasi-periodic Oscillation near 185 HZ. *ApJ*, 517:L127–L130.

Remillard, R. A., Morgan, E. H., McClintock, J. E., Bailyn, C. D., and Orosz, J. A. (1999b). RXTE Observations of 0.1-300 HZ Quasi-periodic Oscillationsin the Microquasar GRO J1655-40. *ApJ*, 522:397–412.

Revnivtsev, M., Churazov, E., Gilfanov, M., and Sunyaev, R. (2001). New class of low frequency QPOs: Signature of nuclear burning or accretion disk instabilities? *A&A*, 372:138–144.

Robitaille, T. and Bressert, E. (2012). APLpy: Astronomical Plotting Library in Python. Astrophysics Source Code Library.

Rodriguez, J., Corbel, S., Caballero, I., Tomsick, J. A., Tzioumis, T., Paizis, A., Cadolle Bel, M., and Kuulkers, E. (2011a). First simultaneous multi-wavelength observations of the black hole candidate IGR J17091-3624. ATCA, INTEGRAL, Swift, and RXTE views of the 2011 outburst. *A&A*, 533:L4.

Rodriguez, J., Corbel, S., and Tomsick, J. A. (2003). Spectral Evolution of the Microquasar XTE J1550-564 over Its Entire 2000 Outburst. *ApJ*, 595:1032–1038.

Rodriguez, J., Corbel, S., Tomsick, J. A., Paizis, A., and Kuulkers, E. (2011b). 0.1 Hz QPOs during RXTE observations of IGR J17091-3624. *The Astronomer's Telegram*, 3168.

Rodriguez, J., Hannikainen, D. C., Shaw, S. E., Pooley, G., Corbel, S., Tagger, M., Mirabel, I. F., Belloni, T., Cabanac, C., Cadolle Bel, M., Chenevez, J., Kretschmar, P., Lehto, H. J., Paizis, A., Varnière, P., and Vilhu, O. (2008). 2 Years of INTEGRAL Monitoring

of GRS 1915+105. I. Multiwavelength Coverage with INTEGRAL, RXTE, and the Ryle Radio Telescope. *ApJ*, 675:1436–1448.

Roming, P. W. A., Hunsberger, S. D., Nousek, J. A., Ivanushkina, M., Mason, K. O., and Breeveld, A. A. (2004). The Swift Ultra-Violet/Optical Telescope (UVOT). In Fenimore, E. and Galassi, M., editors, *Gamma-Ray Bursts: 30 Years of Discovery*, volume 727 of *American Institute of Physics Conference Series*, pages 651–654.

Roy, J., Ray, P. S., Bhattacharyya, B., Stappers, B., Chengalur, J. N., Deneva, J., Camilo, F., Johnson, T. J., Wolff, M., Hessels, J. W. T., Bassa, C. G., Keane, E. F., Ferrara, E. C., Harding, A. K., and Wood, K. S. (2015). Discovery of Psr J1227-4853: A Transition from a Low-mass X-Ray Binary to a Redback Millisecond Pulsar. *ApJ*, 800:L12.

Russell, T., Degenaar, N., Miller-Jones, J., and Tudor, V. (2017). Radio luminosity upper limits of the transient neutron star low-mass X-ray binary GRO J1744-28. *The Astronomer's Telegram*, 10106.

Sanna, A., Bahramian, A., Bozzo, E., Heinke, C., Altamirano, D., Wijnands, R., Degenaar, N., Maccarone, T., Riggio, A., Di Salvo, T., Iaria, R., Burgay, M., Possenti, A., Ferrigno, C., Papitto, A., Sivakoff, G., D'Amico, N., and Burderi, L. (2017a). Discovery of 105 Hz coherent pulsations in the ultracompact binary IGR J16597-3704. *ArXiv e-prints*.

Sanna, A., D'Ai, A., Bozzo, E., Riggio, A., Pintore, F., Burderi, L., Di Salvo, T., and Iaria, R. (2017b). Swift-XRT confirms the renewed X-ray activity of the Bursting Pulsar "GRO J1744-28". *The Astronomer's Telegram*, 10079.

Sanna, A., Riggio, A., Burderi, L., Pintore, F., Di Salvo, T., D'Aì, A., Bozzo, E., Esposito, P., Segreto, A., Scarano, F., Iaria, R., and Gambino, A. F. (2017c). Study of the accretion torque during the 2014 outburst of the X-ray pulsar GRO J1744-28. *MNRAS*, 469:2–12.

Sazonov, S. Y., Sunyaev, R. A., and Lund, N. (1997). Super-Eddington x-ray luminosity of the bursting pulsar GRO J1744-28: WATCH/Granat observations. *Astronomy Letters*, 23:286–292.

Sądowski, A. (2016). Magnetic flux stabilizing thin accretion discs. *MNRAS*, 462:960–965.

Scargle, J. D. (1982). Studies in astronomical time series analysis. II - Statistical aspects of spectral analysis of unevenly spaced data. *ApJ*, 263:835–853.

Scaringi, S., Maccarone, T. J., D'Angelo, C., Knigge, C., and Groot, P. J. (2017). Magnetically gated accretion in an accreting 'non-magnetic' white dwarf. *Nature*, 552:210–213.

Scaringi, S., Maccarone, T. J., Koerding, E., Knigge, C., Vaughan, S., Marsh, T. R., Aranzana, E., Dhillon, V., and Barros, S. C. C. (2015). Accretion-induced variability links young stellar objects, white dwarfs, and black holes. *ArXiv e-prints*.

Schnopper, H. W., Budtz-Joergensen, C. C., Westergaard, N. J., Hornstrup, A., Kamarainen, V. J., Huovelin, J., Vilhu, O., Costa, E., Piro, L., Frontera, F., Manzo, G., Giarrusso, S., Castro-Tirado, A. J., Reglero, V., Svensson, R., Fabian, A. C., Zdziarski, A., Morawski, M., Jahoda, K., Sunyaev, R., and Pavlinsky, M. P. (1996). Joint European x-ray monitor (JEM-X): x-ray monitor for ESA's INTEGRAL mission. In Ramsey, B. D. and Parnell, T. A., editors, *Gamma-Ray and Cosmic-Ray Detectors, Techniques, and Missions*, volume 2806 of *Proc. SPIE*, pages 297–307.

Shakura, N. I. and Sunyaev, R. A. (1973). Black holes in binary systems. Observational appearance. *A&A*, 24:337–355.

Spruit, H. C. and Taam, R. E. (1993). An instability associated with a magnetosphere-disk interaction. *ApJ*, 402:593–604.

Stappers, B. W., Archibald, A. M., Hessels, J. W. T., Bassa, C. G., Bogdanov, S., Janssen, G. H., Kaspi, V. M., Lyne, A. G., Patruno, A., Tendulkar, S., Hill, A. B., and Glanzman, T. (2014). A State Change in the Missing Link Binary Pulsar System PSR J1023+0038. *ApJ*, 790:39.

Stefanov, I. Z. (2014). Confronting models for the high-frequency QPOs with Lense-Thirring precession. *MNRAS*, 444:2178–2185.

Steiner, J. F., Narayan, R., McClintock, J. E., and Ebisawa, K. (2009). A Simple Comptonization Model. *PASP*, 121:1279.

Strader, J., Li, K.-L., Chomiuk, L., Heinke, C. O., Udalski, A., Peacock, M., Shishkovsky, L., and Tremou, E. (2016). A New γ-Ray Loud, Eclipsing Low-mass X-Ray Binary. *ApJ*, 831:89.

Strohmayer, T. and Bildsten, L. (2006). *New views of thermonuclear bursts*, pages 113–156.

Strohmayer, T. E., Jahoda, K., Giles, A. B., and Lee, U. (1997). Millisecond Pulsations from a Low-Mass X-Ray Binary in the Galactic Center Region. *ApJ*, 486:355–362.

Strüder, L., Briel, U., Dennerl, K., Hartmann, R., Kendziorra, E., Meidinger, N., Pfeffermann, E., Reppin, C., Aschenbach, B., Bornemann, W., Bräuninger, H., Burkert, W., Elender, M., Freyberg, M., Haberl, F., Hartner, G., Heuschmann, F., Hippmann, H., Kastelic, E., Kemmer, S., Kettenring, G., Kink, W., Krause, N., Müller, S., Oppitz, A., Pietsch, W., Popp, M., Predehl, P., Read, A., Stephan, K. H., Stötter, D., Trümper, J., Holl, P., Kemmer, J., Soltau, H., Stötter, R., Weber, U., Weichert, U., von Zanthier, C., Carathanassis, D., Lutz, G., Richter, R. H., Solc, P., Böttcher, H., Kuster, M., Staubert, R., Abbey, A., Holland, A., Turner, M., Balasini, M., Bignami, G. F., La Palombara, N., Villa, G., Buttler, W., Gianini, F., Lainé, R., Lumb, D., and Dhez, P. (2001). The European Photon Imaging Camera on XMM-Newton: The pn-CCD camera. *A&A*, 365:L18–L26.

Sturner, S. J. and Dermer, C. D. (1996). On the Nature of the Bursting X-Ray Pulsar GRO J1744-28. *ApJ*, 465:L31.

Sturner, S. J. and Shrader, C. R. (2005). XTE J1550-564: INTEGRAL Observations of a Failed Outburst. *ApJ*, 625:923–930.

Svensson, R. and Zdziarski, A. A. (1994). Black hole accretion disks with coronae. *ApJ*, 436:599–606.

Swank, J. H. (1996). Compact Star Time Scales. In *American Astronomical Society Meeting Abstracts*, volume 28 of *Bulletin of the American Astronomical Society*, page 1342.

Szkody, P., Fraser, O., Silvestri, N., Henden, A., Anderson, S. F., Frith, J., Lawton, B., Owens, E., Raymond, S., Schmidt, G., Wolfe, M., Bochanski, J., Covey, K., Harris, H., Hawley, S., Knapp, G. R., Margon, B., Voges, W., Walkowicz, L., Brinkmann, J., and Lamb, D. Q. (2003). Cataclysmic Variables from the Sloan Digital Sky Survey. II. The Second Year. *AJ*, 126:1499–1514.

Taam, R. E. and Lin, D. N. C. (1984). The evolution of the inner regions of viscous accretion disks surrounding neutron stars. *ApJ*, 287:761–768.

Tagger, M., Varnière, P., Rodriguez, J., and Pellat, R. (2004). Magnetic Floods: A Scenario for the Variability of the Microquasar GRS 1915+105. *ApJ*, 607:410–419.

Takahashi, T., Abe, K., Endo, M., Endo, Y., Ezoe, Y., Fukazawa, Y., Hamaya, M., Hirakuri, S., Hong, S., Horii, M., Inoue, H., Isobe, N., Itoh, T., Iyomoto, N., Kamae, T., Kasama, D., Kataoka, J., Kato, H., Kawaharada, M., Kawano, N., Kawashima, K., Kawasoe, S., Kishishita, T., Kitaguchi, T., Kobayashi, Y., Kokubun, M., Kotoku, J., Kouda, M., Kubota, A., Kuroda, Y., Madejski, G., Makishima, K., Masukawa, K., Matsumoto, Y., Mitani, T., Miyawaki, R., Mizuno, T., Mori, K., Mori, M., Murashima, M., Murakami, T., Nakazawa, K., Niko, H., Nomachi, M., Okada, Y., Ohno, M., Oonuki, K., Ota, N., Ozawa, H., Sato, G., Shinoda, S., Sugiho, M., Suzuki, M., Taguchi, K., Takahashi, H., Takahashi, I., Takeda, S., Tamura, K.-I., Tamura, T., Tanaka, T., Tanihata, C., Tashiro, M., Terada, Y., Tominaga, S., Uchiyama, Y., Watanabe, S., Yamaoka, K., Yanagida, T., and Yonetoku, D. (2007). Hard X-Ray Detector (HXD) on Board Suzaku. *PASJ*, 59:35–51.

Takahashi, T., Kokubun, M., Mitsuda, K., Kelley, R., Ohashi, T., Aharonian, F., Akamatsu, H., Akimoto, F., Allen, S., Anabuki, N., and et al. (2016). The ASTRO-H (Hitomi) x-ray astronomy satellite. In *Space Telescopes and Instrumentation 2016: Ultraviolet to Gamma Ray*, volume 9905 of *Proc. SPIE*, page 99050U.

Tan, J., Lewin, W. H. G., Lubin, L. M., van Paradijs, J., Penninx, W., van der Klis, M., Damen, E., and Stella, L. (1991). The Rapid Burster and its type II burst profiles. *MNRAS*, 251:1–23.

Tomsick, J. A., Kalemci, E., Corbel, S., and Kaaret, P. (2003). X-Ray Flares and Oscillations from the Black Hole Candidate X-Ray Transient XTE J1650-500 at Low Luminosity. *ApJ*, 592:1100–1109.

Turner, M. J. L., Abbey, A., Arnaud, M., Balasini, M., Barbera, M., Belsole, E., Bennie, P. J., Bernard, J. P., Bignami, G. F., Boer, M., Briel, U., Butler, I., Cara, C., Chabaud, C., Cole, R., Collura, A., Conte, M., Cros, A., Denby, M., Dhez, P., Di Coco, G., Dowson, J., Ferrando, P., Ghizzardi, S., Gianotti, F., Goodall, C. V., Gretton, L., Griffiths, R. G., Hainaut, O., Hochedez, J. F., Holland, A. D., Jourdain, E., Kendziorra, E., Lagostina, A., Laine, R., La Palombara, N., Lortholary, M., Lumb, D., Marty, P., Molendi, S., Pigot, C., Poindron, E., Pounds, K. A., Reeves, J. N., Reppin, C., Rothenflug, R., Salvetat, P., Sauvageot, J. L., Schmitt, D., Sembay, S., Short, A. D. T., Spragg, J., Stephen, J., Strüder, L., Tiengo, A., Trifoglio, M., Trümper, J., Vercellone, S., Vigroux, L., Villa, G., Ward, M. J., Whitehead, S., and Zonca, E. (2001). The European Photon Imaging Camera on XMM-Newton: The MOS cameras : The MOS cameras. *A&A*, 365:L27–L35.

Ubertini, P., Lebrun, F., Di Cocco, G., Bazzano, A., Bird, A. J., Broenstad, K., Goldwurm, A., La Rosa, G., Labanti, C., Laurent, P., Mirabel, I. F., Quadrini, E. M., Ramsey, B., Reglero, V., Sabau, L., Sacco, B., Staubert, R., Vigroux, L., Weisskopf, M. C., and Zdziarski, A. A. (2003). IBIS: The Imager on-board INTEGRAL. *A&A*, 411:L131–L139.

Uchiyama, Y., Maeda, Y., Ebara, M., Fujimoto, R., Ishisaki, Y., Ishida, M., Iizuka, R., Ushio, M., Inoue, H., Okada, S., Mori, H., and Ozaki, M. (2008). Restoring the Suzaku Source Position Accuracy and Point-Spread Function. *PASJ*, 60:S35–S42.

Ueda, Y., Ishioka, R., Sekiguchi, K., Ribo, M., Rodriguez, J., Chaty, S., Greiner, J., Sala, G., Fuchs, Y., Goldoni, P., Covino, S., Pooley, G. G., Edwards, P., Tzioumis, A., Lehto, H., Gerard, E., Colom, P., Martin, J., Trushkin, S. A., Castro-Tirado, A. J., Hannikainen, D., Sudo, H., Honma, M., Iwamuro, F., Kubuta, K., Yamaoka, K., Done, C., Naik, S., Fukazawa, Y., Angelini, L., Awaki, H., Ebisawa, K., Iwasawa, K., Kawai, N., Kinugasa, K., Kokubun, M., Kotani, T., Kubota, A., Murakami, T., Namiki, M., Takahashi, H., Yaqoob, T., Yonetoku, D., and Yoshida, A. (2006). The 2005 October Multiwavelength Campaign of GRS 1915+105. In *VI Microquasar Workshop: Microquasars and Beyond*, page 23.1.

van den Eijnden, J., Bagnoli, T., Degenaar, N., Lohfink, A. M., Parker, M. L., in 't Zand, J. J. M., and Fabian, A. C. (2017). A strongly truncated inner accretion disc in the Rapid Burster. *MNRAS*, 466:L98–L102.

van den Heuvel, E. P. J. (1993). Formation and evolution of pulsars. In Phillips, J. A., Thorsett, S. E., and Kulkarni, S. R., editors, *Planets Around Pulsars*, volume 36 of *Astronomical Society of the Pacific Conference Series*, pages 123–147.

van der Klis, M. (1989). Quasi-periodic oscillations and noise in low-mass X-ray binaries. *ARA&A*, 27:517–553.

van der Klis, M. (1997). Kilohertz Quasi-Periodic Oscillations in Low-Mass X-Ray Binaries. In Maoz, D., Sternberg, A., and Leibowitz, E. M., editors, *Astronomical Time Series*, volume 218 of *Astrophysics and Space Science Library*, page 121.

Van der Klis, M. (2004). A review of rapid X-ray variability in X-ray binaries. *ArXiv Astrophysics e-prints*.

Vaughan, S., Edelson, R., Warwick, R. S., and Uttley, P. (2003). On characterizing the variability properties of X-ray light curves from active galaxies. *MNRAS*, 345:1271–1284.

Vedrenne, G., Roques, J.-P., Schönfelder, V., Mandrou, P., Lichti, G. G., von Kienlin, A., Cordier, B., Schanne, S., Knödlseder, J., Skinner, G., Jean, P., Sanchez, F., Caraveo, P., Teegarden, B., von Ballmoos, P., Bouchet, L., Paul, P., Matteson, J., Boggs, S., Wunderer, C., Leleux, P., Weidenspointner, G., Durouchoux, P., Diehl, R., Strong, A., Cassé, M., Clair, M. A., and André, Y. (2003). SPI: The spectrometer aboard INTEGRAL. *A&A*, 411:L63–L70.

Vignarca, F., Migliari, S., Belloni, T., Psaltis, D., and van der Klis, M. (2003). Tracing the power-law component in the energy spectrum of black hole candidates as a function of the QPO frequency. *A&A*, 397:729–738.

Vilhu, O. (1999). Super-Eddington Accretion in GRS 1915+105. In Poutanen, J. and Svensson, R., editors, *High Energy Processes in Accreting Black Holes*, volume 161 of *Astronomical Society of the Pacific Conference Series*, page 82.

von Mises, R. (1964). *Mathematical Theory of Probability and Statistics*.

Walker, M. A. (1992). Radiation Dynamics in X-Ray Binaries. II. Type 2 Bursts. *ApJ*, 385:651.

Walter, R., Rohlfs, R., Meharga, M. T., Binko, P., Morisset, N., Beck, M., Produit, N., Pavan, L., Savchenko, V., Ferrigno, C., Frankowski, A., and Bordas, P. (2010). Integral in Heavens. In *Eighth Integral Workshop. The Restless Gamma-ray Universe (INTEGRAL 2010)*, page 162.

Weisskopf, M. C. (1999). The Chandra X-Ray Observatory (CXO): An Overview. *ArXiv Astrophysics e-prints*.

White, N. E. and Zhang, W. (1997). Millisecond X-Ray Pulsars in Low-mass X-Ray Binaries. *ApJ*, 490:L87–L90.

Wijnands, R., Méndez, M., Markwardt, C., van der Klis, M., Chakrabarty, D., and Morgan, E. (2001). The Erratic Luminosity Behavior of SAX J1808.4-3658 during Its 2000 Outburst. *ApJ*, 560:892–896.

Wijnands, R. and van der Klis, M. (1998). A millisecond pulsar in an X-ray binary system. *Nature*, 394:344–346.

Wijnands, R. and Wang, Q. D. (2002). A Chandra Observation of GRO J1744-28: The Bursting Pulsar in Quiescence. *ApJ*, 568:L93–L96.

Wijnands, R., Yang, Y. J., and Altamirano, D. (2012). The enigmatic black hole candidate and X-ray transient IGR J17091-3624 in its quiescent state as seen with XMM-Newton. *MNRAS*, 422:91–95.

Wilms, J., Allen, A., and McCray, R. (2000). On the Absorption of X-Rays in the Interstellar Medium. *ApJ*, 542:914–924.

Winkler, C. (1996). INTEGRAL. The international gamma-ray astrophysics laboratory. *A&AS*, 120:637–640.

Winkler, C., Courvoisier, T. J.-L., Di Cocco, G., Gehrels, N., Giménez, A., Grebenev, S., Hermsen, W., Mas-Hesse, J. M., Lebrun, F., Lund, N., Palumbo, G. G. C., Paul, J., Roques, J.-P., Schnopper, H., Schönfelder, V., Sunyaev, R., Teegarden, B., Ubertini, P., Vedrenne, G., and Dean, A. J. (2003). The INTEGRAL mission. *A&A*, 411:L1–L6.

Woods, P. M., Kouveliotou, C., van Paradijs, J., Briggs, M. S., Wilson, C. A., Deal, K., Harmon, B. A., Fishman, G. J., Lewin, W. H. G., and Kommers, J. (1999). Properties of the Second Outburst of the Bursting Pulsar (GRO J1744-28) as Observed with BATSE. *ApJ*, 517:431–435.

Woods, P. M., Kouveliotou, C., van Paradijs, J., Koshut, T. M., Finger, M. H., Briggs, M. S., Fishman, G. J., and Lewin, W. H. G. (2000). Detailed Analysis of the Pulsations during and after Bursts from the Bursting Pulsar (GRO J1744-28). *ApJ*, 540:1062–1068.

Xue, L., Sądowski, A., Abramowicz, M. A., and Lu, J.-F. (2011). Studies of Thermally Unstable Accretion Disks Around Black Holes with Adaptive Pseudospectral Domain Decomposition Method. II. Limit-cycle Behavior in Accretion Disks around Kerr Black Holes. *ApJS*, 195:7.

Yadav, J. S., Agrawal, P. C., Paul, B., Rao, A. R., Seetha, S., and Kasturirangan, K. (2000). Different Types of X-ray Bursts During High State of the Superluminal Source GRS 1915+105. *Advances in Space Research*, 25:441–444.

Younes, G., Kouveliotou, C., Grefenstette, B. W., Tomsick, J. A., Tennant, A., Finger, M. H., Fürst, F., Pottschmidt, K., Bhalerao, V., Boggs, S. E., Boirin, L., Chakrabarty, D., Christensen, F. E., Craig, W. W., Degenaar, N., Fabian, A. C., Gandhi, P., Göğüş, E., Hailey, C. J., Harrison, F. A., Kennea, J. A., Miller, J. M., Stern, D., and Zhang, W. W. (2015). Simultaneous NuSTAR/Chandra Observations of the Bursting Pulsar GRO J1744-28 during Its Third Reactivation. *ApJ*, 804:43.

Zdziarski, A. A., Johnson, W. N., and Magdziarz, P. (1996). Broad-band γ-ray and X-ray spectra of NGC 4151 and their implications for physical processes and geometry. *MNRAS*, 283:193–206.

Zhang, Z., Qu, J. L., Gao, H. Q., Zhang, S., Bu, Q. C., Ge, M. Y., Chen, L., and Li, Z. B. (2014). The variability classes in IGR J17091-3624 during the 2011 outburst. *A&A*, 569:A33.

Zheng, S.-M., Yuan, F., Gu, W.-M., and Lu, J.-F. (2011). Revisiting the Thermal Stability of Radiation-dominated Thin Disks. *ApJ*, 732:52.

Zimmerman, E. R., Narayan, R., McClintock, J. E., and Miller, J. M. (2005). Multitemperature Blackbody Spectra of Thin Accretion Disks with and without a Zero-Torque Inner Boundary Condition. *ApJ*, 618:832–844.

Zoghbi, A., Miller, J. M., King, A. L., Miller, M. C., Proga, D., Kallman, T., Fabian, A. C., Harrison, F. A., Kaastra, J., Raymond, J., Reynolds, C. S., Boggs, S. E., Christensen, F. E., Craig, W., Hailey, C. J., Stern, D., and Zhang, W. W. (2016). Disk-Wind Connection during the Heartbeats of GRS 1915+105. *ApJ*, 833:165.

Życki, P. T., Done, C., and Smith, D. A. (1999). The 1989 May outburst of the soft X-ray transient GS 2023+338 (V404 Cyg). *MNRAS*, 309:561–575.

Appendix A

Model-Independent Classification of each Observation of IGR J17091-3624

Observation IDs, and orbit IDs, for every observation and observation segment that was used in our analysis are presented in Table A.1. Note that not all of every observation was used; in many cases, large spikes caused by PCA PCUs switching off or on rendered ~ 100 s unusable. As these often occurred very close to the beginning or end of an observation segment, small sections of data before or after these spikes was also sometimes discarded. Every observation segment is presented along with the variability class assigned to it by this study.

Table A.1: Here is listed the Observation IDs for every *RXTE* observation that was used in this analysis, along with the variability class which has been assigned to it. *Orb.* is the orbit ID (starting at 0) of each observation segment, *Exp.* is the exposure time in seconds and **X** is the prefix 96420-01. This table is continued overleaf in Table A.2.

MJD	OBSID	*Orb.*	Class	*Exp.*	MJD	OBSID	*Orb.*	Class	*Exp.*	MJD	OBSID	*Orb.*	Cla
55622	**X**-01-00	0	I	1840	55676	**X**-09-06	0	VI	3540	55741	**X**-18-05	0	VI
55622	**X**-01-000	0	I	3480	55677	**X**-09-01	0	V	1676	55743	**X**-19-00	0	VI
55622	**X**-01-000	1	I	1656	55678	**X**-09-04	0	V	2090	55744	**X**-19-01	0	VI
55622	**X**-01-000	2	I	3384	55679	**X**-09-02	0	V	2306	55745	**X**-19-02	0	VI
55622	**X**-01-000	3	I	3400	55680	**X**-10-02	0	V	952	55747	**X**-19-03	0	VI
55622	**X**-01-000	4	I	3384	55681	**X**-10-00	0	V	3725	55748	**X**-19-04	0	VI
55623	**X**-01-01	0	I	1240	55682	**X**-10-03	0	V	1157	55749	**X**-19-05	0	VI
55623	**X**-01-01	1	I	752	55684	**X**-10-01	0	III	1504	55751	**X**-20-05	0	VI
55623	**X**-01-01	2	I	992	55686	**X**-10-04	0	III	1127	55752	**X**-20-01	0	VI
55623	**X**-01-01	3	I	1184	55686	**X**-10-05	0	II	2179	55753	**X**-20-02	0	VI
55623	**X**-01-01	4	I	1056	55687	**X**-11-00	0	II	3537	55754	**X**-20-03	0	VI
55623	**X**-01-010	0	I	2080	55688	**X**-11-01	0	II	1153	55756	**X**-20-04	0	VI
55623	**X**-01-010	1	I	1832	55690	**X**-11-02	0	II	1408	55757	**X**-21-00	0	VI
55623	**X**-01-010	2	I	1648	55691	**X**-11-03	0	II	886	55758	**X**-21-01	0	VI
55623	**X**-01-010	4	I	1424	55692	**X**-11-04	0	II	3566	55759	**X**-21-02	0	VI
55623	**X**-01-010	5	I	400	55693	**X**-11-05	0	II	1817	55761	**X**-21-04	0	VI
55623	**X**-01-02	0	I	3056	55694	**X**-12-00	0	II	2761	55762	**X**-21-05	0	VI
55623	**X**-01-02	1	I	2792	55695	**X**-12-01	0	II	1374	55763	**X**-21-06	0	VI
55623	**X**-01-02	2	I	2432	55695	**X**-12-02	0	II	2041	55764	**X**-22-00	0	VI
55623	**X**-01-020	0	I	3456	55696	**X**-12-03	0	II	1456	55765	**X**-22-01	0	VI
55623	**X**-01-020	1	I	3464	55698	**X**-12-04	0	II	1916	55766	**X**-22-02	0	V
55623	**X**-01-020	2	I	3512	55698	**X**-12-05	0	II	3139	55767	**X**-22-03	0	V
55623	**X**-01-020	3	I	3520	55700	**X**-12-06	0	II	1189	55768	**X**-22-04	0	VI
55623	**X**-01-020	4	I	3512	55701	**X**-13-00	0	II	1214	55769	**X**-22-05	0	VI
55623	**X**-01-020	5	I	464	55702	**X**-13-01	0	II	980	55770	**X**-22-06	0	VI
55624	**X**-02-00	0	I	1758	55704	**X**-13-02	0	II	732	55771	**X**-23-00	0	VI
55626	**X**-02-01	0	I	1380	55705	**X**-13-03	0	III	1217	55772	**X**-23-01	0	VI
55628	**X**-02-02	0	I	3305	55706	**X**-13-04	0	III	1161	55773	**X**-23-02	0	VI
55630	**X**-02-03	0	I	1876	55707	**X**-13-05	0	IV	2763	55774	**X**-23-03	0	V
55632	**X**-03-00	0	I	1712	55708	**X**-14-00	0	IV	1188	55775	**X**-23-04	0	V
55634	**X**-03-01	0	III	3590	55709	**X**-14-01	0	IV	3342	55776	**X**-23-05	0	V
55639	**X**-04-00	0	IV	3099	55710	**X**-14-02	0	IV	1094	55777	**X**-23-06	0	IV
55642	**X**-04-02	0	IV	2972	55712	**X**-14-03	0	IV	1404	55777	**X**-23-06	1	IV
55643	**X**-04-01	0	III	1190	55713	**X**-14-04	0	V	871	55778	**X**-24-00	0	IV
55644	**X**-04-03	0	III	2903	55714	**X**-14-05	0	V	1311	55779	**X**-24-01	0	IV
55645	**X**-05-02	0	I	3578	55715	**X**-15-00	0	IV	1241	55779	**X**-24-02	0	IV
55647	**X**-05-00	0	IV	2872	55716	**X**-15-01	0	IV	1262	55782	**X**-24-03	0	V
55647	**X**-05-000	0	IV	3472	55717	**X**-15-02	0	III	1557	55782	**X**-24-04	0	V
55647	**X**-05-000	1	IV	3520	55718	**X**-15-03	0	III	1334	55784	**X**-24-05	0	V
55647	**X**-05-000	2	IV	3512	55720	**X**-15-04	0	IV	1486	55784	**X**-24-06	0	V
55647	**X**-05-000	3	IV	3520	55721	**X**-15-05	0	IV	1500	55785	**X**-25-00	0	V
55647	**X**-05-000	4	IV	3512	55722	**X**-16-00	0	IV	900	55786	**X**-25-01	0	V
55647	**X**-05-000	5	IV	648	55723	**X**-16-01	0	III	1004	55787	**X**-25-02	0	V
55649	**X**-05-03	0	IV	2409	55724	**X**-16-02	0	II	1923	55788	**X**-25-03	0	V
55650	**X**-05-01	0	IV	1473	55725	**X**-16-03	0	II	1919	55789	**X**-25-04	0	V
55651	**X**-05-04	0	IV	2954	55726	**X**-16-04	0	III	1935	55790	**X**-25-05	0	V
55653	**X**-06-00	0	IV	2723	55727	**X**-16-05	0	II	730	55791	**X**-25-06	0	V
55654	**X**-06-01	0	IV	3388	55728	**X**-16-06	0	II	1953	55792	**X**-26-00	0	V
55656	**X**-06-02	0	IV	2908	55729	**X**-17-00	0	II	2735	55794	**X**-26-01	0	V
55657	**X**-06-03	0	V	1842	55730	**X**-17-01	0	II	3556	55795	**X**-26-02	0	VI
55661	**X**-07-00	0	V	1754	55731	**X**-17-02	0	II	3605	55796	**X**-26-03	0	VI
55662	**X**-07-01	0	V	3365	55732	**X**-17-03	0	II	1647	55798	**X**-26-04	0	VI

Table A.2: A continuation of Table A.1. *Orb.* is the orbit ID (starting at 0) of each observation segment, *Exp.* is the exposure time in seconds and **X** is the prefix 96420-01.

MJD	OBSID	*Orb.*	Class	*Exp.*	MJD	OBSID	*Orb.*	Class	*Exp.*	MJD	OBSID
55810	**X**-28-02	0	VI	1251	55836	**X**-32-02	0	IX	1591	55857	**X**-35-01
55811	**X**-28-03	0	VI	2000	55837	**X**-32-03	0	IX	2155	55859	**X**-35-02
55813	**X**-29-00	0	VIII	1309	55838	**X**-32-04	0	IX	2641	55859	**X**-35-02
55819	**X**-29-04	0	VIII	1686	55838	**X**-32-05	0	IX	2077	55860	**X**-35-03
55820	**X**-30-00	0	VI	1488	55840	**X**-32-06	0	IX	3392	55861	**X**-35-04
55821	**X**-30-01	0	VI	1503	55840	**X**-32-06	1	IX	3512	55862	**X**-36-00
55822	**X**-30-02	0	VI	1417	55840	**X**-32-06	2	IX	3934	55863	**X**-36-01
55823	**X**-30-03	0	VI	1290	55840	**X**-32-06	3	IX	3880	55865	**X**-36-03
55824	**X**-30-04	0	VI	1489	55840	**X**-32-06	4	IX	1896	55866	**X**-36-04
55825	**X**-30-05	0	VI	2581	55841	**X**-33-00	0	IX	1188	55867	**X**-36-05
55826	**X**-30-06	0	VI	2747	55842	**X**-33-01	0	IX	855	55868	**X**-36-06
55827	**X**-31-00	0	VI	1559	55843	**X**-33-02	0	IX	1156	55871	**X**-37-00
55828	**X**-31-01	0	VI	2954	55845	**X**-33-04	0	IX	1713	55871	**X**-37-02
55829	**X**-31-02	0	IX	3005	55846	**X**-33-05	0	IX	934	55872	**X**-37-03
55830	**X**-31-03	0	IX	1472	55847	**X**-33-06	0	IX	717	55873	**X**-37-04
55830	**X**-31-03	1	IX	288	55848	**X**-34-00	0	IX	1159	55874	**X**-37-05G
55831	**X**-31-04	0	IX	1586	55849	**X**-34-01	0	IX	973	55875	**X**-37-06
55832	**X**-31-05	0	VI	3812	55851	**X**-34-02	0	IX	2261	55876	**X**-38-00
55833	**X**-31-06	0	IX	3675	55852	**X**-34-03	0	IX	1092	55877	**X**-38-01
55834	**X**-32-00	0	IX	1217	55853	**X**-34-04	0	IX	741	55878	**X**-38-02
55835	**X**-32-01	0	IX	1445	55856	**X**-35-00	0	IX	797	55879	**X**-38-03

Appendix B

List of *RXTE* Observations of the Bursting Pulsar

In Table B.1 we present a table of all *RXTE* observations used in our study of burst evolution in the Bursting Pulsar. The prefixes **A**, **B**, **C**, **D** and **E** correspond to OBSIDs beginning with 10401-01, 20077-01, 20078-01, 20401-01 and 30075-01 respectively.

Obsid	Exp.	Date	Obsid	Exp.	Date	Obsid	Exp.	Date	Obsid	Exp.	Date	Obs
A-01-00	3105	119	**A**-57-00	2432	250	**A**-94-00	3216	332	**C**-16-00	4941	562	**C**-
A-02-00	1655	117	**A**-57-01	894	253	**A**-95-00	9487	333	**C**-16-01	671	561	**C**-
A-03-00	6724	122	**A**-57-02	1408	253	**A**-96-00	2627	337	**C**-16-02	1159	562	**C**-
A-03-000	2372	122	**A**-57-03	1792	253	**A**-97-00	3341	340	**C**-17-00	3537	568	**C**-
A-03-01	768	122	**A**-58-00	1024	255	**A**-98-00	99	343	**C**-18-00	2981	576	**C**-
A-04-00	639	128	**A**-58-01	1401	255	**A**-99-00	2783	345	**C**-18-01	3103	576	**C**-
A-05-00	1990	129	**A**-58-02	1679	255	**A**-99-01	1001	344	**C**-19-00	3286	582	**C**-
A-06-00	1280	134	**A**-58-03	1683	255	**B**-01-00	1664	467	**C**-19-01	2893	582	**C**-
A-08-00	2431	142	**A**-59-00	1152	257	**B**-02-00	1920	468	**C**-19-02	470	582	**D**-
A-09-00	640	138	**A**-59-01	2203	257	**B**-03-00	2982	469	**C**-20-00	3460	589	**D**-
A-10-00	2470	143	**A**-59-02	768	257	**B**-04-00	3530	470	**C**-20-01	1126	589	**D**-
A-11-00	2381	148	**A**-60-00	1907	260	**B**-05-00	2025	472	**C**-21-00	3659	596	**D**-
A-12-00	3352	151	**A**-60-01	3376	260	**B**-06-00	2677	473	**C**-21-01	2907	596	**D**-
A-13-00	3480	155	**A**-60-02	1783	260	**B**-07-00	3365	473	**C**-21-02	1086	596	**D**-
A-14-00	1839	158	**A**-60-03	1559	260	**B**-08-00	3113	475	**C**-22-00	1967	602	**D**-
A-15-00	1595	161	**A**-61-00	3292	262	**B**-09-00	2868	480	**C**-22-01	3086	602	**D**-
A-16-00	3470	156	**A**-61-01	3035	262	**B**-10-00	1009	482	**C**-22-02	1024	602	**D**-
A-17-00	4481	164	**A**-61-02	2013	262	**B**-11-00	2864	487	**C**-23-00	3697	607	**D**-
A-18-00	384	171	**A**-62-00	2390	264	**B**-12-00	1847	489	**C**-23-01	3091	607	**D**-
A-19-00	128	172	**A**-62-01	1703	264	**B**-13-00	2805	497	**C**-24-00	1152	618	**D**-
A-20-00	2087	178	**A**-62-02	2719	264	**B**-14-00	3741	499	**C**-24-01	2300	618	**D**-
A-21-00	2711	181	**A**-63-00	517	266	**B**-15-00	384	501	**C**-24-02	1386	618	**D**-
A-22-00	2816	183	**A**-63-01	3077	266	**B**-16-00	768	503	**C**-25-00	4069	626	**D**-
A-22-01	2911	185	**A**-64-00	2381	268	**B**-17-00	2399	509	**C**-25-01	1920	626	**D**-
A-23-00	1678	187	**A**-64-01	3110	268	**B**-18-00	2306	511	**C**-25-02	768	626	**D**-
A-24-00	2509	189	**A**-65-00	2003	270	**B**-19-00	3477	516	**C**-26-00	2071	633	**D**-
A-25-00	2846	192	**A**-65-01	2744	270	**B**-20-00	1922	520	**C**-26-01	4043	633	**D**-
A-26-00	768	194	**A**-65-02	4331	270	**C**-01-00	8200	389	**C**-27-00	1792	638	**D**-
A-27-00	2923	196	**A**-66-00	2203	272	**C**-02-00	1408	400	**C**-27-01	2495	638	**D**-
A-28-00	6839	199	**A**-66-01	1723	272	**C**-02-01	896	401	**C**-27-02	3082	638	**D**-
A-29-00	3478	201	**A**-66-02	2533	272	**C**-02-02	512	401	**C**-28-00	3454	644	**D**-
A-30-00	5906	203	**A**-67-00	395	274	**C**-03-00	3409	465	**C**-28-01	1359	644	**D**-
A-31-00	6170	206	**A**-67-01	3533	274	**C**-03-01	2635	466	**C**-28-02	756	644	**D**-
A-32-00	2712	209	**A**-67-02	3466	274	**C**-03-02	2645	466	**C**-29-00	1535	652	**D**-
A-34-00	1831	213	**A**-68-00	1841	276	**C**-04-00	2620	478	**C**-30-01	3435	658	**E**-
A-35-00	2563	216	**A**-69-00	3659	278	**C**-04-01	2956	477	**C**-31-00	1920	662	**E**-
A-36-00	3683	219	**A**-70-00	2022	280	**C**-04-02	2515	476	**C**-31-01	1152	662	**E**-
A-37-00	3446	215	**A**-71-00	3474	283	**C**-05-00	1421	484	**C**-31-02	1012	657	**E**-
A-38-00	1536	217	**A**-72-00	5687	285	**C**-05-01	1995	484	**C**-32-00	4646	678	**E**-
A-39-00	2317	218	**A**-73-00	3109	287	**C**-05-02	2505	485	**C**-32-01	2803	678	**E**-
A-40-00	1239	220	**A**-74-00	1659	289	**C**-06-00	2770	492	**C**-33-00	4334	747	**E**-
A-41-00	1363	221	**A**-75-00	1798	291	**C**-06-01	2375	492	**C**-33-01	3534	748	**E**-
A-42-00	2728	224	**A**-76-00	1558	293	**C**-06-02	2203	492	**C**-33-02	2957	748	**E**-
A-43-00	2079	225	**A**-77-00	1738	295	**C**-07-00	1258	494	**C**-34-00	3477	687	**E**-
A-44-00	2076	226	**A**-78-00	463	297	**C**-08-00	3305	505	**C**-34-01	1008	687	**E**-
A-45-00	2050	228	**A**-79-00	1024	299	**C**-08-01	777	505	**C**-34-02	2831	687	**E**-
A-47-00	2687	232	**A**-80-00	5818	301	**C**-09-00	1377	513	**C**-35-00	1497	756	**E**-
A-48-00	2267	234	**A**-81-00	6898	303	**C**-09-01	1536	513	**C**-35-01	1959	755	**E**-
A-49-00	35	236	**A**-82-00	3537	306	**C**-10-00	1664	517	**C**-35-02	2023	755	**E**-
A-50-00	3719	238	**A**-83-00	512	308	**C**-10-01	3796	518	**C**-36-00	2825	702	**E**-
A-51-00	3590	240	**A**-84-00	6361	310	**C**-11-00	2330	527	**C**-36-01	1592	702	**E**-
A-52-00	2518	241	**A**-85-00	10391	312	**C**-11-01	290	527	**C**-37-00	2092	709	**E**-
A-53-00	3063	243	**A**-86-00	9232	314	**C**-11-02	2399	527	**C**-37-01	384	710	**E**-
A-55-00	3328	245	**A**-87-00	3109	316	**C**-12-00	3345	534	**C**-38-00	1752	716	**E**-
A-55-01	3395	245	**A**-88-00	6630	318	**C**-12-01	2048	534	**C**-38-01	1536	716	**E**-
A-55-02	2667	245	**A**-89-00	2569	320	**C**-13-00	1735	541	**C**-38-02	1144	716	**E**-

Appendix C

Normal Burst Histograms

In Figures C.1–C.10, we present histograms showing the distributions of ϕ_B, a_B, σ_B, c, ϕ_d, a_d, d, λ, ϕ_p and a_p we find in our population study of bursts in the Bursting Pulsar. Each of these is a parameter we used to fit the Normal Bursts in our sample: see Section 5.3.3 for a full explanation of these parameters. In Figures C.11–C.16 we show the distributions of ϕ_B, a_B, ϕ_d, a_d, ϕ_p and a_p after being normalised by the persistent emission rate k at the time of each burst.

C.1 Parameter Correlations in Normal Bursts

Before normalizing for persistent rate, we find $> 5\,\sigma$ correlations between 12 pairs of the parameters we use to describe Normal Bursts:

- Persistent emission k correlates with burst fluence ϕ_B ($> 10\,\sigma$), burst amplitude a_b ($> 10\,\sigma$), dip fluence ϕ_D ($> 10\,\sigma$) and dip amplitude a_d ($7.2\,\sigma$).
- Burst fluence ϕ_B also correlates with burst amplitude a_B ($> 10\,\sigma$), dip fluence ϕ_D ($> 10\,\sigma$) and dip amplitude a_d ($7.1\,\sigma$).
- Burst amplitude ϕ_B also correlates with dip fluence ϕ_D ($6.2\,\sigma$) and dip amplitude a_d ($5.7\,\sigma$).
- Burst width σ_B correlates with burst skewness c ($5.8\,\sigma$).
- Dip amplitude a_d anticorrelates with dip recovery timescale λ ($5.0\,\sigma$).
- Plateau fluence ϕ_p correlates with plateau amplitude a_p ($6.6\,\sigma$).

The full correlation matrix can be found in Figure C.17, in which these pairs with $> 5\,\sigma$ correlations are highlighted.

Figure C.1: A histogram showing the distribution of burst fluence ϕ_B amongst our sample of Normal Bursts.

Figure C.2: A histogram showing the distribution of burst amplitude a_B amongst our sample of Normal Bursts.

Figure C.3: A histogram showing the distribution of burst width σ_B amongst our sample of Normal Bursts.

Figure C.4: A histogram showing the distribution of burst skewness c amongst our sample of Normal Bursts.

Figure C.5: A histogram showing the distribution of dip fluence ϕ_d amongst our sample of Normal Bursts.

Figure C.6: A histogram showing the distribution of dip amplitude a_d amongst our sample of Normal Bursts.

Figure C.7: A histogram showing the distribution of dip fall-time d amongst our sample of Normal Bursts.

Figure C.8: A histogram showing the distribution of dip recovery timescale λ amongst our sample of Normal Bursts.

Figure C.9: A histogram showing the distribution of plateau fluence ϕ_p amongst our sample of Normal Bursts.

Figure C.10: A histogram showing the distribution of plateau amplitude a_p amongst our sample of Normal Bursts.

Figure C.11: A histogram showing the distribution of persistent-emission-normalised burst fluence ϕ_B/k amongst our sample of Normal Bursts.

Figure C.12: A histogram showing the distribution of persistent-emission-normalised burst amplitude a_B/k amongst our sample of Normal Bursts.

Figure C.13: A histogram showing the distribution of persistent-emission-normalised dip fluence ϕ_d/k amongst our sample of Normal Bursts.

Figure C.14: A histogram showing the distribution of persistent-emission-normalised dip amplitude a_d/k amongst our sample of Normal Bursts.

Figure C.15: A histogram showing the distribution of persistent-emission-normalised plateau fluence ϕ_p/k amongst our sample of Normal Bursts.

Figure C.16: A histogram showing the distribution of persistent-emission-normalised plateau amplitude a_p/k amongst our sample of Normal Bursts.

Figure C.17: Covariance Matrix with a scatter plot of each of the 66 pairings of the 12 Normal Burst parameters listed in section 5.3.3. Pairings which show a correlation using the Spearman Rank metric with a significance $\geq 5\,\sigma$ are highlighted in red.

Appendix D

Parameter Correlations in Normal Bursts

Before normalizing for persistent rate, we find $> 5\,\sigma$ correlations between 12 pairs of the parameters we use to describe Normal Bursts in the Bursting Pulsar:

- Persistent emission k correlates with burst fluence ϕ_B ($> 10\,\sigma$), burst amplitude a_b ($> 10\,\sigma$), dip fluence ϕ_D ($> 10\,\sigma$) and dip amplitude a_d ($7.2\,\sigma$).
- Burst fluence ϕ_B also correlates with burst amplitude a_B ($> 10\,\sigma$), dip fluence ϕ_D ($> 10\,\sigma$) and dip amplitude a_d ($7.1\,\sigma$).
- Burst amplitude ϕ_B also correlates with dip fluence ϕ_D ($6.2\,\sigma$) and dip amplitude a_d ($5.7\,\sigma$).
- Burst width σ_B correlates with burst skewness c ($5.8\,\sigma$).
- Dip amplitude a_d anticorrelates with dip recovery timescale λ ($5.0\,\sigma$).
- Plateau fluence ϕ_p correlates with plateau amplitude a_p ($6.6\,\sigma$).

The full correlation matrix can be found in Figure C.17, in which these pairs with $> 5\,\sigma$ correlations are highlighted.

Appendix E

PANTHEON suite

Below is the code for the entire suite of *PANTHEON* (Python ANalytical Tools for High-energy Event data manipulatiON) that I used during the work presented in this thesis. This code is also available at `https://github.com/jmcourt/PANTHEON`. PANTHEON makes use of the Astropy (Astropy Collaboration et al., 2013), Matplotlib (Hunter, 2007), Numpy, Scipy (Jones et al., 2001) and Numba (Lam et al., 2015).

E.1 FITS Genie

FITS Genie is a script that allows the user to extract data from raw `FITS` files. The script was designed to interface with *RXTE* data, but there is also limited implementation with *Suzaku*. The script produces `.plotd` and `.speca` files, which can be further processed with *Plot Demon* and *Spec Angel*.

```
#! /usr/bin/env python

# |-----------------------------------------------------------------------|
# |----------------------------FITS GENIE-------------------------------|
# |-----------------------------------------------------------------------|

# Call as ./fitsgenie.py FILE1 PROD_REQ [LCHAN] [HCHAN] [BINNING] [FOURIER RES]
#   [FOURIER SEP] [BGEST] [FLAVOUR]
#
# Takes 1 FITS Event file and produces .speca and .plotd formatted products to be
#   analysed by plotdemon
# and specangel.
#
# Arguments:
#
#  FILE1
#   The absolute path to the file to be used.
#
#  PROD_REQ
#   The products requested by the user.  The following inputs are valid:
#      'spec','speca','s' will cause FITSGenie to produce only a .speca file as output
#      'plot','plotd','p' will cause FITSGenie to produce only a .plotd file as output
```

```
#      'both','all','b','a','sp','ps' will cause both files to be output
#
#  [LCHAN]
#   Optional: The lowest channel on the PCA instrument on RXTE which will be used to
#     populate the data.  Default of 0 (minimum).
#
#  [HCHAN]
#   Optional: The highest channel on the PCA instrument on RXTE which will be used to
#     populate the data.  Default of 255 (maximum).
#
#  [BINNING]
#   Optional: The size, in seconds, of bins into which data will be sorted.  Takes the
#     value of the time resolution of the data if not specified by the user.  Default
#     of 2^-15s
#
#  [FOURIER RES]
#   Optional: The size of the individual time windows in which the data is to be split.
#   Fourier spectra will be made of each of these windows.  Default of 128s.
#
#  [FOURIER SEP]
#   Optional: The separation of the startpoints of individual time windows in which the
#     data is to be split.  Fourier spectra will be made of each of these windows.
#     Default of 128s.
#
#  [BGEST]
#   Optional: The approximate average background count rate during the observation in
#     cts/s.  Default of 30cts/s.
#
#  [FLAVOUR]
#   Optional: A useful bit of text to put on plots to help identify them later on.

#-----User-set Parameters-----------------------------------------------------------------

ptdbinfac=1                 # To save space and time, the time bins for saved
                            #   plotdemon data will be greater than the time bins for
                            #   the not-saved specangel data by this factor.  Must be
                            #   power of 2.
spcbinfac=4096              # The binning factor for SpecAngel data to use when
                            #   searching for data peaks and troughs
usrmin=-13                  # The smallest time resolution to consider is 2^usrmin
                            #   seconds
version=6.1

#-----Welcoming Header--------------------------------------------------------------------

print ''
print '-------Running FITSGenie: J.M.Court, 2015-------'
print ''

#-----Importing Modules-------------------------------------------------------------------

try:
   import sys
   import pan_lib as pan
   from astropy.io import fits
except ImportError:
   print 'Modules missing!  Aborting!'
   print ''
   print '-----------------------------------------------'
```

```
   print ''
   exit()

#-----Checking Validity of Filename-----------------------------------------------------

args=sys.argv
pan.argcheck(args,1)                    # Must give at least 1 args (the function call)
if len(args)<2:
   filename=raw_input('Filename: ')
   print ''
else:
   filename=args[1]                     # Fetch file name from arguments
if len(args)<3:
   print ''
   print 'FitsGenie can produce [P]lotDemon files, [S]pecangel files or [B]oth.'
   print ''
   prod_req=raw_input('Select product(s): ')
   print ''
else:
   prod_req=args[2]                     # Fetch products request from arguments

#-----Identifying Products---------------------------------------------------------------

if prod_req.lower() in ['spec','speca','s','both','all','a','b','ps','sp']:
   spec_on=True
   print '.speca File will be created!'
else:
   spec_on=False
   print '.speca File will NOT be created!'
if prod_req.lower() in ['plot','plotd','p','both','all','a','b','ps','sp']:
   plot_on=True
   print '.plotd File will be created!'
else:
   plot_on=False
   print '.plotd File will NOT be created!'
del prod_req
print ''

#-----Opening FITS file, identifying mission---------------------------------------------

try:
   assert filename[-6:] not in ('.speca','.plotd') # Don't try to open plotd/speca
                                                   #   files please...
   event=fits.open(filename)                       # Unleash the beast! [open the file]
except:
   print 'Could not open file "'+filename+'"!'
   print 'Aborting!'
   pan.signoff()
   exit()

from math import log                               # Import remaining modules.  This is
                                                   #   a slight speedup when running a
                                                   #   a script to attempt to FITSgenie
                                                   #   some valid files and some
                                                   #   invalid files
from numpy import arange, array, histogram, zeros
from numpy import sum as npsum
from scipy.fftpack import fft
import pylab as pl
```

```
try:
   mission=event[1].header['TELESCOP']             # Fetch the name of the telescope
except:
   print 'Could not identify mission!'
   print 'Aborting!'
   pan.signoff()
   exit()
if mission in ['XTE','SUZAKU','SWIFT']:
   print mission,'data detected!'
else:
   print mission,'data not yet supported!'
   pan.signoff()
   exit()
if mission == 'XTE' :
   etype='channel'                                 # XTE requires an input of channel
                                                   #   IDs
   escale=''
   escaleb=''
   #try:
   import xtepan_lib as inst                       # Import XTE extraction functions
   #except:
   #   print 'XTE PANTHEON Library not found!  Aborting!'
   #   pan.signoff()
   #   exit()
elif mission == 'SUZAKU':
   etype='energy'                                  # SUZAKU requires an input of raw
                                                   #   energies
   escale='eV'
   escaleb=' (eV)'
   try:
      import szkpan_lib as inst                    # Import SUZAKU extraction functions
   except:
      print 'Suzaku PANTHEON Library not found!  Aborting!'
      pan.signoff()
      exit()
elif mission == 'SWIFT':
   etype='energy'                                  # SUZAKU requires an input of raw
                                                   #   energies
   escale='eV'
   escaleb=' (eV)'
   try:
      import swfpan_lib as inst                    # Import SUZAKU extraction functions
   except:
      print 'Swift PANTHEON Library not found!  Aborting!'
      pan.signoff()
      exit()
else:
   print "This error shouldn't happen...  sorry 'bout that!"
   pan.signoff()
   exit()
try:
   obsdata=inst.getobs(event,event[1].header['DATAMODE'],filename)       # Fetch object
   print event[1].header['DATAMODE','format detected.'
except:
   print 'Could not identify DATAMODE!'
   print 'Aborting!'
   pan.signoff()
   exit()
```

```
if event[1].header['DATAMODE'][:2] in ['B_','SB']:
   spec_on=False
   bin_dat=True
   print 'No .speca file can be produced!'
   if not plot_on:
      print 'Aborting!'
      pan.signoff()
      exit()
else:
   bin_dat=False

#-----Checking validity of remaining inputs-------------------------------------------

print 'Object =',obsdata[0]
print 'Obs_ID =',obsdata[1]
print ''
maxen=inst.maxen(event[1].header['DATAMODE'])            # Get the value of the
                                                         #   highest energy or channel
                                                         #   for the instrument
print 'Inputs:'
print ''
if len(args)>3:
   lowc=int(args[3])                                     # Collect minimum channel
                                                         #   label from user
   print 'Min Channel =',lowc
else:
   try:
      lowc=int(raw_input("Minimum "+etype+escaleb+": "))
   except:
      lowc=0
      print "Using min "+etype+" of 0"+escale+"!"
if len(args)>4:
   highc=int(args[4])                                    # Collect maximum channel
                                                         #   label from user
   print 'Max Channel =',highc
else:
   try:
      highc=int(raw_input("Maximum "+etype+escaleb+": "))
   except:
      highc=maxen
      print "Using max "+etype+" of "+str(maxen)+escale+"!"
if lowc<0:    lowc=0                                     # Force channels to be in
                                                         #   range 0,255
if highc>maxen: highc=maxen
if lowc>highc:
   print 'Invalid '+etype+'!  Aborting!'                 # Abort if user gives
                                                         #   lowc>highc
   pan.signoff()
   exit()
cs=str(int(lowc))+'-'+str(int(highc))
if len(args)>5:
   bszt=float(args[5])                                   # Collect binsize from inputs
                                                         #   if given, else ask user,
                                                         #   else use resolution
                                                         #   encoded in .fits file
   print 'Bin size (s)=',bszt
else:
   try:
      bszt=float(raw_input("Photon count bin-size (s): "))
```

```
   except:
      bszt=0
      print "Using max time resolution..."
if len(args)>6:
   foures=float(args[6])                                # Collect Fourier resolution
                                                        #   from inputs if given, else
                                                        #   ask user, else use 128s
   print 'Fourier Res.=',foures
elif not spec_on:
   foures=16
else:
   try:
      foures=float(raw_input("Length of time per Fourier Window (s): "))
   except:
      foures=128
      print "Using 128s per spectrum..."
if len(args)>7:
   slide=float(args[7])                                 # Collect Fourier resolution
                                                        #   from inputs if given, else
                                                        #   ask user, else use 128s
   print 'Fourier Sep.=',slide
elif not spec_on:
   slide=16
else:
   try:
      slide=float(raw_input("Separation of Fourier Windows (s): "))
   except:
      slide=foures
      print "Using "+str(slide)+"s per spectrum..."
if len(args)>8:
   bgest=float(args[8])                                 # Collect background estimate
                                                        #   from inputs if given, else
                                                        #   ask user, else use 30c/s
   print 'Background  =',bgest
elif not spec_on:
   bgest=0
else:
   try:
      bgest=float(raw_input("Estimate of background (c/s): "))
   except:
      bgest=30
      print "Using 30c/s background..."
   print ''
if len(args)>9:
   flavour=args[9]                                      # Collect flavour if given,
                                                        #   else flavourless
   print 'Flavour     =',flavour
else:
   flavour=''
print ''
wtype='Boxcar'                                          # Setting all windows to
                                                        #   BoxCar; will make
                                                        #   transition to SpecAngel
                                                        #   4.0 smoother if this
                                                        #   takes a value

#-----Masking data-----------------------------------------------------------------

gti=inst.getgti(event)                                  # Extract GTI indices
```

```
datas=inst.getdat(event)                                    # Extract event data
print 'Discarding photons outside of '+etype+' range '\
   +str(lowc)+escale+'-'+str(highc)+escale+'...'
datas=inst.discnev(datas,event[1].header['DATAMODE'])       # Discarding non-events /
                                                            #   reformatting XTE Binned
                                                            #   data into a less awful
                                                            #   structure
if event[1].header['DATAMODE'][:2] == 'B_':
   olen=str(npsum(array(datas)))
else:
   olen=str(len(datas))
datas=inst.chrange(datas,lowc,highc,event[1].header)
tstart=inst.getini(event)
if event[1].header['DATAMODE'][:2] == 'B_':
   phcts=npsum(datas)
else:
   phcts=len(datas)
if float(olen)==0:
   print 'No photons!  Aborting!'
   pan.signoff()
   exit()
pcg=str(int(100*phcts/float(olen)))+'\%'
print str(phcts)+'/'+olen+' photons fall within '+etype+' range ('+pcg+')!'
if phcts==0:
   print 'Aborting!'
   pan.signoff()
   exit()
print ''

#-----Fetching Bin Size-------------------------------------------------------------------

bsz=inst.getbin(event,event[1].header['DATAMODE']) # Fetch 'Binning' as the time
                                                   #   resolution of the data
ores=bsz
if bszt>bsz:                                       # If user enters a lower binning
                                                   #   resolution than maximum, use
                                                   #   that instead
   bsz=bszt
n=usrmin                                           # Rounding bsz to the nearest
                                                   #   (greater) power of 2
while (2**n)<bsz:
   n+=1
bsz=2**n
bsz=float(bsz)

#-----Fetching Time Axis------------------------------------------------------------------

print 'SpecAngel binsize rounded to 2^'+str(n)+'s ('+str(bsz)+'s)!'
print 'PlotDemon binsize rounded to 2^'+str(n+int(log(ptdbinfac,2)))\
   +'s ('+str(bsz*ptdbinfac)+'s)!'
times,datas=inst.gettim(datas,event[1].data,tstart,ores,event[1].header['DATAMODE'])
                     # ^ Extracting list of photon incident times as a separate object
pcwrds=inst.getwrd(datas,event[1].header['DATAMODE'])
sttim=times[0]
times=times-sttim

#-----Fetching Fourier Range Size---------------------------------------------------------

if foures>max(times):
```

```
   foures=128
if slide>max(times):
   slide=foures
slidelock=slide==foures          # If foures=slide, lock them together
n=0                              # Rounding foures to the nearest (greater) power of 2
while (2**n)<foures:
   n+=1
foures=2**n
if slidelock:
   slide=foures
else:
   plot_on=False
   print 'No .plotd file can be produced!'
   if not spec_on:
      print 'Aborting!'
      pan.signoff()
      exit()
print 'Fourier window length rounded to 2^'+str(n)+'s ('+str(foures)+'s)!'
print 'Fourier window separation rounded to',str(slide)+'s!'
print ''

#-----Rescaling GTI-------------------------------------------------------------------

for j in pan.eqrange(gti):
   gti[j]=gti[j][0]-sttim,gti[j][1]-sttim

#-----Setting up power spectra--------------------------------------------------------

ndat=int(max(times)/bsz)
datres=int(foures/bsz)                      # Work out how many data points corresponds
                                            #   to the user given time interval
                                            #   'foures'
stpres=int(slide/bsz)                       # Work out how many data points corresponds
                                            #   to the user given time separation
                                            #   'slide'
numstep=(ndat/stpres)                       # Calculate how many intervals of 'datres'
                                            #   can be divided into the data length
while ((numstep-1)*slide)+foures>max(times):
   numstep-=1                               # Way to make sure the final bin doesn't
                                            #   exceed the data time limit
print 'Analysing data...'
print ''
fourgrlin=[]                                # Set up matrix
bad=0                                       # Counter to count ranges which fall out of
                                            #   the GTIs
good=[]                                     # Array to keep track of which ranges were
                                            #   good
prates=[]                                   # 'Peak Rates'
trates=[]                                   # 'Trough Rates'
rates=[]                                    # Array of count rates to be populated
npcus=[]
t=arange(0,foures+bsz,bsz)                  # Setting up SpecAngel resolution time
                                            #   series per Fourier bin
tp=arange(0,foures+bsz*spcbinfac,bsz*spcbinfac) # Setting up SpecAngel coarse
                                                #   resolution time series
tc=arange(0,foures+bsz*ptdbinfac,bsz*ptdbinfac) # Setting up PlotDemon coarse
                                                #   resolution time series per
                                                #   Fourier bin
ta=arange(0,(foures*numstep),bsz*ptdbinfac)     # Setting up PlotDemon resolution
```

```
                                                #   full time series

#-----Populating power spectra-------------------------------------------------------

fullhist=[]                                     # Create empty flux array to pass to
                                                #   plotdemon
fullerrs=[]
tcounts=0                                       # Initiate photon counter
pcus=None
if not bin_dat:
   for step in range(numstep):                                  ## For every [foures]s
                                                                ##   interval in the data:
      stpoint=step*slide                                           # Calculate the start
                                                                   #   point of the
                                                                   #   interval
      edpoint=stpoint+foures                                       # Calculate the endpoint
                                                                   #   of the interval
      in_gti=False                                                 # Assume the subrange is
                                                                   #   not in the GTI
      for j in pan.eqrange(gti):
         if gti[j][0]<=stpoint<edpoint<=gti[j][1]: in_gti=True # Change in_gti flag if
                                                                   #   this range is wholly
                                                                   #   within one GTI
      mask=times>=stpoint
      datrow=times[mask]                                           # Take all photons in
                                                                   #   the event data which
                                                                   #   occurred after the
                                                                   #   start point
      wrdrow=inst.getwrdrow(pcwrds,mask,event[1].header['DATAMODE'])
      mask=datrow<edpoint
      datrow=datrow[mask]                                          # Remove all photons
                                                                   #   which occurred after
                                                                   #   the end point
      wrdrow=inst.getwrdrow(wrdrow,mask,event[1].header['DATAMODE'])
      fc,null=histogram(datrow,tc+stpoint)                         # Coarsely bin this sub
                                                                   #   range of event data
      fp,null=histogram(datrow,tp+stpoint)                         #   Very Coarsely bin
                                                                   #   this subrange of
                                                                   #   event data
      del null
      fullhist=fullhist+list(fc)
      fullerrs=fullerrs+list((array(fc)**0.5))
      if in_gti:
         f,txis=histogram(datrow,t+stpoint)                         # Bin well this sub
                                                                    #   range of event data
         pcus=inst.getpcu(wrdrow,event[1].header,t_pcus=pcus)       # Count active PCUs by
                                                                    #   assuming any that
                                                                    #   recorded 0 events
                                                                    #   in the time period
                                                                    #   were inactive
         npcus.append(pcus)
         counts=sum(f)
         peak=max(fp)
         trough=min(fp)
         rates.append(float(counts)/foures)
         prates.append(float(peak)*datres/(foures*spcbinfac))
         trates.append(float(trough)*datres/(foures*spcbinfac))
         tcounts+=counts
         tsfdata=fft(f)                                             # Fourier transform the
```

```
                                                          #   interval
         tsfdata=pan.leahyn(tsfdata,counts,datres)        # Normalise to Leahy
                                                          #   Power
         good.append(True)                                # Flag this column as
                                                          #   good
      else:
         tsfdata=zeros(datres/2)
         npcus.append(0)
         rates.append(0)
         prates.append(0)
         trates.append(0)
         good.append(False)                               #  Flag this column as
                                                          #    bad
      fourgrlin.append(tsfdata)                           #  Append the FT'd data
                                                          #    to the matrix
      prog=step+1
      if (prog \% 5)==0 or prog==numstep:
         print str(prog)+'/'+str(numstep)+' series analysed...' # Display progress
                                                          #   every 5 series
   pcg=str(int(100*tcounts/float(phcts)))+'\%'
   print ''
   print str(tcounts)+'/'+str(phcts)+' ('+pcg+') photons in GTI '\
      +str((gti[0][0],gti[-1][1]))+'!'
   if tcounts==0:
      print 'Aborting!'
      pan.signoff()
      exit()
else:                                                     # Not doing Spectra for
                                                          #   Binned data just
                                                          #   yet...
   print 'Number of PCUs unknown!'
   npcus=[int(raw_input('Number of Active PCUS: '))]      # Ask the user how many
                                                          #   there are
   ta,fullhist,fullerrs=pan.binify(times,datas/ores*bsz*ptdbinfac,
      (datas**0.5)/ores*bsz*ptdbinfac,bsz)

#-----Save .speca and .plotd files-------------------------------------------------

print ''
print 'Saving...'
print ''
filext=(filename.split('.')[-1])          # Identify file extension from the original
                                          #   filename
if filext!=filename:
   print filename
   tfilename=filename[:-len(filext)-1]    # Remove file extension, if present
   if tfilename[-1]!='.':                 # Saving extensionless files with .. in the
                                          #   path name *breaks without this*
      filename=tfilename
filename=filename+'_'+cs+'_'+str(bsz)+'s'
if plot_on:
   pfilename=pan.plotdsv(filename,ta,array(fullhist)/bsz,array(fullerrs)/bsz,tstart,
      bsz*ptdbinfac,gti,max(npcus),bgest,'False',None,flavour,cs,mission,obsdata,
      version)
   print "PlotDemon file saved to "+pfilename
else:
   print "PlotDemon file not saved."
if spec_on:
   sfilename=pan.specasv(filename,fourgrlin,good,rates,prates,trates,tcounts,
```

```
      max(npcus),bsz,bgest,foures,flavour,cs,mission,obsdata,wtype,slide,spcbinfac,
      version)
   print "SpecAngel file saved to "+sfilename
else:
   print "SpecAngel file not saved."

#-----Footer-------------------------------------------------------------------------

pan.signoff()
```

E.2 Plot Demon

```
#! /usr/bin/env python

# |-----------------------------------------------------------------------|
# |------------------------------PLOT DEMON------------------------------|
# |-----------------------------------------------------------------------|

# Call as ./plotdemon.py FILE1 [FILE2] [FILE3] BINNING
#
# Takes 1-3 .plotd files and plots relevant astrometric plots
#
# Arguments:
#
#  FILE1
#   The absolute path to the first file to be used (generally the lowest energy band)
#
#  [FILE2]
#   The absolute path to the second file to be used
#
#  [FILE3]
#   The absolute path to the third file to be used (generally the highest energy band)
#
#  [BINNING]
#   Optional: the size, in seconds, of bins into which data will be sorted.

#-----User-set Parameters------------------------------------------------------------

minbin=0.0078125                  # The minimum bin size the code is allowed to attempt
                                  #   to use.  This can prevent long hang-ups
version=4.3                       # The version of PlotDemon
cbin=32.0                         # The number of bins to use when calculating
                                  #   inhomonogeneity in circfold

#-----Welcoming Header---------------------------------------------------------------

print ''
print '-------Running Plot Demon: J.M.Court, 2014------'
print ''

#-----Importing Modules--------------------------------------------------------------

try:
   import sys,os,imp
   import pylab as pl
   import pan_lib as pan
   import numpy as np
```

```
   import scipy.signal as sig
   import scipy.optimize as optm
   from math import pi
except ImportError:
   print 'Modules missing!  Aborting!'
   print ''
   print '-----------------------------------------------'
   print ''
   exit()
try:
   imp.find_module('PyAstronomy')                        # Check if PyAstronomy exists
   module_pyastro=True
except ImportError:
   module_pyastro=False

#-----Opening Files-----------------------------------------------------------------

args=sys.argv                                          # Fetching arguments; softest
                                                       #    energy band first please
pan.argcheck(args,1)
try:
   float(args[-1])                                     # If the final argument can be
                                                       #   converted to integer, assume
                                                       #   user intends it as a binning
   isbininp=True                                       # "IS BINsize given as an
                                                       #   INPut?"
except:
   isbininp=False
nfiles=max(len(args)-isbininp-1,1)                     # Fetch number of infiles (total
                                                       #   args minus one or two iff
                                                       #   binsize given)
if nfiles>3: nfiles=3
if len(args)<2:
   file1=raw_input('Filename: ')
else:
   file1=args[1]
isplotd1=pan.filenamecheck(file1,'plotd',continu=True) # Work out whether input file is
                                                       #   a plotdemon file or a csv
ch={}                                                  # Save channel info in a library
bg1=0
bg2=0
bg3=0
print 'Opening',file1                                  # Opening file 1
x1r,y1r,ye1r,tst1,bsz1,gti,pcus1,bg1,bsub1,bdata1,flv1,
   ch[1],mis1,obsd1,v1=pan.pdload(file1,isplotd1)
y1r=y1r/float(pcus1)                                   # Normalising flux by dividing
                                                       # by the number of active PCUs
                                                       # and the binsize
ye1r=ye1r/float(pcus1)
flavour=flv1
if flavour=='':
   qflav=''
else:
   qflav=' "'+flavour+'"'
if nfiles>1:
   file2=args[2]
   isplotd2=pan.filenamecheck(file2,'plotd',continu=True)
   print 'Opening',file2                              # Opening file 2
   x2r,y2r,ye2r,tst2,bsz2,gti2,pcus2,bg2,bsub2,bdata2,flv2,ch[2],mis2,obsd2,v2\
```

```
      =pan.pdload(file2,isplotd2)
   y2r=y2r/float(pcus2)                                # Normalising flux by dividing
                                                       # by the number of active PCUs
                                                       # and the binsize
   ye2r=ye2r/float(pcus2)
else: x2r=y2r=ye2r=tst2=bsz2=None
if nfiles>2:
   file3=args[3]
   isplotd3=pan.filenamecheck(file3,'plotd',continu=True)
   print 'Opening',file3                               # Opening file 3
   x3r,y3r,ye3r,tst3,bsz3,gti3,pcus3,bg3,bsub3,bdata3,flv3,ch[3],mis3,obsd3,v3\
      =pan.pdload(file3,isplotd3)
   y3r=y3r/float(pcus3)                                # Normalising flux by dividing
                                                       # by the number of active PCUs
                                                       # and the binsize
   ye3r=ye3r/float(pcus3)
else: x3r=y3r=ye3r=tst3=bsz3=None
bg=(bg1+bg2+bg3)/float(pcus1)
xit1=x1r[-1]
if nfiles>1:
   xit2=x2r[-1]
else:
   xit2=None
if nfiles==3:
   xit3=x3r[-1]
else:
   xit3=None
oet=max(xit1,xit2,xit3)                                # Fetch the observation end time
mint=0                                                 # Save original start and end-
                                                       # points for use in clipping
maxt=oet
if nfiles>1:                                           # Checking that start-times of
                                                       # files 1 & 2 match
   if tst1!=tst2:
      if tst1>tst2:
         while x1r[0]+tst1>x2r[0]+tst2:                # Hack data off of the start of
                                                       # file 2 until its startpoint
                                                       # matches file 1
            if len(x2r)==0:
               print 'Times domains for files 1 & 2 do not overlap!  Aborting!'
               pan.signoff()
               exit()
            x2r=np.delete(x2r,0)
            y2r=np.delete(y2r,0)
            ye2r=np.delete(ye2r,0)
         if tst1+x1r[0]!=tst2+x2r[0]:
            print 'Starting times for files 1 & 2 do not match!  Aborting!'
                                                       # If this overshoots, give up
            pan.signoff()
            exit()
         else:
            tst2+=x2r[0]                               # Amend new start time
            x2r=x2r-x2r[0]
      else:
         while x2r[0]+tst2>x1r[0]+tst1:                # Or Hack data off of the start
                                                       # of file 1 until its startpoint
                                                       # matches file 2
            if len(x1r)==0:
               print 'Times domains for files 1 & 2 do not overlap!  Aborting!'
```

```
            pan.signoff()
            exit()
         x1r=np.delete(x1r,0)
         y1r=np.delete(y1r,0)
         ye1r=np.delete(ye1r,0)
      if tst1+x1r[0]!=tst2+x2r[0]:
         print 'Starting times for files 1 & 2 do not match!  Aborting!'
         pan.signoff()
         exit()
      else:
         tst1+=x1r[0]
         x1r=x1r-x1r[0]
if nfiles>2:                                          # Checking that start-times of
                                                      # files 1 & 3 match (and thus 2 &
                                                      #  3 also match)
   if tst1!=tst3:
      if tst1>tst3:
         while x1r[0]+tst1>x3r[0]+tst3:               # Hack data off of the start of
                                                      # file 3 until its startpoint
                                                      # matches file 1
         if len(x3r)==0:
            print 'Times domains for files 1 & 3 do not overlap!  Aborting!'
            pan.signoff()
            exit()
         x3r=np.delete(x3r,0)
         y3r=np.delete(y3r,0)
         ye3r=np.delete(ye3r,0)
      if tst1+x1r[0]!=tst3+x3r[0]:
         print 'Starting times for files 1 & 3 do not match!  Aborting!'
         pan.signoff()
         exit()
      else:
         tst3+=x3r[0]                                 # Amend new start time
         x3r=x3r-x3r[0]
      else:
         while x3r[0]+tst3>x1r[0]+tst1:               # Or Hack data off of the start
                                                      # of files 1 & 2 until their
                                                      # startpoint matches file 3
         if len(x1r)==0:
            print 'Times domains for files 1 & 3 do not overlap!  Aborting!'
            pan.signoff()
            exit()
         x1r=np.delete(x1r,0)
         y1r=np.delete(y1r,0)
         ye1r=np.delete(ye1r,0)
         x2r=np.delete(x2r,0)
         y2r=np.delete(y2r,0)
         ye2r=np.delete(ye2r,0)
      if tst1+x1r[0]!=tst3+x3r[0]:
         print 'Starting times for files 1 & 3 do not match!  Aborting!'
         pan.signoff()
         exit()
      else:
         tst1+=x1r[0]
         x1r=x1r-x1r[0]

#-----Binning-------------------------------------------------------------------------

if isbininp:
```

```
   binning=float(args[-1])                          # Collect binsize input if given
else:
   while True:                                      # Keep asking until good response
                                                    # is given
      try:
         binning=float(raw_input("Enter bin size (s): "))
                                                    # Ask for binsize in dialogue box
         break
      except:
         print 'Invalid bin size input!'
if minbin>max(binning,bsz1,bsz2,bsz3):
   print 'Warning!  User-entered bin is smaller than the minimum!'
   print 'Minimum can be changed in the user-input section of this code'
binning=max(binning,bsz1,bsz2,bsz3,minbin)          # Prevent overbinning by setting
                                                    # minimum binning to the maximum
                                                    # of the binnings of the files
print ''
print 'Bin size='+str(binning)+'s'
print 'Binning File 1...'
x1,y1,ye1=pan.binify(x1r,y1r,ye1r,binning)          # Bin File 1 using 'binify' in
                                                    # pan_lib
if nfiles>1:
   print 'Binning File 2...'
   x2,y2,ye2=pan.binify(x2r,y2r,ye2r,binning)       # Bin File 2 using 'binify' in
                                                    # pan_lib
   if nfiles>2:
      print 'Binning File 3...'
      x3,y3,ye3=pan.binify(x3r,y3r,ye3r,binning)    # Bin File 3 using 'binify' in
                                                    # pan_lib
print 'Binning complete!'
print ''
wrongsize=False
x3l=len(x1)                                         # Fix to make this work for 2 or
                                                    # 3 mismatched files

#-----Force file lengths to match-------------------------------------------------------

if nfiles>1:                                        # Checking file lengths match
   if len(x1)!=len(x2):
      print 'Warning!  Files 1&2 of different lengths!'
      wrongsize=True
if nfiles==3:
   if len(x1)!=len(x3):
      print 'Warning!  Files 1&3 of different lengths!'
      wrongsize=True
      x3l=len(x3)
if wrongsize:                                       # Forcing file lengths to match
                                                    # if possible
   print 'Attempting to crop files...'
   mindex=min(len(x1),len(x2),x3l)-1
   if x1[mindex]!=x2[mindex]:
      print 'Cannot crop, aborting!'
      pan.signoff()
      exit()
   if nfiles==3:
      if x1[mindex]!=x3[mindex]:
         print 'Cannot crop, aborting!'
         pan.signoff()
         exit()
```

```
   mindex+=1
   x1=x1[:mindex]
   y1=y1[:mindex]
   ye1=ye1[:mindex]
   x2=x2[:mindex]
   y2=y2[:mindex]
   ye2=ye2[:mindex]
   if nfiles==3:
      x3=x3[:mindex]
      y3=y3[:mindex]
      ye3=ye3[:mindex]
   print 'Cropped succesfully!'
   print ''

#-----Fetch GTI Mask-----------------------------------------------------------------

print 'Fetching GTI mask...'
def getmask(xarr,gtis):
   if gtis is None:
       return np.array([True]*len(xarr))              # Assume all is in gtis if
                                                      # loaded from csv
   else:
       return pan.gtimask(xarr,gtis)
gmask=getmask(x1,gti)                                 # A mask to blank values that
                                                      # fall outside of the GTIs
if nfiles>1:
   gmask2=getmask(x2,gti2)                            # 'And' masks for different files
   gmask=gmask&gmask2
if nfiles>2:
   gmask3=getmask(x3,gti3)
   gmask=gmask&gmask3
print str(int(100*sum(gmask)/len(gmask)))+'% of data within GTI!'
print ''

#-----Fetch Colours------------------------------------------------------------------

def colorget(verbose=True):                           # Define colorget to easily re-
                                                      # obtain colours if base data is
                                                      # modified
   if verbose:
      print 'Analysing Data...'
   times=x1[gmask]
   timese=np.zeros(len(times))
   ys={}
   yes={}
   col={}
   cole={}
   if nfiles==1:                                      # If only one file given, flux
                                                      # and flux_error are just the
                                                      # flux and error of this one file
      flux=y1[gmask]                                  # Use gmask to clip out the areas
                                                      # outside of GTI
      fluxe=ye1[gmask]
   elif nfiles==2:
      flux,fluxe,ys,yes,col,cole=pan.pdcolex2(y1,y2,ye1,ye2,gmask)
                                                      # Get 2/1 and 1/2 colour info
                                                      # using PDColEx in pan_lib
   elif nfiles==3:
      flux,fluxe,ys,yes,col,cole=pan.pdcolex3(y1,y2,y3,ye1,ye2,ye3,gmask)
```

```
                                                    # Get ALL colour values with 3D
                                                    # PDColEx
    else:
        print 'Error!  Too much data somehow.'      # This warning should never come
                                                    # up...
        pan.signoff()
        exit()
    return times,timese,flux,fluxe,ys,yes,col,cole
times,timese,flux,fluxe,ys,yes,col,cole=colorget()  # Use colorget
print 'Done!'
print ''

#-----Setting up plot environment-------------------------------------------------------

show_block=False                                    # Do not force plots to stay open
                                                    # by default
plotopt=''
es=True                                             # Options to keep track of what
                                                    # form the data is in.  'es':
                                                    # with error bars.
cs=False                                            # 'cs' with colour key
ls=False                                            # 'ls' with delineation
folded=False                                        # 'folded' has been folded over
                                                    # some period
saveplots=False
def doplot(x,xe,y,ye,ovr=False,ft='-k',per2=False):  # Defining short function to
                                                    # determine whether errorbars are
                                                    # needed on the fly
                                                    # 'ovr' allows to override colour
                                                    # and line options so lightcurves
                                                    # can be made differently
    if ovr: formst=ft                               # If override given, accept input
                                                    # format; if none given, just
                                                    # plot lines
    elif ls: formst='-ok'                           # If deLineate mode on, connect
                                                    # points with lines and mark
                                                    # points
    else: formst='ok'                               # If neither deLineate nor
                                                    # override on, just plot points.
    if ls and not ovr:
        plotx=np.append(x,x[0])
        ploty=np.append(y,y[0])
        plotxe=np.append(xe,xe[0])
        plotye=np.append(ye,ye[0])
    elif per2 and ovr:
        plotx=np.append(x,x+1.0)
        ploty=np.append(y,y)
        plotxe=np.append(xe,xe)
        plotye=np.append(ye,ye)
        pl.axvline(1,color='0.7',linestyle=':')
    else:
        plotx=x
        ploty=y
        plotxe=xe
        plotye=ye
    if cs and not ovr:                              # If coloured mode on, colour
                                                    # first 5 data points unless
                                                    # override given
        if len(x)<5:                                # Abort if less than 5 data
```

```
                                                    # points present
            print 'Not enough data to colour!'
         else:
            pl.plot(x[0],y[0],'or',zorder=1)        # Plot a round marker over each
                                                    # of the first five points with
                                                    # colour ascending red->blue
            pl.plot(x[1],y[1],'oy',zorder=2)
            pl.plot(x[2],y[2],'og',zorder=3)
            pl.plot(x[3],y[3],'oc',zorder=4)
            pl.plot(x[4],y[4],'ob',zorder=5)
      if es:
         pl.errorbar(plotx,ploty,xerr=plotxe,yerr=plotye,fmt=formst,zorder=0)
                                                    # Plot errorbar plot if errors
                                                    # turned on
      else:
         pl.plot(plotx,ploty,formst,zorder=0)       # Else plot regular graph
   def plot_save(saveplots,show_block):             # Add a function to redirect all
                                                    # show calls to savefigs if
                                                    # toggled
      if saveplots:
         pl.savefig(raw_input('Save plot as: '))
         print 'Plot saved!'
      else:
         pl.show(block=show_block)
   def burstplot(key,text,units):
      if bursts is None:
         print 'No burst data to plot!  Run "burst get" first!'
         return
      print 'Plotting Histogram of Burst '+text+'...'
      pl.figure()
      pl.hist(bursts[key],bins=np.arange(min(bursts[key]),max(bursts[key]),
      (max(bursts[key])-min(bursts[key]))/21.0))
      pl.xlabel(text,'('+units+')')
      pl.ylabel('Frequency')
      pl.title('Histogram of Burst '+text)
      plot_save(saveplots,show_block)
   fldtxt=''
   bursts=None
   burst_alg='cubic spline'
   flux_axis=r'Flux (cts s$^{-1}$ PCU$^{-1}$)'
   time_n=0                                         # Normalise dump time

   #-----User Menu-------------------------------------------------------------------

   def give_inst():                                 # Define printing this list of
                                                    # instructions as a function
      print 'COMMANDS: Enter a command to manipulate data.'
      print ''
      print 'DATA:'
      print '* "rebin" to reset the data and load it with a different binning.'
      print '* "clip" to clip the data.'
      print '* "norm time" to renormalise the times by the start time of the data'
      print '* "mask" to remove a range of data.'
      print '* "rms" to return the fractional rms of the data.'
      print '* "fold" to fold data over a period of your choosing'+(' (requires PyAstron'+
            'omy module!)' if not module_pyastro else '')+'.'
      print '* "autofold" to automatically seek a period over which to fold data'+(' (re'+
            'quires PyAstronomy module!)' if not module_pyastro else '')+'.'
      print '* "varifold" to fold over a non-constant period using an algorithm optimise'+
```

```
      'd for high-amplitude quasi-periodic flares.'
print '* "plot bursts" to plot the results of the peak-finding algorithm used in v'+
      'arifold.'
print ''
print '1+ DATASET PLOTS:'
print '* "lc" to plot a simple graph of flux over time.'
print '* "bg" to plot background over time, if background has been estimated for t'+
      'hese files.'
print '* "animate" to create an animation of the lightcurve as the binning is incr'+
      'eased.'
print '* "circanim" to create an animation of the lightcurve circularly folded as '+
      'the period is increased.'
print '* "lombscargle" to create a Lomb-Scargle periodogram of the lightcurve.'
print '* "autocor" to plot the auto-correlation function.'
print '* "rmsflux" to plot the rms-flux relationship of the data.'
if nfiles>1:                                        # Only display 2-data-set inst-
                                                    # -ructions if 2+ datasets given
   print ''
   print '2+ DATASET PLOTS:'
   print '* "hardness21" to plot a hardness/time diagram of file2/file1 colour ove'+
         'r time.'
   print '* "hardness12" to plot a hardness/time diagram of file1/file2 colour ove'+
         'r time.'
   print '* "hid21" to plot a hardness-intensity diagram of file2/file1 colour aga'+
         'inst total flux.'
   print '* "hid12" to plot a hardness-intensity diagram of file1/file2 colour aga'+
         'inst total flux.'
   print '* "calcloop21" to return the probability of a null hysteresis in the 12 '+
         'HID.'
   print '* "col21" to plot file2/file1 colour against time.'
   print '* "col12" to plot file1/file2 colour against time.'
   print '* "band" to plot the lightcurve of a single energy band.'
   print '* "bands" to plot lightcurves of all bands on adjacent axes.'
   print '* "xbands" to plot lightcurves of all bands on the same axes.'
   print '* "compbands21" to plot lightcurves of bands 2 and 1 against each other.'
   print '* "crosscor21" to plot the cross-correlation function of band 1 with ban'+
         'd 2.'
   print '* "timeres crosscor21" to plot the time-resolved cross-correlation funct'+
         'ion of band 1 with band 2'
   print '* "all" to plot all available data products.'
if nfiles==3:                                       # Only display 3-data-set inst-
                                                    # -ructions if 3 datasets given
   print ''
   print '3 DATASET PLOTS:'
   print '* "hardness32" to plot a hardness/time diagram of file3/file2 colour ove'+
         'r time.'
   print '* "hardness23" to plot a hardness/time diagram of file2/file3 colour ove'+
         'r time.'
   print '* "hardness31" to plot a hardness/time diagram of file3/file1 colour ove'+
         'r time.'
   print '* "hardness13" to plot a hardness/time diagram of file1/file3 colour ove'+
         'r time.'
   print '* "hid32" to plot a hardness-intensity diagram of file3/file2 colour aga'+
         'inst total flux.'
   print '* "hid23" to plot a hardness-intensity diagram of file2/file3 colour aga'+
         'inst total flux.'
   print '* "calcloop32" to return the probability of a null hysteresis in the 32 '+
         'HID.'
   print '* "hid31" to plot a hardness-intensity diagram of file3/file1 colour aga'+
```

```
            'inst total flux.'
        print '* "hid13" to plot a hardness-intensity diagram of file1/file3 colour aga'+
            'inst total flux.'
        print '* "calcloop31" to return the probability of a null hysteresis in the 31 '+
            'HID.'
        print '* "col32" to plot file3/file2 colour against time.'
        print '* "col23" to plot file2/file3 colour against time.'
        print '* "col31" to plot file3/file1 colour against time.'
        print '* "col13" to plot file1/file3 colour against time.'
        print '* "compbands31" to plot lightcurves of bands 3 and 1 against each other.'
        print '* "compbands32" to plot lightcurves of bands 3 and 2 against each other.'
        print '* "ccd" to plot a colour-colour diagram (3/1 colour against 2/1 colour).'
        print '* "timeres crosscor31" to plot the time-resolved cross-correlation funct'+
            'ion of band 3 with band 1'
        print '* "timeres crosscor32" to plot the time-resolved cross-correlation funct'+
            'ion of band 3 with band 2'
        print '* "crosscor31" to plot the cross-correlation function of band 3 with ban'+
            'd 1.'
        print '* "crosscor32" to plot the cross-correlation function of band 3 with ban'+
            'd 2.'
    print ''
    print 'BURST ANALYSIS:'
    print '* "burst get" to interactively extract burst data for analysis.'
    print '* "burst peaks" for a histogram of peak heights of extracted bursts.'
    print '* "burst risetimes" for a histogram of rise times of extracted bursts.'
    print '* "burst falltimes" for a histogram of fall times of extracted bursts.'
    print '* "burst lengths" for a histogram of durations of extracted bursts.'
    print '* "burst help" for further information on burst analysis.'
    print ''
    print 'SAVING DATA TO ASCII:'
    print '* "export" to dump the lightcurve and colour data into an ASCII file.'
    print '* "bgdump" to export background lightcurve to an ASCII file.'
    print '* "timenorm" to toggle absolute or relative time values on x-axis.'
    print ''
    print 'TOGGLE OPTIONS:'
    print '* "errors" to toggle whether to display errors in plots.'
    print '* "lines" to toggle lines joining points in graphs.'
    print '* "ckey" to toggle colour key (red-blue) for the first five points in all p'+
        'lots.'
    print '* "save" to save to disk any plots which would otherwise be shown.'
    print ''
    print 'ADVANCED OPTIONS:'
    print '* "burstalg" to select algorithm for finding pulse peaks in lightcurve.'
    print ''
    print 'OTHER COMMANDS:'
    print '* "info" to display a list of facts and figures about the current PlotDemon'+
        ' session.'
    print '* "reflav" to rewrite the flavour text used for graph titles.'
    print '* "help" or "?" to display this list of instructions again.'
    print '* "quit" to quit.'

#give_inst()                                    # Print the list of instructions
print ''
print ' --------------------'

#-----Entering Interactive Mode-------------------------------------------------------

while plotopt not in ['quit','exit']:           # If the previous command given
                                                # was not quit, continue
```

```
print ''
plotopt=raw_input('Give command [? for help]: ').lower() # Fetch command from user
print ''

#-----Aliasing options-----------------------------------------------------------------

if plotopt=='shid':                                # 'shid' refers to the 2/1 HID
   plotopt='hid21'
elif plotopt=='hhid':                              # 'hhid' refers to the 3/1 HID
   plotopt='hid31'
elif plotopt=='hid' and nfiles==2:
   plotopt='hid21'                                 # 'hid' refers to the 2/1 HID if
                                                   # that is the only HID available

#-----Hidden 'stick' option-----------------------------------------------------------

if plotopt=='stick':                               # For use when scripting with
                                                   # Plotdemon.  If turned on, this
                                                   # causes all plots to block when
                                                   # shown.
   show_block=not show_block
   if show_block:
      print 'Sticky Plots on!'
   else:
      print 'Sticky Plots off!'

#-----'save' option-------------------------------------------------------------------

elif plotopt=='save':                              # Causes a plot to be saved when
                                                   # it would otherwise have been
                                                   # shown
   saveplots=not saveplots
   if saveplots:
      print 'Plot saving on!'
   else:
      print 'Plot saving off!'

#-----'rebin' option------------------------------------------------------------------

elif plotopt=='rebin':                             # Rebin data
   bursts=None                                     # Remove burst data
   fldtxt=''
   while True:                                     # Keep asking until a good
                                                   # response is given
      try:
         binning=float(raw_input("Enter bin size (s): "))      # Ask for binsize in
                                                               #dialogue box
         assert binning>=minbin
         break
      except:
         print 'Invalid bin size input!'
   print 'Binning File 1...'
   x1,y1,ye1=pan.binify(x1r,y1r,ye1r,binning)          # Bin File 1 using 'binify'
                                                       # in pan_lib
   if nfiles>1:
      print 'Binning File 2...'
      x2,y2,ye2=pan.binify(x2r,y2r,ye2r,binning)       # Bin File 2 using 'binify'
                                                       # in pan_lib
      if nfiles>2:
```

```
            print 'Binning File 3...'
            x3,y3,ye3=pan.binify(x3r,y3r,ye3r,binning)    # Bin File 3 using 'binify'
                                                          # in pan_lib
      if nfiles>1:                                      # Checking file lengths match
         if len(x1)!=len(x2):
            wrongsize=True
      if nfiles==3:
         if len(x1)!=len(x3):
            wrongsize=True
            x3l=len(x3)
      if wrongsize:                                     # Forcing file lengths to match
                                                        # if possible
         mindex=min(len(x1),len(x2),x3l)-1
         if x1[mindex]!=x2[mindex]:
            print 'Cannot crop, aborting!'
            pan.signoff()
            exit()
         if nfiles==3:
            if x1[mindex]!=x3[mindex]:
               print 'Cannot crop, aborting!'
               pan.signoff()
               exit()
         mindex+=1
         x1=x1[:mindex]
         y1=y1[:mindex]
         ye1=ye1[:mindex]
         x2=x2[:mindex]
         y2=y2[:mindex]
         ye2=ye2[:mindex]
         if nfiles==3:
            x3=x3[:mindex]
            y3=y3[:mindex]
            ye3=ye3[:mindex]
      gmask=getmask(x1,gti)                             # Re-establish gmask
      print 'Binning complete!'
      print ''
      times,timese,flux,fluxe,ys,yes,col,cole=colorget() # Re-get colours
      folded=False                                        # Re-allow clipping
      print 'Done!'
      print ''

   #-----'fold' Option-----------------------------------------------------------------

   elif plotopt=='fold':                                # Fold lightcurve
      bursts=None                                       # Remove burst data
      if folded:
         print 'Data already folded!  Rebin before re-folding.'
         continue
      if not module_pyastro:                            # Only attempt to fold if pyastro
                                                        # is present
         print 'PyAstronomy Module not found!  Cannot perform fold!'# Warn user they
                                                                     # cannot fold as
                                                                     # module is missing
         continue
      while True:                                       # Keep asking user until they
                                                        # give a sensible period
         try:
            period=float(raw_input('Input period to fold over (s): '))   # Fetch period
                                                                         # from user
```

```
            break
        except:
            print "Invalid period!"                 # Keep trying until they give a
                                                    # sensible input
    while True:                                     # Keep asking user until they
                                                    # put a sensible phase resolution
        try:
            phres=float(raw_input('Input phase resolution (0-1): ')) # Fetch phase
                                                                     # resolution from
                                                                     # user
            assert phres<1.0
            assert phres>0.0
            break
        except:
            print "Invalid phase resolution!"       # Keep trying until they give a
                                                    # sensible input
    x1=x1[gmask];y1=y1[gmask];ye1=ye1[gmask]        # Zeroing all data points outside
                                                    # of GTI
    x1,y1,ye1=pan.foldify(x1,y1,ye1,period,binning,phres=phres,name='ch. '+ch[1])
                                                    # Fold using foldify function
                                                    # from pan_lib
    fldtxt='Folded '
    if nfiles>1:
        x2=x2[gmask];y2=y2[gmask];ye2=ye2[gmask]    # Zeroing all data points outside
                                                    # of GTI
        x2,y2,ye2=pan.foldify(x2,y2,ye2,period,binning,phres=phres,name='ch. '+ch[2])
                                                    # Fold data of file 2 if present
    if nfiles==3:
        x3=x3[gmask];y3=y3[gmask];ye3=ye3[gmask]    # Zeroing all data points outside
                                                    # of GTI
        x3,y3,ye3=pan.foldify(x3,y3,ye3,period,binning,phres=phres,name='ch. '+ch[3])
                                                    # Fold data of file 3 if present
    gmask=np.ones(len(x1),dtype=bool)               # Re-establish gmask
    times,timese,flux,fluxe,ys,yes,col,cole=colorget()  # Re-get colours
    folded=True
    print 'Folding Complete!'
    print ''

#-----'norm time' Option-------------------------------------------------------------

elif plotopt=='norm time':
    if folded:
        print 'Cannot renormalise time on folded data!'
        continue
    times=times-times[0]
    print 'Renormalised times!'

#-----'autofold' Option--------------------------------------------------------------

elif plotopt=='autofold':                           # Autofold data lightcurve
    bursts=None                                     # Remove burst data
    if folded:
        print 'Data already folded!  Rebin before re-folding.'
        continue
    if not module_pyastro:                          # Only attempt to fold if pyastro
                                                    # is present
        print 'PyAstronomy Module not found!  Cannot perform fold!'
                                                    # Warn user they cannot fold as
                                                    # module is missing
```

```
        continue
    ls_st=max(4.0/(times[-1]-times[0]),0.005)
    ls_end=0.5/binning
    lsx=np.arange(ls_st,ls_end,(ls_end-ls_st)/2500.0) # Perform Lomb-Scargle Analysis
                                                      # on the data to seek best
                                                      # period
    lsy=pan.lomb_scargle(times,flux,fluxe,lsx)
    period=1.0/(lsx[lsy.tolist().index(max(lsy))])
    while True:                                       # Keep asking user until they put
                                                      # a sensible phase resolution
        try:
            phres=float(raw_input('Input phase resolution (0-1): '))
                                                      # Get phase resolution from user
            assert phres<1.0
            assert phres>0.0
            break
        except:
            print "Invalid phase resolution!"         # Keep trying until they give a
                                                      # sensible input
    print ''
    print 'Using period of '+str(period)+'!'
    x1=x1[gmask];y1=y1[gmask];ye1=ye1[gmask]          # Zeroing all data points outside
                                                      # of GTI
    x1,y1,ye1=pan.foldify(x1,y1,ye1,period,binning,phres=phres,name='ch. '+ch[1])
                                                      # Fold using foldify function
                                                      # from pan_lib
    fldtxt='Folded '
    if nfiles>1:
        x2=x2[gmask];y2=y2[gmask];ye2=ye2[gmask]      # Zeroing all data points outside
                                                      # of GTI
        x2,y2,ye2=pan.foldify(x2,y2,ye2,period,binning,phres=phres,name='ch. '+ch[2])
                                                      # Fold data of file 2 if present
    if nfiles==3:
        x3=x3[gmask];y3=y3[gmask];ye3=ye3[gmask]      # Zeroing all data points outside
                                                      # of GTI
        x3,y3,ye3=pan.foldify(x3,y3,ye3,period,binning,phres=phres,name='ch. '+ch[3])
                                                      # Fold data of file 3 if present
    gmask=np.ones(len(x1),dtype=bool)                 # Re-establish gmask
    times,timese,flux,fluxe,ys,yes,col,cole=colorget()       # Re-get colours
    folded=True
    print 'Folding Complete!'
    print ''

#-----'varifold' Option-----------------------------------------------------------------

elif plotopt=='varifold':
    if folded:
        print 'Cannot perform burst analysis on folded data!'
        continue
    if burst_alg=='cubic spline':
        while True:
            try:
                iq_lo=float(raw_input('Low Threshold:  '))
                iq_hi=float(raw_input('High Threshold: '))
                assert iq_hi>iq_lo
                assert iq_hi<=100
                assert iq_lo>=0
                break
            except AssertionError:
```

```
        print 'Invalid Entry!  Valid entry is of the form High>Low.'
else:
   iq_lo=0
   iq_hi=100
while True:
   try:
      phase_res=float(raw_input('Input phase resolution (0-1): '))
      assert phase_res<1.0
      assert phase_res>0.0
      break
   except AssertionError:
      print 'Invalid Phase Resolution!'
phases,numpeaks,flpeaks=pan.fold_bursts(times,flux,iq_hi,iq_lo,do_smooth=False,
                                        alg=burst_alg,savgol=5)
peaksep=(times[-1]-times[0])/numpeaks
print numpeaks,'flares identified: average separation of',str(peaksep)+'s'
st_time=flpeaks[0]
endtime=flpeaks[1]
intran=np.array(range(len(times)))
ymask1=intran>=st_time
ymask2=intran<endtime
ymask=ymask1&ymask2
nbins=int(1.0/phase_res)
print len(phases),len(x1),len(times)
print len(gmask)
phases=(nbins*phases[ymask]).astype(int)
x1=x1[gmask][ymask];y1=y1[gmask][ymask];ye1=ye1[gmask][ymask]
                                                # Removing all data points
                                                # outside of GTI
newx1=[]
newy1=[]
newye1=[]
for i in range(nbins):
   newx1.append(float(i)/float(nbins))
   newy1.append(np.mean(y1[phases==i]))
   newye1.append((np.sum(ye1[phases==i]**2))**0.5/len(ye1[phases==i]))
x1=np.array(newx1)
#y1=sig.savgol(np.array(newy1),5,3)
#ye1=sig.savgol(np.array(newye1),5,3)
y1=np.array(newy1)
ye1=np.array(newye1)
if nfiles>1:
   x2=x2[gmask][ymask];y2=y2[gmask][ymask];ye2=ye2[gmask][ymask]
                                                # Removing all data points
                                                # outside of GTI
   newx2=[]
   newy2=[]
   newye2=[]
   for i in range(nbins):
      newx2.append(float(i)/float(nbins))
      newy2.append(np.mean(y2[phases==i]))
      newye2.append((np.sum(ye2[phases==i]**2))**0.5/len(ye2[phases==i]))
   x2=np.array(newx2)
   y2=np.array(newy2)
   ye2=np.array(newye2)
if nfiles==3:
   x3=x3[gmask][ymask];y3=y3[gmask][ymask];ye3=ye3[gmask][ymask]
                                                # Removing all data points
                                                # outside of GTI
```

```
        newx3=[]
        newy3=[]
        newye3=[]
        for i in range(nbins):
            newx3.append(float(i)/float(nbins))
            newy3.append(np.mean(y3[phases==i]))
            newye3.append((np.sum(ye3[phases==i]**2))**0.5/len(ye3[phases==i]))
        x3=np.array(newx3)
        y3=np.array(newy3)
        ye3=np.array(newye3)
    gmask=np.ones(len(x1),dtype=bool)                # Re-establish gmask
    times,timese,flux,fluxe,ys,yes,col,cole=colorget()       # Re-get colours
    folded=True
    print 'Folding Complete!'
    print ''
    period='N/A'

#-----Get GTIs-----------------------------------------------------------------------

elif plotopt=='get gtis':
    if folded:
        print 'Cannot perform burst analysis on folded data!'
        continue
    if burst_alg=='cubic spline':
        while True:
            try:
                iq_lo=float(raw_input('Low Threshold:  '))
                iq_hi=float(raw_input('High Threshold: '))
                assert iq_hi>iq_lo
                assert iq_hi<=100
                assert iq_lo>=0
                break
            except AssertionError:
                print 'Invalid Entry!  Valid entry is of the form High>Low.'
    else:
        iq_lo=0
        iq_hi=100
    while True:
        try:
            nphbins=int(raw_input('Number of phase bins: '))
            assert nphbins>1
            break
        except AssertionError:
            print 'Invalid Phase Resolution!'
    spline=pan.get_phases_intp(flux,windows=1,q_lo=iq_lo,q_hi=iq_hi,peaks=None,
                               givespline=True)
    start_valid=times[spline.firstpeak]+tst1
    end_valid=times[spline.lastpeak]+tst1
    numpeaks=int(spline(spline.lastpeak))
    print 'Spline created, extracting phases...'
    print ''
    flnm_prefix=raw_input('Filename Prefix: ')
    gtif={}
    for i in range(nphbins):
        gtif[i]=open(flnm_prefix+'_'+str(i)+'.csv','w')
    guess=spline(0)
    prevcut=optm.fsolve(spline,guess)[0]*binning+tst1
    for i in range(numpeaks):
        for j in range(nphbins):
```

```
            subval=i+((j+1)/float(nphbins))
            def newspline(x):
                return spline(x)-subval
            newcut=optm.fsolve(newspline,guess)[0]*binning+tst1
            if newcut<end_valid and prevcut>start_valid:
               gtif[j].write(str(prevcut)+','+str(newcut)+'\n')
            prevcut=newcut
   for i in range(nphbins):
       gtif[i].close()
   print ''
   print 'GTI files written!'

#-----'Plot Bursts' Option-----------------------------------------------------------

elif plotopt=='plot bursts':
   if folded:
      print 'Cannot perform burst analysis on folded data!'
      continue
   while True:
      try:
         q_lo=float(raw_input('Low Threshold : '))
         assert q_lo<100
         assert q_lo>0
         break
      except:
         pass
   while True:
      try:
         q_hi=float(raw_input('High Threshold: '))
         assert q_hi<100
         assert q_hi>0
         assert q_hi>=q_lo
         break
      except:
         pass
   peaks=pan.get_bursts_windowed(flux,1,q_lo=q_lo,q_hi=q_hi,smooth=False)
   pl.figure()
   doplot(times,timese,flux,fluxe,ovr=True,per2=False)
   for i in peaks:
      pl.plot([times[i]],[flux[i]],'g*',zorder=5)
   pl.axhline(np.percentile(flux,q_lo),color='b',zorder=2)
   pl.axhline(np.percentile(flux,q_hi),color='r',zorder=2)
   plot_save(saveplots,show_block)

#-----'clip' Option-----------------------------------------------------------------

elif plotopt=='clip':                                # Clipping data
   bursts=None                                       # Remove burst data
   if folded:
      print 'Cannot clip folded data!'
   else:
      print 'Clipping data'
      print ''
      print 'Time range is '+str(x1[0])+'s - '+str(x1[-1])+'s'
      print 'Please choose new range of data:'
      mint,maxt,srbool=pan.srinr(x1,binning,'time')# Fetch new time domain endpoints
                                                    # using srinr function from
                                                    # pan_lib
      if srbool:
```

```
         print 'Clipping...'
         x1=x1[mint:maxt]                        # Clip file 1
         y1=y1[mint:maxt]
         ye1=ye1[mint:maxt]
         if nfiles>1:
            x2=x2[mint:maxt]                     # Clip file 2
            y2=y2[mint:maxt]
            ye2=ye2[mint:maxt]
         if nfiles==3:
            x3=x3[mint:maxt]                     # Clip file 3
            y3=y3[mint:maxt]
            ye3=ye3[mint:maxt]
         gmask=getmask(x1,gti)                   # Re-establish gmask
         times,timese,flux,fluxe,ys,yes,col,cole=colorget() # Re-get colours
         print 'Data clipped!'

#-----'mask' Option-----------------------------------------------------------

elif plotopt=='mask':
   bursts=None                                   # Remove burst data
   if folded:
      print 'Cannot mask folded data!'
   else:
      print 'Masking data'
      print ''
      print 'Select time range to mask: '
      mint,maxt,srbool=pan.srinr(x1,binning,'time')# Fetch time domain endpoints of
                                                 # bad window using srinr function
                                                 # from pan_lib
      if srbool:
         print 'Masking...'
         gmask[mint:maxt]=False                  # Force all values inside the bad
                                                 # window to appear as outside of
                                                 # GTIs
         times,timese,flux,fluxe,ys,yes,col,cole=colorget()    # Re-get colours
         print 'Data masked!'

#-----'rms' Option------------------------------------------------------------

elif plotopt=='rms':
   if nfiles>1:                                  # If more than one file loaded,
                                                 # prompt user to select one
      if nfiles==3:
         is_band_3=', 3'
      else:
         is_band_3=''
      selected_band=raw_input('Select Energy Band [1, 2'+is_band_3+', All]: '
                              ).lower()
      if selected_band not in ['1','2','3','all']:
         print 'Invalid band!'
         continue
   else:
      selected_band='all'
   if selected_band=='1':                        # Fetch the rms
      rms=pan.rms(y1)
   elif selected_band=='2':
      rms=pan.rms(y2)
   elif selected_band=='3':
      rms=pan.rms(y3)
```

```
   else:
      rms=pan.rms(flux)
   if rms=='div0':                                  # If the mean of the data is 0,
                                                    # fail safely
      print 'Error!  Div 0 Encountered!  Aborting!'
   else:
      print 'rms =',str(rms*100)+'%'                # Otherwise, print RMS

#-----'rmsflux' Option-----------------------------------------------------------

elif plotopt=='rmsflux':
   try:
      trmsbin=float(raw_input('Time binning: '))
      frmsbin=float(raw_input('Rate binning: '))
   except:
      print 'Invalid value(s)!'
      continue
   ilen=int(trmsbin/binning)
   numbins=int(len(times)/ilen)
   frflux=[]
   frrms_=[]
   frrmse=[]
   for i in range(numbins):
      kst=i*ilen
      ked=(i+1)*ilen
      kflux=flux[kst:ked]
      kfluxe=fluxe[kst:ked]
      arms,armse=pan.rms(kflux,data_err=kfluxe,with_err=True)
      frrms_.append(arms)
      frrmse.append(armse)
      frflux.append(np.mean(kflux))
   frflux=np.array(frflux)
   frrms_=np.array(frrms_)*frflux
   frrmse=np.array(frrmse)*frflux
   cflux=[]
   crms_=[]
   crmse=[]
   top=int(max(frflux)/frmsbin) +1
   for i in range(0,top):
      flow=i*frmsbin
      f_hi=(i+1)*frmsbin
      mask=np.logical_and(frflux>=flow,frflux<f_hi)
      if np.sum(mask)==0:
         continue
      mask=np.logical_and(mask,np.logical_not(np.isnan(frrms_)))
      cflux.append((flow+f_hi)/2.0)
      crms_.append(np.mean(frrms_[mask]))
      crmse.append(np.sqrt(np.sum(frrmse[mask])**2 )/sum(mask))
   pl.figure()
   pl.errorbar(cflux,crms_,yerr=crmse,fmt='x',color='0.7',zorder=20)
   pl.plot(cflux,crms_,'kx',zorder=25)
   pl.xlabel('Rate (cts s$^{-1}$')
   pl.ylabel('RMS (cts s$^{-1}$')
   plot_save(saveplots,show_block)

#-----'lc' Option----------------------------------------------------------------

elif plotopt=='lc':                                 # Plot lightcurve
   taxis='Phase' if folded else 'Time (s)'
```

```
   pl.figure()
   doplot(times,timese,flux,fluxe,ovr=True,per2=folded)     # Plot flux/time using
                                                            # doplot from pan_lib
   pl.xlabel(taxis)
   pl.ylabel(flux_axis)
   pl.ylim(ymin=0)
   pl.title(fldtxt+'Lightcurve'+qflav)
   plot_save(saveplots,show_block)
   print 'Lightcurve plotted!'

#-----'bg' Option-----------------------------------------------------------------

elif plotopt=='bg':                                 # Plot background lightcurve
   is_bdata=(bdata1!=None)
   if nfiles>=2:
      is_bdata=is_bdata and (bdata2!=None)
   if nfiles==3:
      is_bdata=is_bdata and (bdata3!=None)
   if is_bdata:                                     # Only proceed if all infiles
                                                    # have background data
      try:
         bdata_x=bdata1[0]
         bdata_y=bdata1[1]
         if nfiles>=2:
            bdata_y+=bdata2[1]
         if nfiles==3:
            bdata_y+=bdata3[1]                      # Sum counts of all <=3
                                                    # backgrounds
         pl.figure()
         pl.plot(bdata_x,bdata_y)
         pl.xlabel('Time (s)')
         pl.ylabel(flux_axis)
         pl.ylim(ymin=0)
         pl.title(fldtxt+'Lightcurve'+qflav)
         plot_save(saveplots,show_block)
         print 'Background plotted!'
      except:
         print 'Backgrounds inconsistenly formatted!' # Abort if backgrounds are
                                                      # somehow of different lengths
   else:
      print 'Not all files have background data!'

#-----'bgdump' Option-------------------------------------------------------------

elif plotopt=='bgdump':                             # Export background lightcurve
   is_bdata=(bdata1!=None)
   if nfiles>=2:
      is_bdata=is_bdata and (bdata2!=None)
   if nfiles==3:
      is_bdata=is_bdata and (bdata3!=None)
   if is_bdata:                                     # Only proceed if all infiles
                                                    # have background data
      try:
         bdata_x=bdata1[0]
         bdata_y=bdata1[1]
         if nfiles>=2:
            bdata_y+=bdata2[1]
         if nfiles==3:
            bdata_y+=bdata3[1]                      # Sum counts of all <=3
```

```
                                                # backgrounds
      except:
         print 'Backgrounds inconsistenly formatted!' # Abort if backgrounds are
                                                      # somehow of different lengths
         continue
      ofilename=raw_input('Save textfile as: ')    # Fetch filename from user
      ofil = open(ofilename, 'w')                  # Open file
      for i in range(len(bdata_x)):
         row=['0.0']*3                             # Create a row of strings reading
                                                   # 0.0, append data into it
         row[0]=str(bdata_x[i]+time_n)+' '         # Column 01: Time
         row[1]=str(bdata_y[i])+' '                # Column 02: Rate
         row[2]='\n'
         ofil.writelines(row)
      print 'Background saved as '+ofilename+'!'
   else:
      print 'Not all files have background data!'

#-----'export' Option-----------------------------------------------------------------

elif plotopt=='export':                            # Export lightcurve to ASCII file
   ofilename=raw_input('Save textfile as: ')       # Fetch filename from user
   ofil = open(ofilename, 'w')                     # Open file
   if folded:
      itr=[0,1]
   else:
      itr=[0]
   for ti in itr:
      for i in range(len(times)):
         row=['0.0 ']*15                           # Create a row of strings reading
                                                   #  0.0, append data into it
         row[0]=str(times[i]+time_n+ti)+' '        # Column 00: Time
         row[1]=str(timese[i])+' '                 # Column 01: Time Error
         row[2]=str(flux[i])+' '                   # Column 02: Total Flux
         row[3]=str(fluxe[i])+' '                  # Column 03: Total Flux Error
         row[4]=str(y1[gmask][i])+' '              # Column 04: Band 1 Flux
         row[5]=str(ye1[gmask][i])+' '             # Column 05: Band 1 Flux Error
         row[14]='\n'                              # Column 14: Return (so further
                                                   #  data will be appended to a new
                                                   #  line)
         if nfiles>1:                              # If 2+ bands are given:
            row[6]=str(y2[gmask][i])+' '           # Column 06: Band 2 Flux
            row[7]=str(ye2[gmask][i])+' '          # Column 07: Band 2 Flux Error
            row[10]=str(col[21][i])+' '            # Column 10: [2/1] Colour
            row[11]=str(cole[21][i])+' '           # Column 11: [2/1] Colour Error
         if nfiles==3:                             # If 3 bands are given:
            row[8]=str(y3[gmask][i])+' '           # Column 08: Band 3 Flux
            row[9]=str(ye3[gmask][i])+' '          # Column 09: Band 3 Flux Error
            row[12]=str(col[31][i])+' '            # Column 12: [3/1] Colour
            row[13]=str(cole[31][i])+' '           # Column 13: [3/1] Colour Error
         ofil.writelines(row)                      # Append row of data into open
                                                   #  file
   ofil.close()                                    # Close file
   print 'Data saved!'

#-----'timenorm' Option---------------------------------------------------------------

elif plotopt=='timenorm':
   print ''
```

```
   if time_n==0:
      time_n=tst1
      print 'Using absolute time!'
   else:
      time_n=0
      print 'Using relative time!'
   print ''

#-----'animate' Option-----------------------------------------------------------------

elif plotopt=='animate':
   animsloc=raw_input('Folder to save images: ')
   print ''
   if os.path.exists(animsloc):                    # Create the folder
      print 'Folder "'+animsloc+'" already exists...'
   else:
      print 'Creating folder "'+animsloc+'"...'
      os.makedirs(animsloc)
   here=os.getcwd()                                # Get current working directory
                                                   # (to move back to later)
   os.chdir(animsloc)                              # Change working directory to
                                                   # animation location
   animbin=0.0025                                  # Start with an arbitrarily low
                                                   # binsize
   while animbin<max(bsz1,bsz2,bsz3,minbin):       # Find lowest allowable binsize
                                                   # of the form 0.01*2^N
      animbin=animbin*2
   anstep=1                                        # Track the number of steps taken
   dst=times[0]                                    # Fetch largest and smallest
                                                   # times to use to force same
                                                   # scale on all graphs
   det=times[-1]
   while animbin<(det-dst)/4.0:                    # Set the maximum binsize at one
                                                   # quarter of the observation
                                                   # length
      print "Creating",str(animbin)+"s binned lightcurve"
      x1,y1,ye1=pan.binify(x1r,y1r,ye1r,animbin)   # Bin File 1 using 'binify' in
                                                   # pan_lib
      if nfiles>1:
         x2,y2,ye2=pan.binify(x2r,y2r,ye2r,animbin)# Bin File 2 using 'binify' in
                                                   # pan_lib
         if nfiles>2:
            x3,y3,ye3=pan.binify(x3r,y3r,ye3r,animbin)
                                                   # Bin File 3 using 'binify' in
                                                   # pan_lib
      mina,maxa,srbool=pan.srinr(x1,binning,'time',minv=dst,maxv=det)
                                                   # Clip each individual lightcurve
      if srbool:
         x1=x1[mina:maxa]                          # Clip file 1
         y1=y1[mina:maxa]
         ye1=ye1[mina:maxa]
         if nfiles>1:
            x2=x2[mina:maxa]                       # Clip file 2
            y2=y2[mina:maxa]
            ye2=ye2[mina:maxa]
         if nfiles==3:
            x3=x3[mint:maxt]                       # Clip file 3
            y3=y3[mint:maxt]
            ye3=ye3[mint:maxt]
```

```
      gmask=getmask(x1,gti)                          # Re-establish gmask
      times,timese,flux,fluxe,ys,yes,col,cole=colorget(verbose=False)
                                                     # Re-get colours
      if anstep==1:
         if es:
           merr=max(fluxe)
         else:
           merr=0
         maxany=max(flux)+merr                       # Calculate the scale of all
                                                     # plots based on the range of the
                                                     # first plot
         minany=min(flux)-merr
         if minany<0: minany=0
      taxis='Phase' if folded else 'Time (s)'
      pl.figure()
      doplot(times,timese,flux,fluxe,ovr=True)       # Plot the graph using doplot
      pl.xlabel(taxis)
      pl.ylabel(flux_axis)
      pl.title('Lightcurve ('+str(animbin)+'s binning)')
      pl.xlim(dst,det)
      pl.ylim(max(minany,0),maxany)
      pl.savefig(str("%04d" % anstep)+'.png')        # Save the figure with leading
                                                     # zeroes to preserve order when
                                                     # int convereted to string
      pl.close()
      anstep+=1                                      # Increment the step tracker
      animbin=animbin*2                              # Double the binsize
   print 'Cleaning up...'
   os.system ("convert -delay 10 -loop 0 *.png animation.gif")
                                                     # Use the bash command 'convert'
                                                     # to create the animated gif
   x1,y1,ye1=pan.binify(x1r,y1r,ye1r,binning)        # Reset binning of File 1 using
                                                     # 'binify' in pan_lib
   if nfiles>1:
      x2,y2,ye2=pan.binify(x2r,y2r,ye2r,binning)     # Reset binning of File 2 using
                                                     # 'binify' in pan_lib
      if nfiles>2:
         x3,y3,ye3=pan.binify(x3r,y3r,ye3r,binning)# Reset binning File 3 using
                                                     # 'binify' in pan_lib
   gmask=getmask(x1,gti)                             # Re-establish gmask
   times,timese,flux,fluxe,ys,yes,col,cole=colorget(verbose=False) # Re-get colours
   print ''
   print "Animation saved to",animsloc+'/animation.gif!'
   os.chdir(here)


#-----'hidxy' Option-----------------------------------------------------------------

elif plotopt[:3]=='hid':                             # Plot x/y HID
   ht=plotopt[3:]                                    # Collect the xy token from the
                                                     # user
   if nfiles>1:
      if not (ht in ['12','13','21','23','31','32']):
                                                     # Check that the token is 2 long
                                                     # long and contains two different
                                                     # characters of the set [1,2,3]
         print 'Invalid command!'
         print ''
         print 'Did you mean...'
```

```
            print ''
            print 'HID options:'
            print '* "hid21" for 2/1 colour'
            print '* "hid12" for 1/2 colour'
            if nfiles==3:
               print '* "hid32" for 3/2 colour'
               print '* "hid23" for 2/3 colour'
               print '* "hid31" for 3/1 colour'
               print '* "hid13" for 1/3 colour'
         elif ('3' in ht) and (nfiles<3):
            print 'Not enough infiles for advanced HID!'
                                                  # If token contains a 3 but only
                                                  # 2 infiles are used, abort!
         else:
            h1=int(ht[0])                         # Extract numerator file number
            h2=int(ht[1])                         # Extract denominator file number
            ht=int(ht)
            pl.figure()
            doplot(col[ht],cole[ht],flux,fluxe)   # Collect colours from col
                                                  # library and plot
            pl.ylabel(flux_axis)
            pl.xlabel('('+ch[h1]+'/'+ch[h2]+') colour')
            pl.title(fldtxt+'Hardness Intensity Diagram'+qflav)
            plot_save(saveplots,show_block)
            print 'File'+str(h1)+'/File'+str(h2)+' HID plotted!'
      else:
         print 'Not enough infiles for HID!'

#-----'calcloopxy' Option-------------------------------------------------------------

elif plotopt[:8]=='calcloop':                     # Calculate likelihood of loop in
                                                  # HID
   ht=plotopt[8:]                                 # Collect the xy token from the
                                                  # user
   if nfiles>1:
      if not (ht in ['12','13','21','23','31','32']):
                                                  # Check that the token is 2 long
                                                  # and contains two different
                                                  # characters of the set [1,2,3]
         print 'Invalid command!'
         print ''
         print 'Did you mean...'
         print ''
         print 'CalcLoop options:'
         print '* "calcloop21" for 2/1 HID loop calculation'
         print '* "calcloop12" for 1/2 HID loop calculation'
         if nfiles==3:
            print '* "calcloop32" for 3/2  HID loop calculation'
            print '* "calcloop23" for 2/3  HID loop calculation'
            print '* "calcloop31" for 3/1  HID loop calculation'
            print '* "calcloop13" for 1/3  HID loop calculation'
         elif ('3' in ht) and (nfiles<3):
            print 'Not enough infiles for advanced HID!'
                                                  # If token contains a 3 but only
                                                  # 2 infiles are used, abort!
         else:
            h1=int(ht[0])                         # Extract numerator file number
            h2=int(ht[1])                         # Extract denominator file number
            ht=int(ht)
```

```
            lkl=pan.calcloop(flux,col[ht],fluxe,cole[ht])
                                                    # Collect likelihood of loop from
                                                    # pan_lib
            print 'Null hypothesis (no hysteresis) probability = '+str(lkl)
      else:
         print 'Not enough infiles for HID!'

#-----'hardnessxy' Option-------------------------------------------------------

elif plotopt[:8]=='hardness':                       # Plot x/y hardness/time plot
      ht=plotopt[8:]                                # Collect the xy token from the
                                                    # user
      if nfiles>1:
         if not (ht in ['12','13','21','23','31','32']):
                                                    # Check that the token is 2 long
                                                    # and contains two different
                                                    # characters of the set [1,2,3]
            print 'Invalid command!'
            print ''
            print 'Did you mean...'
            print ''
            print 'HID options:'
            print '* "hardness21" for 2/1 hardness over time'
            print '* "hardness12" for 1/2 hardness over time'
            if nfiles==3:
               print '* "hardness32" for 3/2 hardness over time'
               print '* "hardness23" for 2/3 hardness over time'
               print '* "hardness31" for 3/1 hardness over time'
               print '* "hardness13" for 1/3 hardness over time'
         elif ('3' in ht) and (nfiles<3):
            print 'Not enough infiles for advanced HID!'
                                                    # If token contains a 3 but only
                                                    # 2 infiles are used, abort!
         else:
            h1=int(ht[0])                           # Extract numerator file number
            h2=int(ht[1])                           # Extract denominator file number
            ht=int(ht)
            pl.figure()
            doplot(times,timese,col[ht],cole[ht],ovr=True,per2=folded)
                                                    # Collect colours from col
                                                    # library and plot
            pl.ylabel('('+ch[h1]+'/'+ch[h2]+') colour')
            pl.xlabel(taxis)
            pl.title(fldtxt+'Hardness/Time plot'+qflav)
            plot_save(saveplots,show_block)
            print 'File'+str(h1)+'/File'+str(h2)+' Hardness/time diagram plotted!'
      else:
         print 'Not enough infiles for HID!'

#-----'compbandsxy' Option------------------------------------------------------

elif plotopt[:9]=='compbands':                      # Plot two bands against each
                                                    # other
      ht=plotopt[9:]                                # Collect the xy token from the
                                                    # user
      if nfiles>1:
         if not (ht in ['12','13','21','23','31','32']):
                                                    # Check that the token is 2 long
                                                    # and contains two different
```

```
                                                # characters of the set [1,2,3]
         print 'Invalid command!'
         print ''
         print 'Did you mean...'
         print ''
         print 'HID options:'
         print '* "compbands21" to plot lightcurve 2 against 1'
         print '* "compbands12" to plot lightcurve 1 against 2'
         if nfiles==3:
            print '* "compbands32" to plot lightcurve 3 against 2'
            print '* "compbands23" to plot lightcurve 2 against 3'
            print '* "compbands31" to plot lightcurve 3 against 1'
            print '* "compbands13" to plot lightcurve 1 against 3'
      elif ('3' in ht) and (nfiles<3):
         print 'Not enough infiles for advanced lightcurve comparison!'
                                                # If token contains a 3 but only
                                                # 2 infiles are used, abort!
      else:
         h1=int(ht[0])                          # Extract numerator file number
         h2=int(ht[1])                          # Extract denominator file number
         pl.figure()
         doplot(ys[h1],yes[h1],ys[h2],yes[h2])  # Collect colours from col
                                                # library and plot
         pl.ylabel(r'Band '+str(h1)+r' rate (cts s$^{-1}$ PCU$^{-1}$)')
         pl.xlabel(r'Band '+str(h2)+r' rate (cts s$^{-1}$ PCU$^{-1}$)')
         pl.title(fldtxt+'Lightcurve Comparison Diagram'+qflav)
         plot_save(saveplots,show_block)
         print 'File'+str(h1)+'/File'+str(h2)+' LC Comparison plotted!'
   else:
      print 'Not enough infiles for lightcurve comparison!'

#-----'autocor' Option-----------------------------------------------------------

elif plotopt=='autocor':                        # Plot autocorrelation function
   ccor=sig.correlate(flux-np.mean(flux),flux-np.mean(flux),mode='same')/
                     (len(flux)*(np.std(flux)**2))
   nlen=len(times)
   cct=range(nlen)                              # Set up the lag axis
   cct=((np.array(cct)-nlen/2.0)*binning).tolist()
   pl.figure()
   pl.plot(cct,ccor)
   pl.xlabel('Lag (s)')
   pl.ylabel('Cross-Correlation')
   plot_save(saveplots,show_block)
   print 'Autocorrelation diagram plotted!'

#-----'crosscor' Option----------------------------------------------------------

elif plotopt[:8]=='crosscor':                   # Plot cross-correlation function
   ht=plotopt[8:]                               # Collect the xy token from the
                                                # user
   if nfiles>1:
      if not (ht in ['12','13','21','23','31','32']):
                                                # Check that the token is 2 long
                                                # and contains two different
                                                # characters of the set [1,2,3]
         print 'Invalid command!'
         print ''
         print 'Did you mean...'
```

```
            print ''
            print 'HID options:'
            print '* "crosscor21" for 2 against 1 cross-correlation'
            if nfiles==3:
               print '* "crosscor32" for 3 against 2 cross-correlation'
               print '* "crosscor31" for 3 against 1 cross-correlation'
         elif ('3' in ht) and (nfiles<3):
            print 'Not enough infiles for advanced cross-correlation!'
                                                   # If token contains a 3 but only
                                                   # 2 infiles are used, abort!
         else:
            nlen=len(times)
            h1=int(ht[0])                          # Extract file 1 number
            h2=int(ht[1])                          # Extract file 2 number
            h1a=ys[h1]                             # Extract dataset 1
            h2a=ys[h2]                             # Extract dataset 2
            if nlen%2==0:                          # Cross-correlation only gives a
                                                   # point at zero lag if an even
                                                   # number of points are input
               nlen-=1
               h1a=h1a[:-1]
               h2a=h2a[:-1]
            ccor=sig.correlate(h1a-np.mean(h1a),h2a-np.mean(h2a),mode='same')/
                              (nlen*np.std(h1a)*np.std(h2a))
            cct=range(nlen)                        # Set up the lag axis
            cct=(((np.array(cct)-nlen/2.0)*binning)+(binning/2.0)).tolist()
            pl.figure()
            pl.plot(cct,ccor)
            pl.axvline(0,linestyle=':',color='0.7')
            pl.xlabel('Band '+str(h1)+' lag (s) wrt Band '+str(h2))
            pl.ylabel('Cross-Correlation')
            plot_save(saveplots,show_block)
            print 'Band '+str(h1)+' lag against band '+str(h2)+
                  ' (cross-correlation) diagram plotted!'
      else:
         print 'Not enough infiles for cross-correlation!'

#-----'timeres crosscor' Option-----------------------------------------------------

elif plotopt[:16]=='timeres crosscor':             # Plot time-resolved cross-
                                                   # correlation function
   ht=plotopt[16:]                                 # Collect the xy token from the
                                                   # user
   if nfiles>1:
      if not (ht in ['12','13','21','23','31','32']):
                                                   # Check that the token is 2 long
                                                   # and contains two different
                                                   # characters of the set [1,2,3]
         print 'Invalid command!'
         print ''
         print 'Did you mean...'
         print ''
         print 'HID options:'
         print '* "timeres crosscor21" for 2 against 1 time-resolved cross-correla'+
               'tion'
         if nfiles==3:
            print '* "timeres crosscor32" for 3 against 2 time-resolved cross-corr'+
                  'elation'
            print '* "timeres crosscor31" for 3 against 1 time-resolved cross-corr'+
```

```
                'elation'
      elif ('3' in ht) and (nfiles<3)
         print 'Not enough infiles for advanced cross-correlation!'
                                                # If token contains a 3 but only
                                                # 2 infiles are used, abort!
      else:
         try:
            nlen=int(float(raw_input('Length of segments (s): '))/binning)
            if nlen%2!=0: nlen-=1
            frang=float(raw_input('Max lag (s) : '))
         except:
            print 'Not a number!  Aborting!'
            continue
         maxtims=[]
         maxlocs=[]
         n=len(times)/nlen
         matrix=[]                              # Set up the 2d matrix
         mtimes=[]                              # Set up the time axis
         xn=range(nlen)                         # Set up the lag axis
         xn=((np.array(xn)-nlen/2.0)*binning).tolist()
         h1=int(ht[0])                          # Extract file 1 number
         h2=int(ht[1])                          # Extract file 2 number
         for i in range(n):
            y1clip=np.array(ys[h1][nlen*i:nlen*(i+1)]) # Clip the first lightcurve
                                                       # into chunks
            y2clip=np.array(ys[h2][nlen*i:nlen*(i+1)]) # Clip the second lightcurve
                                                       # into chunks
            cclip=sig.correlate(y1clip-np.mean(y1clip),y2clip-np.mean(y2clip),
                           mode='same')/(len(y2clip)*np.std(y1clip)*np.std(y2clip))
                                                       # Do the cross-correlation
            cclip=np.array(cclip)[np.array(xn)<=frang] # Cut the matrix line to the
                                                       # user requested lag range
            txn=np.array(xn)[np.array(xn)<=frang]
            cclip=cclip[txn>-frang]
            txn=txn[txn>-frang]
            maxc=max(cclip)
            maxloc=txn[cclip.tolist().index(maxc)]
            matrix.append(cclip)                # Grow the matrix!
            mtimes.append(nlen*i*binning)
            maxtims.append(nlen*(i+0.5)*binning)
            maxlocs.append(maxloc)
         pl.figure()
         pl.pcolor(np.array(mtimes+[mtimes[-1]+binning*nlen])-binning/2.0,txn,
                   np.array(matrix).T)
         pl.xlabel('Time (s)')
         pl.ylabel('Band '+str(h1)+' lag (s) wrt Band '+str(h2))
         pl.plot(maxtims,maxlocs,':k',label='Peak lag')
         pl.legend()
         plot_save(saveplots,show_block)
         print 'Band '+str(h1)+' lag against band '+str(h2)+' (time-resolved cross'+
               '-correlation) diagram plotted!'
   else:
      print 'Not enough infiles for cross-correlation!'

#-----'colxy' Option-----------------------------------------------------------------

elif plotopt[:3]=='col':                        # Plot x/y colour/t
   ht=plotopt[3:]                               # Collect the xy token from the
                                                # user
```

```
   if nfiles>1:
      if not (ht in ['12','13','21','23','31','32']):
                                                # Check that the token is 2 long
                                                # and contains two different
                                                # characters of the set [1,2,3]
         print 'Invalid command!'
         print ''
         print 'Did you mean...'
         print ''
         print 'Col/t plot options:'
         print '*"col21" for 2/1 colour'
         print '*"col12" for 1/2 colour'
         if nfiles==3:
            print '*"col32" for 3/2 colour'
            print '*"col23" for 2/3 colour'
            print '*"col31" for 3/1 colour'
            print '*"col13" for 1/3 colour'
      elif ('3' in ht) and (nfiles<3):
         print 'Not enough infiles for advanced Col/t plot!'
                                                # If token contains a 3 but only
                                                # 2 infiles are used, abort!
      else:
         taxis='Phase' if folded else 'Time (s)'
         h1=int(ht[0])                          # Extract numerator file number
         h2=int(ht[1])                          # Extract denominator file number
         ht=int(ht)
         pl.figure()
         doplot(times,timese,col[ht],cole[ht],ovr=True,per2=folded)
                                                # Collect colours from col
                                                # library and plot
         pl.xlabel(taxis)
         pl.ylabel('('+ch[h1]+'/'+ch[h2]+') colour')
         pl.ylim(ymin=0)
         pl.title(fldtxt+'Colour over Time Diagram'+qflav)
         plot_save(saveplots,show_block)
         print 'File'+str(h1)+'/File'+str(h2)+' Colour over Time Diagram plotted!'
   else:
      print 'Not enough infiles for Col/t plot!'

#-----'ccd' Option-----------------------------------------------------------------

elif plotopt=='ccd':                            # Plot Colour-Colour diagram
   if nfiles==3:
      pl.figure()
      doplot(col[31],cole[31],col[21],cole[21])
      pl.xlabel('('+ch[2]+'/'+ch[1]+') colour')
      pl.ylabel('('+ch[3]+'/'+ch[1]+') colour')
      pl.xlim(0,2)
      pl.ylim(0,2)
      pl.title(fldtxt+'Colour-Colour Diagram'+qflav)
      plot_save(saveplots,show_block)
      print 'CCD plotted!'
   else:
      print 'Not enough infiles for CCD!'

#-----'all' Option-----------------------------------------------------------------

elif plotopt=='all':
   pl.figure()
```

```
   if   nfiles==3: colexp=2; rowexp=2; gexp=4      # If 3 files given, 4 graphs will
                                                   # be plotted in a 2x2 grid
   elif nfiles==2: colexp=1; rowexp=2; gexp=2      # If 2 files given, 2 grapgs will
                                                   # be plotted in a 2x1 grid
   else:           colexp=1; rowexp=1; gexp=1      # If 1 file given, only one graph
                                                   # can be plotted
   print 'Plotting Lightcurve...'
   taxis='Phase' if folded else 'Time (s)'
   pl.subplot(rowexp,colexp,1)                     # Create subplot in the first slot
   doplot(times,timese,flux,fluxe,ovr=True)        # Always plot the lightcurve
   pl.xlabel(taxis)
   pl.ylabel(flux_axis)
   pl.ylim(ymin=0)
   pl.title(fldtxt+'Lightcurve'+qflav)
   if nfiles>1:                                    # If 2+ files given, plot 2+ file
                                                   # data products
      print 'Plotting Soft Hardness-Intensity Diagram...'
      pl.subplot(rowexp,colexp,2)                  # Create subplot in the second
                                                   # slot
      doplot(col[21],cole[21],flux,fluxe)          # Plot Soft HID
      pl.xlim(0,2)
      pl.ylim(0,300)
      pl.ylabel(flux_axis)
      pl.xlabel('('+ch[2]+'/'+ch[1]+') colour')
      pl.title(fldtxt+'Soft HID'+qflav)
   if nfiles==3:                                   # If 3 files given, plot 3 file
                                                   # data products
      print 'Plotting Hard Hardness-Intensity Diagram...'
      print 'Plotting Colour-Colour Diagram...'
      pl.subplot(rowexp,colexp,3)                  # Create subplot in the 3rd slot
      doplot(col[31],cole[31],flux,fluxe)          # Plot Hard HID
      pl.xlim(0,2)
      pl.ylim(0,300)
      pl.ylabel(flux_axis)
      pl.xlabel('('+ch[3]+'/'+ch[1]+') colour')
      pl.title(fldtxt+'Hard HID'+qflav)
      pl.subplot(rowexp,colexp,4)                  # Create subplot in the 4th slot
      doplot(col[31],cole[31],col[21],cole[21])    # Plot CCD
      pl.xlim(0,2)
      pl.ylim(0,2)
      pl.ylabel('('+ch[2]+'/'+ch[1]+') colour')
      pl.xlabel('('+ch[3]+'/'+ch[1]+') colour')
      pl.title(fldtxt+'CCD'+qflav)
   print ''
   plot_save(saveplots,show_block)
   print 'All products plotted!'

#-----'band' Option-----------------------------------------------------------------

elif plotopt=='band':                              # Plot lightcurve of individual
                                                   # band
   if nfiles==1:
      user_b_band='1'                              # Select energy band to plot
   else:
      if nfiles==3:
         is_band_3=', 3'
      else:
         is_band_3=''
      user_b_band=raw_input('Select Energy Band [1, 2'+is_band_3+']: ')
```

```
   avail_b_band=['1','2']                          # Define valid user inputs
   if nfiles==3:
      avail_b_band.append('3')                     # Add '3' as a valid input if 3
                                                   # bands present
   if user_b_band in avail_b_band:
      taxis='Phase' if folded else 'Time (s)'
      pl.figure()
      if user_b_band=='1':
         doplot(times,timese,y1[gmask],ye1[gmask],ovr=True,per2=folded)
                                                   # Plot flux/time using doplot
                                                   # from pan_lib
         b_band_name='Band 1'
      elif user_b_band=='2':
         doplot(times,timese,y2[gmask],ye2[gmask],ovr=True,per2=folded)
         b_band_name='Band 2'
      else:
         b_band_name='Band 3'
         doplot(times,timese,y3[gmask],ye3[gmask],ovr=True,per2=folded)
      pl.xlabel(taxis)                             # Format plot
      pl.ylabel(flux_axis)
      pl.ylim(ymin=0)
      pl.title(fldtxt+b_band_name+'Lightcurve'+qflav)
      plot_save(saveplots,show_block)
      print b_band_name+' lightcurve plotted!'

#-----'bands' Option-----------------------------------------------------------------

elif plotopt=='bands':                             # Plot lightcurves of individual
                                                   # bands apart
   taxis='Phase' if folded else 'Time (s)'
   pl.figure()
   pl.subplot(nfiles,1,1)
   doplot(times,timese,y1[gmask],ye1[gmask],ovr=True,per2=folded)
                                                   # Plot the lowest band
   pl.xlabel(taxis)
   pl.ylabel(flux_axis)
   pl.title(fldtxt+ch[1]+' Lightcurve'+qflav)
   if nfiles>1:
      pl.subplot(nfiles,1,2)
      doplot(times,timese,y2[gmask],ye2[gmask],ovr=True,per2=folded)
                                                   # Plot the second band
      pl.xlabel(taxis)
      pl.ylabel(flux_axis)
      pl.title(fldtxt+ch[2]+' Lightcurve'+flv2)
   if nfiles>2:
      pl.subplot(nfiles,1,3)
      doplot(times,timese,y3[gmask],ye3[gmask],ovr=True,per2=folded)
                                                   # Plot the third band
      pl.xlabel(taxis)
      pl.ylabel(flux_axis)
      pl.title(fldtxt+ch[3]+' Lightcurve'+flv3)
   plot_save(saveplots,show_block)
   print 'Banded lightcurves plotted!'

#-----'xbands' Option-----------------------------------------------------------------

# 'Same axes bands'

elif plotopt=='xbands':                            # Plot lightcurves of individual
```

```
                                                # bands together
   donorm=raw_input('Normalise bands? : ')      # Fetch whether user wants to
                                                # normalise
   donorm=donorm in ('y','yes')
   taxis='Phase' if folded else 'Time (s)'
   pl.figure()
   leg=[ch[1]]                                  # Create a legend array to
                                                # populate with channel names
   if donorm:
      n1=max(y1[gmask])
   else:
      n1=1.0
   doplot(times,timese,y1[gmask]/n1,ye1[gmask]/n1,ovr=True,ft='-b',per2=folded)
                                                # Plot the lowest band
   if nfiles>1:
      if donorm:
         n2=max(y2[gmask])
      else:
         n2=1.0
      doplot(times,timese,y2[gmask]/n2,ye2[gmask]/n2,ovr=True,ft='-g',per2=folded)
                                                # Plot the second band
      leg.append(ch[2])                         # Append name of second channel
                                                # to key
   if nfiles>2:
      if donorm:
         n3=max(y3[gmask])
      else:
         n3=1.0
      doplot(times,timese,y3[gmask]/n3,ye3[gmask]/n3,ovr=True,ft='-r',per2=folded)
                                                # Plot the third band
      leg.append(ch[3])                         # Append name of third channel to
                                                # key
   if folded:
      pl.axvline(1,linestyle=':',color='0.7')
   pl.legend(leg)                               # Create key on plot
   pl.xlabel(taxis)
   pl.ylabel(flux_axis)
   pl.title(fldtxt+'Lightcurve'+qflav)
   plot_save(saveplots,show_block)
   print 'Banded lightcurves plotted!'

#-----'burst get'-------------------------------------------------------------

elif plotopt in ['burst get','burstget','burst_get','bursts get','burstsget',
                 'bursts_get']:
   if folded:
      print 'Cannot perform burst analsysis on folded data!'
      continue
   while True:
      try:
         iq_lo=float(raw_input('Low Threshold:  '))
         iq_hi=float(raw_input('High Threshold: '))
         assert iq_hi>iq_lo
         assert iq_hi<=100
         assert iq_lo>=0
         break
      except:
         print 'Invalid Entry!  Valid entry is of the form High>Low.'
   bursts={}
```

```
    bursts['endpoints']=pan.get_bursts(flux,q_lo=iq_lo,q_hi=iq_hi,just_peaks=False,
                                       alg=burst_alg)
    pl.figure()
    doplot(times,timese,flux,fluxe,ovr=True)        # Plot flux/time using doplot
                                                    # from pan_lib
    t_lo=np.percentile(flux,iq_lo)                  # Fetch thresholds used in
                                                    # get_bursts
    t_hi=np.percentile(flux,iq_hi)
    col_toggle=True
    for i in bursts['endpoints']:
       if col_toggle:
          col_toggle=False
          burst_colour='#c7c7c7'
       else:
          col_toggle=True
          burst_colour='#e7e7e7'
       pl.axvspan(times[i[0]],times[i[1]], facecolor=burst_colour, edgecolor='none')
    pl.plot([times[0],times[-1]],[t_lo,t_lo],'g')
    pl.plot([times[0],times[-1]],[t_hi,t_hi],'b')
    pl.legend(['Flux','Low Pass Threshold','High Pass Threshold'])
    pl.xlabel('Time (s)')
    pl.ylabel(flux_axis)
    pl.ylim(ymin=0)
    pl.title(fldtxt+'Lightcurve with Bursts Highlighted'+qflav)
    plot_save(saveplots,show_block)
    print ''
    print 'Bursts plotted!'
    print len(bursts['endpoints']),'bursts found!'
    print ''
    print 'Analysing Bursts...'
    bursts['peaks']=[]
    bursts['rises']=[]
    bursts['falls']=[]
    bursts['duras']=[]
    for endpoints in bursts['endpoints']:
       burst=flux[endpoints[0]:endpoints[1]]
       btime=times[endpoints[0]:endpoints[1]]
       peak,trough,pk_time,rise_time,fall_time=pan.eval_burst(btime,burst)
       bursts['peaks'].append(peak)
       #bursts['trghs'].append(troughs)
       bursts['rises'].append(rise_time)
       bursts['falls'].append(fall_time)
       bursts['duras'].append(btime[-1]-btime[0])
    print ''
    print 'Analysis Complete!'
    print 'Burst Products now available!'

#-----'burst peaks'-----------------------------------------------------------------

elif plotopt in ['burst peaks','burstpeaks','burst_peaks','bursts peaks',
                 'burstspeaks','bursts_peaks']:
    burstplot('peaks','Peak Heights','cts/s/PCU')

#-----'burst risetimes'-------------------------------------------------------------

elif plotopt in ['burst risetimes','burstrisetimes','burst_risetimes',
                 'bursts risetimes','burstsrisetimes','bursts_risetimes']:
    burstplot('rises','Rise Times','s')
```

```
#-----'burst falltimes'-----------------------------------------------------------------

elif plotopt in ['burst falltimes','burstfalltimes','burst_falltimes',
                 'bursts falltimes','burstsfalltimes','bursts_falltimes']:
   burstplot('falls','Fall Times','s')

#-----'burst lengths'-------------------------------------------------------------------

elif plotopt in ['burst lengths','burstlengths','burst_lengths','bursts lengths',
                 'burstslengths','bursts_lengths']:
   burstplot('duras','Durations','s')

#-----'bursts help'---------------------------------------------------------------------

elif plotopt in ['burst help','bursthelp','burst_help','bursts help','burstshelp',
                 'bursts_help']:
   print 'Help coming soon.'

#-----'burst alg'-----------------------------------------------------------------------

elif plotopt in ['burst alg','burstalg','burst_alg','bursts alg','burstsalg',
                 'bursts_alg']:
   print 'Available Burst-Finding Algorithms:'
   print ' * CUBIC SPLINE'
   print ' * LOAD'
   print ''
   inp_burst_alg=raw_input('Select Burst Algorithm: ').lower()
   if inp_burst_alg in ('cubic spline','load'):
      burst_alg=inp_burst_alg
      print 'Burst-Finding Algorithm set to "'+burst_alg+'"!'
   else:
      print 'Invalid Burst-Finding Algorithm!'

#-----'lombscargle' Option--------------------------------------------------------------

elif plotopt=='lombscargle':
   if folded:                                        # If data is folded, abort
      print 'Cannot perform Lomb-Scargle on folded data!'
      continue
   if nfiles==1:
      user_scargl_bands='1'                          # Select energy bands to
                                                     # LombScargle
   else:
      if nfiles==3:
         is_band_3=', 3'
      else:
         is_band_3=''
      user_scargl_bands=raw_input('Select Energy Band [1, 2'+is_band_3+', All]: '
                                  ).lower()
   ls_st=max(4.0/(times[-1]-times[0]),0.005)
   ls_end=0.5/binning
   lsx=np.arange(ls_st,ls_end,(ls_end-ls_st)/2500.0)
   avail_scargl_bands=['1','2','all']                # Define valid user inputs
   if nfiles==3:
      avail_scargl_bands.append('3')                 # Add '3' as a valid input if 3
                                                     # bands present
   if user_scargl_bands in avail_scargl_bands:
      if user_scargl_bands=='1':
         lsy=pan.lomb_scargle(times,y1[gmask],ye1[gmask],lsx)
```

```
                                                # Perform LombScargle of band 1
                                                # using lombscargle function
                                                # defined in header
            s_band_name='band 1'
         elif user_scargl_bands=='2':
            lsy=pan.lomb_scargle(times,y2[gmask],ye2[gmask],lsx)
                                                # Perform LombScargle of band 2
                                                # using lombscargle function
                                                # defined in header
            s_band_name='band 2'
         elif user_scargl_bands=='3':
            lsy=pan.lomb_scargle(times,y3[gmask],ye3[gmask],lsx)
                                                # Perform LombScargle of band 3
                                                # using lombscargle function
                                                # defined in header
            s_band_name='band 3'
         else:
            lsy=pan.lomb_scargle(times,flux,fluxe,lsx)# Perform LombScargle of all
                                                # bands using lombscargle
                                                # function defined in header
            s_band_name='all bands'
            if user_scargl_bands!='all':
               print 'Invalid band!  Using all.'
         pl.figure()
         pl.plot(lsx,lsy,'k')                   # Plot lombscargle
         pl.xlabel('Frequency (Hz)')
         pl.ylabel('Power')
         pl.xlim(0,max(lsx))
         pl.ylim(1,100000)
         pl.yscale('log')
         pl.title('Lomb-Scargle Periodogram of '+s_band_name+qflav)
         plot_save(saveplots,show_block)
         print ''
         print 'Lomb-Scargle Diagram of '+s_band_name+' plotted!'
      else:
         print 'Invalid energy band!'

#-----'errors' Option-------------------------------------------------------------

elif plotopt in ['errors','error']:             # Toggle Errors
   if es:
      es=False
      print 'Errors suppressed!'
   else:
      es=True
      print 'Errors displayed!'

#-----'ckey' Option---------------------------------------------------------------

# 'Colour Key'

elif plotopt=='ckey':                           # Toggle Colour-key
   if cs:
      cs=False
      print 'Colour key suppressed!'
   else:
      cs=True
      print 'Colour key displayed!'
```

```
#-----'lines' Option-----------------------------------------------------------------

elif plotopt=='lines':                                  # Toggle Delineation
   if ls:
      ls=False
      print 'Plot Lines suppressed!'
   else:
      ls=True
      print 'Plot Lines displayed!'

#-----'info' Option------------------------------------------------------------------

elif plotopt=='info':
   dst=times[0]
   det=times[-1]
   print 'PlotDemon.py version',version
   print ''
   print nfiles,'files loaded:'
   print ''
   filn1,loca1=pan.xtrfilloc(file1)
   print 'File 1:'
   print ' Filename       = ',filn1
   print ' Location       = ',loca1
   print ' Mission        = ',mis1
   print ' Object         = ',obsd1[0]
   print ' Obs_ID         = ',obsd1[1]
   if mis1 in ['SUZAKU']:
      print ' Energy         = ',ch[1],'eV'
   else:
      print ' Channel        = ',ch[1]
   print ' Resolution     = ',str(bsz1)+'s'
   print ' BG Subtracted  = ',bsub1
   print ' No. of PCUs    = ',pcus1
   print ' Flavour        = ',flv1
   print ' FITSGenie Ver. = ',v1
   if nfiles>1:
      filn2,loca2=pan.xtrfilloc(file2)
      print ''
      print 'File 2:'
      print ' Filename       = ',filn2
      print ' Location       = ',loca2
      print ' Mission        = ',mis2
      print ' Object         = ',obsd2[0]
      print ' Obs_ID         = ',obsd2[1]
      if mis2 in ['SUZAKU']:
         print ' Energy         = ',ch[2],'eV'
      else:
         print ' Channel        = ',ch[2]
      print ' Resolution     = ',str(bsz2)+'s'
      print ' BG Subtracted  = ',bsub2
      print ' No. of PCUs    = ',pcus2
      print ' Flavour        = ',flv2
      print ' FITSGenie Ver. = ',v2
   if nfiles==3:
      filn3,loca3=pan.xtrfilloc(file3)
      print ''
      print 'File 3:'
      print ' Filename       = ',filn3
      print ' Location       = ',loca3
```

```
      print ' Mission        = ',mis3
      print ' Object         = ',obsd3[0]
      print ' Obs_ID         = ',obsd3[1]
      if mis3 in ['SUZAKU']:
         print ' Energy         = ',ch[3],'eV'
      else:
         print ' Channel        = ',ch[3]
      print ' BG Subtracted  = ',bsub3
      print ' Resolution     = ',str(bsz3)+'s'
      print ' No. of PCUs    = ',pcus3
      print ' Flavour        = ',flv3
      print ' FITSGenie Ver. = ',v3
   print ''
   print 'Other Info:'
   print ' Global Flavour = ',flavour
   print ' Obs. Starttime = ',str(tst1)+'s (0.0s)'
                                                 # The start of the observation
   print ' Obs. Endtime   = ',str(oet+tst1)+'s ('+str(oet)+'s)'
                                                 # The end of the observation
   print ' Data Starttime = ',str(dst+tst1)+'s ('+str(dst)+'s)'
                                                 # The start of the data set (i.e.
                                                 # after GTI considerations and
                                                 # clipping)
   print ' Data Endtime   = ',str(det+tst1)+'s ('+str(det)+'s)'
                                                 # The end of the data set
   print ' Bin-size       = ',str(binning)+'s'
   print ' Background     = ',str(bg)+'cts/s/PCU'
   print ' Folded         = ',folded
   if folded:
      print ' Fold Period    = ',period
   print ' Errorbars      = ',es
   print ' Delineated     = ',ls
   print ' Colour-coded   = ',cs

#-----'reflav' Option-----------------------------------------------------------------

elif plotopt=='reflav':
   print 'Please give a new flavour.'
   try:
      nflavour=raw_input('Flavour: ')
      assert nflavour!=''
      flavour=nflavour
      if flavour==='':
         qflav=''
      else:
         qflav=' "'+flavour+'"'
      print 'Flavour set to "'+flavour+'"'
   except:
      print 'Invalid flavour!  Flavour remains "'+flavour+'"'

#-----'help' Option-------------------------------------------------------------------

elif plotopt in ['help','?']:                    # Display instructions
   print 'Instructions:'
   print ''
   give_inst()                                   # Re-call the instructions list,
                                                 # defined as the get_inst()
                                                 # function in initialisation
```

```
   #-----'quit' Option-----------------------------------------------------------------

   elif plotopt not in ['quit','exit']:                # Invalid command if none of the
                                                       # if statements triggered and no
                                                       # 'q' given
      print 'Invalid command!'
   if plotopt not in ['quit','exit']:
      print ''
      print ' --------------------'

#-----Exiting Interactive Mode-------------------------------------------------------

print ''
print 'Goodbye!'

#-----Footer------------------------------------------------------------------------

pan.signoff()
```

E.3 Spec Angel

```
#! /usr/bin/env python

# |-----------------------------------------------------------------------|
# |------------------------------SPEC ANGEL------------------------------|
# |-----------------------------------------------------------------------|

# Call as ./specangel.py FILE1 [LBINNING]

# Takes 1 RXTE FITS Event file and produces an interactive spectrogram
#
# Arguments:
#
#  FILE1
#   The absolute path to the file to be used.
#
#  [LBINNING]
#   Optional- the logarithmic binning factor 'x'; frequency data will be binned into
#   bins which have their lefthand edges defined by the formula 10**(ix) for integer i.
#

#-----User-set Parameters-----------------------------------------------------------

logfreqres_default=0.005                            # The best resolution, in log10
                                                    # space, in which the data will
                                                    # be analysed
version=4.2                                         # The version of SpecAngel

#-----Welcoming Header--------------------------------------------------------------

print ''
print '-------Running Spec Angel: J.M.Court, 2015------'
print ''

#-----Importing Modules-------------------------------------------------------------

try:
```

```
   import sys,os
   import pylab as pl
   import pan_lib as pan
   from astropy.io import fits
   from numpy import array, arange, mean, meshgrid, transpose, zeros
   from numpy import exp, histogram, linspace, log10, nonzero, sqrt
   from numpy import append as npappend              # Importing numpy append as
                                                     # npappend to avoid confusion
                                                     # with in-built append function
   from numpy import min as npmin
   from numpy import max as npmax
   from numpy import sum as npsum
   from scipy import delete
   from scipy.fftpack import fft
   from scipy.stats import spearmanr
   from scipy.optimize import brentq
except ImportError:
   print 'Modules missing!  Aborting!'
   print ''
   print '-----------------------------------------------'
   print ''
   exit()

#-----Checking Validity of Arguments, Fetching File-----------------------------------

args=sys.argv
pan.argcheck(args,2)                                 # Must give at least 2 args
                                                     # (Filename and the function
                                                     # call)
filename=args[1]                                     # Fetch file name from arguments
pan.filenamecheck(filename,'speca')

#-----Extracting data from file-------------------------------------------------------

print 'Opening '+str(filename)                       # Use SpecaLd from pan_lib to
                                                     # load data from file
loadmatrix,good,rates,pk_rates,tr_rates,ph_cts,bg,binsize,four_res,bg_est,load_flavour,
          cs,mis,obsid,wtype,slide,binfac,v=pan.specald(filename)
flavour=load_flavour
if flavour=='':
   q_flav=''
else:
   q_flav=' "'+flavour+'"'

#-----Initially normalising data------------------------------------------------------

print ''
print 'Normalizing and binning...'
if len(args)>2:                                      # Check for logfreqres input,
                                                     # else request one
   try:
      logfreqres=float(args[2])
      assert logfreqres>0
   except:
      logfreqres=logfreqres_default
else:
   try:
      logfreqres=float(raw_input('Logarithmic binning factor: '))
   except:
```

```
      logfreqres=logfreqres_default
if len(args)>3:                                          # Check for normalization input,
                                                         # else request one
   try:
      norm=str(args[3])
   except:
      norm='nupnu'
else:
   try:
      norm=str(raw_input('Input normalisation [leahy, rms, nupnu]: '))
   except:
      norm='nupnu'
numstep=len(loadmatrix)
nleahy=float(sum(good))
lspec=npsum(loadmatrix,axis=0)/nleahy                   # Create the average Leahy
                                                        # spectrum
const=pan.lhconst(lspec)                                # Calculate the normalisation of
                                                        # noise
def constmi(k):                                         # Define nuP(nu) noise average as
                                                        # a function of  Leahy constant.
   nlspec=pan.lh2rms(lspec,mean(rates),bg,k)
   nlrang=arange(len(nlspec))
   nlrang=nlrang[int(4*len(nlspec)/5.0):]
   nlspec=nlspec[int(4*len(nlspec)/5.0):]
   return (mean(nlrang*nlspec))
const=(brentq(constmi,const-0.1,const+0.1))             # Minimise the above function to
                                                        # improve the constant
datres=int(four_res/binsize)
tfl=linspace(0.0, (1.0/2.0)*datres/float(four_res), (datres/2)+1)
                                                        # Create linearly spaced freq.
                                                        # domain up to the Nyquist freq.
                                                        # 1/2 (N/T)
tfl=tfl[:-1]
nulldat=zeros((datres/2))                               # Create null data with the same
                                                        # number of points as tfl
tf,null,null=pan.lbinify(tfl[1:],nulldat[1:],nulldat[1:],logfreqres)
                                                        # Fetch new array of bins to be
                                                        # output after lbinning
del null
print ''
def lbin(logfreqres,pow_norm='nupnu',prt=False):        # Defining a log-binning function
                                                        # that just depends on bin
                                                        # resolution and normalisation
   if True:                                             # Instructions with a list of
                                                        # possible normalisations
      if pow_norm not in ['rms','nupnu','leahy']:
         print 'Unknown normalisation selected!'
         print ''
         print 'Available normalisations are:'
         print '* "leahy" for Leahy-normalised power'
         print '* "rms" for RMS-normalised power'
         print '* "nupnu" for RMS-normalised power multiplied by frequency'
         print ''
         print 'Using "nupnu" normalisation:'
         pow_norm='nupnu'
      else:
         print 'Using "'+pow_norm+'" normalisation:'
   errgr=[]                                             # Set up matrix of errors
   fourgr=[]
```

```
    for i in range(numstep):
        tsfdata=loadmatrix[i]                          # Load a row of data
        errs=pan.lh2rms(tsfdata,rates[i],bg,0)         # Errors of a Leahy spectrum =
                                                       # the Leahy spectrum
        if pow_norm in ['rms','nupnu']:
            if pow_norm=='nupnu':
                sconst=const                           # For nuP(nu) normalisation, the
                                                       # Leahy constant will need
                                                       # subtracting
            else:
                sconst=0                               # In RMS norm, it can stay
            tsfdata=pan.lh2rms(tsfdata,rates[i],bg,sconst)
                                                       # Convert to RMS-normalised data
                                                       # using the LH2RMS function from
                                                       # pan_lib
            if pow_norm=='nupnu':
                tsfdata=tsfdata*tfl                    # Multiply by frequency if nupnu
                                                       # normalisation requested
                errs=errs*tfl
        tf,fours,errs=pan.lbinify(tfl[1:],tsfdata[1:],errs[1:],logfreqres)
                                                       # Logarithmically bin the data
                                                       # using lbinify from pan_lib
        fourgr.append(fours)                           # Populate the data matrix
        errgr.append(abs(errs))                        # Populate the error matrix
        prog=i+1
        if prt and ((prog % 5)==0 or prog==numstep):
            print str(prog)+'/'+str(numstep)+' series re-binned...'
                                                       # Display progress every 5 series
    fourgr=transpose(fourgr)                           # Flip the matrices (makes them
                                                       # easier to plot the correct way
                                                       # round in spectrogram)
    errgr=transpose(errgr)
    return fourgr,errgr,pow_norm
fourgr,errgr,norm=lbin(logfreqres,prt=True,pow_norm=norm)
print ''
print 'Preparing spectrogram...'
deftitle='Spectrogram'+q_flav                          # Define default title for
                                                       # spectrogram
if norm=='leahy':                                      # Define default key label for
                                                       # spectrogram
    defzlabl='Leahy-Normalised Power (Hz^-1)'
elif norm=='rms':
    defzlabl='RMS Normalised Power (Hz^-1)'
else:
    defzlabl='Frequency x RMS Normalised Power'
fourgrm=fourgr                                         # Storing a copy of the matrix in
                                                       # memory so it can be reset
errgrm=errgr
td=arange(0,(numstep+1)*slide,slide)                   # Creating the time domain as an
                                                       # array
tdg, tfg = meshgrid(td, tf)                            # Making a grid from the time and
                                                       # frequency domains
specopt=''                                             # Force spectrogram manipulation
                                                       # mode to trigger
speclog=False                                          # Indicate that the spectrogram
                                                       # is not initially logarithmic
stitle=deftitle                                        # Give an initial title
rtlabl=defzlabl                                        # Give an initial key label
tmdbin=four_res                                        # Initial time binning
```

```
frqbin=(tf[-1]-tf[0])/(len(tf)-1)                    # Initial freq binning
tdgd=tdg                                             # Saving default grid [Time
                                                     # Domain Grid- Default]
tfgd=tfg
tdlm=td                                              # Saving 1D arrays to re-form
                                                     # grids [time-domain linear,
                                                     # modifiable]
tflm=tf
ogood=good                                           # Save copy of the 'good' list
fudge=npmin(abs(fourgr[nonzero(fourgr)]))            # Obtain smallest nonzero value
                                                     # in array to add on when using
                                                     # logarithm to prevent log(0)
print 'Done!'
print ''

#-----Setting up Spectrogram Environment-----------------------------------------------

es=True                                              # Start with errors on by default
saveplots=False
show_block=False
def spectrogram(td,tfc,fourgr,zlabel=defzlabl,title=deftitle):
                                                     # Defining the creation of the
                                                     # spectrogram plot 's' as a
                                                     # function for clarity later
   pl.close('Spectrogram')                           # Close any previous spectrograms
                                                     # that may be open
   fg=pl.figure('Spectrogram')
   ax=fg.add_subplot(1,1,1)
   if speclog:
      pl.pcolor(td,tfc,(fourgr))
   else:
      pl.pcolor(td,tfc,fourgr,vmin=sgfloor,vmax=sgceil) # Plot spectrogram
   cbar=pl.colorbar()                                # Create colourbar key
   cbar.set_label(zlabel)
   pl.xlabel('Time(s)')
   pl.ylabel('Frequency(Hz)')
   pl.title(title)
   ax.set_yscale('log')                              # Make the freq-axis logarithmic
                                                     # too
   pl.ylim(tfc[0,0],tfc[-1,-1])                      # Resize axes to fit the
                                                     # spectrogram
   pl.xlim(td[0,0],td[-1,-1])
   plot_save(saveplots,show_block)
sxlab='Frequency (Hz)'
sylab=defzlabl
szlab=defzlabl                                       # Give an initial key label,
                                                     # storing second copy
def give_inst():                                     # Define printing this list of
                                                     # instructions as a function
   print 'COMMANDS: Enter a command to manipulate data.'
   print ''
   print 'DATA:'
   print '* "rebin" to reset the data and load it with a different normalisation and '+
         ' binning.'
   print '* "clip" to clip the range of data.'
   print '* "reset" to reset data.'
   print ''
   print 'SPECTROGRAM:'
   print '* "sg plot" to plot the spectrogram currently being worked on.'
```

```
   print '* "sg floor" to set a minimum value for the spectrogram'+"'"+'s z-axis colo'+
         'ur key.'
   print '* "sg ceil" to set a maximum value for the spectrogram'+"'"+'s z-axis colou'+
         'r key.'
   print '* "sg auto" to automatically set colour floor and ceiling.'
   print '* "sg log" to toggle logarithmic spectrogram plotting.'
   print ''
   print 'POWER SPECTRA:'
   print '* "aspec" to plot the average spectrum and return the frequency of its high'+
         'est peak.'
   print '* "gspec" to get an individual spectrum at any time and plot it.'
   print '* "peaks" to plot a graph of the frequency of the strongest oscillation aga'+
         'inst time.'
   print '* "rates" to get a simple lightcurve of the data.'
   print '* "fqflux" to plot "peaks" against "rates".'
   print ''
   print 'TOGGLE OPTIONS:'
   print '* "errors" to toggle errorbars on power spectra plots.'
   print '* "save" to save to disk any plots which would otherwise be shown.'
   print ''
   print 'OTHER COMMANDS:'
   print '* "info" to display a list of facts and figures about the current SpecAngel'+
         ' session.'
   print '* "reflav" to rewrite the flavour text used for graph titles.'
   print '* "export" to create an ASCII file of the average power density spectrum.'
   print '* "help" or "?" to display this list of instructions again.'
   print '* "quit" to Quit'

#give_inst()                                          # Print the list of instructions
print ''
print ' --------------------'
sgfloor=max(npmin(fourgrm),0)
sgceil=npmax(fourgrm)
def plot_save(saveplots,show_block):                  # Add a function to redirect all
                                                      # show calls to savefigs if
                                                      # toggled
   if saveplots:
      pl.savefig(raw_input('Save plot as: '))
      print 'Plot saved!'
   else:
      pl.show(block=show_block)

#-----Entering Interactive Mode-------------------------------------------------------

while specopt not in ['quit','exit']:                 # If the previous command give
                                                      # was not quit, continue
   print ''
   specopt=raw_input('Give command [? for help]: ').lower()   # Fetch command from user
   print ''

   #-----Hidden 'stick' option-----------------------------------------------------

   if specopt=='stick':                               # For use when scripting with
                                                      # Specangel.  If turned on, this
                                                      # causes all plots to block when
                                                      # shown.
      show_block=not show_block
      if show_block:
         print 'Sticky Plots on!'
```

```
   else:
      print 'Sticky Plots off!'

#-----'save' option-----------------------------------------------------------------

elif specopt=='save':                                # Causes a plot to be saved when
                                                     # it would otherwise have been
                                                     # shown
   saveplots=not saveplots
   if saveplots:
      print 'Plot saving on!'
   else:
      print 'Plot saving off!'

#-----'rebin' Option----------------------------------------------------------------

elif specopt=='rebin':                               # Rebinning data
   print 'Data currently binned in '+str(tmdbin)+'s and 10^'+str(logfreqres)+'Hz b'+
         'ins.'
   speclog=False                                   # Indicate that the spectrogram is
                                                   # not logarithmic
   stitle=deftitle                                 # Restore initial title
   tmdbin=four_res                                 # Initial time binning
   frqbin=(tf[-1]-tf[1])/(len(tf)-2)               # Initial freq binning
   fourgrm=fourgr                                  # Reload original, unmodified data
   errgrm=errgr
   tdgd=tdg                                        # Reloading default grid
   tfgd=tfg
   tdlm=td                                         # Loading 1D arrays to re-form
                                                   # grids [time-domain linear,
                                                   # modifiable]
   tflm=tf
   good=ogood
   norm=raw_input('Select normalisation [leahy, rms, nupnu]: ')
   try:
      tbinmult=int(raw_input('Input time-domain binning factor: '))
      if tbinmult<1: tbinmult=1                     # Prevent bin-sizes smaller than
                                                    # current bin-size
      if tbinmult>len(fourgrm)/2.0:tbinmult=1       # Forces there to be at least 2
                                                    # bins
   except:
      tbinmult=1                                    # Cancel binning if a non-number
                                                    # is entered
      print 'Invalid time binning!'
   tmdbin=four_res*tbinmult                         # Recover current binning
   try:
      newfbin=float(raw_input('Input new freq-domain exponent: '))
      if newfbin>0: logfreqres=newfbin              # Prevent bin-sizes smaller than
                                                    # zero
   except:
      newfbin=0                                     # Cancel binning if a non-number
                                                    # is entered
      print 'Invalid frequency binning!'
   print ''
   print 'Re-binning...'
   tflm,null,null=pan.lbinify(tfl[1:],nulldat,nulldat,logfreqres)
                                                   # Fetch new array of bins to be
                                                   # output after lbinning
   del null
```

```
   fourgrm,errgrm,norm=lbin(logfreqres,prt=True,pow_norm=norm)
                                                # Re log-bin data
   if tbinmult!=1:                              # Cancel binning if new bin is
                                                # not greater than old bin
      fourgrm,errgrm,tdlm,good=pan.mxrebin(fourgrm,errgrm,tdlm,good,tbinmult)
   tdgd,tfgd=meshgrid(tdlm,tflm)                # Recreate grid from rescaled
                                                # axes
   print ''
   print 'Data rebinned by '+str(tmdbin)+'s, [10^'+str(logfreqres)+'n]Hz.'
   print str(int(sum(good)))+'/'+str(len(good))+' power spectra are good'
   if norm=='leahy':
      sylab='Leahy-Normalised Power (Hz^-1)'
   elif norm=='rms':
      sylab='RMS Normalised Power (Hz^-1)'
   else:
      sylab='Frequency x RMS Normalised Power'
   defzlabl=sylab                               # Restore root z label for
                                                # spectrogram
   rtlabl=defzlabl                              # Restore initial key label
   szlab=defzlabl                               # Restore initial key label,
                                                # storing second copy
   sgfloor=max(npmin(fourgrm),0)                # Reset spectrogram colour floor
                                                # & ceil
   sgceil=npmax(fourgrm)

#-----'sg plot' Option-----------------------------------------------------------

# 'Spectrogram'

elif specopt=='sg plot':                        # Plotting data
   proce=True                                   # Assume plot will be made
   npl=len(fourgrm[0,:])*len(fourgrm[:,0])      # Check how large this plot'll be
   if npl>1000000:                              # Check for very large plots, ask
                                                # user whether to proceed
       try:
          print "LARGE DATA WARNING! Plot will contain "+"{:,}".format(npl)+" elem"+
                "ents."
          proc=raw_input("Proceed? [y/n]: ")
          if proc!='y': proce=False             # Cancel plot is user doesn't
                                                # explicity say 'y'
       except:
          proce=False
   if proce==True:                              # If all is ok...
      print 'Plotting...'
      spectrogram(tdgd,tfgd,fourgrm,szlab,stitle)  # Plot spectrogram

#-----'sg floor' Option----------------------------------------------------------

# 'Colour Floor'

elif specopt=='sg floor':                       # Setting floor of spectrogram
                                                # colour scale:
   sgfloor=raw_input('Input spectrogram colour floor: ')
                                                # Ask user to input floor
   try:
      sgfloor=float(sgfloor)                    # Check floor is a number
      print 'Colour floor set!'
   except:
      sgfloor=max(npmin(fourgrm),0)
```

```
      print 'Invalid floor!'

#-----'sg ceil' Option-----------------------------------------------------------------

# 'Colour Ceiling'

elif specopt=='sg ceil':                          # Setting ceiling of spectrogram
                                                  # colour scale:
   sgceil=raw_input('Input spectrogram colour ceiling: ')
                                                  # Ask user to input floor
   try:
      sgceil=float(sgceil)                        # Check ceil is a number
      print 'Colour ceiling set!'
   except:
      sgceil=npmax(fourgrm)
      print 'Invalid ceiling!'

#-----'sg auto' Option-----------------------------------------------------------------

# 'Auto Recolour'

elif specopt=='sg auto':
   sgfloor=max(npmin(fourgrm),0)                  # Reset spectrogram colour floor
                                                  # & ceil
   sgceil=npmax(fourgrm)
   print 'Colour floor and ceiling automatically set!'

#-----'sg log' Option------------------------------------------------------------------

# 'Logarithm'

elif specopt=='sg log':                           # Taking or undoing log of data
   if speclog:                                    # If the spectrogram was already
                                                  # logged, undo this with exp
      print 'Exponentiating spectrogram...'
      stitle=deftitle                             # Reset title to default
      rtlabl=rtlabl[4:]                           # Remove 'log ' from the start of
                                                  # both saved z-labels
      szlab=szlab[4:]
      fourgrm=10**(fourgrm)-fudge                 # Exponentiate every element of
                                                  # every power spectrum
      speclog=False                               # Indicate that spectrogram is no
                                                  # longer logged
      print 'Done!'
   else:
      print 'Taking logarithm of spectrogram...'
      stitle='Log '+deftitle                      # Add 'log ' to start of titles
                                                  # and labels
      rtlabl='Log '+rtlabl
      szlab='Log '+szlab
      fourgrm=log10(abs(fourgrm)+fudge)           # Take the log of every element
                                                  # of every power spectrum
      speclog=True                                # Indicate that spectrum is
                                                  # logged
      print 'Done!'

#-----'sg' Catch-All Help Message-------------------------------------------------------

elif specopt[:2]=='sg':
```

```
   print 'SPECTROGRAM COMMANDS:'
   print '* "sg plot" to plot the spectrogram currently being worked on.'
   print '* "sg floor" to set a minimum value for the spectrogram'+"'"+'s z-axis c'+
         'olour key.'
   print '* "sg ceil" to set a maximum value for the spectrogram'+"'"+'s z-axis co'+
         'lour key.'
   print '* "sg auto" to automatically set colour floor and ceiling.'
   print '* "sg log" to toggle logarithmic spectrogram plotting.'

#-----'clip' Option-----------------------------------------------------------------

elif specopt=='clip':                                # Clipping data
   print 'Clipping data'
   print ''
   print 'Time range is '+str(tdlm[0])+'s - '+str(tdlm[-2]+four_res)+'s'
   print 'Freq range is '+str(tflm[0])+'Hz- '+str(tflm[-1]+four_res)+'Hz'
   print ''
   print 'Please choose new range of data:'
   mint,maxt,null=pan.srinr(tdlm,tmdbin,'time')     # Fetch new time domain endpoints
                                                    # using srinr function from
                                                    # pan_lib
   minf,maxf,null=pan.srinr(tflm,frqbin,'freq')     # Fetch new freq domain endpoints
                                                    # using srinr function from
                                                    # pan_lib
   print 'Clipping...'
   tdlm=tdlm[mint:maxt]                             # Clip the time-domain array
   tflm=tflm[minf:maxf]                             # Clip the freq-domain array
   fourgrm=fourgrm[minf:maxf,mint:maxt]             # Clip the spectrogram data
   errgrm=errgrm[minf:maxf,mint:maxt]
   tdgd,tfgd=meshgrid(tdlm,tflm)                    # Recreate grid from rescaled
                                                    # axes
   print 'Data clipped!'

#-----'reset' Option----------------------------------------------------------------

elif specopt=='reset':                               # Resetting data
   print 'Resetting spectrogram...'
   speclog=False                                    # Indicate that the spectrogram
                                                    # is not logarithmic
   stitle=deftitle                                  # Restore initial title
   rtlabl=defzlabl                                  # Restore initial key label
   szlab=defzlabl                                   # Restore initial key label,
                                                    # storing second copy
   tmdbin=four_res                                  # Initial time binning
   frqbin=(tf[-1]-tf[1])/(len(tf)-2)                # Initial freq binning
   fourgrm=fourgr                                   # Reload original, unmodified
                                                    # data
   errgrm=errgr
   tdgd=tdg                                         # Reloading default grid
   tfgd=tfg
   tdlm=td                                          # Loading 1D arrays to re-form
                                                    # grids [time-domain linear,
                                                    # modifiable]
   tflm=tf
   good=ogood                                       # Reload original 'good' list
   print 'Spectrogram reset!'

#-----'aspec' Option----------------------------------------------------------------
```

```
# 'Average Spec'

elif specopt=='aspec':                          # Find the time-averaged spectrum
   if slide!=four_res:
      print "Warning!  Data taken with sliding window: data points not independent!"
      print ''
   print "Fetching time-averaged power spectrum..."
   spec=npsum(fourgrm, axis=1)/sum(good)        # Sum all spectra in the matrix
                                                # and divide by the number of
                                                # good columns
   err=sqrt(npsum( array(errgrm)**2, axis=1))/sum(good)
   ttl='Average power density spectrum'+q_flav
   pan.slplot(tflm,spec,err,sxlab,sylab,ttl,'spc',errors=es,typ='log')
                                                # SLPlot from the pan_lib plots
                                                # data on standard and log-log
                                                # axes
   plot_save(saveplots,show_block)
   scerr=spec-(err**0.5)
   print 'Maximum power found at '+str(tflm[scerr.argmax()])+'Hz!'
                                                # Suggest a peak location
   print '  (Period of '+str(1.0/tflm[scerr.argmax()])+'s)'

#-----'gspec' Option-------------------------------------------------------------

# 'Get Spec'

elif specopt=='gspec':                          # Find the power spectrum at a
                                                # specific point in time
   if slide!=four_res:
      print "Warning!  Data taken with sliding window: data points not independent!"
      print ''
   print 'Getting a spectrum at a time (since start of observation) of your choice.'
   print ''
   while True:
      try:
         specid=float(raw_input('Enter time: '))   # Fetch raw time suggestion from
                                                # user
         break
      except:
         print 'Invalid time!'
   specid=int((specid-tdlm[0])/tmdbin)          # Work out which time bin this
                                                # would correlate to
   if 0<=specid<len(fourgrm[0,:])-1:            # Check that this time bin
                                                # actually exists in the matrix
      if good[specid]:
         print "Fetching power spectrum at "+str(specid*tmdbin)+"s: "

         gsp=fourgrm[:,specid]                  # Extract the lightcurve from
                                                # this bin
         ger=errgrm[:,specid]
         ttl='Power density spectrum "'+flavour+'" at +'+str(specid*tmdbin)+'s'
         pan.slplot(tflm,gsp,ger,sxlab,sylab,ttl,'spc',typ='log',errors=es)
         plot_save(saveplots,show_block)
         scerr=spec-(err**0.5)
         print 'Maximum power found at '+str(tflm[scerr.argmax()])+'Hz!'
                                                # Suggest a peak location
      else:
         print 'Time not in GTI!'
   else:
```

```
        print 'Time not in range!'

#-----'peaks' Option-----------------------------------------------------------------

elif specopt=='peaks':
    peaks=[]
    for i in range(len(fourgrm[1])):
        row=fourgrm[:,i]
        peaks.append(tflm[list(row).index(max(row))])
    pl.close('pk')
    pl.figure('pk')
    pl.semilogy(array(tdlm[:-1])[good],array(peaks)[good],'-ok')
    pl.xlabel('Time (s)')
    pl.ylabel('Frequency (Hz)')
    pl.title('Peak Frequency/Time Plot'+q_flav)
    plot_save(saveplots,show_block)

#-----'rates' Option-----------------------------------------------------------------

elif specopt=='rates':
    print 'Average rate of',str(mean(rates[ogood]))+'c/s.'
    print str(ph_cts),'total counts.'
    print ''
    datasel=raw_input('Select Rates to plot [ave/peak/trough]: ')
    brates={}
    titles={}
    titles['ave']='Flux (photons/s/PCU)'
    titles['peak']='Peak '+str(binfac*binsize)+'s Flux (photons/s/PCU)'
    titles['trough']='Trough '+str(binfac*binsize)+'s Flux (photons/s/PCU)'
    if datasel not in ('ave','peak','trough'):
        print 'Invalid selection!  Using Average flux.'
        datasel='ave'
    brates['ave']=rates
    brates['peak']=pk_rates
    brates['trough']=tr_rates
    pl.close('lc')
    pl.figure('lc')
    pl.plot(td[:-1][ogood],brates[datasel][ogood],'-ok')
    pl.xlabel('Time (s)')
    pl.ylabel('Flux (photons/s/PCU)')
    pl.title('Lightcurve'+q_flav)
    plot_save(saveplots,show_block)

#-----'fqflux' Option----------------------------------------------------------------

elif specopt=='fqflux':
    peaks=[]
    for i in range(len(fourgrm[1])):
        row=fourgrm[:,i]
        peaks.append(tflm[list(row).index(max(row))])
    brates={}
    titles={}
    titles['ave']='Flux (photons/s/PCU)'
    titles['peak']='Peak '+str(binfac*binsize)+'s Flux (photons/s/PCU)'
    titles['trough']='Trough '+str(binfac*binsize)+'s Flux (photons/s/PCU)'
    brates['ave']=pan.vcrebin(rates,len(rates)/len(peaks))
    brates['peak']=pan.vcrebin(pk_rates,len(pk_rates)/len(peaks))
    brates['trough']=pan.vcrebin(tr_rates,len(tr_rates)/len(peaks))
    datasel=raw_input('Select Rates to plot against [ave/peak/trough]: ')
```

```
   if datasel not in ('ave','peak','trough'):
      print 'Invalid selection!  Using Average flux.'
      datasel='ave'
   print 'Spearman Rank Coefficient: ',spearmanr(array(peaks)[good],
                                                 brates[datasel][good])[1]
   pl.close('pr')
   pl.figure('pr')
   pl.semilogx(array(peaks)[good],brates[datasel][good],'ok')
   pl.ylabel(titles[datasel])
   pl.xlabel('Frequency (Hz)')
   pl.title('Flux/Peak Frequency Plot'+q_flav)
   plot_save(saveplots,show_block)

#-----'errors' Option-----------------------------------------------------------------

elif specopt in ['error', 'errors']:                # Toggle Errors
   if es:
      es=False
      print 'Errors suppressed!'
   else:
      es=True
      print 'Errors displayed!'

#-----'info' Option-------------------------------------------------------------------

elif specopt=='info':
   print 'SpecAngel.py version',version
   print ''
   print '1 file loaded:'
   print ''
   local_name,file_path=pan.xtrfilloc(filename)
   print 'File 1:'
   print ' Filename       = ',local_name
   print ' Location       = ',file_path
   print ' Mission        = ',mis
   print ' Object         = ',obsid[0]
   print ' Obs_ID         = ',obsid[1]
   if mis in ['SUZAKU']:
      print ' Energy         = ',cs,'eV'
   else:
      print ' Channel        = ',cs
   print ' Resolution     = ',str(binsize)+'s'
   print ' Flavour        = ',load_flavour
   print ' FITSGenie Ver. = ',v
   print ''
   print 'Windowing:'
   print ' Shape          = ',wtype
   print ' Sliding        = ',slide!=four_res
   print ' Length         = ',str(four_res)+'s'
   if slide!=four_res:
      print ' Separation     = ',str(slide)+'s'
   print ''
   print 'Normalisation:'
   print ' Normalisation  = ',norm
   print ' Leahy constant = ',const
   print ''
   print 'Other Info:'
   print ' Global Flavour = ',flavour
   print ' Obs length     = ',str(four_res*numstep)+'s'
```

```
   print ' Time. Bin-size = ',str(tmdbin)+'s'
   print ' Freq. Bin-size = ',logfreqres
   print ' Num. Time Bins = ',str(int(len(good)))
   print ' Good Time Bins = ',str(int(sum(good)))
   print ' Avg. Rates     = ',str(mean(rates[ogood]))
   print ' Total photons  = ',ph_cts
   print ' Background     = ',str(bg_est)+'cts/s/PCU'
   print ' Errorbars      = ',es

#-----'reflav' Option-----------------------------------------------------------------

elif specopt=='reflav':
   print 'Please give a new flavour.'
   try:
      nflavour=raw_input('Flavour: ')
      assert nflavour!=''
      flavour=nflavour
      if flavour==='':
         q_flav=''
      else:
         q_flav=' "'+flavour+'"'
      print 'Flavour set to "'+flavour+'"'
   except:
      print 'Invalid flavour!  Flavour remains "'+flavour+'"'

#-----'export' Option-----------------------------------------------------------------

elif specopt=='export':
   aflname=raw_input('Filename: ')
   try:
      assert len(aflname)>0
      fle=open(aflname,'w')
      spec=npsum(fourgrm, axis=1)/sum(good)        # Sum all spectra in the matrix
                                                   # and divide by the number of
                                                   # good columns
      err=sqrt(npsum( array(errgrm)**2, axis=1))/sum(good)
      for i in range(len(tflm)):
         a=[str(tflm[i]),' ',str(spec[i]),' ',str(err[i]),'\n']
                                                   # Frequency, Power, Error
         fle.writelines(a)
      fle.close()
      print 'ASCII Leahy-normalised spectrum saved to',aflname+'!'
   except:
      print 'Invalid filename!'

#-----'help' Option-------------------------------------------------------------------

elif specopt in ['help','?']:                      # Display instructions
   print 'Instructions:'
   print ''
   give_inst()                                     # Re-call the instructions list,
                                                   # defined as the get_inst()
                                                   # function in initialisation

#-----'quit' Option-------------------------------------------------------------------

elif specopt not in ['quit','exit']:               # Invalid command if none of the
                                                   # if statements triggered and no
                                                   # 'q' given
```

```
      print 'Invalid command!'

   if specopt not in ['quit','exit']:
      print ''
      print ' --------------------'

#-----Exiting Interactive Mode-------------------------------------------------------

print ''
print 'Goodbye!'

#-----Footer-------------------------------------------------------------------------

pan.signoff()
```

E.4 Back Hydra

```
#! /usr/bin/env python

# |-----------------------------------------------------------------------|
# |-----------------------------BACK HYDRA-------------------------------|
# |-----------------------------------------------------------------------|

# Call as ./bckghydra.py DATA_FILE BACK_FILE SAVE_FILE

# Takes a .plotd file and a background file created with PCABACKEST and returns
#
# Arguments:
#
#  DATA_FILE
#   The absolute path to the file to be used as data.
#
#  BACK_FILE
#   The file to be used as background; does not need to be the same binning as File 1.
#   suggest using pcabackest from FTOOLS to produce this file.
#   FTOOLS can be found at http://heasarc.gsfc.nasa.gov/ftools/
#
#  SAVE_FILE
#   The location to save the resultant background-subtracted file
#

#-----Welcoming Header---------------------------------------------------------------

print ''
print '-------Running BackHydra: J.M.Court, 2015-------'
print ''

#-----Importing Modules--------------------------------------------------------------

try:
   import sys
   import pan_lib as pan
   from astropy.io import fits
   import numpy as np
   import pylab as pl
except ImportError:
   print 'Modules missing!  Aborting!'
```

```
   print ''
   print '-----------------------------------------------'
   print ''
   exit()

#-----Checking Validity of Filenames-----------------------------------------------

args=sys.argv
pan.argcheck(args,4)                                # Must give at least 3 args (Both
                                                    # filenames function call)
data_filename=args[1]                               # Fetch datafile name from
                                                    # arguments
back_filename=args[2]                               # Fetch background file name from
                                                    # arguments
save_filename=args[3]
print 'Loading Data...'
datafile_packed=pan.plotdld(data_filename)          # Load datafile
print 'Loading Background...'
backfile_packed=fits.open(back_filename)

#-----Unpack Data-------------------------------------------------------------------

# Collect all data from the data file
b_sub='True'
data_x        = datafile_packed[0]
data_f        = datafile_packed[1]
data_fe       = datafile_packed[2]
data_t_start  = datafile_packed[3]
data_bin_size = datafile_packed[4]
data_gti      = datafile_packed[5]
data_maxpcus  = datafile_packed[6]
data_flavour  = datafile_packed[10]
data_channels = datafile_packed[11]
data_mission  = datafile_packed[12]
data_obs_data = datafile_packed[13]
data_fitsg_v  = datafile_packed[14]
data_obsid=data_obs_data[1]
# Collect only relevant data from the background file
backfile_data=backfile_packed[1].data
back_mission=backfile_packed[1].header['TELESCOP']
if back_mission == 'XTE' :
   try:
      import xtepan_lib as inst                     # Import XTE extraction functions
   except:
      print 'XTE PANTHEON Library not found!  Aborting!'
      pan.signoff()
      exit()
elif back_mission == 'SUZAKU':
   try:
      import szkpan_lib as inst                     # Import SUZAKU extraction
                                                    # functions
   except:
      print 'Suzaku PANTHEON Library not found!  Aborting!'
      pan.signoff()
      exit()
low_chan,high_chan=data_channels.split('-')
back_x,back_f,back_fe = inst.getbg(backfile_data,int(low_chan),int(high_chan))
back_x=back_x[back_f>0]
back_fe=back_fe[back_f>0]
```

```
back_f=back_f[back_f>0]
back_t_start  = back_x[0]
back_bin_size = inst.getbin(backfile_packed,None)

#-----Check Background and Data files are compatible----------------------------------

same_mission  = ( data_mission == back_mission )      # Check the mission names match

if not same_mission:                                  # Abort if missions differ
   print 'Files are from different missions!'
   print 'Aborting!'
   pan.signoff()
   exit()

#-----Shift Arrays---------------------------------------------------------------------

back_x=pan.tnorm(back_x,back_bin_size)                # Force background x array to
                                                      # start at 0
shifted_data_x=data_x+data_t_start-back_t_start       # Create shifted data axis to
                                                      # align with a background
                                                      # starting at 0s
shifted_back_x=back_x+back_t_start-data_t_start       # Create shifted background axis
                                                      # to align with data starting at
                                                      # 0s
if back_x[0]>shifted_data_x[-1] or shifted_data_x[0]>back_x[-1]:
                                                      # Abort if the timescales don't
                                                      # overlap
   print 'WARNING! Files times do not overlap!'
   print ''
   print 'Estimating constant background.'
   print ''
   b_sub='Estimate'
   dump_file=open('backhydra_log.txt','w')
   dump_file.write('File and background times did not overlap!')
   dump_file.close()

#-----Define Background Subtraction----------------------------------------------------

def backgr(i):                                        # Function that returns the
                                                      # appropriate background counts
                                                      # at each point in the datafile
   timestamp=shifted_data_x[i]                        # Collect the timestamp of the
                                                      # ith data element
   st_i=int(timestamp/back_bin_size)                  # Collect the start and endpoints
                                                      # of the bg bin in which the
                                                      # timestamp falls
   ed_i=st_i+1
   if st_i<0:
      return back_f[0],back_fe[0]                     # Return startpoint background if
                                                      # sampling before bg range
   elif ed_i>=len(back_x):
      return back_f[-1],back_fe[-1]                   # Return endpoint background if
                                                      # sampling after bg range
   else:
      posit_in_bin=(timestamp % back_bin_size)/back_bin_size
                                                      # Work out where in the bin the
                                                      # timestamp falls
      f_est  = back_f[st_i]+posit_in_bin*(back_f[ed_i]-back_f[st_i])
                                                      # Linearly interpolate between
```

```
                                                        # two background points to return
                                                        # background estimate
      fe_est = back_fe[st_i]+posit_in_bin*(back_fe[ed_i]-back_fe[st_i])
                                                        # Collect error too
      return f_est,fe_est

#-----Perform Background Subtraction-----------------------------------------------------

print 'Subtracting Background...'
for i in pan.eqrange(data_x):
   back,back_e=backgr(i)
   data_f[i]-=back
   data_fe[i]=(data_fe[i]**2+back_e**2)**0.5

#-----Re-save Data-----------------------------------------------------------------------

print 'Saving...'
new_bg_est=np.mean(back_f)/data_maxpcus
bg_data=(shifted_back_x[(shifted_back_x>=data_x[0]) | (shifted_back_x<=data_x[-1])],
         back_f[(shifted_back_x>=data_x[0]) | (shifted_back_x<=data_x[-1])])
pan.plotdsv(save_filename,data_x,data_f,data_fe,data_t_start,data_bin_size,
            data_gti,data_maxpcus,new_bg_est,b_sub,bg_data,data_flavour,
            data_channels,data_mission,data_obs_data,data_fitsg_v)
print ''
print 'Background Subtracted file saved as "'+save_filename+'.plotd"!'
```

E.5 PAN Lib

```
#! /usr/bin/env python

# |-----------------------------------------------------------------------|
# |-------------------------------PAN_LIB--------------------------------|
# |-----------------------------------------------------------------------|

# A selection of useful functions which are placed here to reduce clutter in the other
#  files of

# PANTHEON.
#
# Contents:
#
# ARGCHECK   - compares the list of arguments against a value given as the minimum
#              allowed number of arguments.  If the list of arguments is too short,
#              throw a warning and kill the script.
#
#  BINIFY    - takes a x-series with its associated y-axis data and y-axis errors.
#              Rebins the data into larger linear bins with a width of the user's
#              choosing, and returns the tuple x,y,y_error.
#
#  BOOLVAL   - takes a list of Boolean values and, interpreting it as binary, returns
#              its integer value.
#
#  EQRANGE   -
#
#  EVAL_BURST-
#
#  FILENAMECHECK  - checks to see whether a proposed input file has the correct file
```

```
#                   extension.
#
#  FOLDIFY   - takes a time series with its associated y-axis data and y-axis errors.
#              Folds this data over a time period of the user's choosing, and returns
#              them as the tuple x,y,y_error.
#
#  FOLD_BURSTS - uses GET_BURSTS to obtain burst locations then interpolates to
#                populate phase information for all other points
#
#  GET_BURSTS- takes an array of data, looks for bursts and returns an array of tuples
#              containing the start and end points of these bursts.
#
#  GET_DIP   - returns the index of the lowest point between two user-defined flags in
#              a dataset.
#
#  GTIMASK   - returns a data mask when given a time series and a GTI object
#
#  LBINIFY   - takes a linearly binned x-series with associated y-axis data and y-axis
#              errors and rebins them into bins of a constant width in logx space.
#              In places where the logarithmic bins would be finer than the linear
#              bins, the linear bins are retained.
#
#  LEAHYN    - takes the raw power spectrum output from the scipy FFT algorithm and
#              normalises it using Leahy normalisation.
#
#  LH2RMS    - takes a Leahy-normalised power spectrum and converts it to an
#              (RMS/Mean)^2-normalised power spectrum.
#
#  LHCONST   - returns the normalisation of the white noise component in a Leahy-
#              normalised power spectrum with no features in the range 1.5kHz - 4kHz.
#
#
#  MXREBIN   - takes a 2-dimensional set of data and corresponding errors linearly
#              binned on the x-axis and rebins them by an integer binning factor of
#              the user's choice.
#
#  NONES     - like np.zeros, but with None.
#
#  PDCOLEX   - extracts colours from a set of 2 or 3 lightcurves
#
#  PLOTDLD   - load and unpickle a .plotd file and extract its data.
#
#  PLOTDSV   - collect a selection of data products as a library, pickle it and save
#              as a .plotd file.
#
#  RMS_N     - takes the raw power spectrum output from the scipy FFT algorithm and
#              normalises it using (RMS/Mean)^2 normalisation.
#
#  SAFE_DIV  - Divides two arrays by each other, replacing NaNs that would be caused
#              by div 0 errors with zeroes.
#
#  SIGNOFF   - prints an dividing line with some space.  That's all it does.
#
#  SINFROMCOS- calculates the sines of an array of values when also passed their co-
#              sines.  If both sines and cosines of the array are required, this method
#              is faster than calling both trig functions.
#              Also contains function COSFROMSIN.
#
#  SLPLOT    - plots an x-y line plot of two sets of data, and then below plots the
```

```
#              same data on another set of axes in log-log space.
#
#  SPECALD   - load and unpickle a .speca file and extract its data.
#
#  SPECASV   - collect a selection of data products as a library, pickle it and save
#              as a .speca file.
#
#  SRINR     - calculates whether a value given by a user is within an existant evenly
#              spaced array and, if it is, returns the index value of the closest
#              match to this value within the array.
#              Intended for validating subranges specified by user.
#
#  TNORM     - takes a list of times, and subtracts the lowest value from each entry
#              such that a new list starting with 0 is produced.  Large number
#              subtraction errors are avoided by checking that every entry is an
#              integer number of time-resolution steps from zero.
#
#  UNIQFNAME - checks if a proposed filename is currently in use and, if so, proposes
#              an alternative filename to prevent overwrite.
#
#  XTRFILLOC - takes a filepath and outputs the file name and its absolute(ish)
#              location
#

#-----Importing Modules-----------------------------------------------------------------

import os
try:
   import cPickle as pickle
except ImportError:
   import pickle
from matplotlib import pyplot as pl
import warnings
import scipy.optimize as optm
import scipy.interpolate as intp
import scipy.stats as stt
import scipy.signal as sgnl
import numpy as np
import operator as ope
try:
   import numba as nb
   gotnumba=True
except ImportError:
   print 'Warning: numba module not found!  May run slow.'
   gotnumba=False
from math import pi
from numpy import random as rn

# =========== CLASSES ==================================================================

#-----ObsData---------------------------------------------------------------------------

class obsdata(object):
   def __init__(self,filename,tcol,ycol,ecol,name=''):
      self.name=name
      xs=[]
      ys=[]
      es=[]
      f=open(filename)
```

```
        for line in f:
            l=line.split()
            try:
                t=float(l[tcol])
                y=float(l[ycol])
                e=float(l[ecol])
                xs.append(t)
                ys.append(y)
                es.append(e)
            except (IndexError, ValueError):
                pass
        f.close()
        xs=np.array(xs)
        ys=np.array(ys)
        es=np.array(es)
        assert len(xs)==len(ys)
        assert len(xs)==len(es)
        self.t=xs
        self.y=ys
        self.e=es
    def get_len(self):
        return self.t[-1]-self.t[0]
    def get_start(self):
        return self.t[0]
    def get_end(self):
        return self.t[-1]
    def plot(self):
        pl.figure()
        pl.errorbar(self.t,self.y,yerr=self.e,fmt='0.7',zorder=-1)
        pl.plot(self.t,self.y,'k')
        pl.title(self.name)
        pl.show(block=True)
    def plot_on_ax(self,ax,zorder=0,errcolor='0.7',color='k'):
        ax.errorbar(self.t,self.y,yerr=self.e,fmt=errcolor,zorder=zorder-0.5)
        ax.plot(self.t,self.y,color,zorder=zorder,label=self.name)

# =========== FUNCTIONS ================================================================

#-----Setup Conditional Wrapper to use jit when available------------------------------

class mjit(object):
    def __call__(self, f):
        if not gotnumba:
            return f
        else:
            return nb.jit(f)

#-----ArgCheck-------------------------------------------------------------------------

def argcheck(x,y):
    '''Argument Checker

    Description:
     Takes a list (of arguments passed into a script), and a value representing the
     smallest allowable number of arguments.  If the length of the list is smaller than
     the criterion, an error is returned and the script is killed.
    Inputs:
     x - LIST   : The list of arguments passed into the current working script, i.e.
                  x=sys.argv.
```

```
    y - INTEGER: The minimum allowable number of arguments.  Caution; if called from
                 bash, sys.argv returns the function call as the x[0] element, so y is
                 1 greater than the number of user inputs.
   Outputs:
    [none]

   -J.M.Court, 2014'''
   if len(x)<y:
      print 'Not enough arguments!'
      signoff()
      exit()

#-----Binify-----------------------------------------------------------------------------

@mjit()
def binify(x,y,ye,binsize):                          # Defining 'binify' subscript
   '''Binify

   Description:
    Takes a 2-dimenstional set of data which has already been evenly binned on the
    x-axis, and re-bins it into larger, evenly spaced bins on the x-axis.
   Inputs:
    x       -  LIST: The x-values of the two-dimensional data.
    y       -  LIST: The y-values of the two-dimensional data, must be the same length
                     as x.
    ye      -  LIST: The errors associated with the y-values of the two-dimensional
                     data, must be the same length as x and y.
    binsize - FLOAT: the size of the new x-axis bins in which to re-bin the data.
   Outputs:

    xb      -  LIST: The re-binned x-values of the two-dimensional data, i.e. an array
                     of the left-hand edges of the new bins.
    yb      -  LIST: The re-binned y-values of the two-dimensional data.
    yeb     -  LIST: The errors associated with the rebinned y-values of the 2D data.

   -J.M.Court, 2014'''
   binlx=binsize*np.floor(x[0]/binsize)              # Initialising 'bin lowest x',
                                                     # or the lowest x value of the
                                                     # current bin
   binct=0.0                                         # Initialising 'bin count', or
                                                     # number of values sorted into
                                                     # the current bin
   xb=[x[0]]                                         # Setting up arrays to append
                                                     # binned values into
   yb=[0]
   yeb=[0]
   for xid in range(len(x)):
      if x[xid]-binlx < binsize:                     ## If the difference between the
                                                      #  current x and bin start x is
                                                      #  less than bin width:
         binct+=1
         yb[-1]=yb[-1]+y[xid]                         #  Add y to current bin
         yeb[-1]=yeb[-1]+(ye[xid]**2)                 #  Add y error in quadrature to
                                                      #  current bin
      else:                                          ## Otherwise:
         binlx+=binsize                               #  Create new bin with minimum x
                                                      #  equal to current x
         xb.append(binlx)                             #  Append next x value into new
                                                      #  array element
```

```
        yb[-1]=yb[-1]/binct                          #  Divide y in previous bin by
                                                     #  bincount to get the average
        yeb[-1]=(np.sqrt(yeb[-1]))/binct             #  Sqrt error and divide by bin
                                                     #  count
        yb.append(y[xid])                            #  Append current y value into
                                                     #  new array element
        yeb.append((ye[xid])**2)
        binct=1                                      #  Reset bin count to 1
    yb[-1]=yb[-1]/binct                             ## Clean up final bin
    yeb[-1]=(np.sqrt(yeb[-1]))/binct
    return np.array(xb),np.array(yb),np.array(yeb)

#-----BoolVal-------------------------------------------------------------------------

def boolval(data,reverse=True):
    '''Boolean Evaluator

    Description:
     Given a list of Boolean values, interprets them as binary and returns the
     corresponding integer.  By default reads lists as having the highest-value digit
     first.
    Inputs:
     data    - LIST: a list of lists Boolean values.
     reverse - BOOL: If set to False, then the Boolean strings will be interpreted as
                     binaries with the lowest value (1) first.
    Outputs:
     data    - LIST: the integer values represented by 'data' if its values are
                     interpreted as binary.

    -J.M.Court, 2015'''
    keyr=range(len(data[0]))                         # Set up the 'key range' to
                                                     # convert Bool list into int
    if reverse:
        keyr=keyr[::-1]                              # Reverse the key range
    mult=[1<<i for i in keyr]                        # Mult is a list of all the
                                                     # powers of 2 from 2^0 to 2^(len
                                                     # of data)
    data=np.array(data)*mult
    data=np.sum(data,axis=1)                         # Multiply Boolean list by mult,
                                                     # sum per row
    return data

#-----CalcLoop------------------------------------------------------------------------

def calcloop(flux,color,fluxe,colore):
    y=flux
    ye=fluxe
    x=color
    xe=colore
    maxyp=y.tolist().index(max(y))
    minyp=y.tolist().index(min(y))
    if maxyp>minyp:
        interv=(minyp,maxyp)
    else:
        interv=(maxyp,minyp)
    newx={}
    newxe={}
    newy={}
    newye={}
```

```
    i1=1
    i2=2
    newx[False]=x[interv[0]:interv[1]+1]
    newxe[False]=xe[interv[0]:interv[1]+1]
    newy[False]=y[interv[0]:interv[1]+1]
    newye[False]=ye[interv[0]:interv[1]+1]
    newx[True]=np.append(x[interv[1]:],x[:interv[0]+1])
    newxe[True]=np.append(xe[interv[1]:],xe[:interv[0]+1])
    newy[True]=np.append(y[interv[1]:],y[:interv[0]+1])
    newye[True]=np.append(ye[interv[1]:],ye[:interv[0]+1])
    if len(newy[True])>=len(newy[False]):
        master=True
    else:
        master=False
    slave = not master
    newmaster=[list(x) for x in zip(*sorted(zip(newy[master], newx[master],
                newxe[master]), key=ope.itemgetter(0)))]
    sp=intp.UnivariateSpline(newmaster[0],newmaster[1],1/(np.array(newmaster[2])**2),
                             ext=0,k=1)
    se=intp.UnivariateSpline(newmaster[0],newmaster[2],ext=0,k=3)
    aves=0.0
    for q in range(len(newy[slave])-2):
        i=q+1
        aves+=( (sp(newy[slave][i])-newx[slave][i]) / np.sqrt(newxe[slave][i]**2+
                se(newy[slave][i])**2) )**2
    loopc=stt.chisqprob(aves,len(newy[slave])+len(newy[master])-2)
    return loopc

#-----CCor-----------------------------------------------------------------------------

# Cross-Correlate

def ccor(data1,data2):
    assert len(data1)==len(data2)
    data1=np.array(data1)
    data2=np.array(data2)
    crosscor=[]
    crosscore=[]
    for i in range(len(data1)):
        r1=data1[(-i-1):]
        r2=data2[:i+1]
        mr1=np.mean(r1)
        mr2=np.mean(r2)
        sr1=np.std(r1)
        sr2=np.std(r2)
        ccorarray=(r1-mr1)*(r2-mr2)/(sr1*sr2)
        crosscor.append(np.mean(ccorarray))
        crosscore.append(np.std(ccorarray))
    for i in range(len(data1)-1):
        r1=data1[:(-i-1)]
        r2=data2[i+1:]
        mr1=np.mean(r1)
        mr2=np.mean(r2)
        sr1=np.std(r1)
        sr2=np.std(r2)
        ccorarray=(r1-mr1)*(r2-mr2)/(sr1*sr2)
        crosscor.append(np.mean(ccorarray))
        crosscore.append(np.std(ccorarray))
    times=np.array(range(len(crosscor)))
```

```
    times=times+1-len(data1)
    return times,crosscor,crosscore

#-----Eddington-----------------------------------------------------------------------

def eddington(M):
    '''Eddington

    Description:
     Returns the Eddington Limit for a given black hole mass, assuming Hydrogen accreta.
    Inputs:
     M - FLOAT: The mass of the black hole in solar masses
    Outputs:
     L - FLOAT: The Eddington Luminosity in ergs/s

    -J.M.Court, 2016'''
    return 1.26E38*M

#-----EqRange--------------------------------------------------------------------------

# Equal Length Range

def eqrange(array):
    '''Equal Length Array

    Description:
     Creates a range with an equal length to that of the array or list given as an
     argument
    Inputs:
     array - ARRAYLIKE: The array to be used as comparison.
    Outputs:
     A range with equal length to the input

    -J.M.Court, 2015'''
    return range(len(array))

#-----Eval_Burst-----------------------------------------------------------------------

# Evaluate Burst

@mjit()
def eval_burst(t,y):
    '''Evaluate Burst

    Description:
     Takes a piece of data and its associated time array, and treats it as a single
     'burst'-like pattern.  Returns the peak flux, peak time, rise time, fall time of
     this burst.  Works best with Get_Bursts.
    Inputs:
     t         - ARRAY: The t (time) co-ordinates of the data.
     y         - ARRAY: The y co-ordinates of the data.
    Outputs:
     peak      - FLOAT: The highest y-value in the burst
     peak_time - FLOAT: The time at which the peak occurs
     rise_time - FLOAT: The time between the start of the burst and its peak
     fall_time - FLOAT: The time between the peak of the burst and its end
    '''
    peak=max(y)
    trough=min(y)
```

```
    p_ind=np.array(y).tolist().index(peak)
    rise_time=t[p_ind]-t[0]
    fall_time=t[-1]-t[p_ind]
    peak_time=t[p_ind]
    return peak,trough,peak_time,rise_time,fall_time

#-----FlnCheck-----------------------------------------------------------------------

def filenamecheck(filename,validext,continu=False):
    '''Filename Checker

    Description:
     Takes a filename and a string representing the expected file extension.  If the
     extension of the file does not match expectations, either kill the script or return
     'False'.
    Inputs:
     filename - STRING: The filename to be checked.
     validext - STRING: The extenstion expected for the file (WITHOUT the leading '.')
     continu  -   BOOL: [Optional: Default=False] If True, returns a value of False for
                        an incorrect file extension.  If False (default), kills the
                        script upon finding an incorrect file extension.
    Outputs:
     iscorr   -   BOOL: True if the file has the correct extension, False otherwise

    -J.M.Court, 2015'''
    flext=(filename.split('.')[-1])
    if flext != validext:
       if continu:
          return False
       else:
          print 'Invalid input file!  Must use .'+str(validext)+' file!'
          signoff()
          exit()
    else:
       return True

#-----Foldify-------------------------------------------------------------------------

def foldify(t,y,ye,period,binsize,phres=None,name='',compr=False,verb=True):
    '''Foldify

    Description:
     Folds a two-dimensional set of data over a period in the first dimension using the
     folding script which is provided in PyAstronomy.  Also calculates how much the peak
     trough difference of the data has been compressed by the fold, and tells the user.
    Inputs:
     t       -   LIST: The t- or x-values of the two-dimensional data.  The period to be
                       folded over is a period in this dimension.
     y       -   LIST: The y-values of the two-dimensional data, must be the same length
                       as x.
     ye      -   LIST: The errors associated with the y-values of the two-dimensional
                       data, must be the same length as x and y.
     period  -  FLOAT: The period of the data, in the same units as the data's x-values.
     binsize -  FLOAT: The size of the new x-axis bins in which to re-bin the data.
     phres   -  FLOAT: [Optional: Default=None] The resolution, in frequency space, at
                       which data will be shown.
     name    - STRING: [Optional: Default=''] the name of the file to be folded.  This
                       name will be used in text outputs printed to screen.
     compr   -   BOOL: [Optional: Default=False] if set True, gives a fourth output
```

```
                    which is the amount by which the min-max difference of the
                    data-set was compressed by folding.
    verb     -   BOOL: [Optional: Default=True] if set false, suppresses all non-error
                    text output.
   Outputs:
    newt     -  LIST: A clipped version of the input t which now only corresponds to one
                    period.
    newy     -  LIST: The folded y-values of the two-dimensional data for one period.
    newye    -  LIST: The errors associated with the folded y-values of the
                    two-dimensional data for one period.
    compr    - FLOAT: see 'compr' input option

   -J.M.Court, 2015'''
   try:
      from PyAstronomy.pyasl import foldAt
   except:
      print 'PyAstronomy module not found; aborting!'
      return t,y,ye
   tn=tnorm(t,binsize)
   if verb:
      print 'Folding File '+name+'...'
   ptdiff=max(y)-min(y)                              # Flux range before folding
   phases=foldAt(tn,period)
   if phres==None:
      try:
         phres=float(raw_input('Input Phase Resolution (0-1): '))
         assert phres<=1.0
      except:
         print 'Invalid phase resolution!  Aborting!'
         return t,y,ye
   npbins=int(1.0/phres)
   phasx =np.arange(0,1,phres)
   phasy =np.zeros(npbins)
   phasye=np.zeros(npbins)
   ny=np.zeros(npbins)
   for i in range(len(y)):
      k=int(phases[i]*npbins)
      phasy[k]+=y[i]
      phasye[k]+=(ye[i]**2)
      ny[k]+=1
   phasy=phasy/ny
   phasye=np.sqrt(phasye)/ny
   afdiff=max(phasy)-min(phasy)                      # Flux range after folding
   if verb:
      print 'Flattened by '+str(100-afdiff/ptdiff*100)+'%'
   if compr:
      return phasx,phasy,phasye,(afdiff/ptdiff)
   else:
      return phasx,phasy,phasye

#-----Fold_Bursts-------------------------------------------------------------------

@mjit()
def fold_bursts(times,data,q_lo=50,q_hi=90,do_smooth=False,alg='cubic spline',
                savgol=5):
   '''Return Phases Using Bursts as Reference Points

   Description:
    Takes a lightcurve and identifies 'bursts' in the data.  The peak of each burst is
```

```
    considered to be at zero phase, and all other points are assigned phases by
    linearly interpolating between them.
   Inputs:
    data       - ARRAY: The data in which bursts are sought.
    q_low      - FLOAT: [Optional: Default=30] The percentile value of the data which
                        will be used as the low-pass threshold.  This threshold
                        determines the edges of already-located bursts, and thus
                        changing it will change the quality of bursts but not the
                        quantity.
    q_mid      - FLOAT: [Optional: Default=50] The percentile value of the data which
                        will be used as the med-pass threshold.  This threshold
                        determines how deep an trough must be before the regions either
                        side of it are considered separate bursts-candidate regions.
                        Must be larger than q_low.
    q_hi       - FLOAT: [Optional: Default=70] The percentile value of the data which
                        will be used as the hi-pass threshold.  This threshold
                        determines how much peak flux a previously defined
                        burst-candidate region must be before it is considered a true
                        burst.  larger than q_med.
    smooth     -  BOOL: [Optional: Default=False] Apply a Savitsky-Golay Filter to
                        smooth the lightcurve before fetching peaks.
    alg        -STRING: [Optional: Default='cubic spline'] The algorithm to use to
                        obtain peaks.
   Outputs:
    burst_locs -  LIST: A list of tuples containing the start and end indices of each
                        burst.

   -J.M.Court, 2015'''
   assert len(times)==len(data)
   peaks=get_bursts(data,q_lo,q_hi,just_peaks=True,smooth=do_smooth,alg=alg,
                    times=times,savgol=savgol)
   peaks.sort()
   p0=peaks[0]
   pe=peaks[-1]
   #data=data[p0:pe+1]
   #times=times[p0:pe+1]
   phases=np.zeros(len(times))
   peaks=np.array(peaks)
   npeaks=[0]
   for i in range(len(peaks)-1):
      if peaks[i+1]-peaks[i]>0.1*(peaks[-1]/float(len(peaks))):
                                                    # Remove any peaks separated by
                                                    # less than 10% of average
                                                    # separation
         npeaks.append(peaks[i+1])
   peaks=npeaks
   numpeaks=len(peaks)
   phases=get_phases_intp(data,windows=1,q_lo=20,q_hi=90,peaks=peaks,givespline=False)
   return np.array(phases),numpeaks,(peaks[0],peaks[-1])

#-----Gauss-------------------------------------------------------------------------

@mjit()
def gauss(mean,standev,x):
   '''Gauss.  Returns a Gaussian'''
   return (1.0/(standev*(2*np.pi)**0.5))*np.exp(-(x-mean)**2/(2*(standev**2)))

#-----Get_Bursts--------------------------------------------------------------------
```

```
def get_bursts(data,q_lo=50,q_hi=90,just_peaks=False,smooth=False,savgol=5,
               alg='cubic spline',times=None):
   '''Get Bursts

   Description:
    Takes a lightcurve and identifies 'bursts' in the data; short, discrete regions of
    increased flux.  Returns the locations of all peaks identified as a list of tuples,
    each of which consist of two integers which correspond to the indices of the start
    and end of a peak in the original data.
   Inputs:
    data       - ARRAY: The data in which bursts are sought.
    q_low      - FLOAT: [Optional: Default=50] The percentile value of the data which
                        will be used as the low-pass threshold.  This threshold
                        determines the edges of already-located bursts, and thus
                        changing it will change the quality of bursts but not the
                        quantity.
    q_hi       - FLOAT: [Optional: Default=90] The percentile value of the data which
                        will be used as the hi-pass threshold.  This threshold
                        determines how much peak flux a previously defined burst-
                        candidate region must be before it is considered a true burst.
                        larger than q_med.
    just_peaks -  BOOL: [Optional: Default=False] If set to true, the function will
                        return a list of peak indices instead of a list of peak
                        datasets.
    smooth     -  BOOL: [Optional: Default=False] If set to true, applies a univariate
                        spline to the data to smooth it.
    savgol     -   INT: [Optional: Default=5] The window size for the Savitsky-Golay
                        filter
    alg        -STRING: [Optional: Default='cubic spline'] The algorithm to use to
                        obtain peaks.
    times      - ARRAY: [Optional: Default=None] If 'load' selected for algorithm, user
                        must also give the time array associated with the data.
   Outputs:
    burst_locs -  LIST: A list of tuples containing the start and end indices of each
                        burst.

   -J.M.Court, 2015'''
   if smooth:                                          # If user has requested
                                                       # smoothing...
      savgol=int(savgol)
      if savgol%2==0:
         savgol+=1
      data=sgnl.savgol_filter(data,savgol,3)           #  Smooth it!
   high_thresh=np.percentile(data,q_hi)
   low_thresh=np.percentile(data,q_lo)
   over_thresh=data>low_thresh                         # Create a Boolean array by
                                                       # testing whether the input array
                                                       # is above the mid threshold.
                                                       # Each region of consecutive
                                                       # 'True' objects is considered
                                                       # a burst-candidate region.
   peak_locs=[]
   burst_locs=[]
   if alg=='cubic spline':
      while True:
         masked=np.array(data)*over_thresh             # Reduce all data outside of
                                                       # burst-candidate regions to zero

         if max(masked)<high_thresh:                   # If highest peak in all
```

```
                                                    # remaining burst-candidate
                                                    # regions is below the high
                                                    # threshold, assume there are no
                                                    # more bursts to be found.
            break
        peak_loc=masked.tolist().index(max(masked))  # Find peak in remaining data
        peak_locs.append(peak_loc)                  # Construct list of peak location
        i=peak_loc
        while i<len(data) and over_thresh[i]:       # Scrub the True objects in the
                                                    # Boolean array corresponding to
                                                    # that peak's candidate region,
                                                    # thus removing it
            over_thresh[i]=False
            i+=1
        i=peak_loc-1
        while i>=0 and over_thresh[i]:
            over_thresh[i]=False
            i-=1
    elif alg=='load':
        if times==None:
            raise Exception('Must provide data times when using "load" mode!')
        times=np.array(times)
        loadfilename=raw_input('Burst File Name: ')
        loadfile=open(loadfilename,'r')
        with open(loadfilename,'r') as f:
            first_line=f.readline()
        if len(first_line.split(','))>6:
            dataex=np.array(first_line.split(','))
            for l in dataex:
                peak_locs.append(float(l))
        else:
            for line in loadfile:
                l=line.split(',')
                peak_locs.append(float(l[0]))
        loadfile.close()
        peak_locs=(np.array(peak_locs)-times[0])/(times[1]-times[0])
        peak_locs=peak_locs.astype(int)
    if just_peaks: return peak_locs
    peak_locs.sort()                                # Sort the list so peaks can be
                                                    # returned in chronological order
    start_col=get_dip(data,0,peak_locs[0])
    for i in range(len(peak_locs)):
        if i==len(peak_locs)-1:
            n_peak=len(data)-1
        else:
            n_peak=peak_locs[i+1]
        end_col=get_dip(data,peak_locs[i],n_peak)
        if end_col>start_col:                       # Force any null-length 'bursts'
                                                    # to be removed
            burst_locs.append((start_col,end_col))
            start_col=end_col
    return burst_locs

#-----Get_Bursts_Windowed-----------------------------------------------------------

def get_bursts_windowed(data,windows,q_lo=50,q_hi=90,smooth=False):
    '''Get Bursts

    Description:
```

```
    Takes a lightcurve and identifies 'bursts' in the data; short, discrete regions of
    increased flux.  Returns the locations of all peaks identified as a list of tuples,
    each of which consist of two integers which correspond to the indices of the start
    and end of a peak in the original data.  Currently can only return the location of
    all burst peaks.
   Inputs:
    data       - ARRAY: The data in which bursts are sought.
    windows    -   INT: The number of windows into which to divide the data
    q_low      - FLOAT: [Optional: Default=50] The percentile value of the data which
                        will be used as the low-pass threshold.  This threshold
                        determines the edges of already-located bursts, and thus
                        changing it will change the quality of bursts but not the
                        quantity.
    q_hi       - FLOAT: [Optional: Default=90] The percentile value of the data which
                        will be used as the hi-pass threshold.  This threshold
                        determines how much peak flux a previously defined
                        burst-candidate region must be before it is considered a true
                        burst.  larger than q_med.
    smooth     -  BOOL: [Optional: Default=False] If set to true, applies a univariate
                        spline to the data to smooth it.
    savgol     -   INT: [Optional: Default=1] The window size for the Savitsky-Golay
                        filter
   Outputs:
    burst_locs -  LIST: A list of tuples containing the peak of each burst.

   -J.M.C.Court, 2016'''
   windows=int(windows)
   windowlength=len(data)/windows
   win_starts=[windowlength*i for i in range(windows)]
   win_ends=[windowlength*(i+1) for i in range(windows-1)]
   win_ends.append(len(data)+1)
   peak_locs=[]
   for i in range(windows):
      new_bursts=np.array(get_bursts(data[win_starts[i]:win_ends[i]],q_lo=q_lo,
                          q_hi=q_hi,just_peaks=True,smooth=smooth,
                          savgol=(win_ends[i]-win_starts[i])/4.0))
      new_bursts+=win_starts[i]
      peak_locs+=new_bursts.tolist()
   return peak_locs

#-----Get Dip-------------------------------------------------------------------------

def get_dip(data,start,finish,smooth=False,savgol=1):
   '''Get Dips

   Description:
    Returns the index of the lowest value between two given points in a dataset
   Inputs:
    data    - LIST:  The dataset in which a trough is to be found
    start   -  INT:  The index of the startpoint of the user-defined sub-range
    finish  -  INT:  The index of the endpoint of the user-defined sub-range
    smooth  -  BOOL: [Optional: Default=False] If set to true, applies a univariate
                     spline to the data to smooth it.
   Outputs:
    key_col -  INT: The index of the lowest value in the user-defined sub-range

   -J.M.Court, 2015'''
   data=np.array(data)
   data_l=np.arange(len(data))
```

```
    if smooth:
        savgol=int(savgol)
        if savgol%2==0:
            savgol+=1
        data=sgnl.savgol_filter(data,savgol,3)
    data=data*(data_l>=start)*(data_l<finish)
    data[data==0]=max(data)
    keycol_loc=data.tolist().index(min(data))
    return keycol_loc

#-----Get Phase-----------------------------------------------------------------------

@mjit()
def get_phases(data,windows=1,q_lo=20,q_hi=90):
    '''Get Phases

    Description:
     Given a set of data points evenly spaced in time, returns the phase coordinate of
     each point.
    Inputs:
     data    -  ARRAY: The data in which bursts are sought.
     windows -    INT: [Optional: Default=1] The number of windows into which to divide
                       the data for analysis.
     q_low   - FLOAT: [Optional: Default=20] The percentile value of the data which will
                       be used as the low-pass threshold.  This threshold determines the
                       edges of already-located bursts, and thus changing it will change
                       the quality of bursts but not the quantity.
     q_hi    - FLOAT: [Optional: Default=90] The percentile value of the data which will
                       be used as the hi-pass threshold.  This threshold determines how
                       much peak flux a previously defined burst-candidate region must be
                       before it is considered a true burst.  larger than q_med.
    Outputs:
     phases - ARRAY: The phase coordinates of the input data

    -J.M.C.Court, 2016'''
    data_keys=range(len(data))                          # Generate the data indices
    data=np.array(data)                                 # Format the input data
    data_phas=np.zeros(len(data),dtype=float)           # Create the phase array
    peak_keys=get_bursts_windowed(data,windows,q_lo=50,q_hi=90,smooth=False)
    peak_keys.sort()                                    # Fetch and sort the indices
                                                        # corresponding to peaks
    dip_keys=[]                                         # Create empty array for indices
                                                        # of dips
    if peak_keys[0]!=0:
        dip_keys.append(get_dip(data,0,peak_keys[0]))   # If there is no peak in the
                                                        # first slot, prepend a false dip
                                                        # at element 0
    for i in range(len(peak_keys)-1):
        dip_keys.append(get_dip(data,peak_keys[i],peak_keys[i+1]))
                                                        # For every pair of peaks, fetch
                                                        # the index of the dip between
                                                        # them
    if peak_keys[-1]!=data_keys[-1]:
        dip_keys.append(get_dip(data,peak_keys[-1],data_keys[-1]))
                                                        # If there is no peak in the last
                                                        # slot, append a false dip at
                                                        # element -1
    data_keys=np.array(data_keys)
    peak_keys=np.array(peak_keys)
```

```
dip_keys=np.array(dip_keys)
if peak_keys[0]==0:                                 # Prepend false dips/peaks at 0
                                                    # or at a negative index to force
                                                    # all comparisons to be valid.
                                                    # This has the effect of making
                                                    # the fit messy at the beginning
                                                    # and end of the time series,
                                                    # but this is not aviodable.
   dip_keys=np.hstack((-1,dip_keys))
elif dip_keys[0]==0:
   peak_keys=np.hstack((-1,peak_keys))
                                                    #         ^
elif peak_keys[0]<dip_keys[0]:                      #         |
   dip_keys=np.hstack((0,dip_keys))                 #         |
   peak_keys=np.hstack((-1,peak_keys))              #         |
else:                                               #         |
   peak_keys=np.hstack((0,peak_keys))               #         |
   dip_keys=np.hstack((-1,dip_keys))                #         |
                                                    #         |
if peak_keys[-1]==data_keys[-1]:                    # Appending false dips/peaks, see
                                                    # above
   dip_keys=np.hstack((dip_keys,data_keys[-1]+1))
elif dip_keys[-1]==data_keys[-1]:
   peak_keys=np.hstack((peak_keys,data_keys[-1]+1))
elif peak_keys[-1]>dip_keys[-1]:
   dip_keys =np.hstack((dip_keys ,data_keys[-1]))
   peak_keys=np.hstack((peak_keys,data_keys[-1]+1))
else:
   peak_keys=np.hstack((peak_keys,data_keys[-1]))
   dip_keys =np.hstack((dip_keys ,data_keys[-1]+1))
for key in data_keys:                               # For each key, extract phase
   if key in peak_keys:                             # If key is a peak, set phase to
                                                    # 0.5
      data_phas[key]=0.5
      continue
   elif key in dip_keys:                            # If key is a dip, set phase to
                                                    # 0.0
      continue
   prev_peak=peak_keys[peak_keys<key][-1]           # Otherwise, collect the keys of
                                                    # the previous/next peaks and
                                                    # dips
   prev_dip =dip_keys[ dip_keys<key][-1]
   next_peak=peak_keys[peak_keys>key][0]
   next_dip =dip_keys[ dip_keys>key][0]
   is_in_fall=(prev_peak>prev_dip)                  # If the considered element's
                                                    # most recent peak was more
                                                    # recent than the most recent dip
                                                    # then it is in the 'fall' regime
                                                    # (phase>0.5)
   prev_marker=float(max(prev_peak,prev_dip))       # Collect the indices of the
                                                    # previous/next significant
                                                    # points of either type
   next_marker=float(min(next_peak,next_dip))
   phase=0.5*(float(key)-prev_marker)/(next_marker-prev_marker)
                                                    # Define phase as half the
                                                    # considered point's fractional
                                                    # distance between them
   if is_in_fall:
      phase+=0.5                                    # Add 0.5 in the fall regime
```

```
        data_phas[key]=phase
    return data_phas

#-----Get Phase INTP-----------------------------------------------------------------

def get_phases_intp(data,windows=1,q_lo=20,q_hi=90,peaks=None,givespline=False):
    if peaks==None:
        peak_keys=get_bursts(data,q_lo=q_lo,q_hi=q_hi,just_peaks=True,smooth=False,
                             savgol=5)
    else:
        peak_keys=peaks
    peak_keys.sort()
    peak_keys=np.array(peak_keys)
    data=np.array(data)
    p_phases=np.arange(0,len(peak_keys),1.0)
    spline=intp.PchipInterpolator(peak_keys, p_phases, extrapolate=True)
    spline.firstpeak=peak_keys[0]                       # Store the keys of the first and
                                                        # last peaks in the spline object
    spline.lastpeak=peak_keys[-1]
    if givespline:
        return spline
    phases=spline(range(len(data)))
    phases=np.remainder(phases,1)
    return phases

#-----GTIMask-------------------------------------------------------------------------

@mjit()
def gtimask(times,gtis):

    '''GTI Mask

    Description:
     Takes a time series and a GTI file taken from FITS and creates a mask to place over
     the time series with 'True' values over timestamps within the GTI and 'False'
     values elsewhere.
    Inputs:
     times -       LIST: A list of times, the x-axis of the data to be considered.  Must
                         be in same physical units as GTI.
     gtis  - FITS TABLE: A FITS table object containing a list of 2-element lists, each
                         of which is the start and endpoint respectively of a good time
                         index.
    Outputs:
     mask  -       LIST: A mask to put over data to hide values outside of the GTI.

    -J.M.Court, 2015'''
    times=np.array(times)
    mask=np.zeros(len(times),dtype=bool)                # Set up initial blank list of
                                                        # False
    for gti in gtis:                                    # For every GTI index:
        smask=(times>gti[0]) & (times<gti[1])           # Create a submask which is the
                                                        # 'and'ed product of
                                                        # times>gti_start and
                                                        # times<gti_end
        mask=mask|smask                                 # 'or' the submask with the main
                                                        # mask
    return mask

#-----LBinify-------------------------------------------------------------------------
```

```
@mjit()
def lbinify(x,y,ye,logres):

    '''Logarithmic Binify

    Decription:
     Takes a 2-dimenstional set of data which has already been evenly binned on the
     x-axis, and re-bins it into larger bins on the x-axis which are evenly spaced in
     log-space.  The original linear binning is retained in regions of the data where
     the logarithmic binning would be smaller than the data resolution.
    Inputs:
     x      -  LIST: The x-values of the two-dimensional data.
     y      -  LIST: The y-values of the two-dimensional data, must be the same length
                     as x.
     ye     -  LIST: The errors associated with the y-values of the two-dimensional
                     data, must be the same length as x and y.
     logres - FLOAT: the size of the new x-axis bins, in log10-space, in which to re-bin
                     the data.
    Outputs:
     xb     -  LIST: The re-binned x-values of the two-dimensional data, i.e. an array
                      of the left-hand edges of the new bins.
     yb     -  LIST: The re-binned y-values of the two-dimensional data.
     yeb    -  LIST: The errors associated with the rebinned y-values of the
                     two-dimensional data.
    Outputs:

    -J.M.Court, 2015'''
    hinge=((x[1]-x[0])*10**logres)/((10**logres)-1)     # Find the 'hinge' point at which
                                                        # to switch between linear and
                                                        # logarithmic binning
    lbin=np.log10(x[0])
    xb =10**(np.arange(lbin,np.log10(x[-1]),logres))    # Setting up arrays to append
                                                        # binned values into
    yb =np.zeros(len(xb))
    yeb=np.zeros(len(xb))
    hingel=sum((xb)<=hinge)                             # Getting the ID of the hinge
                                                        # point in the log
    xbl=len(xb)
    for i in range(hingel,xbl):
        lowid=int(((10**((i*logres)+lbin))-x[0])/(x[1]-x[0]))
                                                        # Calculate the ID of the lowest
                                                        # linear bin that corresponds to
                                                        # this log bin
        uppid=int(((10**(((i+1)*logres)+lbin))-x[0])/(x[1]-x[0]))
                                                        # Calculate the ID of the highest
                                                        # linear bin that corresponds to
                                                        # this log bin
        if uppid>lowid:
            yb[i]=np.mean(y[lowid:uppid])
            yeb[i]=(np.sqrt(sum(np.array(ye[lowid:uppid])**2)))/int(uppid-lowid)
        else:
            yb[i]=0                                     # If no data found, error=power=0
            yeb[i]=0
    mask=x<hinge
    lmask=xb>hinge
    xf=np.append(x[mask],xb[lmask])
    yf=np.append(y[mask],yb[lmask])
    yef=np.append(ye[mask],yeb[lmask])
```

```
    return xf,yf,yef

#-----LeahyN-------------------------------------------------------------------------

@mjit()
def leahyn(data,counts,datres):
    '''Leahy Normaliser

    Description:
     Takes a raw FFT-algorithm output spectrum and normalises it by the Leahy
     convention.
    Inputs:
     data   - LIST: the spectrum to be normalised.
     counts -  INT: the number of photon events in the data sample for which the
                    spectrum was created.
     datres -  INT: the number of time bins in the data sample for which the spectrum
                    was created.
    Outputs:
     leahy  - LIST: the Leahy-normalised spectrum.

    -J.M.Court, 2015'''
    leahy=2*(abs(data[0:int(datres/2.0)])**2)/counts
    return leahy

#-----Lh2RMS--------------------------------------------------------------------------

@mjit()
def lh2rms(leahy,rate,bg,const):

    '''Leahy 2 RMS Converter
    Description:
     Takes a Leahy-normalised spectrum and re-normalises it by the (RMS/Mean)^2
     convention.
    Inputs:
     leahy -  LIST: the leahy-normalised spectrum to be re-normalised.
     rate  - FLOAT: the source + background count rate (per second) of the data sample
                    from which the spectrum was created.
     bg    - FLOAT: the background count rate (per second) of the data sample from which
                    the spectrum was created.
     const - FLOAT: the average power given in Leahy normalisation for pure white noise.
                    Theoretically const=2, but in practice is slightly lower and varies
                    between telescopes.
    Outputs:
     rms   -  LIST: the (RMS/Mean)^2-normalised spectrum.

    -J.M.Court, 2015'''
    denom=(rate-bg)**2
    if denom==0:
        mult=0.0
    else:
        mult=1.0/denom
    rms= (leahy-const)*(rate)*mult
    return rms

#-----LhConst-------------------------------------------------------------------------

def lhconst(data):
    '''LHS Const
```

```
    Decription:
     Finds the normalisation of white noise in a given power Leahy-normalised power
     spectrum.  Assumes no spectral features in the last 20% of the spectrum.
    Inputs:
     data  -  LIST: the data to be converted.
    Outputs:
     const - FLOAT: the normalisation of white noise.

    -J.M.Court, 2015'''
    def leahynoise(x,a):
       return a
    olen=len(data)
    olen=int((4/5.0)*olen)
    datav=data[olen:]
    const=optm.curve_fit(leahynoise,range(len(datav)),datav)
    const=const[0][0]
    if const>2.5 or const<1.5:
       print "WARNING: Leahy constant of "+str(const)+" outside of accepted range!"
    return const

#-----Lomb_Scargle-----------------------------------------------------------------

@mjit()
def lomb_scargle(x,y,ye,freqs):
    '''Lomb Scargle

    Decription:
     Returns the Lomb-Scargle periodogram of data provided by the user, scanning over a
     set of frequencies also provided by the user.
    Inputs:
     x     -  LIST: the time-array for the data to be converted.
     y     -  LIST: the data associated with time array x.
     ye    -  LIST: the errors associated with each point in y.
     freqs -  LIST: the list of discrete frequencies over which the user wants the L-S
                    periodogram to scan.
    Outputs:
     pgram - ARRAY: the Lomb-Scargle power associated with each frequency provided in
             freqs.

    -J.M.Court, 2015'''
    assert len(x)==len(y)
    x=np.array(x)
    y=np.array(y)
    ye=np.array(ye)
    x=x[y>0]                                          # Weed out any negative values
                                                      # here before they break things

    ye=ye[y>0]
    y=y[y>0]
    w=safe_div(np.ones(len(ye)),ye**2)
    y=y-(sum(y*w)/sum(w))
    freqs=np.array(freqs)*2*pi
    wt=np.multiply.outer(x,freqs)
    sin2wt=np.sin(2*wt)
    cos2wt=cosfromsin(2*wt,sin2wt)
    tau=(np.arctan(np.sum(sin2wt,axis=0)/np.sum(cos2wt,axis=0)))/(2*freqs)
    wttau=wt-(freqs*tau)
    sinw=np.sin(wttau)
    cosw=cosfromsin(wttau,sinw)
```

```
    yT=np.vstack(y)
    ysin=yT*sinw
    ycos=yT*cosw
    norm=2*np.sum(w*(y**2)/(len(y)-2))
    w=np.vstack(w)
    pgram=((np.sum(w*ycos,axis=0)**2)/np.sum((w*cosw)**2,axis=0)+
           (np.sum(w*ysin,axis=0)**2)/np.sum((w*sinw)**2,axis=0)) / norm
    return pgram

#-----MCErrorCalc-----------------------------------------------------------------------

@mjit()
def mcerrorcalc(func,data,data_e,ntrials,verbose=False):
    '''Monte Carlo Error Calculation

    Description:
     For a function which takes a set of 1-dimensional values, and for values with
     associated errors, simulates new datasets from the given datavalues and errors and
     uses these simulations to estimate the error on the output value.
    Inputs:
     func    - FUNCTION: the function to be used.  MUST take arguments of the form
                         (data,data_error).
     data    -    ARRAY: the measured data from which new data is to be simulated.
     data_e  -    ARRAY: the error on the measured data.
     ntrials -      INT: the number of simulations to run.

    -J.M.Court, 2017'''
    vals=[]
    for i in range(ntrials):
        new_data=rn.normal(data,data_e)
        val=(func(new_data,data_e))
        if not np.isnan(val):
            vals.append(val)
        if verbose:
            print ' Trial: '+str(i)+'/'+str(ntrials)
    serr=np.std(vals)
    return serr

#-----MXRebin----------------------------------------------------------------------------

@mjit()
def mxrebin(spcdata,spcerrs,xaxis,good,bfac):
    '''Matrix X-Rebin

    Description:
     Takes 2-Dimensional arrays of data and errors which are linearly binned on the
     x-axis and rebins them by a factor of the user's choosing, returning new data
     array, error array, x-axis and Boolean 'good' array.
    Inputs:
     spcdata   - ARRAY: some 2 dimensional array of values.
     spcerrs   - ARRAY: a 2 dimensional array containing the errors associated with the
                        values in spcdata.  Must have the same dimensions as spcdata.
     xaxis     - ARRAY: an array of values corresponding to the x-values of columns in
                        spcdata.  Must be the same length as a row of spcerrs and must
                        be linearly spaced.
     good      - ARRAY: A Boolean array with as many elements as spcdata has columns.
                        Its entries are True unless the corresponding row in spcdata
                        does not correspond to a valid time within the GTIs of the
                        photon count data.
```

```
    bfac      -   INT: the binning factor, i.e. the ratio between new bin size and old
                       bin size.
   Outputs:
    b_spcdata - ARRAY: the rebinned 2-dimensional data array
    b_spcerrs - ARRAY: the rebinned 2-dimensional error array
    b_xaxis   - ARRAY: the rebinned x-axis
    b_good    - ARRAY: the rebinned 'good' array

   -J.M.Court, 2015'''
   spx=len(spcdata[:,0])                            # x-dimension of new matrix
                                                    # matches old matrix
   spy=int(len(spcdata[0,:])/bfac)                  # y-dimension of new matrix
                                                    # equals the y dimension of the
                                                    # old matrix divided by the
                                                    # binning factor
   b_good=np.zeros(spy,dtype=bool)                  # array to label good columns
   b_spcdata=np.zeros([spx,spy])                    # Create the new matrix
   b_spcerrs=np.zeros([spx,spy])                    # Create the new error matrix
   for i in range(spx):                             # For each freq row of fourgr:
      for j in range(spy):                          # For each time col of fourgr:
         celltot=0                                  # Create a running total to put
                                                    # into the matrix cell
         errtot=0                                   # Create a running total to put
                                                    # into the error matrix cell
         for k in range(bfac):                      # Calculate extent of current bin
            b_good[j]=(True)                        # Assume column is good
            if not good[j*bfac+k]:                  # If any of the columns being
                                                    # summed were flagged as bad,
                                                    # flag new column as bad
               celltot=0
               b_good[j]=(False)
               break                                # Don't attempt to find spectrum
                                                    # in column if any constituent
                                                    # columns are bad
            celltot+=spcdata[i,j*bfac+k]            # Sum all values that fall within
                                                    # the given bin
            errtot+=((spcerrs[i,j*bfac+k])**2)
         b_spcdata[i,j]=celltot/bfac                # Divide the cell total by the
                                                    # bin multiplier to convert to a
                                                    # mean
         b_spcerrs[i,j]=np.sqrt(errtot)/bfac
   b_xaxis=[]                                       # Calculate new x-axis
   for i in range(int(len(xaxis)/bfac)):
      b_xaxis.append(xaxis[i*bfac])
   return b_spcdata,b_spcerrs,b_xaxis,b_good

#-----Nones-------------------------------------------------------------------------

def nones(shape):
    '''Function to basically do what np.zeros and np.ones do, but creating an array of Nones.

    -J.Coxon, 2015'''
    makeNone = np.vectorize(lambda x: None)
    return makeNone(np.zeros(shape))

#-----PDColEx-----------------------------------------------------------------------

@mjit()
def pdcolex2(y1,y2,ye1,ye2,gmask):
```

```
    '''Plot Demon Colour Extract (2D)

    Description:
     Takes two equally spaced flux series in the same set of time co-ordinates and
     constructs an array representing the file2 / file1 colour over the same time
     period.  Also returns the sum of the flux of the two series.
    Inputs:
     y1,y2   - ARRAYS: The first and second flux series respectively
     ye1,ye2 - ARRAYS: The errors of the first and second flux series respectively
     gmask   -  ARRAY: A mask of Boolean values with False at every point outside of a
                       GTI in this time series
    Outputs:
     flux    -  ARRAY: The sum total flux from the two bands
     fluxe   -  ARRAY: The errors of 'flux'
     col21   -  ARRAY: The file2/file1 colour
     col21e  -  ARRAY: The error on "col21"

    -J.M.Court, 2015'''
    warnings.filterwarnings("ignore")                   # Div 0 errors are a real
                                                        # possibility.  This is me
                                                        # ignoring them...
    y=   {}                                             # Prepare libraries for data
    ye=  {}
    col= {}
    cole={}
    y[1]=y1[gmask]                                      # Mask data, store in library
    y[2]=y2[gmask]
    ye[1]=ye1[gmask]
    ye[2]=ye2[gmask]
    flux=y[1]+y[2]                                      # Get total flux
    fluxe=np.sqrt(ye[1]**2+ye[2]**2)                    # Get flux error
    for i in range(1,3):                                # For the ith possible numerator
                                                        # band
       for j in range(1,3):                             # For the jth possible
                                                        # denominator band
          if j!=i:                                      # Prevents taking x/x colour
             ld=int(str(i)+str(j))
             col[ld]=(y[i]/y[j])                        # Fetch colour
             cole[ld]=col[ld]*np.sqrt(((ye[i]/y[i])**2)+((ye[j]/y[j])**2))
                                                        # Fetch colour error
    return flux,fluxe,y,ye,col,cole

@mjit()
def pdcolex3(y1,y2,y3,ye1,ye2,ye3,gmask):
    '''Plot Demon Colour Extract (3D)

    See help for Plot Demon Colour Extract (2D)

    -J.M.Court, 2015'''
    warnings.filterwarnings("ignore")                   # Div 0 errors are a real
                                                        # possibility.  This is me
                                                        # ignoring them...
    y=   {}                                             # Prepare libraries for data
    ye=  {}
    col= {}
    cole={}
    y[1]=y1[gmask]                                      # Mask data, store in library
    y[2]=y2[gmask]
    y[3]=y3[gmask]
```

```
    ye[1]=ye1[gmask]
    ye[2]=ye2[gmask]
    ye[3]=ye3[gmask]
    flux=y[1]+y[2]+y[3]                                  # Get total flux
    fluxe=np.sqrt(ye[1]**2+ye[2]**2+ye[3]**2)            # Get flux error
    for i in range(1,4):                                 # For the ith possible numerator
                                                         # band
        for j in range(1,4):                             # For the jth possible
                                                         # denominator band
            if j!=i:                                     # Prevents taking x/x colour
                ld=int(str(i)+str(j))
                col[ld]=(y[i]/y[j])                      # Fetch colour
                cole[ld]=col[ld]*np.sqrt(((ye[i]/y[i])**2)+((ye[j]/y[j])**2))
                                                         # Fetch colour error
    return flux,fluxe,y,ye,col,cole

#-----PDload-------------------------------------------------------------------------

def pdload(filename,isplotd):
    '''PlotDemon loader

    Description:
     Calls plotdld or csvload depending on the type of file to be opened.
    Inputs:
     filename - STRING: The absolute or relative path to the location of the file that
                        will be opened.
     isplotd  - BOOL  : Whether the file to be opened has been identified as a .plotd
                        file
    Outputs:
     times    -      ARRAY: An array, the elements of which are the left-hand edges of
                            the time bins into which counts have been binned.  Units of
                            seconds.
     counts   -      ARRAY: The number of counts in each bin defined in 'times'.
     errors   -      ARRAY: The 1-sigma errors associated with each value in 'counts'
     binsize  -      FLOAT: The size of each bin in 'times'.  Saved for speed upon
                            loading.
     gti      - FITS TABLE: The table of GTI values from the event data .fits file.
     mxpcus   -        INT: The maximum number of PCUs active at any one time during the
                            observation.
     bgpcu    -      FLOAT: An estimate of the count rate of the background flux during
                            the full observation, in counts per second per PCU,
                            multiplied by the number of PCUs.
     flavour  -     STRING: A useful bit of text to put on plots to help identify them
                            later on.
     chanstr  -     STRING: A string containing the high and low channel numbers
                            separated by a dash.
     mission  -     STRING: The name of the satellite
     obsdata  -      TUPLE: The first element is the name of the object, the second is
                            the observation ID.
     version  -     STRING: The Version of FITSGenie in which the file was created

    -J.M.Court, 2015'''
    if isplotd:
        return plotdld(filename)
    else:
        return csvload(filename)

#-----PlotdLd-------------------------------------------------------------------------
```

```
def plotdld(filename):
   '''.Plotd Load

   Description:
    Opens a .plotd file and retrieves the useful data from it.
   Inputs:
    filename - STRING: The absolute or relative path to the location of the file that
                       will be opened.
   Outputs:
    times    -      ARRAY: An array, the elements of which are the left-hand edges of
                           the time bins into which counts have been binned.  Units of
                           seconds.
    counts   -      ARRAY: The number of counts in each bin defined in 'times'.
    errors   -      ARRAY: The 1-sigma errors associated with each value in 'counts'
    binsize  -      FLOAT: The size of each bin in 'times'.  Saved for speed upon
                           loading.
    gti      - FITS TABLE: The table of GTI values from the event data .fits file.
    mxpcus   -        INT: The maximum number of PCUs active at any one time during the
                           observation.
    bgpcu    -      FLOAT: An estimate of the count rate of the background flux during
                           the full observation, in counts per second per PCU,
                           multiplied by the number of PCUs.
    flavour  -     STRING: A useful bit of text to put on plots to help identify them
                           later on.
    chanstr  -     STRING: A string containing the high and low channel numbers
                           separated by a dash.
    mission  -     STRING: The name of the satellite
    obsdata  -      TUPLE: The first element is the name of the object, the second is
                           the observation ID.
    version  -     STRING: The Version of FITSGenie in which the file was created

   -J.M.Court, 2015'''
   try:
      readfile=open(filename,'rb')                    # Open the .plotd file
   except:
      print ''
      print 'File '+filename+' not found!  Aborting!'
      signoff()
      exit()
   data=pickle.load(readfile)                         # Unpickle the .plotd file
   times=np.array(data['time'])                       # Unleash the beast! [extract the
                                                      # file]
   rates=np.array(data['rate'])
   errors=np.array(data['errs'])
   tstart=data['tstr']
   binsize=data['bsiz']
   gti=data['gtis']
   mxpcus=data['pcus']
   bgest=data['bkgr']
   bgsub=data['bsub']
   bgdata=data['bdat']
   flavour=data['flav']
   chanstr=data['chan']
   mission=data['miss']
   obsdata=data['obsd']
   version=data['vers']
   bgpcu=bgest*mxpcus                                 # Collect background * PCUs
   readfile.close()
   return times,rates,errors,tstart,binsize,gti,mxpcus,bgpcu,bgsub,bgdata,flavour,
```

```
                chanstr,mission,obsdata,version

#-----csvLoad-------------------------------------------------------------------------

def linesplit(lx,iscomma):
   if iscomma:
      return lx.split(',')
   return lx.split()

def csvload(filename):
   '''.csv Loader

   Description:
    Extracts information from a csv file.  Column delimiters automatically set using
    python string object's .split() function.  Assumes the file is of format
    'time,rate' if two columns, 'time,rate,rate_error' if three columns of 'time,
    time_error,rate,rate_error; if four or more columns.
   Inputs:
    filename - STRING: The absolute or relative path to the location of the file that
                       will be opened.
   Outputs:
    times    -      ARRAY: An array, the elements of which are the left-hand edges of
                           the time bins into which counts have been binned.  Units of
                           seconds.
    counts   -      ARRAY: The number of counts in each bin defined in 'times'.
    errors   -      ARRAY: The 1-sigma errors associated with each value in 'counts'
    binsize  -      FLOAT: The size of each bin in 'times'.  Saved for speed upon
                           loading.
    gti      - FITS TABLE: The table of GTI values from the event data .fits file.
    mxpcus   -        INT: The maximum number of PCUs active at any one time during the
                           observation.
    bgpcu    -      FLOAT: An estimate of the count rate of the background flux during
                           the full observation, in counts per second per PCU,
                           multiplied by the number of PCUs.
    flavour  -     STRING: A useful bit of text to put on plots to help identify them
                           later on.
    chanstr  -     STRING: A string containing the high and low channel numbers
                           separated by a dash.
    mission  -     STRING: The name of the satellite
    obsdata  -      TUPLE: The first element is the name of the object, the second is
                           the observation ID.
    version  -     STRING: The Version of FITSGenie in which the file was created

   -J.M.Court, 2015'''
   f=open(filename,'r')
   times=[]
   rates=[]
   errors=[]
   got_firstline=False
   haserrs=False
   xind=0
   for line in f:
       if not got_firstline:
          if ((len(line.split())<2) and (len(line.split(','))<2)):
                                                    # Assume lines with <2 columns,
                                                    # or a non-number in col zero are
                                                    # part of the header, skip 'em
             continue
          if ',' in line:
```

```
            iscomma=True
         else:
            iscomma=False
         l=linesplit(line,iscomma)
         try:
            float(l[0].strip('"'))
         except:
            continue
         llen=len(l)
         if llen<3:                                  # Interpret csv data depending on
                                                     # how many columns
            yind=1
            haserrs=False
         elif llen<4:
            yind=1
            haserrs=True
            eind=2
         elif l[0][0]=='"':
            yind=1
            haserrs=True
            eind=2
         else:
            yind=2
            haserrs=True
            eind=3
         got_firstline=True
      l=linesplit(line,iscomma)
      times.append(float(l[xind].strip('\n').strip('"')))
      rates.append(float(l[yind].strip('\n').strip('"')))
      if haserrs:
         errors.append(float(l[eind].strip('\n').strip('"')))
   if not haserrs:
      errors=np.sqrt(np.array(rates))                # Assume root flux errors if none
   times=np.array(times)
   rates=np.array(rates)
   errors=np.array(errors)
   tstart=times[0]
   times=times-times[0]
   binsize=times[1]-times[0]
   gti=None
   mxpcus=1
   bgpcu=1
   bgsub=0
   bgdata=None
   flavour='csv'
   chanstr='Unknown'
   mission='Unknown'
   obsdata=['Unknown','Unknown']
   version='From csv'
   return times,rates,errors,tstart,binsize,gti,mxpcus,bgpcu,bgsub,bgdata,flavour,
               chanstr,mission,obsdata,version

#-----PlotdSv-------------------------------------------------------------------------

@mjit()
def plotdsv(filename,times,rates,errors,tstart,binsize,gti,mxpcus,bgest,bgsub,bgdata,
            flavour,chanstr,mission,obsdata,version):
   '''.Plotd Save
```

```
    Description:
     Takes the input of the data products required to create a .plotd file (to read with
     plotdemon) and creates a .plotd file at a location given as the first input.
    Inputs:
     filename -     STRING: The absolute or relative path to the location of the file
                            that will be created.
     times    -      ARRAY: An array, the elements of which are the left-hand edges of
                            the time bins into which counts have been binned.  Units of
                            seconds.
     counts   -      ARRAY: The number of counts in each bin defined in 'times'.
     binsize  -      FLOAT: The size of each bin in 'times'.  Saved for speed upon
                            loading.
     gti      - FITS TABLE: The table of GTI values from the event data .fits file.
     mxpcus   -        INT: The maximum number of PCUs active at any one time during the
                            observation.
     bgest    -      FLOAT: An estimate of the count rate of the background flux during
                            the full observation, in counts per second per PCU.
     flavour  -     STRING: A useful bit of text to put on plots to help identify them
                            later on.
     chanstr  -     STRING: A string containing the high and low channel numbers
                            separated by a dash.
     mission  -     STRING: The name of the satellite
     obsdata  -      TUPLE: The first element is the name of the object, the second is
                            the observation ID.
     version  -     STRING: The Version of FITSGenie in which the file was created
    Outputs:
     [none]

    -J.M.Court, 2015'''
    savedata={}                                         # Open library object to save in
                                                        # file
    savedata['time']=times                              # Dump each piece of data into an
                                                        # appropriate library element
    savedata['rate']=np.array(rates)
    savedata['errs']=np.array(errors)
    savedata['tstr']=tstart
    savedata['bsiz']=binsize
    savedata['gtis']=gti
    savedata['pcus']=mxpcus
    savedata['bkgr']=bgest
    savedata['flav']=flavour
    savedata['chan']=chanstr
    savedata['miss']=mission
    savedata['bsub']=bgsub
    savedata['bdat']=bgdata
    savedata['obsd']=obsdata
    savedata['vers']=version
    filename=uniqfname(filename,'plotd')                # Get the next available name of
                                                        # form filename(x).plotd
    wfile = open(filename, 'wb')                        # Open file to write to
    pickle.dump(savedata,wfile)                         # Pickle the data (convert into
                                                        # bitstream) and dump to file
    wfile.close()                                       # Close file
    return filename

#-----RMS_N-------------------------------------------------------------------------

@mjit()
def rms_n(data,counts,datres,rate,bg,const):
```

```
    '''RMS Normaliser

    Description:
     Takes a raw FFT-algorithm output spectrum and normalises it by the (RMS/Mean)^2
     convention.
    Inputs:
     data   -  LIST: the spectrum to be normalised.
     counts -   INT: the number of photon events in the data sample for which the
                     spectrum was created.
     datres -   INT: the number of time bins in the data sample for which the spectrum
                     was created.
     rate   - FLOAT: the source + background count rate (per second) of the data sample
                     from which the
                     spectrum was created.
     bg     - FLOAT: the background count rate (per second) of the data sample from
                     which the spectrum was created.
     const  - FLOAT: the average power given in Leahy normalisation for pure white
                     noise.  Theoretically const=2, but in practice is slightly lower
                     and varies between telescopes.
    Outputs:
     rms    -  LIST: the Leahy-normalised spectrum.

    -J.M.Court, 2015'''
    leahy=leahyn(data,counts,datres)                    # Leahy normalise the data
    rms=lh2rms(leahy,rate,bg,const)                     # Convert to RMS
    return rms

#-----RMS-------------------------------------------------------------------------------

#@mjit()
def rms(data,data_err=None,with_err=False):
    '''RMS

    Description:
     Returns the fractional RMS of a 1-dimensional data set
    Inputs:
     data     - LIST: The data to find the RMS of.
     data_err - LIST: The errors on those data points
    Outputs:
     rms  - FLOAT: The rms of the data

    -J.M.Court, 2015'''
    if type(data_err)==type(None):
       data_err=np.array([0]*len(data))
    if np.mean(data)==0:
       return 'div0'
    nr=1.0/len(data)
    mn=np.mean(data)
    vd=np.sum((data-mn)**2)-np.sum((data_err)**2)
    orms=((vd*nr)**0.5)/abs(mn)

    if not with_err:
       return orms
    else:
       rmserr=mcerrorcalc(rms,data,data_err,100,verbose=False)
       return orms,rmserr

#-----Safe_Div---------------------------------------------------------------------------
```

```
@mjit()
def safe_div(x,y):
   '''Safe Div

   Description:
    Divides the first inputs by the second inputs if the latter is nonzero.  If an
    element of the second input is zero, the corresponding element in the output is
    zero.
   Inputs:
    x - ARRAY: The numerator
    y - ARRAY: The denominator
   Outputs:
    r - ARRAY: The result of division, or zero

   -J.M.Court, 2015'''
   r=np.zeros(len(y))
   r[y!=0]=x[y!=0]/y[y!=0]
   return r

#-----SignOff-------------------------------------------------------------------------

def signoff():
   '''Sign Off

   Description:
    Prints an underline with some spaces.  That's all.

   -J.M.Court, 2015'''
   print ''
   print '-----------------------------------------------'
   print ''

#-----sinfromcos----------------------------------------------------------------------

@mjit()
def sinfromcos(x,cosx):
   '''Sine from Cosine

   Description:
    Returns the sine of an array of values when also given their cosines.  Using this
    function is faster than using sine if the cosine values are already stored.
   Inputs:
    x    - ARRAY: the array of values to calculate the sine of.
    cosx - ARRAY: the cosines array of values to calculate the sines of.
   Outputs:
    sinx - ARRAY: the sines of x.

   -J.M.Court, 2015'''
   sinx=np.absolute((1-cosx**2)**0.5)
   signx=np.sign(((x+pi)%(2*pi))-pi)
   return sinx*signx

@mjit()
def cosfromsin(x,sinx):
   '''Cosine from Sine

   Description:
    Returns the cosine of an array of values when also given their sines.  Using this
    function is faster than using cosine if the sine values are already stored.
```

```
    Inputs:
     x    - ARRAY: the array of values to calculate the cosine of.
     sinx - ARRAY: the sines array of values to calculate the cosines of.
    Outputs:
     cosx - ARRAY: the sines of x.

    -J.M.Court, 2015'''
    cosx=np.absolute((1-sinx**2)**0.5)
    signx=np.sign(((x-pi/2)%(2*pi))-pi)
    return cosx*signx

#-----SLPlot-------------------------------------------------------------------------

def slplot(x,y,ye,xlabel,ylabel,title,figid="",typ='both',errors=True):
    '''Standard/Log Plotter

    Description:
     Takes a two-dimensional array of data and plots it twice on the same figure; once
     on standard linear axes, and once on logarithmic axes.
    Inputs:
     x      -   LIST: The x-values of the two-dimensional data.
     y      -   LIST: The y-values of the two-dimensional data, must be the same length
                      as x.
     xlabel - STRING: The title of the x-axis on the graphs.
     ylabel - STRING: The title of the y-axis on the graphs.
     title  - STRING: The title of the figure.
     figid  - STRING: Gives the Figure a name such that it can be easily manipulated or
                      closed outside of this function.
     typ    - STRING: User can input 'log' or 'lin' to only display plot of that type.
     errors -   BOOL: True or False: whether to display errorbars on plots
    Outputs:
     filename - STRING: The filename actually used when saving

    -J.M.Court, 2015'''
    pl.close(figid)                                     # Close any previous plot of this
                                                        # type
    pl.figure(figid)                                    # Create spectrum plot
    if typ in ('lin','both'):
       if typ=='lin':
          pl.subplot(111)                               # If 'lin' passed as typ word,
                                                        # only create one subplot
       else:
          pl.subplot(211)                               # If 'both' passed as typ word,
                                                        # make 1st of 2 subplots
       pl.grid(True,which="both")
       pl.xlabel(xlabel)
       pl.ylabel(ylabel)
       pl.title(title)
       if errors:
          pl.errorbar(x,y,ye)                           # Plot data
       else:
          pl.plot(x,y)
       pl.plot([x[0],x[-1]],[0,0])
    if typ in ('log','both'):
       if typ=='log':
          ax=pl.subplot(111)                            # If 'log' passed as typ word,
                                                        # only create one subplot
       else:
          ax=pl.subplot(212)                            # If 'both' passed as typ word,
```

```
                                                # make 2nd of 2 subplots
      pl.xlabel(xlabel)
      pl.ylabel(ylabel)
      pl.title(title)
      if errors:
         pl.errorbar(x,abs(y),ye,fmt='k')       # Plot data
      else:
         pl.plot(x,abs(y),'k')                  # Plot log-log data
      ax.set_xscale('log')
      ax.set_yscale('log')
      pl.grid(True,which="both")
   #if typ in ('lin','log','both'):
   #   pl.show(block=False)                     # Show both plots together
   else:
      print 'Invalid typ!  No plot shown.'      # Complain if none of 'lin',
                                                # 'log' or 'both are given as typ
                                                # word

#-----Spliner-------------------------------------------------------------------------

@mjit()
def spliner(data,errors=None):
   '''Spliner

   Description:
    Smooths evenly-sampled data using a first-order Univariate spline algorithm
   Inputs:
    data        - ARRAY: The data to be smoothed
    errors      - ARRAY: [Optional] The errors associated with the errors
   Outputs:
    smooth_data - ARRAY: The smoothed data values

   - J.M.C.Court,2016'''
   if type(errors)==type(None):
      errors=np.ones(len(data))                 # If no errors given, do not
                                                # weight points
   else:
      assert len(errors)==len(data)
   spline=intp.UnivariatrSpline(range(len(data)),data,w=errors**-1,k=2)
                                                # Construct a univariate spline
                                                # function from the data
   #pl.figure()                                 # Uncomment these 4 lines to
                                                # allow for display of smoothed
                                                # and pre-smoothed data
   #pl.plot(data_keys,data,'0.5')
   smooth_data=spline(range(len(data)))         # Reconstruct the data using the
                                                # spline
   #pl.plot(data_keys,data,'k')
   #pl.show(block=True)
   return smooth_data

#-----SpecaLd-------------------------------------------------------------------------

def specald(filename):
   '''.Speca Load

   Description:
    Opens a .speca file and retrieves the useful data from it.
   Inputs:
```

```
 filename - STRING: The absolute or relative path to the location of the file that
                    will be opened.
Outputs
 spcdata  -  ARRAY: An array, the elements of which are the non-normalised power
                    spectra taken over different equal-length time intervals of the
                    photon count data being considered.
 good     -  ARRAY: A Boolean array with as many elements as spcdata has columns.
                    Its entries are True unless the corresponding row in spcdata
                    does not correspond to a valid time within the GTIs of the
                    photon count data.
 rates    -  ARRAY: The count rate per second of photons in the time interval
                    represented by the corresponding row of spcdata.
 prates   -  ARRAY: An array of floats with as many elements as spcdata has columns.
                    Denotes the highest number of total counts in each timing window
                    when the counts are binned on a time of binsize * binfac
 trates   -  ARRAY: An array of floats with as many elements as spcdata has columns.
                    Denotes the lowest number of total counts in each timing window
                    when the counts are binned on a time of binsize * binfac
 phcts    -    INT: The total number of photons detected, overall.
 bg       -  ARRAY: An estimate of the count rate of the background flux during the
                    full observation, in counts per second per PCU, multiplied by
                    the number of PCUs active in the corresponding row of specdata.
 binsize  -  FLOAT: The size in seconds of the time bins used when converting event
                    data into time binned photon count data.
 foures   -  FLOAT: The length of time corresponding to a single row of spcdata, in
                    seconds.
 bgest    -  FLOAT: An estimate of the count rate of the background flux during the
                    full observation, in counts per second per PCU.
 flavour  - STRING: A useful bit of text to put on plots to help identify them later
                    on.
 cs       - STRING: A string containing the high and low channel numbers separated
                    by a dash.
 mission  - STRING: The name of the satellite
 obsdata  -  TUPLE: The first element is the name of the object, the second is the
                    observation ID.
 wtype    - STRING: A string denoting the functional geometry of the windows used
                    for Fourier analysis.
 slide    -  FLOAT: The time separation of the start times of windows represented
                    by two consecutive rows of spcdata, in seconds
 binfac   -    INT: See trates or prates
 version  - STRING: The Version of FITSGenie in which the file was created

-J.M.Court, 2015'''
try:
   readfile=open(filename,'rb')                    # Open the .speca file
except:
   print ''
   print 'File not found!  Aborting!'
   signoff()
   exit()
data=pickle.load(readfile)                         # Unpickle the .speca file
spcdata=data['data']                               # Unleash the beast! [extract the
                                                   # file]
good=np.array(data['good'])
rates=np.array(data['rate'])
prates=np.array(data['prts'])
trates=np.array(data['trts'])
phcts=data['phct']
n_pcus=np.array(data['npcu'])
```

```
    binsize=data['bsiz']
    bgest=data['bkgr']
    foures=data['fres']
    flavour=data['flav']
    cs=data['chan']
    mission=data['miss']
    obsdata=data['obsd']
    wtype=data['wndw']
    slide=data['slid']
    binfac=data['sbin']
    version=data['vers']
    readfile.close()
    print ''
    print 'Power spectra taken over '+str(foures)+'s of data each'
    print str(sum(good))+'/'+str(len(spcdata))+' power spectra are good'
    if flavour!='':
       print "Flavour is '"+str(flavour)+"'"           # Only print flavour if there is
                                                        # a flavour to print
    bg=n_pcus*bgest
    return spcdata,good,rates,prates,trates,phcts,bg,binsize,foures,bgest,flavour,cs,
                   mission,obsdata,wtype,slide,binfac,version

#-----SpecaSv-------------------------------------------------------------------------

@mjit()
def specasv(filename,spcdata,good,rates,prates,trates,phcts,npcus,binsize,bgest,foures,
            flavour,cs,mission,obsdata,wtype,slide,spcbinfac,version):
    '''.Speca Save

    "Should've gone to SpecaSaver..."

    Description:
     Takes the input of the data products required to create a .speca file (to read with
     specangel) and creates a .speca file at a location given as the first input.
    Inputs:
     filename  - STRING: The absolute or relative path to the location of the file that
                         will be created.
     spcdata   -  ARRAY: An array, the elements of which are the non-normalised power
                         spectra taken over different equal-length time intervals of the
                         photon count data being considered.
     good      -  ARRAY: A Boolean array with as many elements as spcdata has columns.
                         Its entries are True unless the corresponding row in spcdata
                         does not correspond to a valid time within the GTIs of the
                         photon count data.
     rates     -  ARRAY: An array of floats with as many elements as spcdata has
                         columns.  Denotes the total number of photon counts in the time
                         interval represented by the corresponding row in spcdata,
                         divided by the width of a column in seconds.
     prates    -  ARRAY: An array of floats with as many elements as spcdata has
                         columns.  Denotes the highest number of total counts in each
                         timing window when the counts are binned on a time of
                         binsize * spcbinfac
     trates    -  ARRAY: An array of floats with as many elements as spcdata has
                         columns.  Denotes the lowest number of total counts in each
                         timing window when the counts are binned on a time of
                         binsize * spcbinfac
     phcts     -    INT: The total number of photons detected, overall.
     npcus     -  ARRAY: An array of ints with as many elements as spcdata has columns.
                         States the number of detectors that were active when the data
```

```
                    represented by the corresponding row of spcdata was recorded.
    binsize   -  FLOAT: The size in seconds of the time bins used when converting event
                        data into time binned photon count data.
    bgest     -  FLOAT: An estimate of the count rate of the background flux during the
                        full observation, in counts per second per PCU.
    foures    -  FLOAT: The length of time corresponding to a single row of spcdata, in
                        seconds.
    flavour   - STRING: A useful bit of text to put on plots to help identify them
                        later on.
    cs        - STRING: A string containing the high and low channel numbers separated
                        by a dash.
    mission   - STRING: The name of the satellite
    obsdata   -  TUPLE: The first element is the name of the object, the second is the
                        observation ID.
    wtype     - STRING: A string denoting the functional geometry of the windows used
                        for Fourier analysis.
    slide     -  FLOAT: The time separation of the start times of windows represented
                        by two consecutive rows of spcdata, in seconds
    spcbinfac -    INT: See trates or prates
    version   - STRING: The Version of FITSGenie in which the file was created
   Outputs:
    filename - STRING: The filename actually used when saving

   -J.M.Court, 2015'''
   savedata={}                                          # Open library object to save in
                                                        # file
   savedata['data']=spcdata                             # Dump each piece of data into an
                                                        # appropriate library element
   savedata['good']=good
   savedata['rate']=rates
   savedata['prts']=prates
   savedata['trts']=trates
   savedata['phct']=phcts
   savedata['npcu']=npcus
   savedata['bsiz']=binsize
   savedata['bkgr']=bgest
   savedata['fres']=foures
   savedata['flav']=flavour
   savedata['chan']=cs
   savedata['miss']=mission
   savedata['obsd']=obsdata
   savedata['wndw']=wtype
   savedata['slid']=slide
   savedata['sbin']=spcbinfac
   savedata['vers']=version
   filename=uniqfname(filename,'speca')                 # Get the next available name of
                                                        # form filename(x).speca
   wfile = open(filename, 'wb')                         # Open file to write to
   pickle.dump(savedata,wfile)                          # Pickle the data (convert into
                                                        # bitstream) and dump to file
   wfile.close()                                        # Close file
   return filename

#-----SRinR-------------------------------------------------------------------------

def srinr(t,binning,domain,minv=None,maxv=None):
   '''Subrange in Range: Subrange Creator

   Description:
```

```
    Takes an array of ordered values and asks the user to select a subrange.  Converts
    the users entries (in the same units as the data) into data IDs and checks that
    they are valid and within the array.
   Inputs:
    t       -   LIST: An evenly spaced list of values.
    binning -   LIST: The spacing of the list of values.
    domain  - STRING: The name of the physical quantity represented by the data in t.
   Outputs:
    new_mn  -    INT: The array index of the subrange minimum.
    new_mx  -    INT: The array index of the subrange maximum.
    boolv   -   BOOL: A flag to denote whether range was clipped.

   -J.M.Court, 2015'''
   old_mn=0
   old_mx=len(t)
   try:
      if minv==None:
         new_mn=float(raw_input('Minimum '+domain+': '))
                                                    # Fetch new min value from user
      else:
         new_mn=minv
      new_mn=len(t[t<new_mn])
   except:
      new_mn=old_mn                                 # Treat garbage input as 'no
                                                    # change'
   try:
      if maxv==None:
         new_mx=float(raw_input('Maximum '+domain+': '))
                                                    # Fetch new max value from user
      else:
         new_mx=maxv
      new_mx=len(t[t<=new_mx])-1
   except:
      new_mx=old_mx                                 # Treat garbage input as 'no
                                                    # change'
   if new_mn>=new_mx:
      print 'Invalid clipping!  Resetting to full.'
      boolv=False                                   # Set the flag to propagate the
                                                    # fact no clipping took place
      new_mn=old_mn
      new_mx=old_mx
   else:
      boolv=True
   return new_mn,new_mx,boolv

#-----TNorm-------------------------------------------------------------------------

def tnorm(t,res):
   '''Time-Normer

   Description:
    Takes an array of evenly-spaced time values and normalises them by subtracting the
    lowest value from each.  Then forces each time to equal an integer multiple of the
    pre-normalising resolution to deal with computational inaccuracies in the
    subtraction process.
   Inputs:
    t   -  LIST: The list of evenly spaced time-values.  They need not be ordered, but
                 the lowest value must be first.
    res - FLOAT: The resolution of the list t; entered by hand again to negate
```

```
                 computational error.
    Outputs:
     nt  -  LIST: The normalised time values.

    -J.M.Court, 2014'''
    t=np.array(t)
    sttime=t[0]
    for i in range(len(t)):
       t[i]=res*np.floor((t[i]-sttime)/float(res))     # Rescaling time to start at 0,
                                                       # rounding to deal with errors
                                                       # incurred by subtraction of
                                                       # large numbers
    return t

#-----TokenLoc-------------------------------------------------------------------------

def tokenloc(enigma,token):
    envals=[]
    entoks=[]
    for i in range(len(enigma)):
       if enigma[i][0]==',':
          et=enigma[i][1:]                             # Extract the tokens and how many
                                                       # bytesare allocated to each in
                                                       # the datawords
       else:
          et=enigma[i]
       entoks.append(et[0])
       if et[0]=='C':
          tk=et.split('[')[1].split(']')[0].split(',')
          envals.append(int(enigma[i].split('{')[1]))
    if not (token in entoks):
       return None,None
    cloc=entoks.index(token)                           # Fetch the 'channels' token
    r1=sum(envals[:cloc])
    r2=sum(envals[cloc+1:])
    return tk,(r1,r2)

#-----UniqFName-------------------------------------------------------------------------

def uniqfname(filename,extension):
    '''Unique Filename Generator

    Description:
     When given the path to a file, ascertains whether a file already exists in that
     location.  If it does, this function finds the lowest value of n such that
     'filename(n).extension' is unique and returns this new name.  If the file does not
     exist, returns 'filename.extension'.
    Inputs:
     filename  - STRING: The full path to the proposed file location, excluding the
                         extension.
     extension - STRING: The extension of the proposed file location, excluding the
                         leading '.'.
    Outputs:
     uniqname  - STRING: A string containing the best available unique filename.

    -J.M.Court, 2015'''
    filenamex=filename
    n=1
    while os.path.isfile(filenamex+'.'+extension):
```

```
      # print filename+'.'+extension+' already exists!# Can uncomment this for more
                                                      # verbosity.
      filenamex=filename+'('+str(n)+')'
      n+=1
   uniqname=filenamex+'.'+extension
   return uniqname

#-----VCRebin-----------------------------------------------------------------------

@mjit()
def vcrebin(vecdata,bfac):
   '''Vector X-Rebin

   Description:
    Takes 1-Dimensional array of data and which is linearly binned and rebins it by a
    factor of the user's choosing, returning new data array.
   Inputs:
    vecdata   - ARRAY: some 2 dimensional array of values.
   Outputs:
    b_vecdata - ARRAY: the rebinned 2-dimensional data array

   -J.M.Court, 2015'''
   bfac=int(bfac)
   spx=len(vecdata)                                   # x-dimension of new vector
   b_vecdata=np.zeros(spx)                            # Create the new vector
   for i in range(spx):                               # For each freq row of fourgr:
      celltot=0                                       # Create a running total to
                                                      # deposit into the vector cell
      for k in range(bfac):                           # Calculate extent of current bin
         celltot+=vecdata[i*bfac+k]                   # Sum all values that fall within
                                                      # the given bin
      b_vecdata[i]=celltot/bfac                       # Divide the cell total by the
                                                      # bin multiplier to convert to a
                                                      # mean
   return b_vecdata

#-----XtrFilLoc---------------------------------------------------------------------

@mjit()
def xtrfilloc(filepath):

   '''Extract File Location
   Description:
    Given the relative path to a file, extracts the file name and it's location.
   Inputs:
    filepath - STRING: The path to a file
   Outputs:
    filename - STRING: The name of the file
    fileloca - STRING: The location of the file

   -J.M.Court, 2015'''
   filename=(filepath.split('/')[-1])                 # Identify filename without
                                                      # directory
   if filename!=filepath:
      fileloca=filepath[:-len(filename)]              # Fetch directory
   else:
      fileloca=os.getcwd()

   if fileloca[0]!='/':
```

```
        fileloca=os.getcwd()+fileloca[1:]              # Dump current directory onto the
                                                       # front to make this absolute
    return filename,fileloca
```

E.6 XTE-PAN Lib

```
#! /usr/bin/env python

# |-----------------------------------------------------------------------|
# |----------------------------XTEPAN_LIB-------------------------------|
# |-----------------------------------------------------------------------|

#
# A selection of useful XTE-specific functions which are placed here to reduce clutter
# in the other files of PANTHEON.  Supports GoodXenon_2s and E_125us_64M_0_1s
#  DATAMODEs.

#
#
# Contents:
#
#  CHRANGE   - clips a set of event data given to it to screen out photons outside of
#              some energy range given by the user.
#
#  DISCNEV   - Discards non-photon events from a table of data
#
#  BIHCHAN   - converts an RXTE channel ID into a channel range ID.
#
#  GETBIN    - gets the binning time of a FITS data table
#
#  GETDAT    - gets the data of a FITS data table
#
#  GETGTI    - gets the GTIs associated with a FITS data table
#
#  GETINI    - gets the start time of the observation associated with a FITS data
#              table
#
#  GETPCU    - gets the number of PCUs that contributed events to a FITS data table
#
#  MAXEN     - returns the highest energy or channel valid for the instrument.

# Modes supported:
# 'E'
# 'B''
# 'SB'
# 'GoodXenon'

#-----Importing Modules-------------------------------------------------------------

import pan_lib as pan
from numpy import array, zeros, arange
from numpy import sum as npsum

#-----ChRange-----------------------------------------------------------------------

def chrange(data,low,high,header):
    '''Channel Ranger
```

```
    Description:
     Takes raw event or GoodXenon data from PCA on RXTE and removes all photons outside
     of an energy channel range given by the user.
    Inputs:
     data     - FITS DATA: the event data; raw photon arrival times and data words.
     low      -       INT: the lowest channel the user wants to consider.
     high     -       INT: the highest channel the user wants to consider.
     datamode -    STRING: the datamode in which the data was taken.
    Outputs:
     ch_data  - FITS DATA: the same as data, but with all photons outside of the given
                           channel range removed.

    -J.M.Court, 2015'''
    datamode=header['DATAMODE']
    if datamode[0]=='B':
       chconv=header['TDDES2'].split('&')[2].split('[')[1].split(']')[0].split(',')
       low=bihchan(datamode,low,chconv)
       high=bihchan(datamode,high,chconv)+1
       data=data[:,low:high,:]
       data=npsum(data,axis=1)
       data=data.reshape([len(data)*len(data[0])])
       return data
    elif datamode[:2]=='SB':
       print 'No energy information in this datamode!' # No energy information in SB
                                                       # datamodes, so don't filter
       return data
    else:
       words=data.field(1)
       if low<=0 and high>=255:
          return data                                  # Don't bother searching through
                                                       # if the user wants full range
       if datamode[:10]=='GoodXenon_':
          r=(17,25)                                    # GoodXenon data contains the
                                                       # channels as written, no need to
                                                       # convert
       elif datamode[:2]=='E_':
          enigma=header['TEVTB2'].split('^')[0][1:-1].split('}')[:-1]
                                                       # Fetch (and decode) the
                                                       # enigmatic TEVTB2 word
          tk,r=pan.tokenloc(enigma,'C')
          if tk==None:
             print 'No energy information in this datamode!'   # If no channels token is
                                                       # present, can't filter on energy
             return data
          low=bihchan(datamode,low,tk)
          high=bihchan(datamode,high,tk)+1
       else:
          print datamode,'not yet supported, using full range!'
          return data
       words=array(pan.boolval((words[:,r[0]:r[1]]).tolist()))
       mask1=(words>=low)
       mask2=(words<=high)
       ch_data=data[mask1&mask2]
       return ch_data

#-----DiscNEv-------------------------------------------------------------------------

def discnev(datas,datamode):
```

```
    '''Discard Non-Events

    Decription:
     Given a FITS data table, discards all non-photon events from the table.
    -J.M.Court, 2014'''
    if datamode[:2] in ['B_','SB']:
        return discnevb(datamode,datas.field(1))
    mask=datas['Event'][:,0]==True                    # Creating a mask to obscure any
                                                      # data not labelled as photons
    datas=datas[mask]                                 # Applying the mask
    return datas

#-----DiscNEvB-----------------------------------------------------------------------

def discnevb(datamode,datas):

    '''Discard Non-Events: Binned Data Version
    Decription:
     Given a FITS binned data table, discards all non-photon events from the table.

    -J.M.Court, 2014'''
    if datamode[:2]=='SB':
        datas=datas.reshape([len(datas)*len(datas[0])])
        return datas
    return datas

#-----BiHChan------------------------------------------------------------------------

def bihchan(datamode,chan,chanconv):
    '''Channel-Get

    Description:
     Converts a channel number into the ID of the range which contains that channel in
     B_2ms_4B_0_35_H or B_8ms_16A_0_35_H data from PCA on RXTE.
    Inputs:
     chan   - INT: the real channel number for PCA data from RXTE.
    Outputs:
     n_chan - INT: the ID of the range containing the relevant channel.

    -J.M.Court, 2015'''
    n_nchan=len(chanconv)
    chan=int(chan)
    t_chan=[0]*n_nchan
    for i in range(len(chanconv)):
        t_chan[i]=map(int,chanconv[i].split('~'))
    if chan>t_chan[-1][-1]:                           # Simple sanity check to prevent
                                                      # messy accidents
        print 'This data type does not store photons above Channel '+str(t_chan[-1][-1])+'!'
        pan.signoff()
        exit()
    elif chan<t_chan[0][0]:
        print 'This data type does not store photons below Channel '+str(t_chan[0][0])+'!'
        pan.signoff()
        exit()
    for i in range(len(t_chan)):
        if chan<=t_chan[i][-1]:                       # This is just a list of ifs.
                                                      # It checks if the value falls
                                                      # into each and, if not, carries
                                                      # on.
```

```
        return i

#-----GetBG-------------------------------------------------------------------------

@pan.mjit()
def getbg(data,low_channel,high_channel):
   '''Get BG

   Description:
    Returns the counts from a standard XTE Background file created with pcabackest
   Inputs:
    event        - FITS OBJECT: The data array of FITS file that has been opened.
    low_channel  -         INT: The highest channel data is to be collected from.
    high_channel -         INT: The lowest channel data is to be collected from.
   Outputs:
    times  -              LIST: The time array associated with the fluxes
    fluxes -              LIST: The fluxes at each time

   -J.M.Court,2015'''
   times=data.field(0)
   PCU0=data.field(1)                                # Collect data from PCU0
   PCU1=data.field(2)                                # Collect data from PCU1
   PCU2=data.field(3)                                # Collect data from PCU2
   PCU3=data.field(4)                                # Collect data from PCU3
   PCU4=data.field(5)                                # Collect data from PCU4
   PCU=array([PCU0,PCU1,PCU2,PCU3,PCU4])
   PCU[:,:,:low_channel]=0                           # Remove data from outside of
                                                     # channel arrays
   PCU[:,:,(high_channel+1):]=0
   PCU=npsum(PCU,axis=2)
   fluxes=npsum(PCU,axis=0)
   flux_ers=fluxes**0.5
   fluxes=fluxes/16.0                                # Change from counts to counts/s
   flux_ers=flux_ers/16.0                            # Change from counts to counts/s
   return times,fluxes,flux_ers

#-----GetBin-------------------------------------------------------------------------

def getbin(event,datamode):
   '''Get Bin

   Description:
    Returns the time binning of a given set of FITS data.
   Inputs:
    event - FITS OBJECT: The FITS file that has been opened.
   Outputs:
    bsz   -       FLOAT: The bin size of the observation (s).

   -J.M.Court, 2014'''
   if datamode[:2] in ['B_','SB']:
      bsz=event[1].header['1CDLT2']
   else:
      bsz=event[1].header['TIMEDEL']
   return bsz

#-----GetDat-------------------------------------------------------------------------

def getdat(event):
   '''Get data
```

```
    Description:
     Returns the data table of a given set of FITS data.
    Inputs:
     event - FITS OBJECT: The FITS file that has been opened.
    Outputs:
     data  -  FITS TABLE: The data table from the FITS file.

    -J.M.Court, 2014'''
    data=event[1].data                                  # Extract GTI indices
    return data

#-----GetGTI-------------------------------------------------------------------------

def getgti(event):
    '''Get GTI

    Description:
     Returns the GTI of a given set of FITS data.
    Inputs:
     event - FITS OBJECT: The FITS file that has been opened.
    Outputs:
     gti   -     2D LIST: A list, each element of which contains the start and end point
                 of a GTI.

    -J.M.Court, 2014'''
    gti=event[2].data                                   # Extract GTI indices
    return gti

#-----GetIni-------------------------------------------------------------------------

def getini(event):
    '''Get Ini

    Description:
     Returns the initial (lowest) time of a given set of FITS data.
    Inputs:
     event - FITS OBJECT: The FITS file that has been opened.
    Outputs:
     ini   -       FLOAT: The starting time of the observation (s).

    -J.M.Court, 2014'''
    ini=event[1].header['TSTART']
    return ini

#-----Get Obs------------------------------------------------------------------------

def getobs(event,datamode,filepath):
    '''Get Obs

    Description:
     Fetches a tuple consisting of the object and obs_id of the observation.'''
    if datamode[:10]=='GoodXenon_':
        try:
            obsid=(filepath.split('/')[-4])             # GoodXenon for XTE doesnt store
                                                        #  obs_id for some reason

        except:
            obsid=''
```

```
    else:
        try:
            obsid=event[1].header['OBS_ID']
        except:
            obsid=''
    obsdata=(event[1].header['OBJECT'],obsid)
    return obsdata

#-----Get PCU-------------------------------------------------------------------------

def getpcu(words,header,t_pcus=None,pculist=False):
    '''Get PCUs

    Description:
     If given a list of RXTE event words, and the data-mode in which their associated data is stored,
     returns the number of PCUs that contributed photons to the dataset.  Currently works for two
     datamodes: GoodXenon_2s and E_125us_64M_0_1s with the intention to eventually generalise it to
     any event data.
    Inputs:
     words    - 2D ARRAY: The array of event words, each of which is an array of True/False statements.
     datamode -   STRING: The DATAMODE in which the data to be analysed is stored.
    Outputs:
     pcus     -      INT: The number of PCUs active when the data was taken.  If a non-recognised event
                          word is given, 1 is returned instead along with an error message.
    -J.M.Court, 2015'''
    datamode=header['DATAMODE']
    if datamode[0]=='E':
        enigma=header['TEVTB2'].split('^')[0][1:-1].split('}')[:-1]
                                                        # Again fetch (and decode) the
                                                        # enigmatic TEVTB2 word
        tk,r=pan.tokenloc(enigma,'D')
    elif datamode=='GoodXenon_2s':
        r=7,10
    else:
        r=None
    if r==None and t_pcus==None:
        goodpcus=False
        while not goodpcus:
            try:
                pcus=int(raw_input('Number of Active PCUS: '))
                                                        # Ask the user how many there are
                assert pcus<6                           # Check they give a number between
                                                        # 1 and 5 (inclusive)
                assert pcus>0
                goodpcus=True
            except:
                'Invalid number of PCUs!'
        return pcus                                     # Use the number they give as the
                                                        # number of PCUs
    elif t_pcus!=None:
        return t_pcus
    words=(words[:,r[0]:r[1]]).tolist()
    pcus=[0,0,0,0,0]
    if [False,False,False] in words: pcus[0]=1
    if [False,False,True] in words: pcus[1]=1
    if [False,True,False] in words: pcus[2]=1
    if [False,True,True] in words: pcus[3]=1
    if [True,False,False] in words: pcus[4]=1
    if sum(pcus)==0:
```

```
      print 'Error!  0 PCUs present in data!'
      return 1
   elif not pculist:
      return sum(pcus)
   else:
      return (sum(pcus), pcus)

#-----Get Tim-----------------------------------------------------------------------

def gettim(data,event,tstart,res,datamode):
   '''Get Times

   Description: Returns the DATA Word or table column containing data on arrival times.

   -J.M.Court, 2015'''
   if datamode[0]=='B':
      times=[]
      for i in range(len(data)):
         times+=[(i*res)+tstart]
      return array(times),data
   elif datamode[:2]=='SB':
      data=data.tolist()
      timeseeds=event.field(0)-event.field(0)[0]+tstart
      times=[]
      indx=0
      prevtimeseed=timeseeds[0]
      for i in timeseeds:
         nx=prevtimeseed+1
         while i-nx>0.9:                              # When a 1s chunk is missing,
                                                      # zero pad the data here

            times+=(arange(0,1.0,res)+nx).tolist()
            data=data[:indx]+([(data[indx]+data[indx-1])/2.0]*int(1.0/res))+data[indx:]
            nx+=1
            indx+=int(1.0/res)
         for j in arange(0,1.0,res):
            times+=[i+j]
         indx+=int(1.0/res)
         prevtimeseed=i
      return array(times),array(data)

   else:
      return data.field(0),data

#-----Get Wrd-----------------------------------------------------------------------

def getwrd(data,datamode):
   '''Get Words

   Description: Returns the DATA Word or table column containing data on PCUs.

   -J.M.Court, 2015'''
   if datamode[:2] in ['B_','SB']:
      return None
   else:
      return data.field(1)

#-----Get Wrd Row-------------------------------------------------------------------
```

```
def getwrdrow(words,mask,datamode):
   '''Get Word Rows

   Description: Returns Data Words filtered by a mask, for data that has datawords.

   -J.M.Court, 2015'''
   if datamode[:2] in ['B_','SB']:
      return None
   else:
      return words[mask]

#-----MaxEn-------------------------------------------------------------------------

def maxen(datamode):
   '''Max Energy

   Description:
    Returns the highest energy or channel valid for the instrument.

   -J.M.Court, 2015'''
   if datamode[:2] in ['B_','SB']:
      mcha=datamode.split('_')[-2]
      return int(mcha)
   else:
      return 255
```

E.7 Mode Get

```
#! /usr/bin/env python

# |-----------------------------------------------------------------------|
# |-----------------------------MODE_GET---------------------------------|
# |-----------------------------------------------------------------------|

# A script to extract the datamode, start time and end time of all .FITS files in a
# folder and save this data to a .txt file.

#-----Importing Modules-------------------------------------------------------------

try:
   import sys
   import os
   import pan_lib as pan
   from astropy.io import fits
   from math import log
   from numpy import array
except:
   print 'Modules missing!  Aborting!'
   print ''
   print '-----------------------------------------------'
   print ''
   exit()

#-----Welcoming Header--------------------------------------------------------------

print ''
print '-------Running Mode Get: J.M.Court, 2015------'
```

```
print ''

#-----Opening Files-----------------------------------------------------------------

read=0
outname=pan.uniqfname('modes','txt')
tmpname=pan.uniqfname('temp','txt')
info = open(outname, 'w')
files=sorted(os.listdir(os.getcwd()))

#-----Excluding Readings of .plotd and .speca Files-----------------------------------

for fileid in range(len(files)):
   filename=files[fileid]
   if '.plotd' in filename or '.speca' in filename:
      files[fileid]=tmpname

#-----Creating Header---------------------------------------------------------------

info.write('{0:26} {1:22} {2:12} {3:12} {4:6} \n \n'.format('File','Datamode','Start',
                                                            'End','Length'))


#-----Extracting Data--------------------------------------------------------------

files=array(files)
for filename in files[files!=tmpname]:
   try:
      cfile=fits.open(filename)[1].header
      read+=1
      try:
         dmode=cfile['DATAMODE']
      except:
         dmode=None
      try:
         stime=float(cfile['TSTART'])
         etime=float(cfile['TSTOP'])
         dtime=etime-stime
      except:
         stime=None
         etime=None
         dtime=None
      info.write('{0:26} {1:22} {2:12} {3:12} {4:6} \n'.format(filename,dmode,
                                                                 str(stime),str(etime),
                                                                 str(dtime)))
   except:
      None
info.close()
if read==0:
   print 'No FITS files found!'
   os.remove(outname)
else:
   print 'Information saved as '+outname+'!'
pan.signoff()
exit()
```