Master thesis, University of Copenhagen

Dark Cosmology Centre

Supervisor: Dr. Darach Watson

Swift XRT pile-up correction and the canonical gamma ray burst lightcurve

Jón Hafsteinn Guðmundsson

Kópavogur, 16th December 2007

Contents

Abstract

In this thesis we present a new function to correct for pile-up in Swift XRT gamma ray burst data and use that correction to try to find empirically the parameters to describe the canonical GRB X-ray afterglow lightcurve. The correction function is arrived at by plotting the measured countrate against the corrected (by hand) countrate for various bursts at various times and fitting a function to the dataset. The purpose of finding this function was to be able to correct a large number of lightcurves quickly because we wished to search for an empirical description of the canonical GRB afterglow lightcurve. Armed with this function we then find the main parameters of the X-ray afterglow lightcurve (slopes and breaktimes) for chosen bursts and use them in correlation plots to try to determine a function to describe the canonical lightcurve.

Acknowledgements

This work was written as a Masters thesis at the University of Copenhagen. It was done under the guidance and supervision of Dr. Darach Watson at the Dark Cosmology Centre of the University of Copenhagen and would not have been possible without his help any more than without his lightcurve extraction program the code of which is included in an appendix at the end.

This work makes use of data supplied by the UK Swift Science Data Centre at the University of Leicester (http://www.swift.ac.uk). Also it makes use of data from the Swift lightcurve collection available from the website of Nat Butler (http://astro.berkeley.edu/~nat/).

Chapter 1

Introduction

Gamma Ray Bursts (GRBs) were first discovered by the Vela satellite in 1967 but it took quite a long time before any real progress was made in understanding their nature. We now know that these sharp flashes of gamma radiation come from cosmological distances and are associated with the most energetic events in the Universe. There have been shown to be two different types of bursts, short duration ones (< 2s) which release harder radiation and long duration ones (> 2s) which release softer radiation. Here we will only discuss the longer type. In these spectacular events enormous amounts of energy are released (roughly 10^{51} ergs for long GRB's, perhaps rather less for short ones) in the form of high energy γ-ray photons over a very short time period (usually less than 10^2 s for long bursts). When it was finally established that the bursts were cosmological in origin a model was developed which seems to describe them properly and has come to be known as the *standard fireball model.* According to the standard fireball model GRBs arise from the collapse of massive stars to form black holes leading to the expulsion of jets of gas and radiation at relativistic speeds into the surrounding medium. The radiation emitted during the bursts is emitted in two main phases; the prompt emission, caused by internal shocks in the outflow of jets of matter, and the afterglow which is caused by the decelaration of the forward shock in the ambient medium. One of the as yet unsolved problems of these tremendous events is finding a function to describe the afterglow decay lightcurve despite the fact that it seems to follow more or less the same pattern every time. One particular reason for seeking this function is to be able to determine when a burst is detected how bright it will be after a given period of time (an hour, 24 hours and so on). This would in turn make it significantly easier to decide if it will be worth the effort of trying to observe the burst with particular satellites (that may require a long time to focus on the target) or if the remaining afterglow will by that time be too faint to be useful for that satellites purpose. This was in fact the original motivation for the investigation presented here.

The Swift satellite has observed hundreds of GRBs with great precision since its launch and the quick response time of this satellite has made it possible to observe many of these GRBs almost from the onset of the prompt emission until the afterglow has as good as disappeared. From observing all these afterglows a certain picture of the emission decay has emerged, showing that the decay usually proceeds in three distinct stages, each of which can be described by a power-law but with sharp breaks in between them. As mentioned above there has been effort for some time to find an empirical function to describe the decay with more precision, both in order to facilitate the search for details of the underlying mechanism and for the aforementioned reason of hoping to predict the brightness of a new GRB after a given amount of time. In order to perform such a search one must examine a large number of Swift lightcurves but here one encounters a problem that must be solved first. This problem is that of piled-up data or measurements where the CCD chip of the Swift XRT camera has been exposed to a greater flux of photons than its readout time resolution is designed to handle and therefore some of these photons are not counted. This leads both to lower countrates and an artificial hardening of the spectrum as the energy of all photons that hit a single pixel within one element of readout time is attributed to one artificial photon [7]. Obviously this problem must be corrected for in order to use the data for any kind of research. This can be an extremely time consuming process when many bursts are involved as there has not been available any simple and quick method to correct many datasets at once. Here it was decided to try to find one function that the measured countrate could be plugged into in order to make correcting a large number of datasets a much simpler and less time consuming process. The details of the search for such a correction method will form the greater part of this thesis followed by an investigation into the canonical GRB X-ray lightcurve in which the pile-up correction method developed in the first part of this project was used.

In the search for the canonical X-ray afterglow lightcurve it was decided to look for correlation between the parameters of each of the power-law decay segments and the break times between them. The collection of lightcurves generated using the correction method and the code used to fully extract them from the measurement data can be found in the appendices at the end of this thesis.

Chapter 2

Data analysis

The Swift website (*http://swift.gsfc.nasa.gov*) contains all the observation data from the Swift mission available for download as well as a large cache of tools specially developed to deal with this data. We used *xselect*, *lcurve* and *lcmath* from the *heasoft ftools* package to extract images, spectra and lightcurves and to subtract background noise from the data as well as to determine the proper binning for the lightcurves. The ftool *fkeyprint* was used to determine the right ratio between the source and background regions. During the search for the pile-up correction function this was done by hand, one burst at a time in order to compare the actual countrate observed with the true (pile-up corrected) countrate. The process was as follows: When the data package for a burst had been downloaded and unpacked the events files for photon counting mode and windowed timing mode (for each mode separately) were read into *xselect*. First of all an image was extracted and plotted using the program *ds9* to locate the source and a convenient area for a background extraction region (when a background subtraction was deemed necessary). The image was used to create filters for the source, both a circle and an annulus (30 arcseconds radius for the circle and outer radius for the annulus, 8 pixels inner radius for the annulus) and a larger circle for background extraction. If a background subtraction was required it was done using *lcmath* having obtained the appropriate scaling factor for source and background using *fkeyprint*. When investigating the pile-up correction *xselect* was used to make time filters so the effects of pile-up at several different countrates could be compared. When preparing the lightcurves that were used for the pile-up investigation binning was uniformly chosen to be 50s. It should also be noted that during this part background subtraction was skipped since when pile-up does occur it will also affect what background noise is there, even if this effect is probably negligeble.

Chapter 3

Pile-up correction

In order to use Swift XRT data for research it is necessary to first correct the data for the pile-up which inevitably occurs when detected bursts are brighter than the equipment is designed for. A precise definition would be the coincidence of two or more photons per CCD time-resolution element, or frame-time, within an event-detection cell [2]. This happens, in other words, when photons come in faster than the readout time of the instrument can cope with so it will measure fewer counts than actually hit it resulting in lightcurves that are somewhat flattened at the top. All photons that hit a given pixel during a single integration period will be counted as one and so the end result will not only be reduced countrates during the brightest period but also an artificially hardened spectrum since the energy of all the photons will be attributed to this artificial one. This is a prominent problem with single-photon-counting charge-coupled device (CCD) cameras and must be dealt with for each such device as it is strongly dependent on readout time, pixel size and other parameters of the device as well as general construction quality. Here we will deal with two of the available modes of operation of the Swift XRT instrument, windowed timing mode and photon counting mode. Of those two the windowed timing mode can cope with much higher countrates (one can expect pile-up to become significant at about 100 counts/s) but only gives a 1D image whereas the photon counting mode gives a 2D image (but here pile-up becomes significant at about 0.7 counts/s). If there is pile-up it is concentrated in the center of the image (since the GRBs are point sources) and so one can exclude the piled-up region by picking an annulus shaped extraction area rather than a circle when extracting spectra and lightcurves. This of course excludes the brightest part of the image and must be corrected for to get the actual countrate. The ftool *xrtmkarf* is used to create an Ancillary Response Function (ARF) file which includes the point spread function of the chip and using this function with the countrate from the annulus region one can find the true countrate. This process is very time consuming however if many bursts are needed and so we set out to find a simple and quick method to correct a large number of bursts for pile-up without having to examine each and every one individually. Most of all it was important to avoid the need to take an annular extraction region for every single burst. The idea was to compare measured countrates with corrected ones for several bursts, choosing several different time slices for each burst. This would then be plotted up with the purpose of fitting a function to the data. If successful one could then use this function to correct any number of bursts quickly and simply for pile-up, requiring no more than plugging in the raw countrate from a normal circular extraction region.

To get a good image of how strongly pile-up influences countrates several bursts with high countrates were chosen, downloaded and unpacked from the Swift archives. Then, using *xselect* and the *ds9* imaging program, two region filters (circle and annulus) were made for each burst. This way of analysing the pile-up required the extraction regions to be uniform in size. As previously mentioned it was decided to use 30 arcseconds radius for the circular extraction region and the outer radius of the annular extraction region. The inner radius of the annuli was chosen to be 8 pixels (1 pixel = $2''.36$) in order to be sure of excluding the pile-up of the brightest sources used for this investigation without cutting too much out from those not so bright where pile-up is hardly noticable. To find what the true countrate would be in the absence of pile-up the countrate for the annulus was multiplied by the ratio between ARF files for circle and annulus, the function being

$$c_t = c_a \frac{A_c}{A_a} \tag{3.1}$$

where c_t is the true countrate, c_a is the measured countrate using an annulus filter, A_c is the ARF file value for a circle and A_a is the ARF file value for an annulus. For each burst several different timeslices were chosen for the comparison and so a plot of the ratio between measured (circle) countrate and corrected (annulus multiplied by ARF ratio) countrate was constructed for each mode of operation (windowed timing and photon counting, see figures 3.2, 3.3, 3.4 and 3.5). Also we made sure to choose bursts captured on different locations on the CCD chip as the point spread function is dependent on the distance from the middle of the chip as well as the shape and size of the extraction region. It has been shown that the point spread function can be well fitted by a King function [6]

$$P(r) = \left(1 + \left(\frac{r}{r_c}\right)^2\right)^{-\beta} \tag{3.2}$$

where the parameters r_c and β are functions of the images location on the chip as well as the energy of the incoming photons.

As the objective was to find one function to use in correcting the countrates of every burst automatically it was necessary to first determine if position on the chip had significant influence on the correction offered by the point spread function as this would make the task much more complicated if not entirely futile. An investigation into the variation of ratio between the ARFs for circle and annulus as a function of distance from the center of the chip confirmed that while there was a difference it was small enough to be absorbed in what error one must always expect anyway (around 10% can generally be assumed here unless otherwise stated). Also there is the fact that the Swift satellite seems to "aim" very well so the image is nearly always within half of the distance from the center to the edge and usually much closer to the center than that. Therefore the aforementioned effort to choose bursts captured on different locations on the chip was deemed enough to make sure that the correction would be quite applicable to any burst without sacrificing precision. The question of energy dependence was even more vital to the project since, if the correction were significantly energy dependent, the process of correcting each burst with the derived function would hardly be much more practical than doing it by hand if one needed the energy of each photon to use it. An investigation into this also revealed that the energy dependence was within the aforementioned acceptable error (see figure 3.1).

Having confirmed that the use of a single function of measured countrate would be acceptable to correct quickly for pile-up the next step was to find that function. As previously mentioned we compared the measured countrates (using the circular extraction region) of different bursts at different times during the afterglow decay (3-5 different times for each burst) with the corresponding corrected countrates (using the annular extraction region and multiplying by the ARF ratio). This data along with the function fit to each set of points (pc-mode and wt-mode) can be seen in the following plots (3.2, 3.3, 3.4 and 3.5). To fit a function to the data we used the MPFIT routine for IDL (available from Craig Markwandts homepage, *http://cow.physics.wisc.edu/~craigm/idl/fitting.html*). Originally the plan was to fit a power-law to the data but it soon became clear that a power-law did not represent the available data very well. Plotted in logspace both datasets show a small yet significant curvature which must be taken into account so the function used for fitting the data was of the form

$$R_m = aR_t^b R_t^{c \log R_t} \tag{3.3}$$

where a, b, and c are the parameters to be fitted, R_m is the measured countrate and R_t is the true countrate. As the objective is of course to transform measured countrates into correct ones the final correction function was therefore of the form

$$R_t = 10^{\frac{-b \pm \sqrt{b^2 - 4c(\log a - \log R_m)}}{2c}}. \tag{3.4}$$

For the photon counting mode the fit eventually gave the function

$$R_m = 0.906696 R_t^{0.699773} R_t^{-0.129157 \log R_t} \tag{3.5}$$

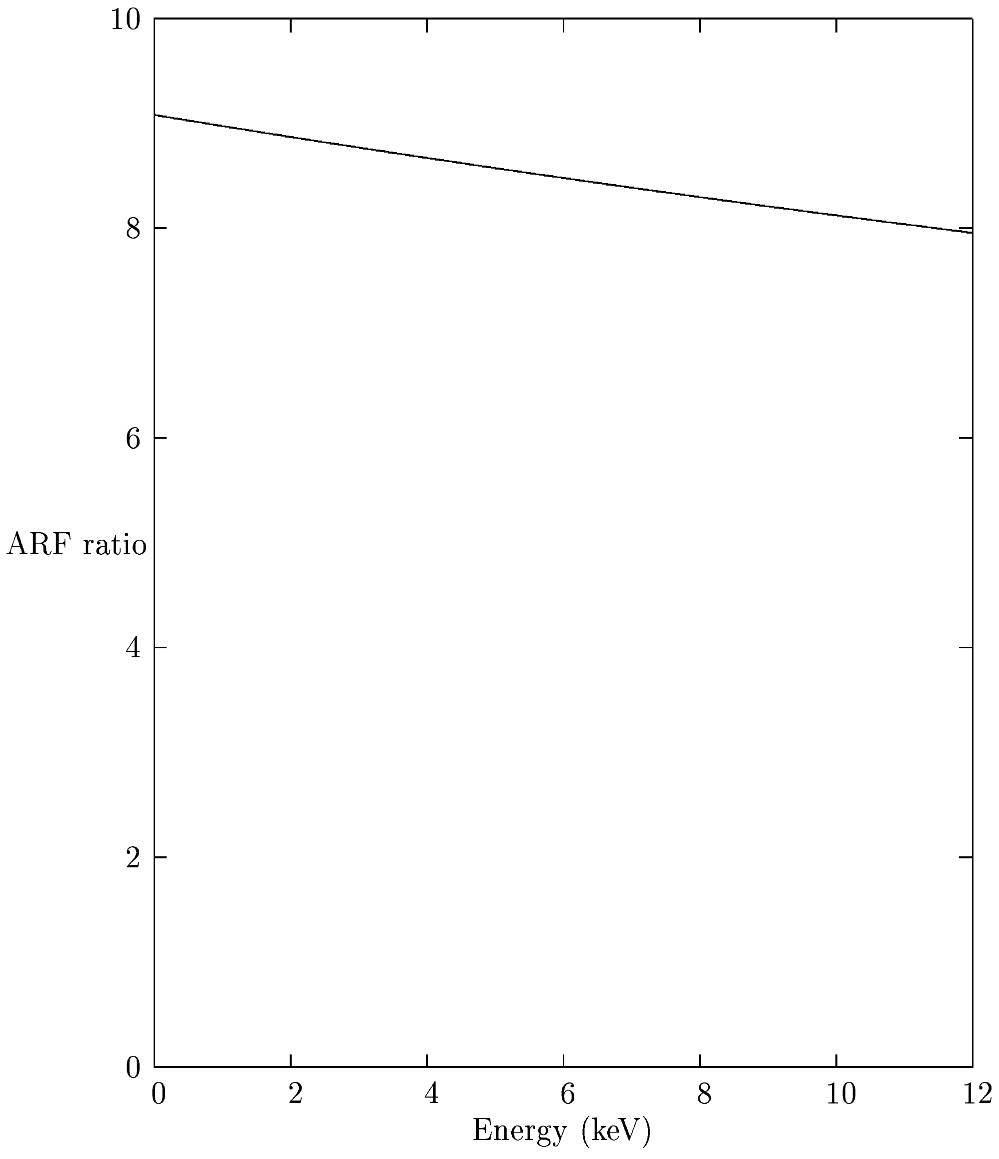


Figure 3.1: The circle/annulus ratio for Ancillary Responce Function files as a function of energy using photon counting mode. By far the greatest part of the radiation for each burst is in the lower half of the energy range presented here.

with $\chi^2_{reduced} = 1.7162$. Converting this to the much more useful format of having the correct countrate as a function of the measured countrate gives

$$R_t = 10^{2.70900 + \frac{\sqrt{0.467706 - 0.516628 \log R_m}}{-0.258314}}. \tag{3.6}$$

For the windowed timing mode the fitting resulted in the function

$$R_m = 0.561183 R_t^{1.28976} R_t^{-0.0932849 \log R_t} \tag{3.7}$$

with $\chi^2_{reduced} = 1.7157$. Converting this to the form of correct countrate as function of measured countrate gives

$$R_t = 10^{6.91301 + \frac{\sqrt{1.56986 - 0.373140 \log R_m}}{-0.186570}}. \tag{3.8}$$

Once these correction functions were available it was necessary to test them on a number of bursts that had not been used to extract the points for the fitting process. This was of course done by applying the correction functions to whole lightcurves extracted from the measurement data, and then comparing them to pile-up corrected lightcurves that were available elsewhere. We compared this correction to the lightcurves made available by Nat Butler (see Butler and Kocevski [1]) and those available from the Swift website by Evans et al. at the UK Swift Science Centre at the University of Leicester [4]. This yielded some interesting results as our corrected lightcurves seemed to consistently agree very well with Butler's lightcurves but frequently did not agree with the lightcurves from Evans et al. in Leicester. Burst after burst showed the same pattern, generally the disagreement between the different correction methods becoming visible when countrates started getting very high and pile-up therefore very significant. In general the most obvious difference came at very high countrates (upwards of 100 counts/s) where the lightcurves from Evans et al. show rather lower countrates than those from Butler and those presented here. Of course all three agree much better when countrates are too low for pile-up to appear because then there is nothing to correct and it should be kept in mind that outside the realm of very high countrates disagreements are generally minor and irregular. The following plots (3.6, 3.7 and 3.8) show representative cases of the similarities and differences between the three correction methods.

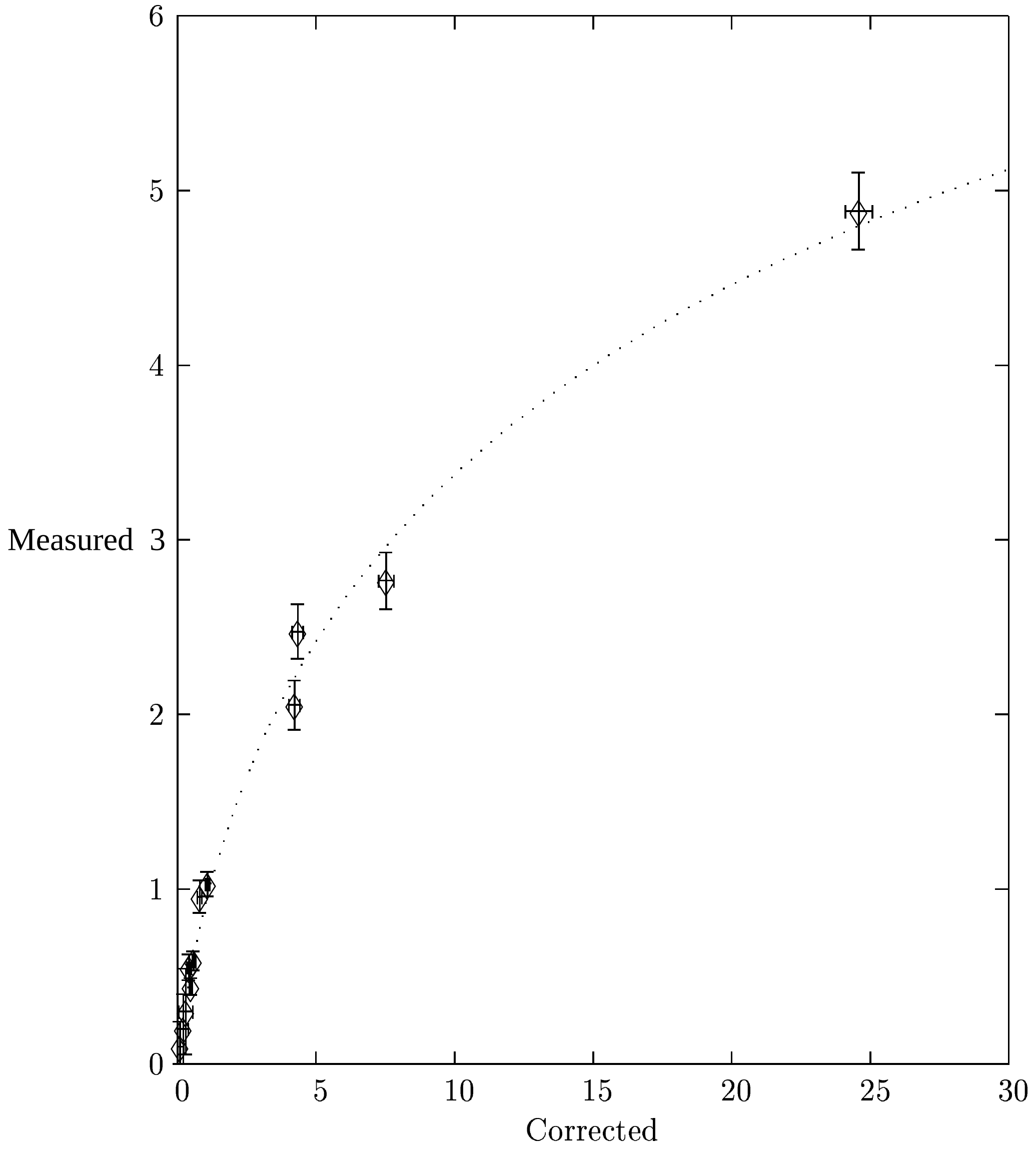


Figure 3.2: The relation between measured countrates and corrected countrates for photon counting mode. The curve represents the fitted function as described in the text.

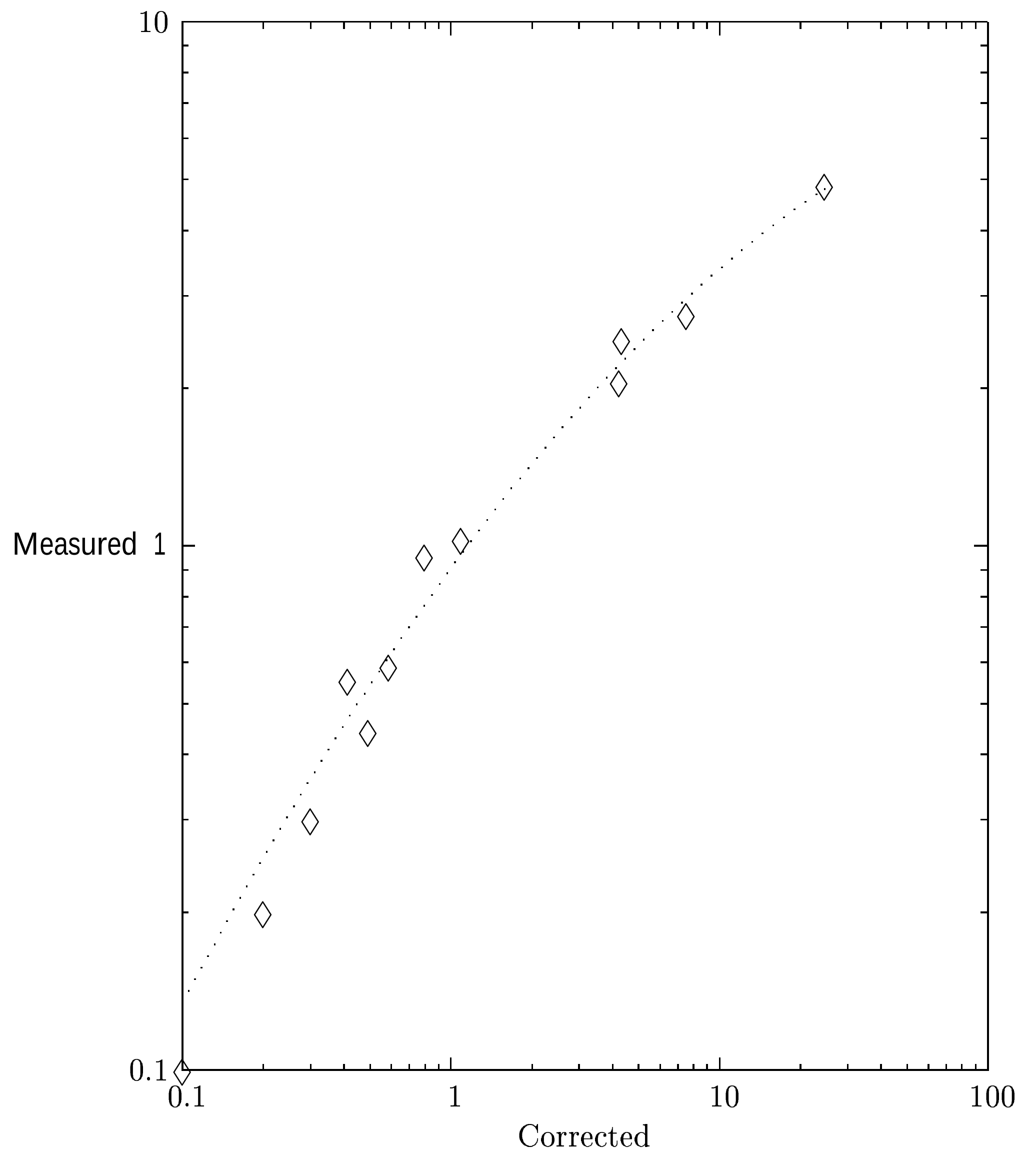


Figure 3.3: The relation between measured countrates and corrected countrates for photon counting mode shown in logspace. The curve represents the fitted function as described in the text.

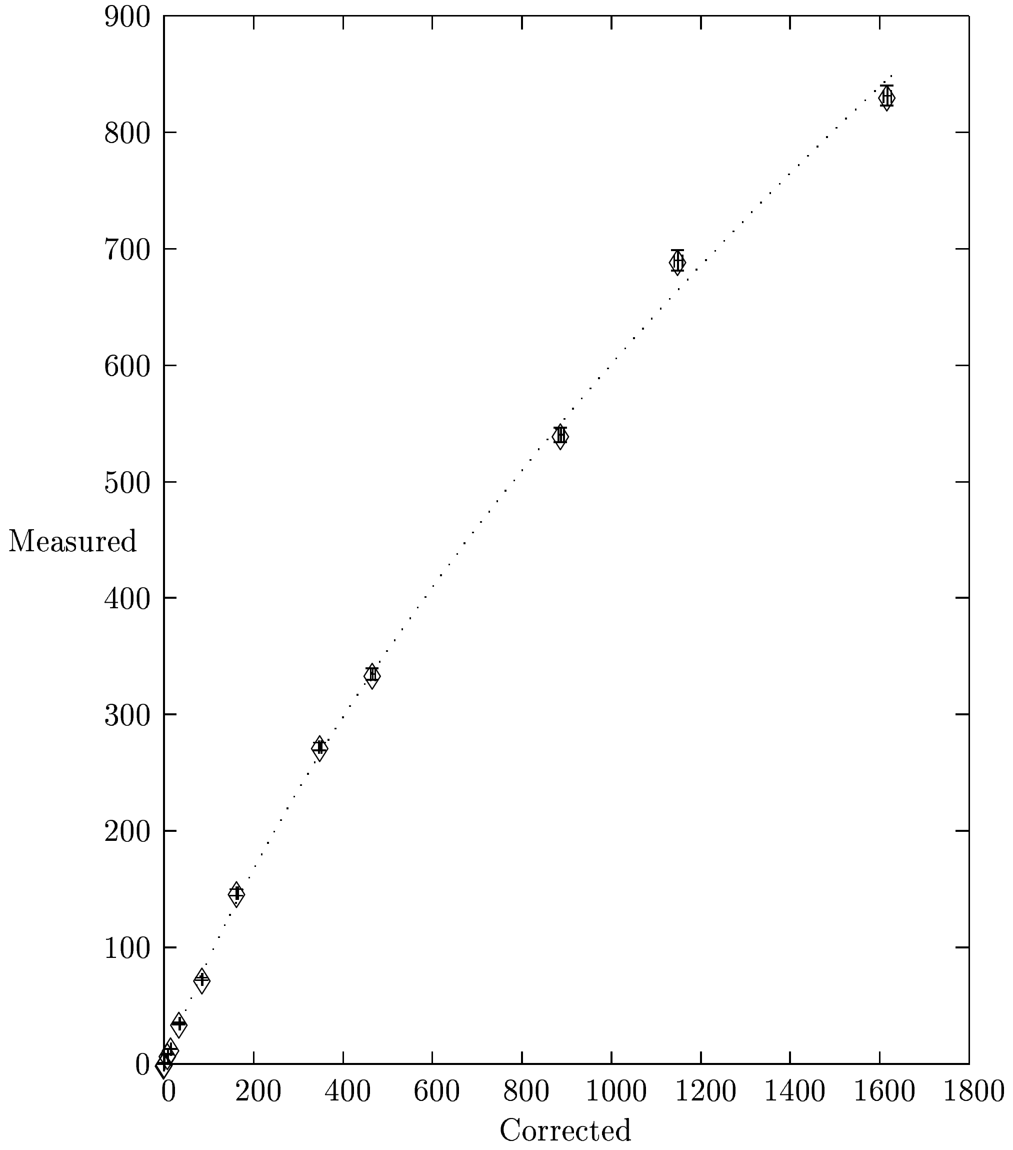


Figure 3.4: The relation between measured countrates and corrected countrates for windowed timing mode. The curve represents the fitted function as described in the text.

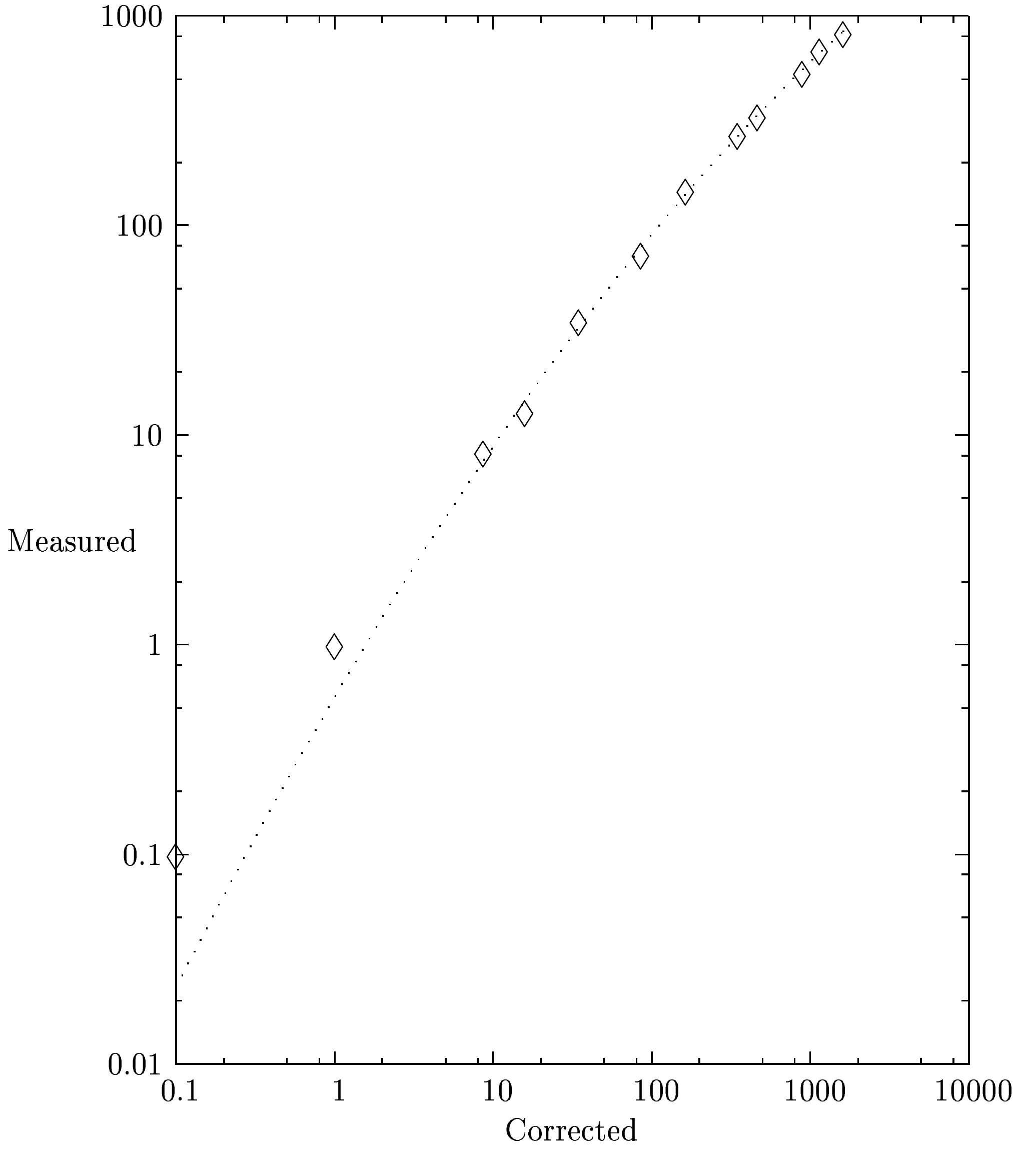


Figure 3.5: The relation between measured countrates and corrected countrates for windowed timing mode shown in logspace. The curve represents the fitted function as described in the text.

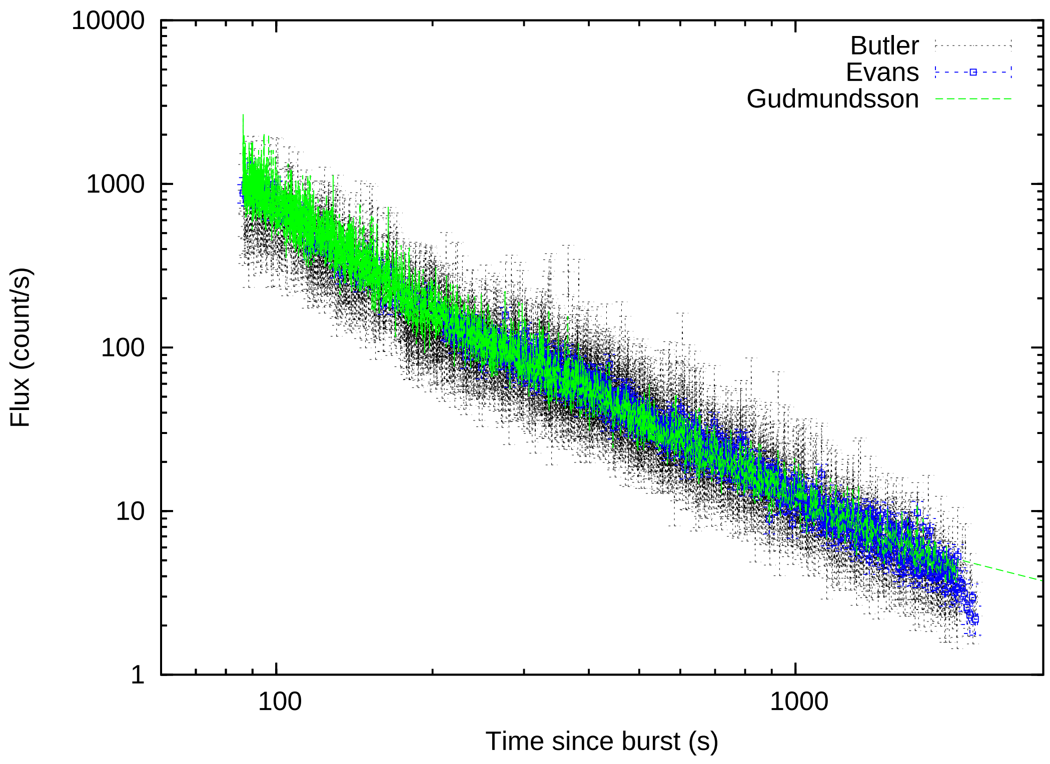


Figure 3.6: GRB061007. Here one can see how the three lightcurves are in good agreement at lower countrates. Appraching 1000 counts/s our lightcurve shows rather higher countrates than the other two.

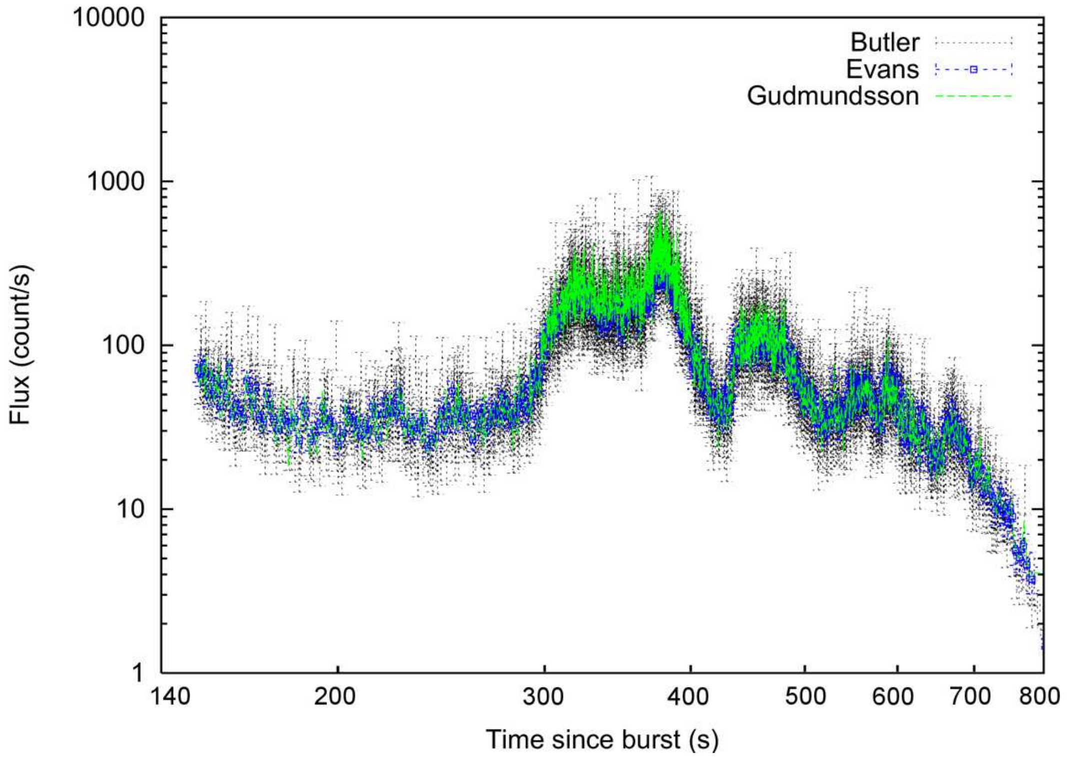


Figure 3.7: GRB070129. Here one can see how our lightcurve is in good agreement with Butler throughout but disagrees with Evans at high countrates.

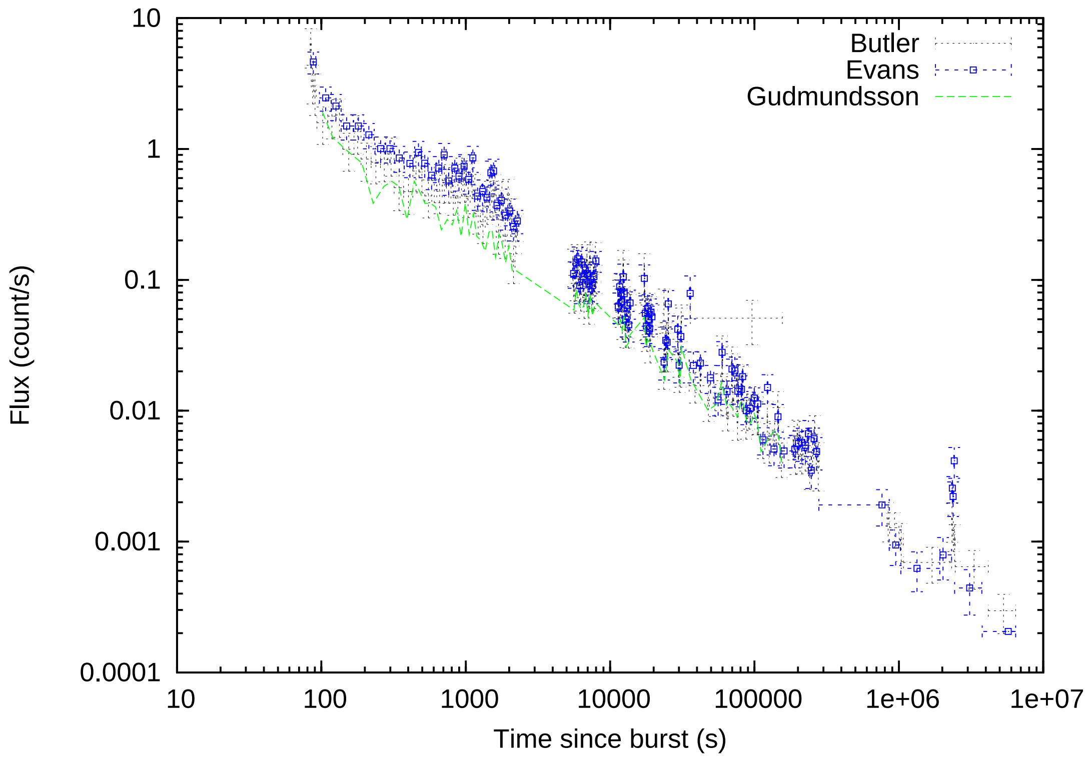


Figure 3.8: GRB050416A. Here one can see decent agreement with Butler (though rather low this time) but hardly any with Evans which is much higher.

Chapter 4

The canonical GRB afterglow lightcurve

Observations of GRBs have shown that the afterglow decay lightcurve usually follows three basic stages. These stages may not all be apparent in every lightcurve but usually one can find at least some of them fairly easily. Each stage can be described by a power-law $N \propto t^{\alpha}$, where N is the countrate and α is the decay index, and between these stages there are break points t_1 and t_2. The first stage is a very steep decay slope, the second a very flat one and the third is again steeper (but not as steep as the first stage slope). During the very steep decay of the first stage one generally finds that $-3 {>}{\sim} \alpha_1 {>}{\sim} -5$, the flatter decay slope of the second stage shows $-0.5 {>}{\sim} \alpha_2 {>}{\sim} -1.0$ and the steeper decay slope of the third stage $-1 {>}{\sim} \alpha_3 {>}{\sim} -1.5$ [7]. What has eluded researchers so far however is determining whether or not there is a clear relationship between the parameters describing these different stages of the decay and, if so, what function describes that relationship. It should be made clear here that bursts exhibiting significant X-ray flares or other strong deviations from the underlying decay process were generally excluded from this investigation to reduce the risk of confusion or error in determining the various decay rates and break times. It is often the case with X-ray flares that they obscure the structure of the lightcurve, for example if the onset of the flare happens late in a stage and the flare lasts past the next break time and well into the next stage. In such a scenario (and many similar ones) it becomes very difficult to determine the relevant parameters of the lightcurve with sufficient accuracy to be entirely reliable. Due to this kind of uncertainty the number of bursts that eventually were chosen for this investigation is very small compared to the vast number of bursts available.

The method chosen for this investigation was to look for correlation between the measured parameters of lightcurves where some or all of the stages are very clear and therefore easy to fit a power-law to. Here the fitting was done in the simplest way possible, the data was plotted in logspace and straight lines fitted to relevant parts of the lightcurves. Since the idea of performing this investigation arose when considering the possibility of observing new Swift GRBs with other satellites the problem was approached from the viewpoint of having just received notice of a new GRB along with its brightness for the first several minutes and wishing to know whether training another satellite on it (knowing roughly how long that process will take) will be worth it or whether the afterglow of the burst will by then have faded below useful brightness for the satellite in question. Therefore it was decided to look for correlation between the three main stages, mainly consecutive ones. Therefore data from bursts that clearly displayed some or all of the described stages was collected and compiled into correlation plots between (always in the order the stage or event appears in the lightcurve) α_1 and t_1, α_1 and α_2, t_1 and α_2, α_2 and t_2 and finally t_2 and α_3 (figures 4.1, 4.2, 4.3, 4.4 and 4.5). The plan was to use these plots to try to find out if there was noticable correlation between the measured decay indices and break times. The bursts chosen for this search and their relevant parameters can be seen in the following table.

Burst	α_1	t_1	α_2	t_2	α_3
GRB050505			−0.194	2460.3	−1.32
GRB050716			−0.168	175.3	−1.90
GRB050730	−2.08	176.7	−0.454	4293.0	−1.87
GRB050801			−0.316	147.7	−0.810
GRB050803	−5.63	258.2	−0.896		
GRB050922C			−0.469	116.0	−1.15
GRB060105			−1.16	6371.7	−1.90
GRB060109	−4.07	436.8	−0.0908		
GRB060202	−3.38	974.5	−0.974		
GRB060204B		180.7	−0.657		
GRB060413	−1.80	2968.7	−0.316		
GRB060502A	−3.00	272.7	−0.598		
GRB060614	−2.48	5079.2	−0.280		
GRB060729		238.4	−0.418		
GRB060813			−0.310	430.8	−1.13
GRB060814	−3.09	657.0	−0.370		
GRB061110A			−0.444	138.1	−3.87
GRB061121	−8.03	177.9	−0.0811	2018.8	−1.51
GRB061202	−6.40	335.2	−0.261		

These bursts were chosen for this investigation on the basis of the stages of decay being clearly defined and data being available from two consecutive stages (necessary to find the breaktime between them). Certainly one could choose more bursts but this would require including data that is much less clear and rather more questionable. As can be seen from the correlation plots in this chapter (figures 4.1, 4.2, 4.3, 4.4 and 4.5) this would probably not improve matters very much, simply because there does not really seem to be much correlation between the lightcurve parameters.

The correlation plots in this chapter, showing the relationships between the various parameters, are all without any plotted error bars. The reason for that is very simply that after fitting lines to slopes that are often very hard to define properly (where does the slope really start and stop for instance?, how much deviation is not just deviation but actually a small flare or break or some other actual change in the lightcurve?) and then using those fits to calculate the breaktimes in curves where every point generally has a fairly large error to begin with, any calculated error would probably be rather meaningless by this point. In attempts to fit functions to the data a 10% error was assumed. As it turned out the correlation plots did not reveal any clear connections between the slopes during the various stages and the break times. The only plot that prompted any real attempts to fit curves to it is figure 4.1 (showing what correlation there is between α_1 and t_1). For this dataset we experimented with fitting an inverse proportion function but very quickly found out that it simply did not describe the data at all. Excluding the highest two points gave marginally better results but not nearly good enough to make this pursuit worth continuing with. The other plots mostly present two problems, either there are no signs of any trends in them or they show two wildly different trends.

One can only conclude that this method of searching for a function to describe the canonical lightcurve of GRB afterglows is not particularly likely to yield any results. In fact we feel that while a very high percentage of GRBs certainly show more or less the same qualitative behavior during their decay period the whole process seems to be simply too chaotic and too dependent on many different factors (such as various qualities of the GRB progenitor and its immediate environment) to make a single function to describe the decay lightcurve attainable empirically. It seems likely that in order to predict the decay behavior it will be necessary to 'go the other way round', that is to first gain a more precise understanding of the mechanics of the prompt emission and afterglow emission as well as of any influence the surroundings of the progenitor can have on the proceedings.

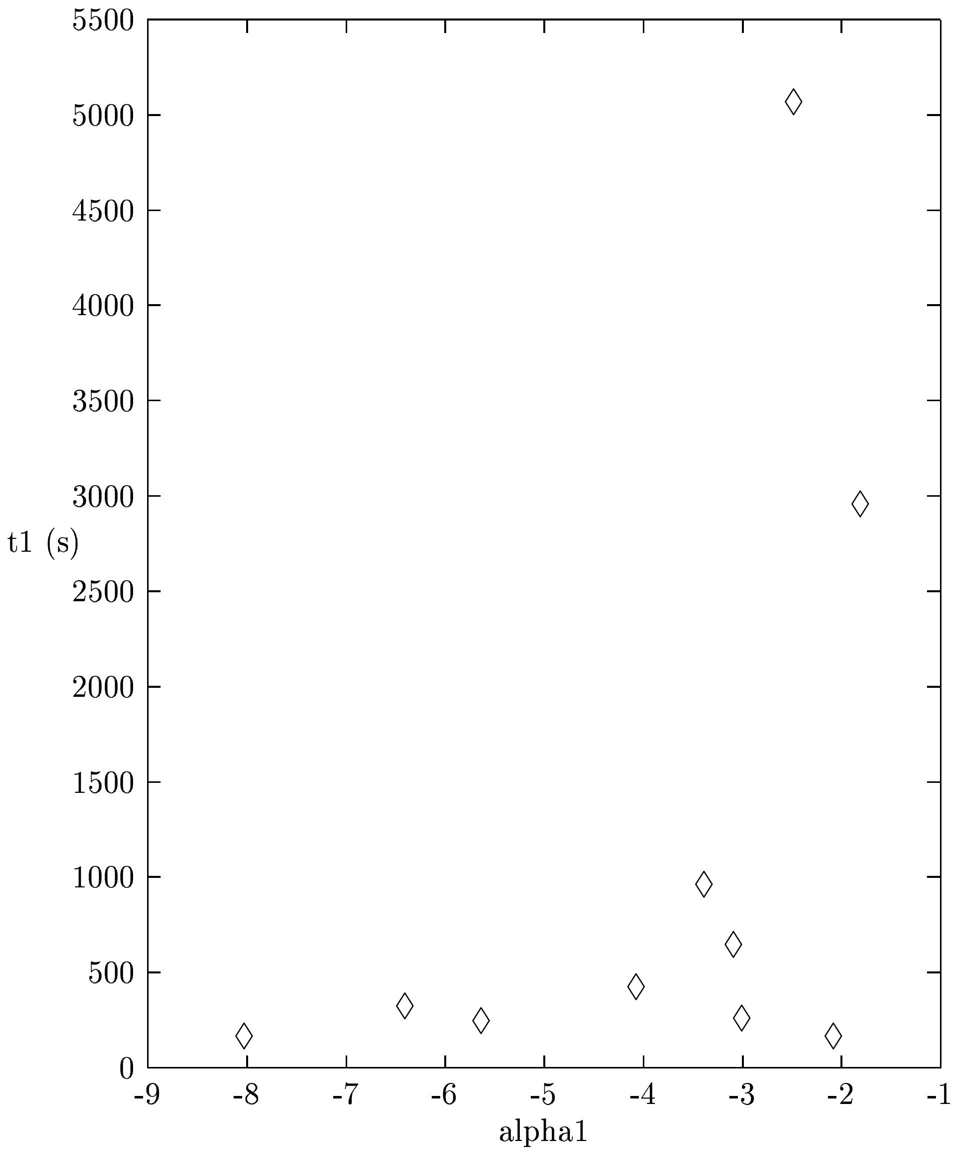


Figure 4.1: The correlation between the parameters α_1 and t_1 for the tested bursts.

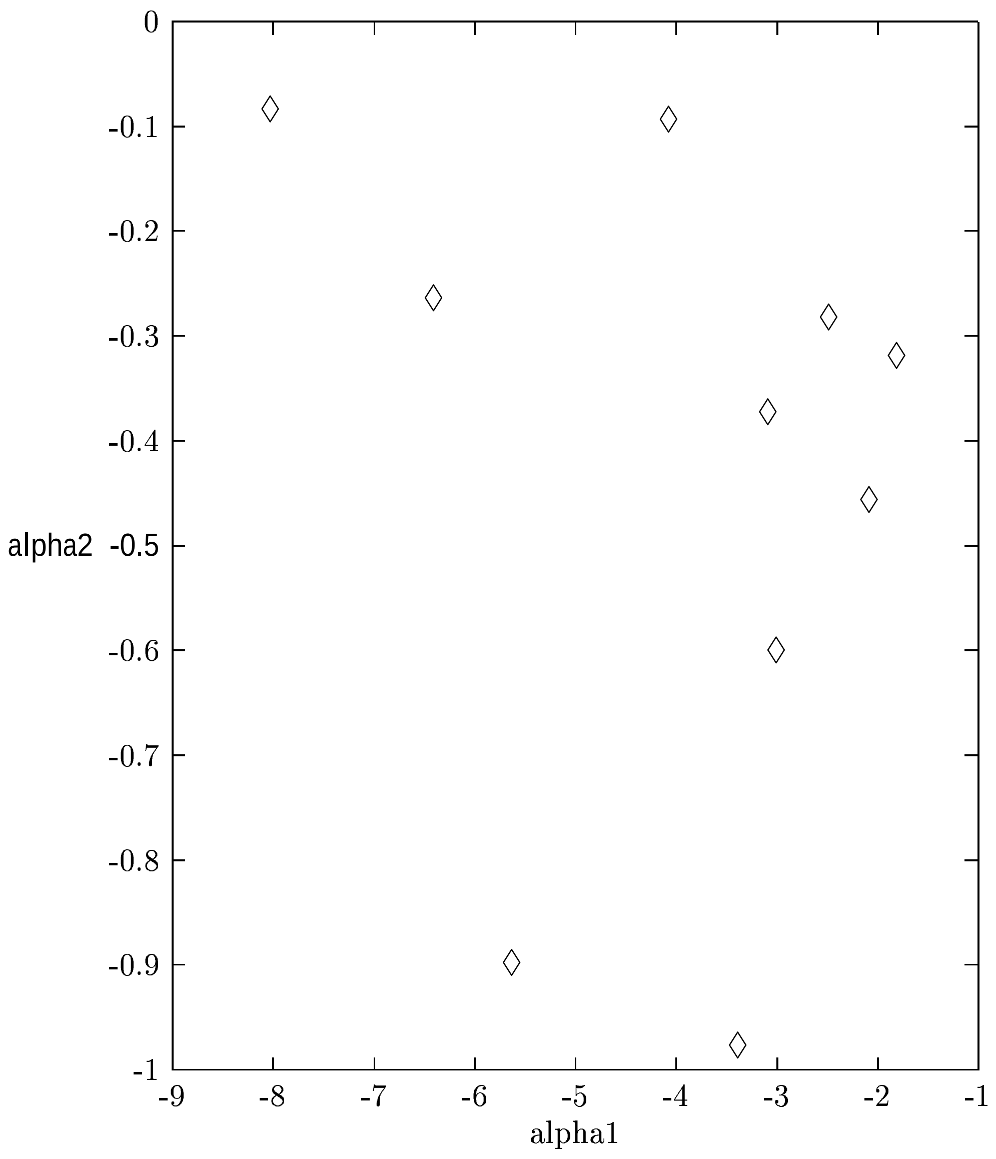


Figure 4.2: The correlation between the parameters α_1 and α_2 for the tested bursts.

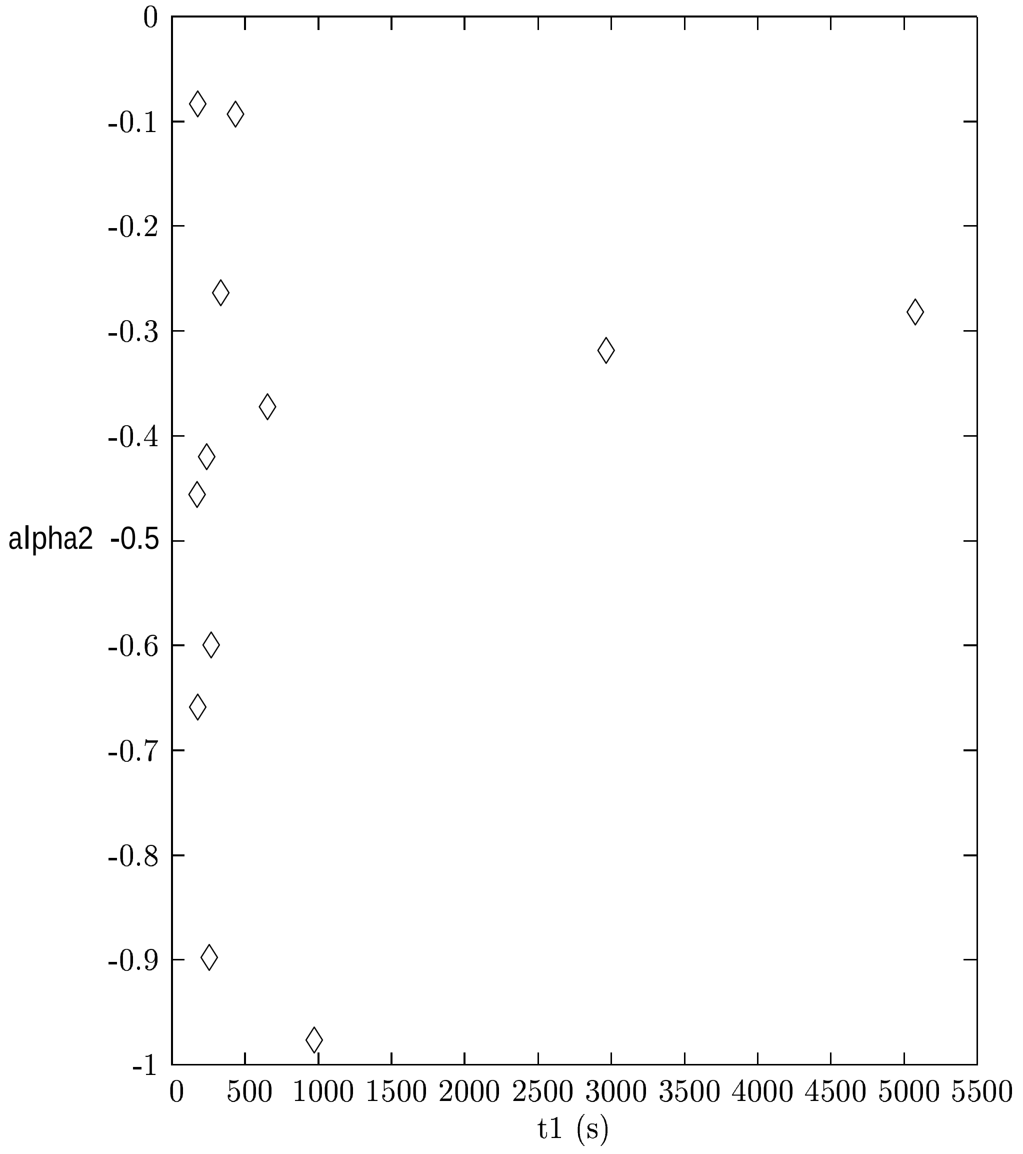


Figure 4.3: The correlation between the parameters t_1 and α_2 for the tested bursts.

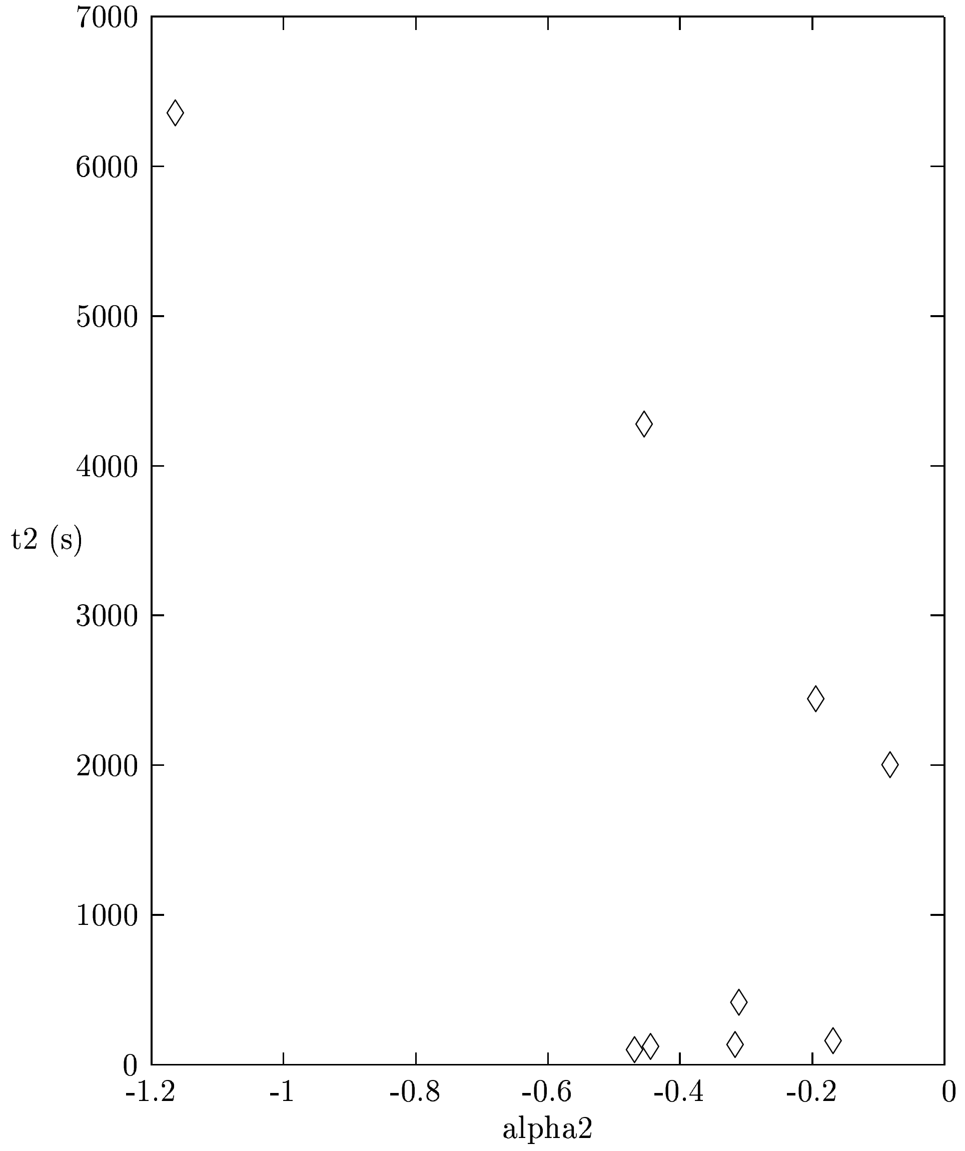


Figure 4.4: The correlation between the parameters α_2 and t_2 for the tested bursts.

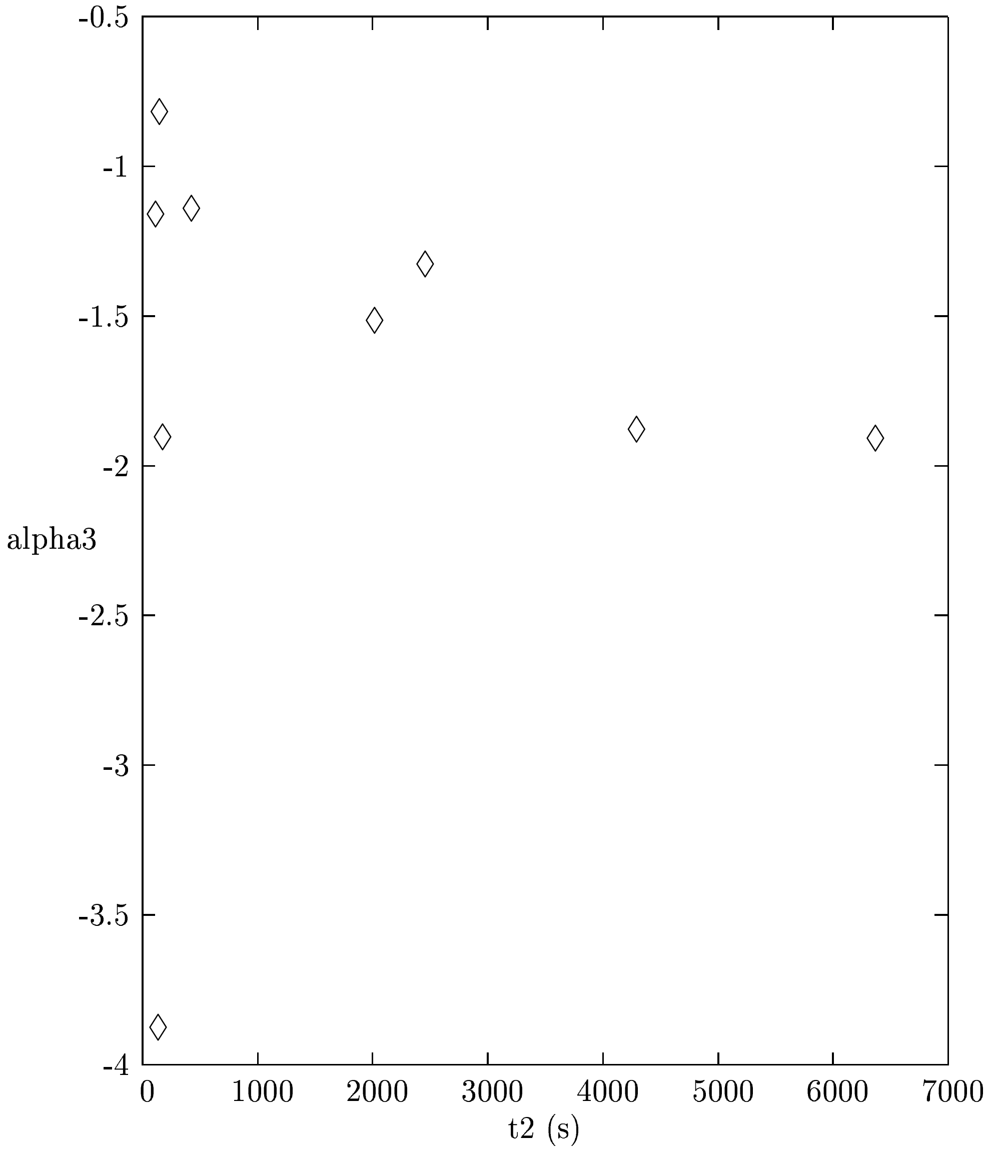


Figure 4.5: The correlation between the parameters t_2 and α_3 for the tested bursts.

Chapter 5

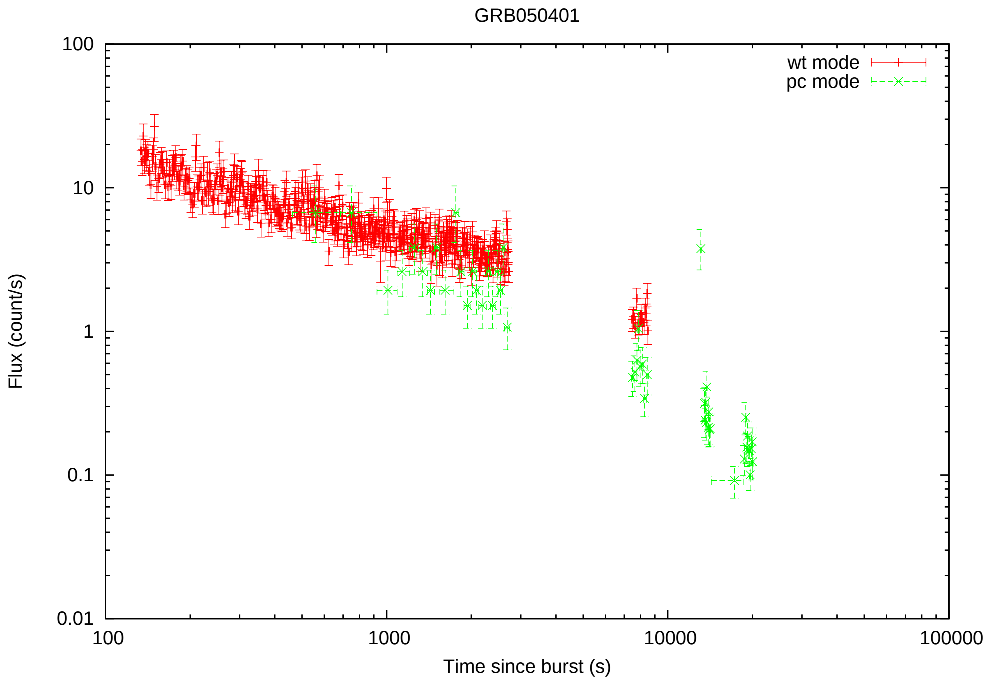
Conclusion

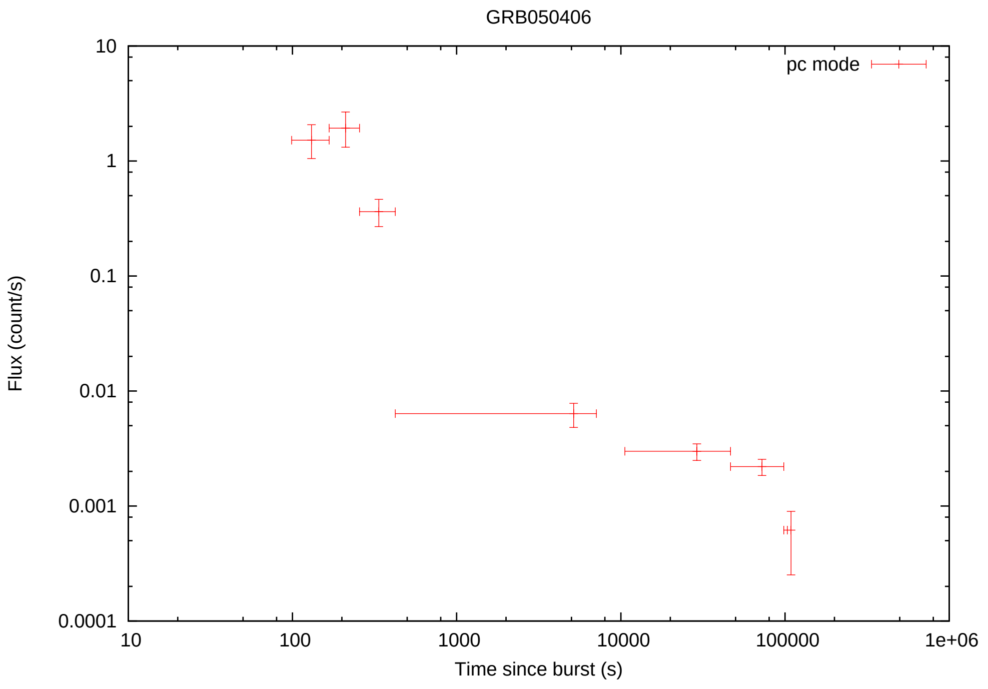
We have presented here a new method to correct Swift GRB lightcurves for pile-up in the form of one function (for each mode of operation of the Swift XRT instrument) for transforming the observed countrates into corrected ones. This function was arrived at by comparing the observed countrates to countrates corrected by using an annular extraction region to avoid pile-up and fitting a function to this dataset. When this correction became available it was used to correct almost all (a few were skipped due to insufficient data or other similar problems) Swift lightcurves dated from the beginning of April 2005 to the end of July 2007 (plots of these lightcurves are included in Appendix A). From this collection we then chose convenient bursts to use in attempting to empirically determine a function to describe the canonical GRB afterglow lightcurve by making correlation plots between the different stages of the afterglow decay. This effort did not yield positive results, in fact one must draw the conclusion that determining such a function empirically is not a particularly viable method and that it will probably be necessary to derive it from what information can be discovered about the GRB progenitor and its surroundings.

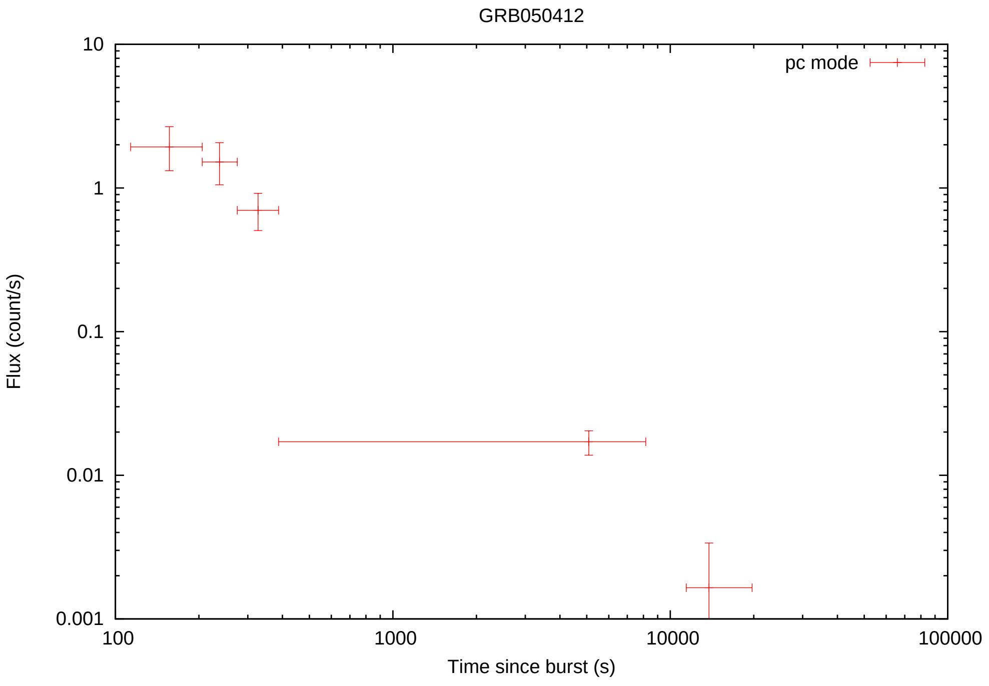
Appendix A

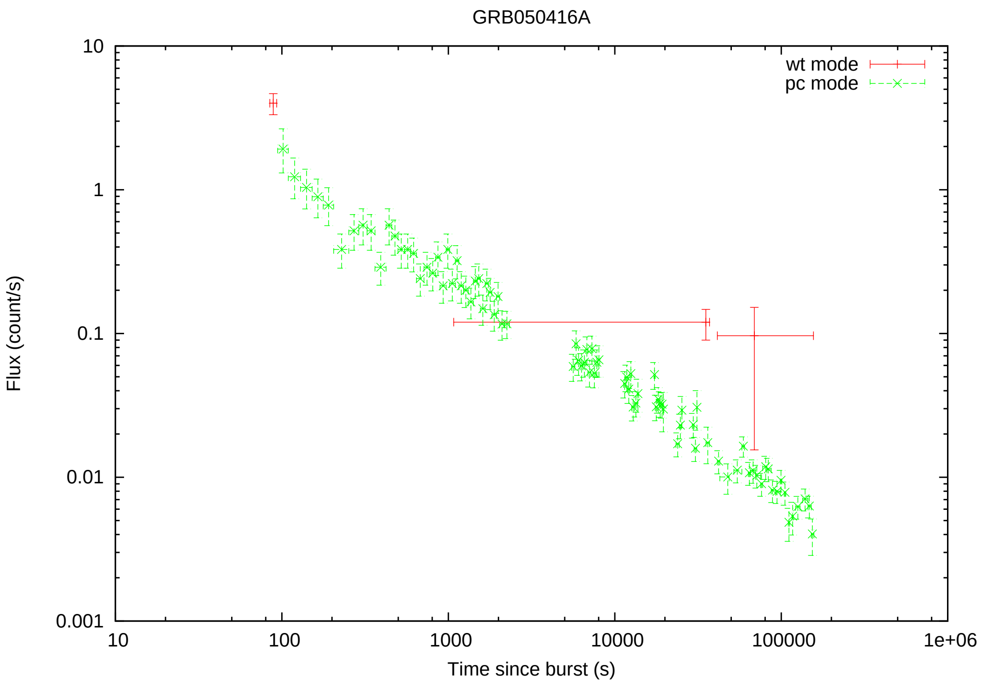
Lightcurves

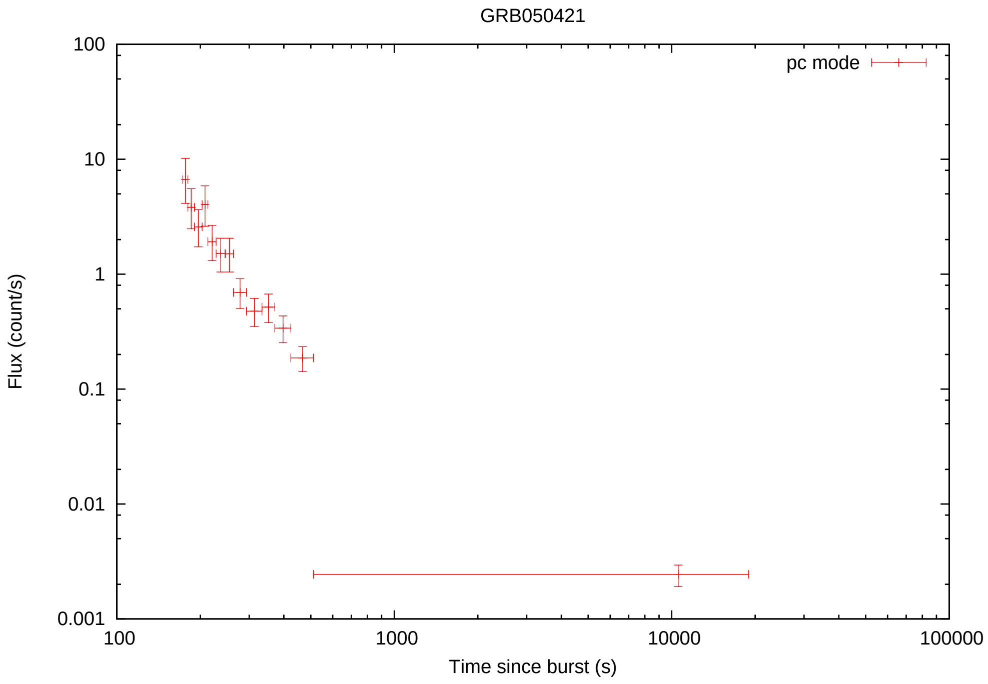
Here follow all the lightcurves generated for this work. They were made using the lightcurve extraction and plotting program written by Darach Watson with the pile-up correction presented in this paper added into the code.

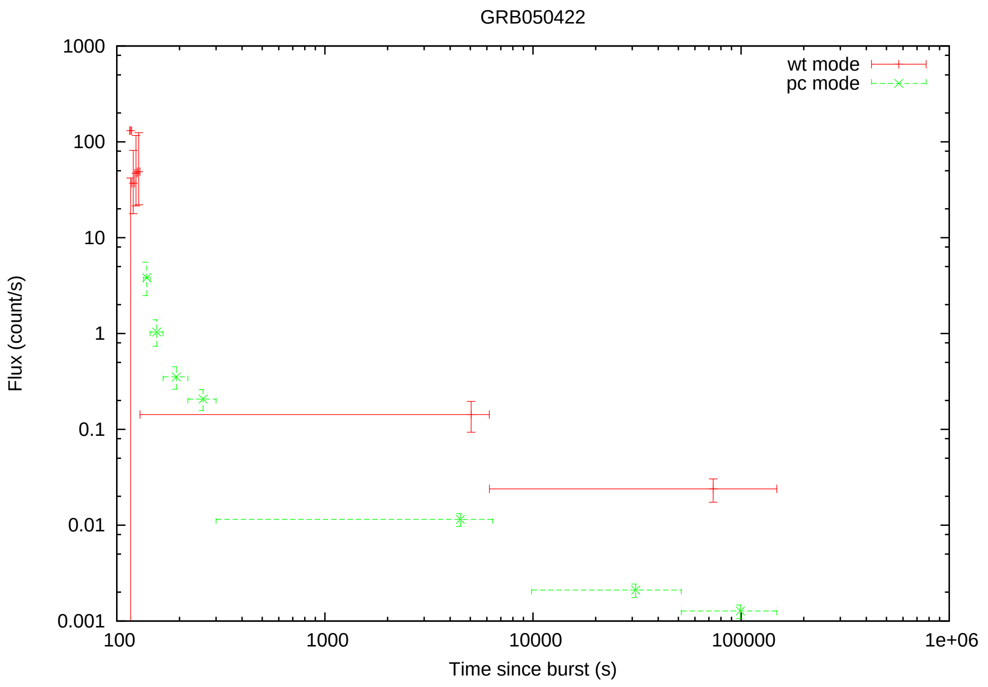

GRB050401
wt mode
pc mode
Flux (count/s)
Time since burst (s)
100
10
1
0.1
0.01
100
1000
10000
100000

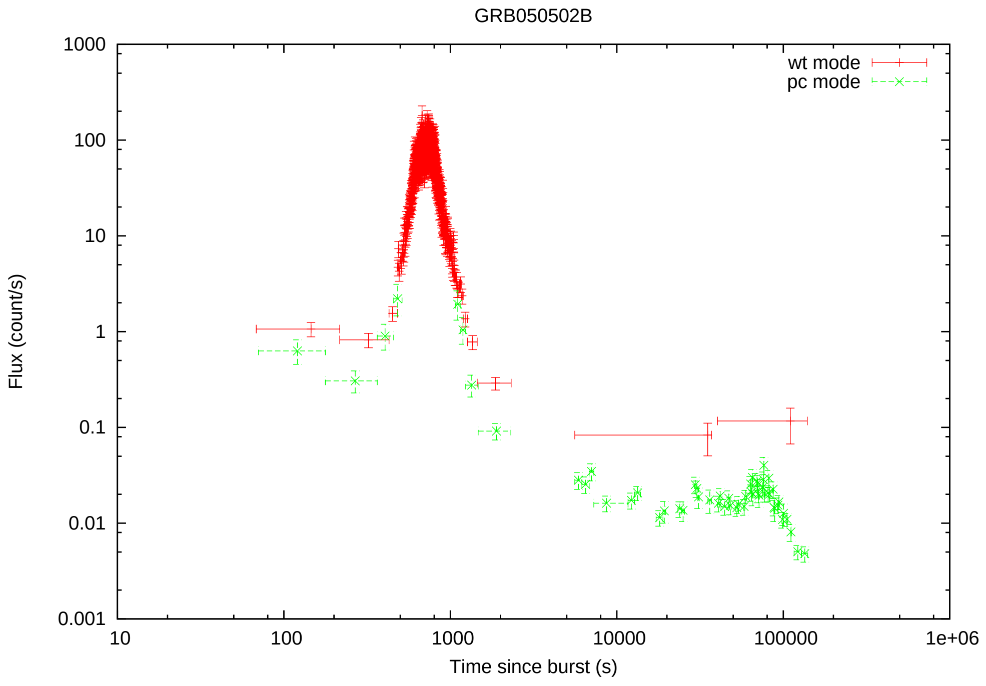

GRB050406
pc mode
Flux (count/s)
Time since burst (s)
10
1
0.1
0.01
0.001
0.0001
10
100
1000
10000
100000
1e+06

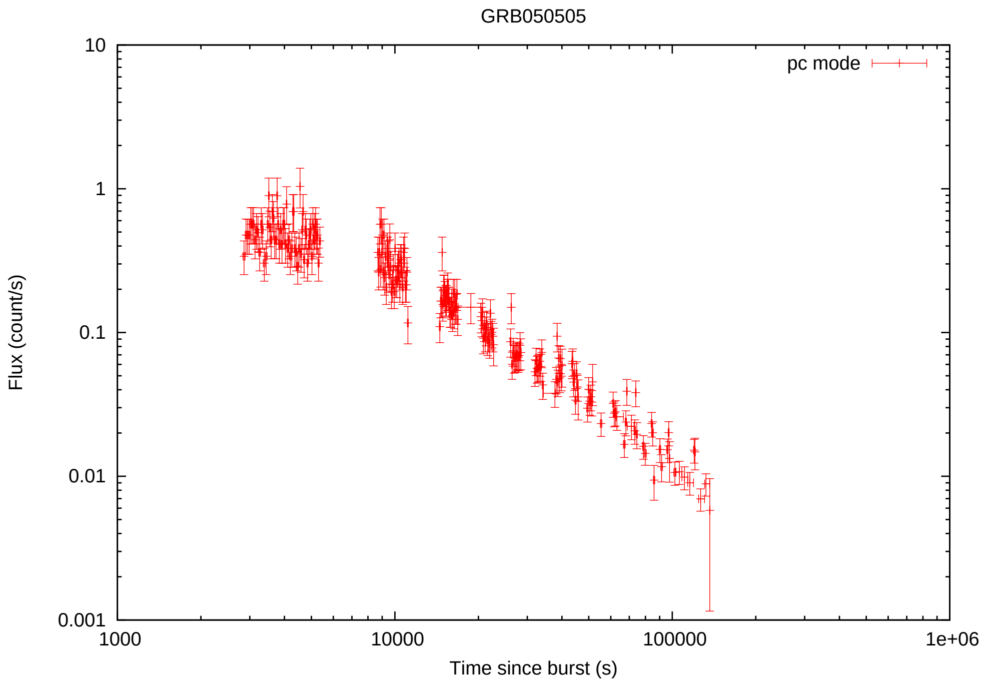

GRB050412
pc mode
Flux (count/s)
10
1
0.1
0.01
0.001
100
1000
10000
100000
Time since burst (s)

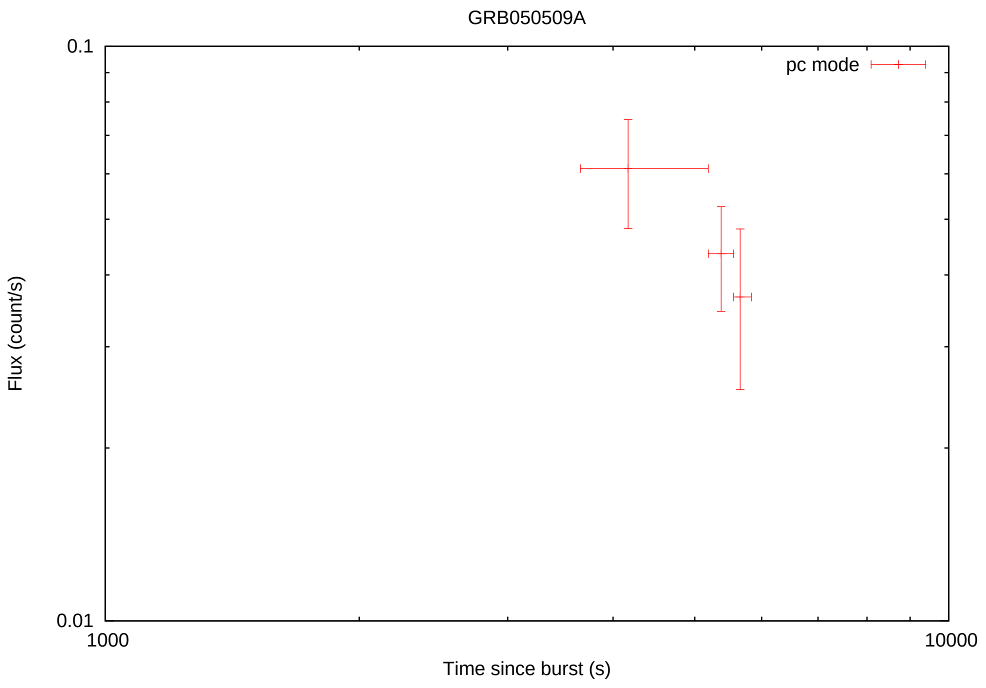

GRB050416A
wt mode
pc mode
Flux (count/s)
10
1
0.1
0.01
0.001
10
100
1000
10000
100000
1e+06
Time since burst (s)

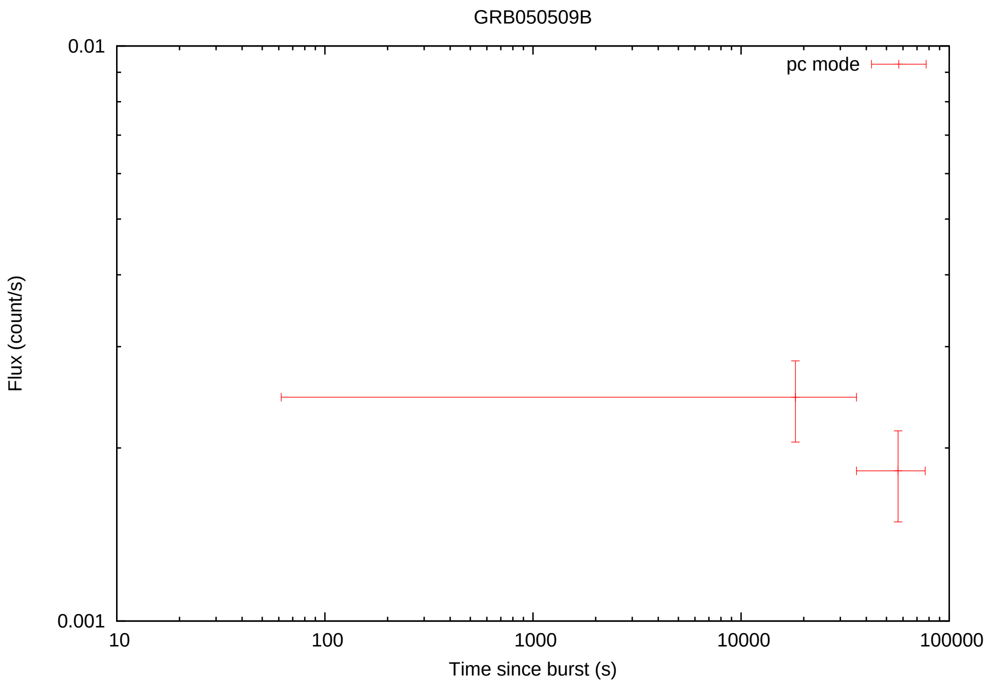

GRB050421
pc mode
Flux (count/s)
Time since burst (s)
100
10
1
0.1
0.01
0.001
100
1000
10000
100000

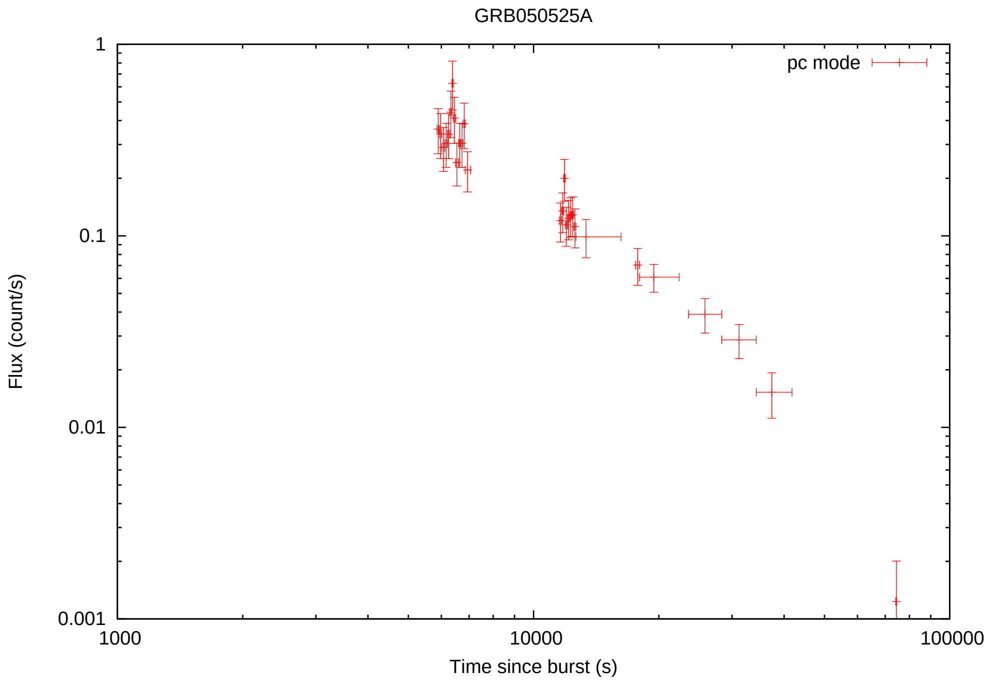

GRB050422
wt mode
pc mode
Flux (count/s)
Time since burst (s)
1000
100
10
1
0.1
0.01
0.001
100
1000
10000
100000
1e+06

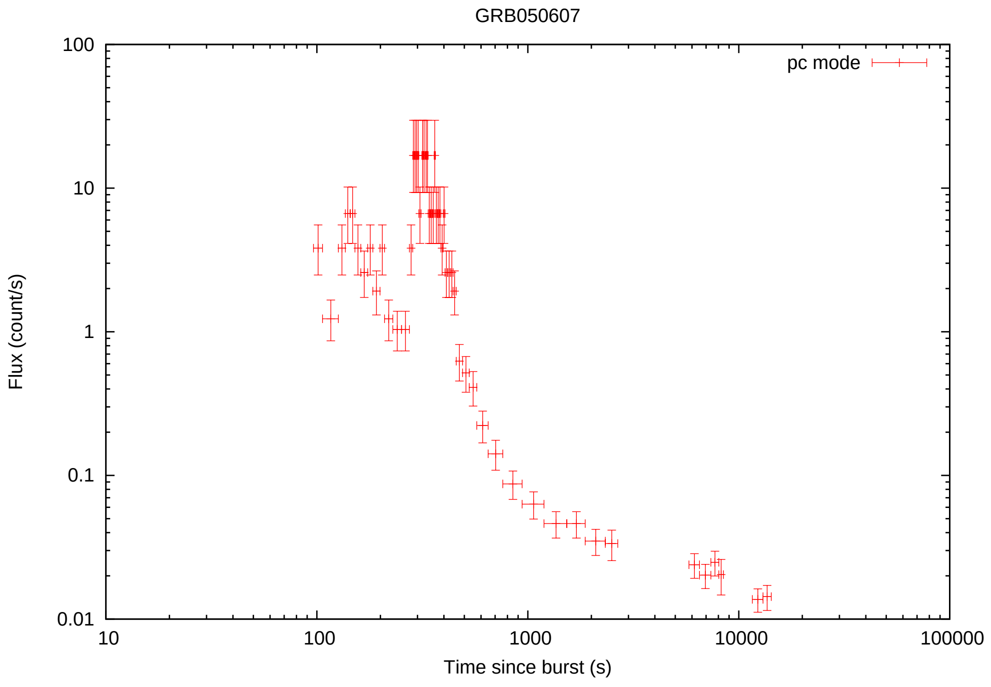

GRB050502B
wt mode
pc mode
Flux (count/s)
Time since burst (s)
1000
100
10
1
0.1
0.01
0.001
10
100
1000
10000
100000
1e+06

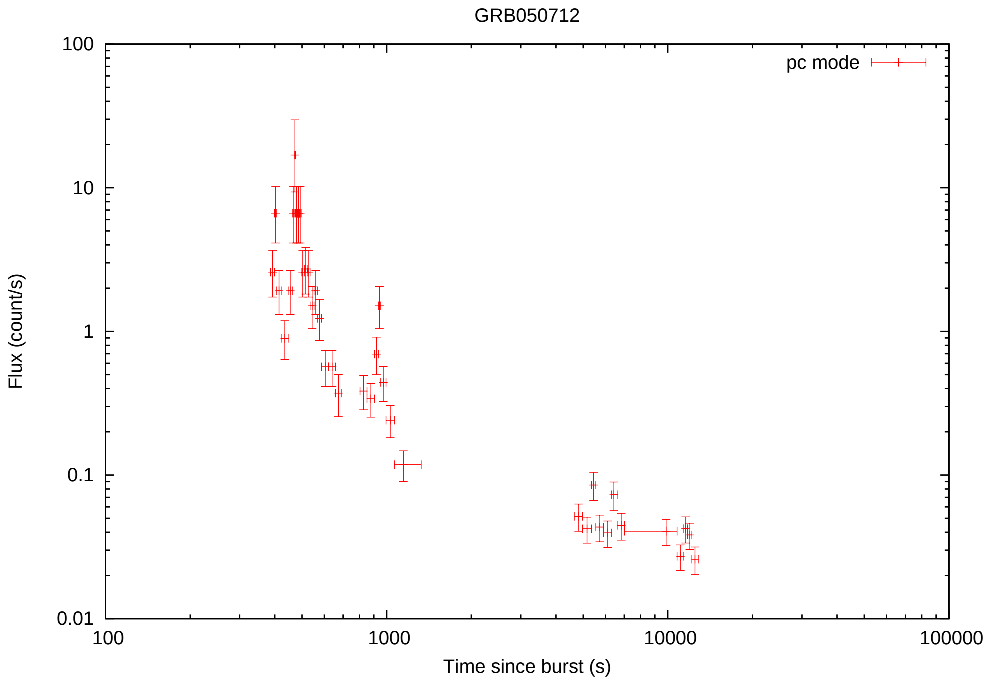

GRB050505
pc mode
Flux (count/s)
Time since burst (s)
10
1
0.1
0.01
0.001
1000
10000
100000
1e+06

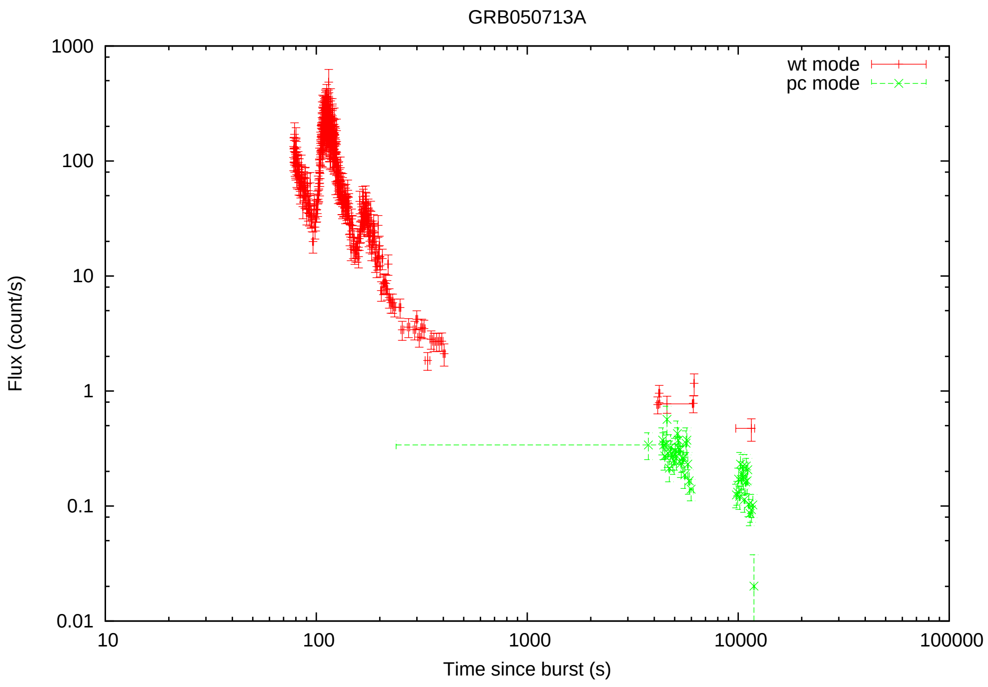

GRB050509A
pc mode
0.1
0.01
1000
10000
Flux (count/s)
Time since burst (s)

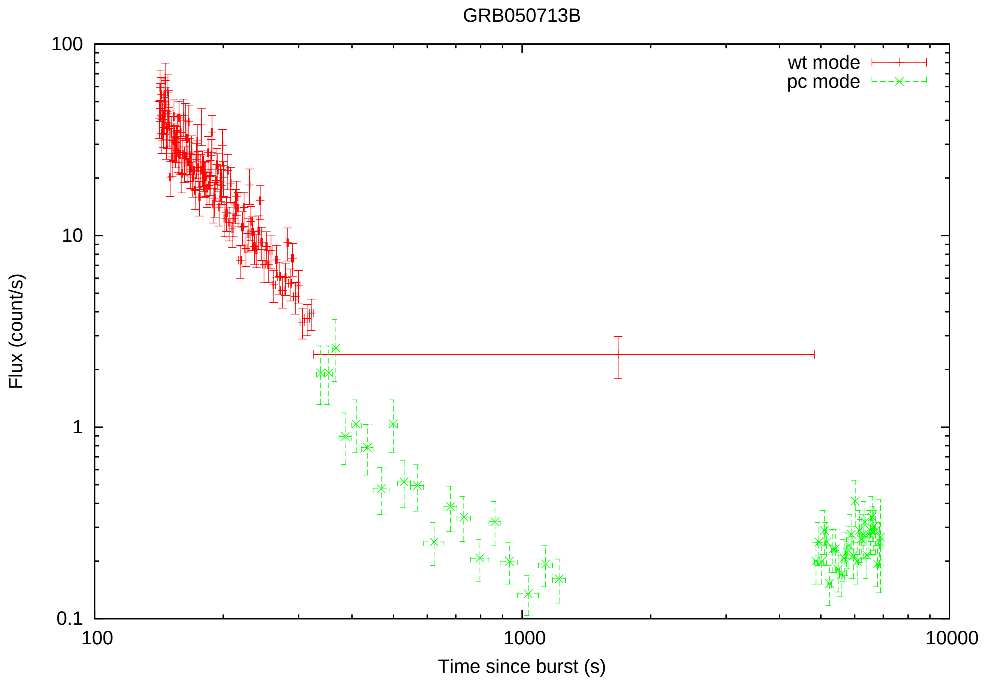

GRB050509B
pc mode
0.01
0.001
10
100
1000
10000
100000
Flux (count/s)
Time since burst (s)

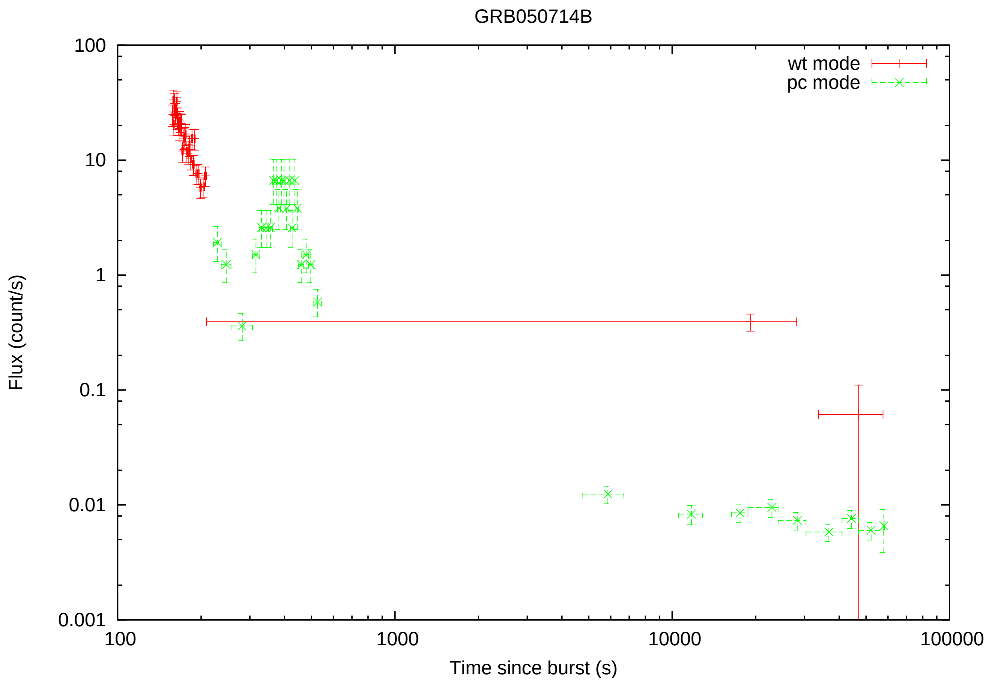

GRB050525A
pc mode
Flux (count/s)
1
0.1
0.01
0.001
1000
10000
100000
Time since burst (s)

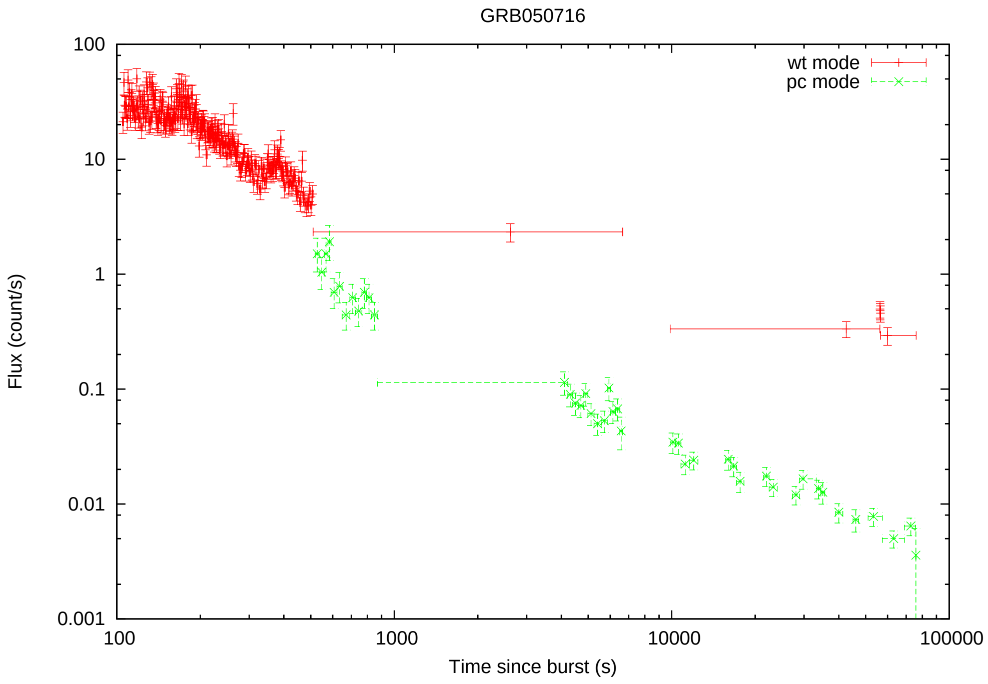

GRB050607
pc mode
Flux (count/s)
100
10
1
0.1
0.01
10
100
1000
10000
100000
Time since burst (s)

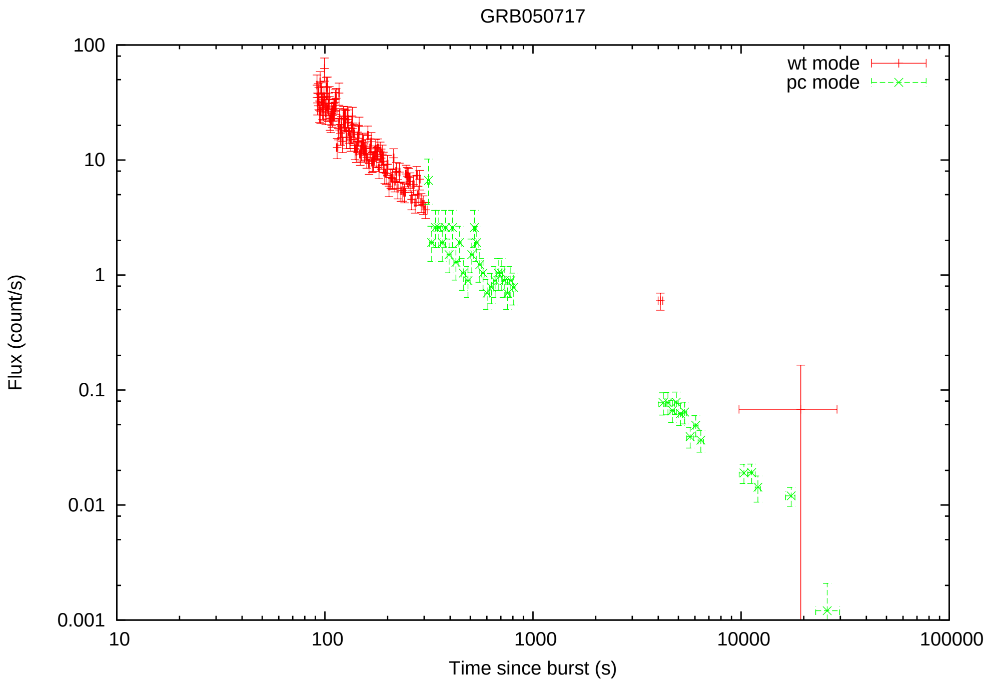

GRB050712
pc mode
Flux (count/s)
Time since burst (s)
100
10
1
0.1
0.01
100
1000
10000
100000

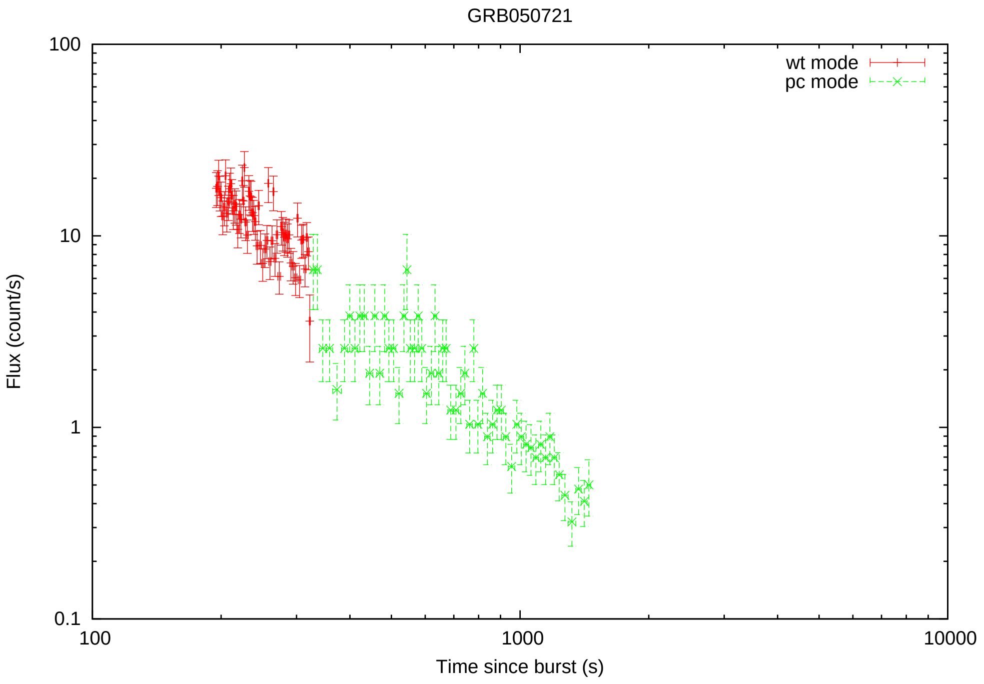

GRB050713A
wt mode
pc mode
Flux (count/s)
Time since burst (s)
1000
100
10
1
0.1
0.01
10
100
1000
10000
100000

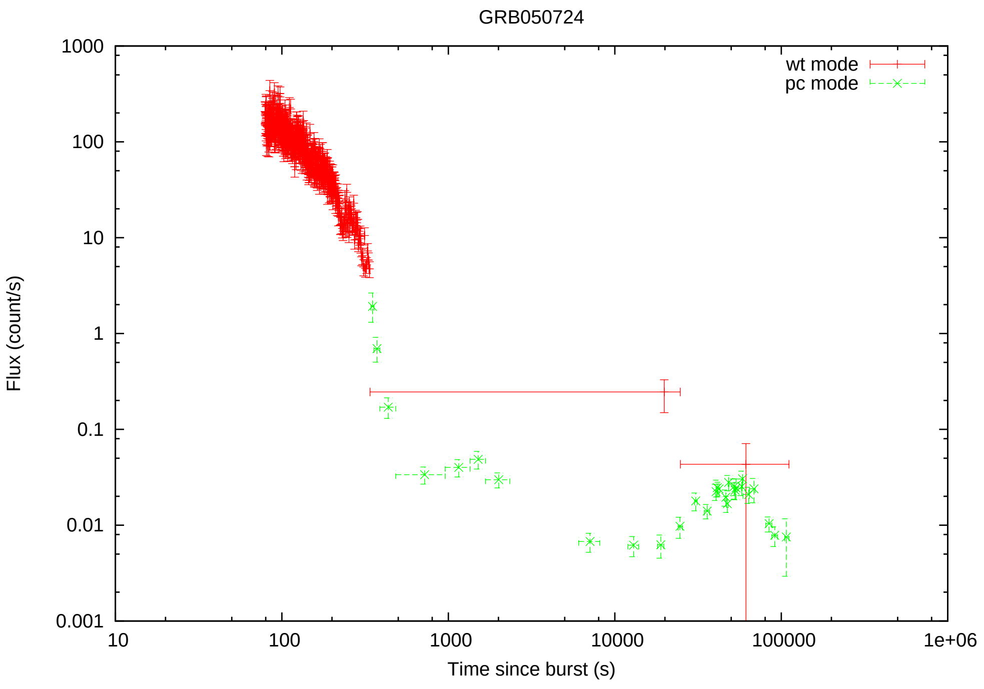

GRB050713B
wt mode
pc mode
Flux (count/s)
Time since burst (s)
100
10
1
0.1
100
1000
10000

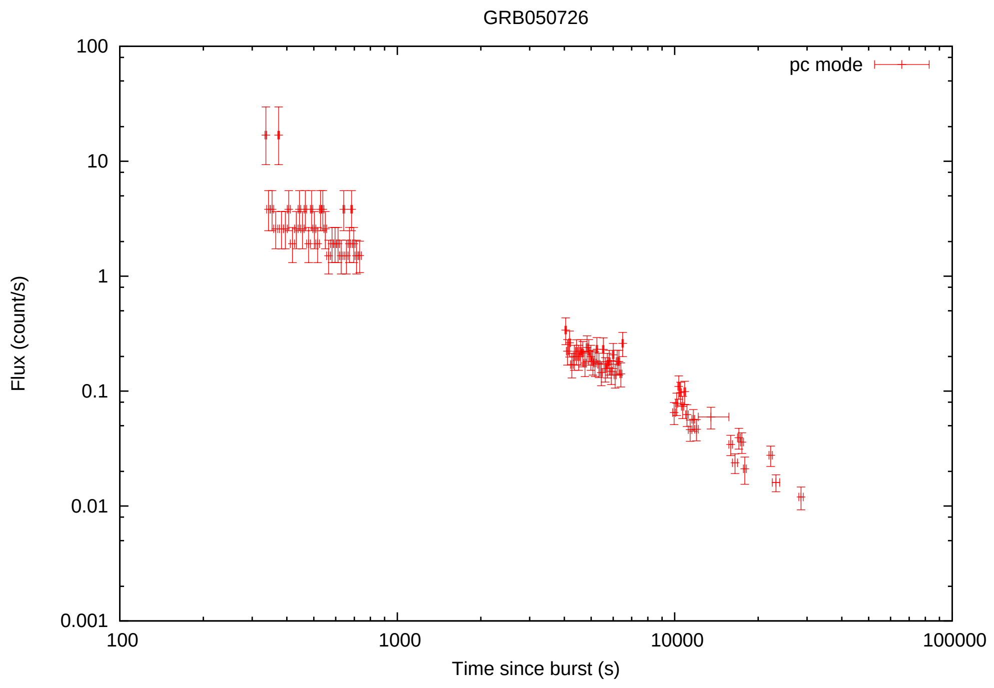

GRB050714B
wt mode
pc mode
Flux (count/s)
Time since burst (s)
100
10
1
0.1
0.01
0.001
100
1000
10000
100000

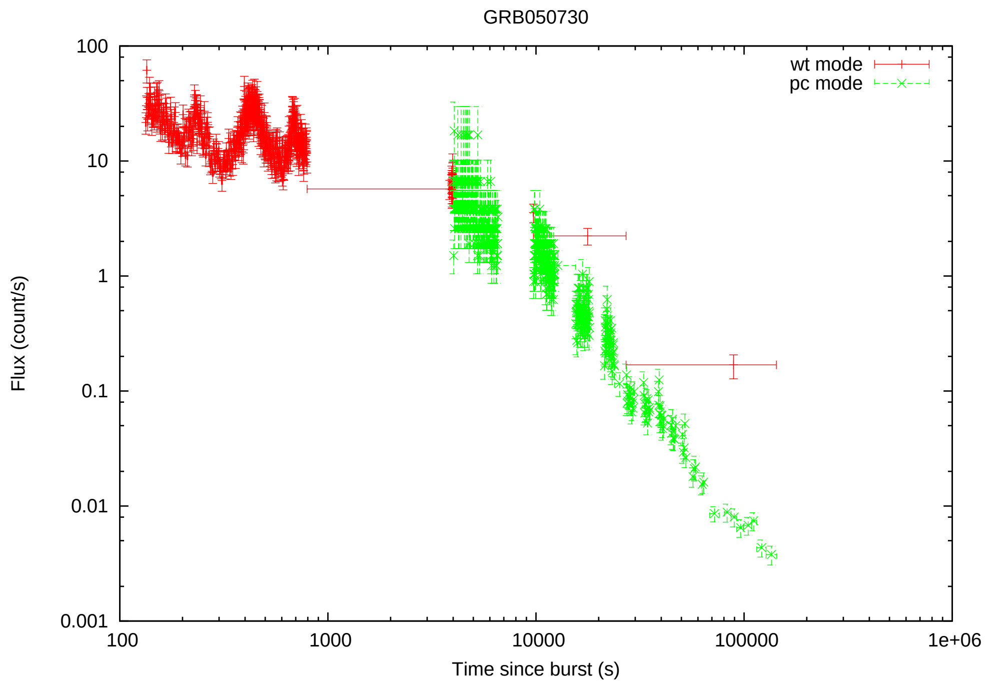

GRB050716
Flux (count/s)
Time since burst (s)
wt mode
pc mode
100
10
1
0.1
0.01
0.001
100
1000
10000
100000

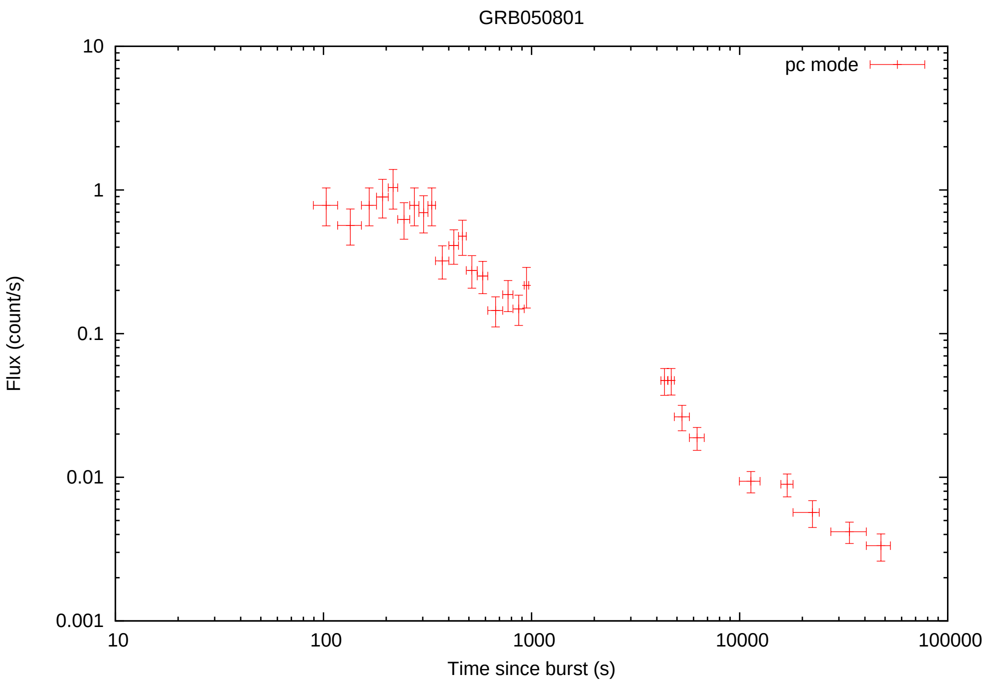

GRB050717
Flux (count/s)
Time since burst (s)
wt mode
pc mode
100
10
1
0.1
0.01
0.001
10
100
1000
10000
100000

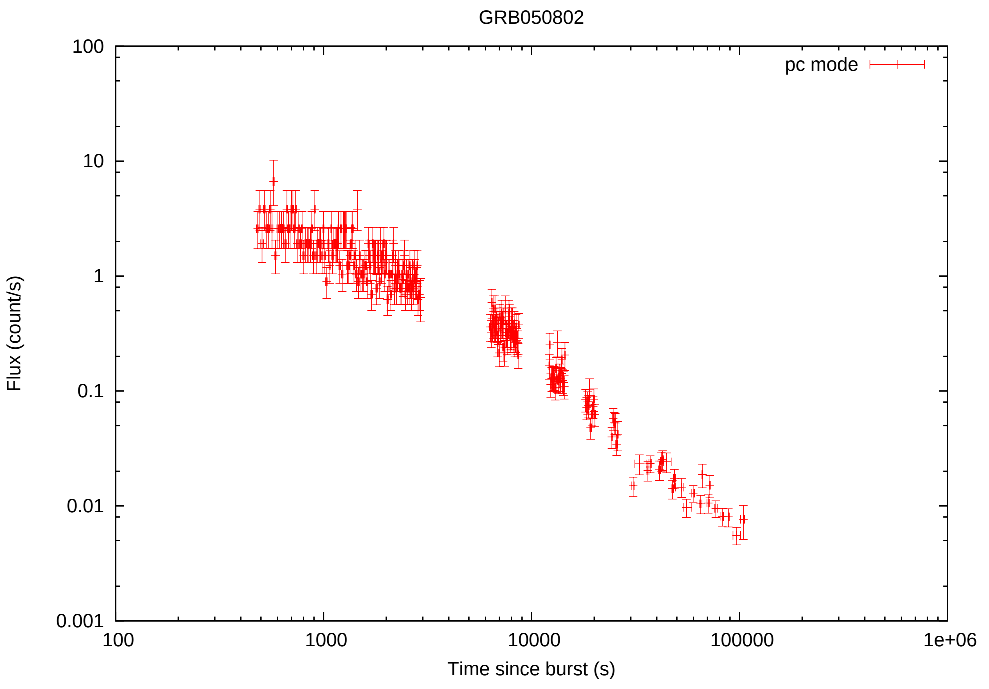

GRB050721
wt mode
pc mode
Flux (count/s)
Time since burst (s)
100
10
1
0.1
100
1000
10000

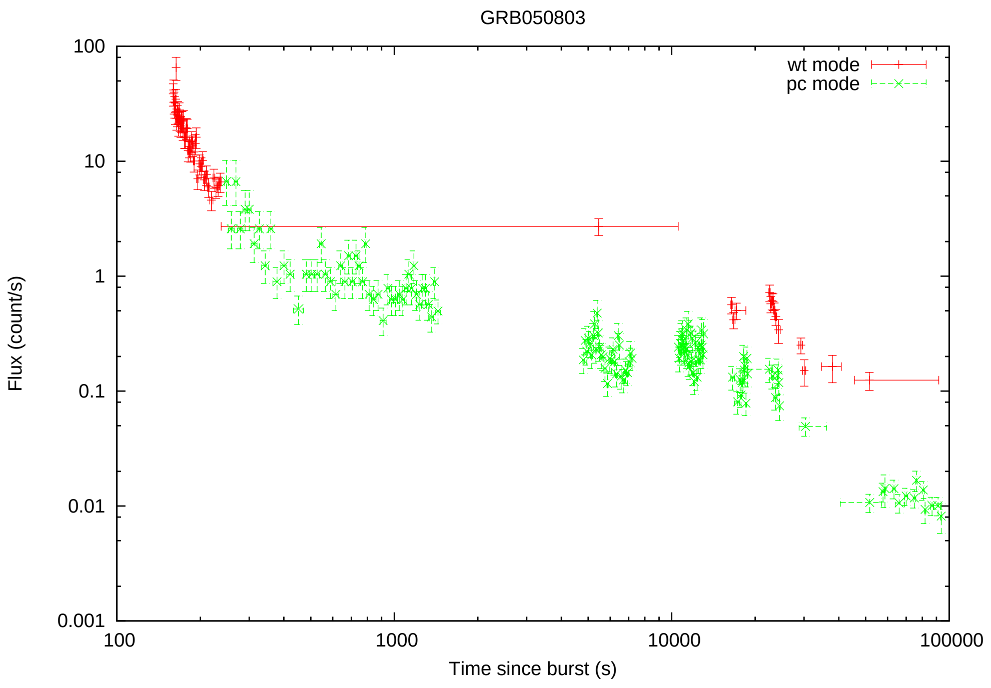

GRB050724
wt mode
pc mode
Flux (count/s)
Time since burst (s)
1000
100
10
1
0.1
0.01
0.001
10
100
1000
10000
100000
1e+06

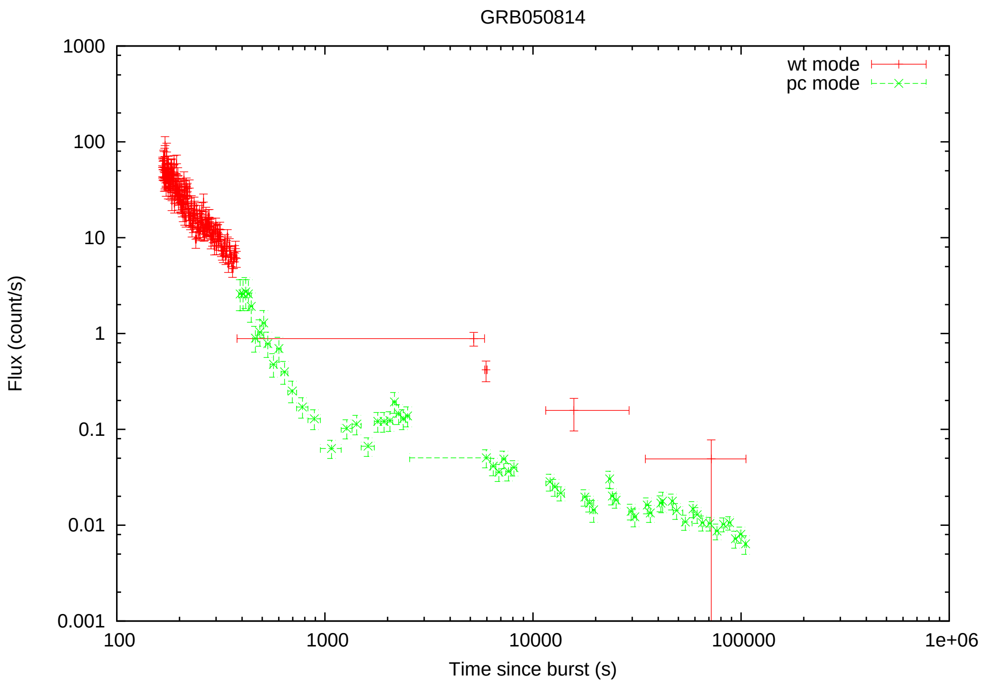

GRB050726
pc mode
Flux (count/s)
100
10
1
0.1
0.01
0.001
100
1000
10000
100000
Time since burst (s)

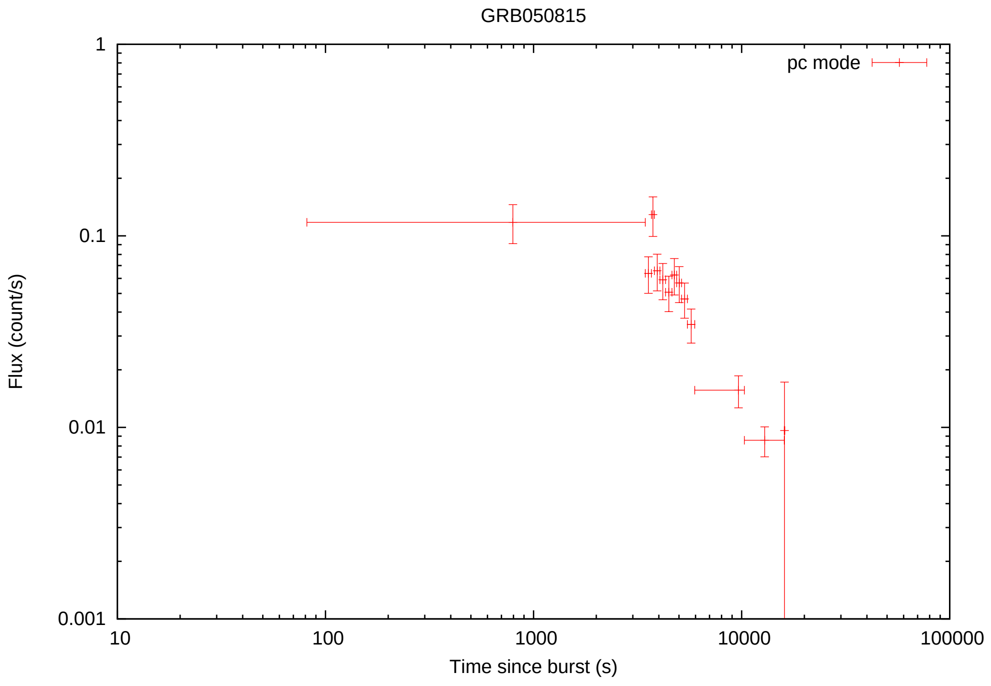

GRB050730
wt mode
pc mode
Flux (count/s)
100
10
1
0.1
0.01
0.001
100
1000
10000
100000
1e+06
Time since burst (s)

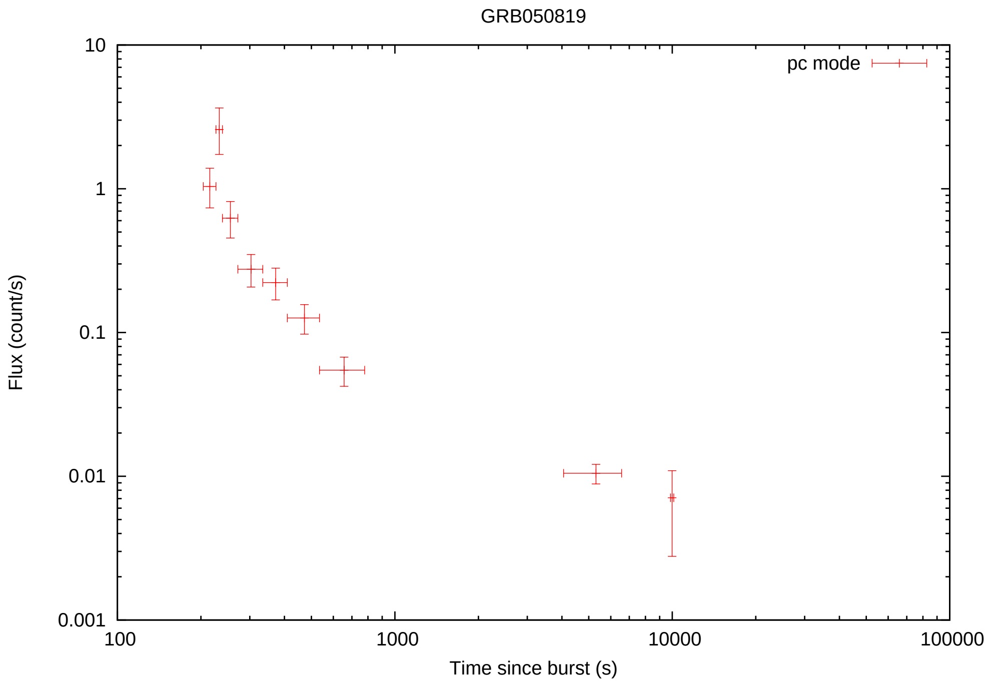

GRB050801
pc mode
Flux (count/s)
Time since burst (s)
10
1
0.1
0.01
0.001
10
100
1000
10000
100000

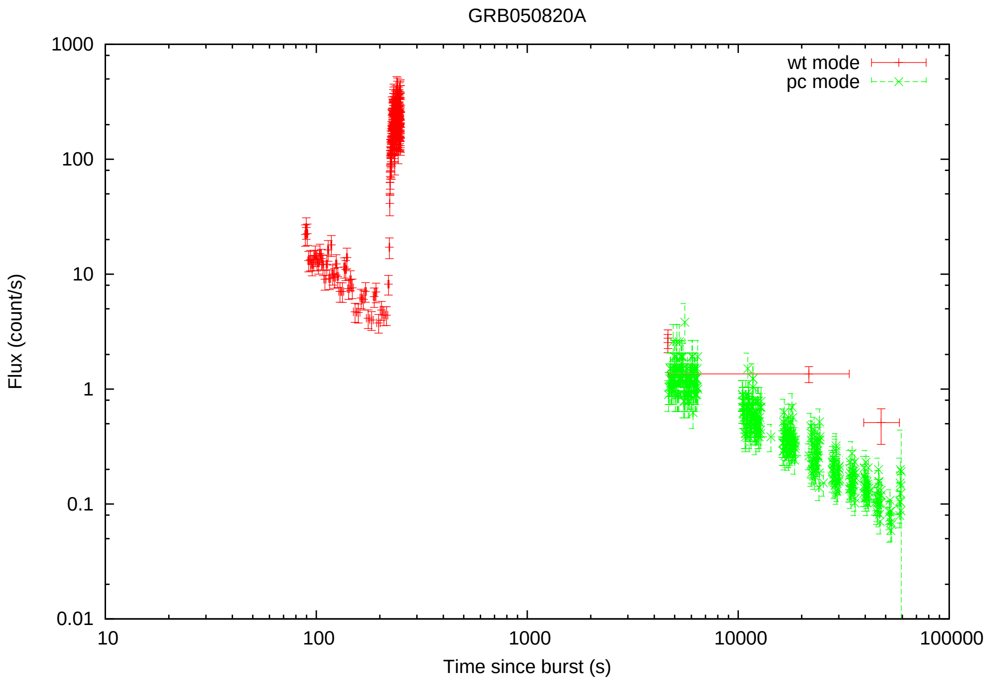

GRB050802
pc mode
Flux (count/s)
Time since burst (s)
100
10
1
0.1
0.01
0.001
100
1000
10000
100000
1e+06

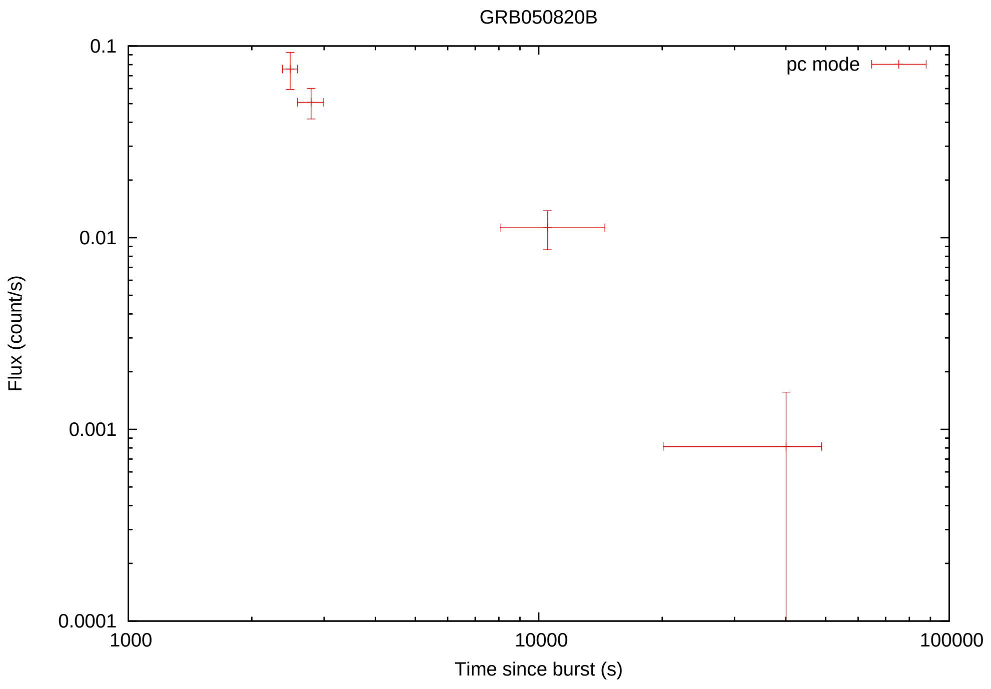

GRB050803
wt mode
pc mode
Flux (count/s)
Time since burst (s)
100
10
1
0.1
0.01
0.001
100
1000
10000
100000

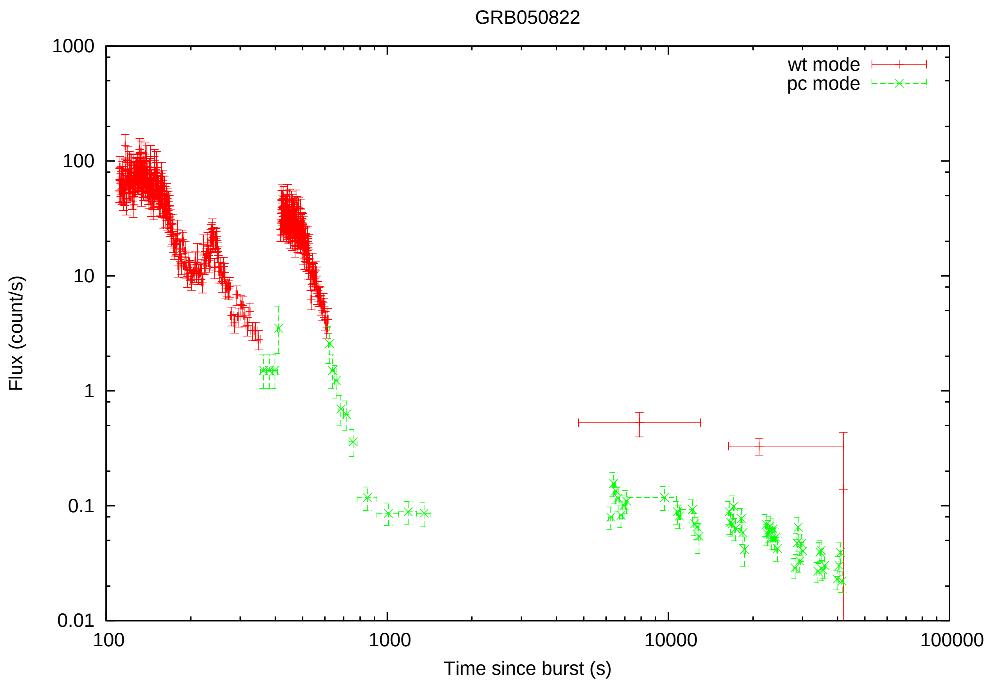

GRB050814
wt mode
pc mode
Flux (count/s)
Time since burst (s)
1000
100
10
1
0.1
0.01
0.001
100
1000
10000
100000
1e+06

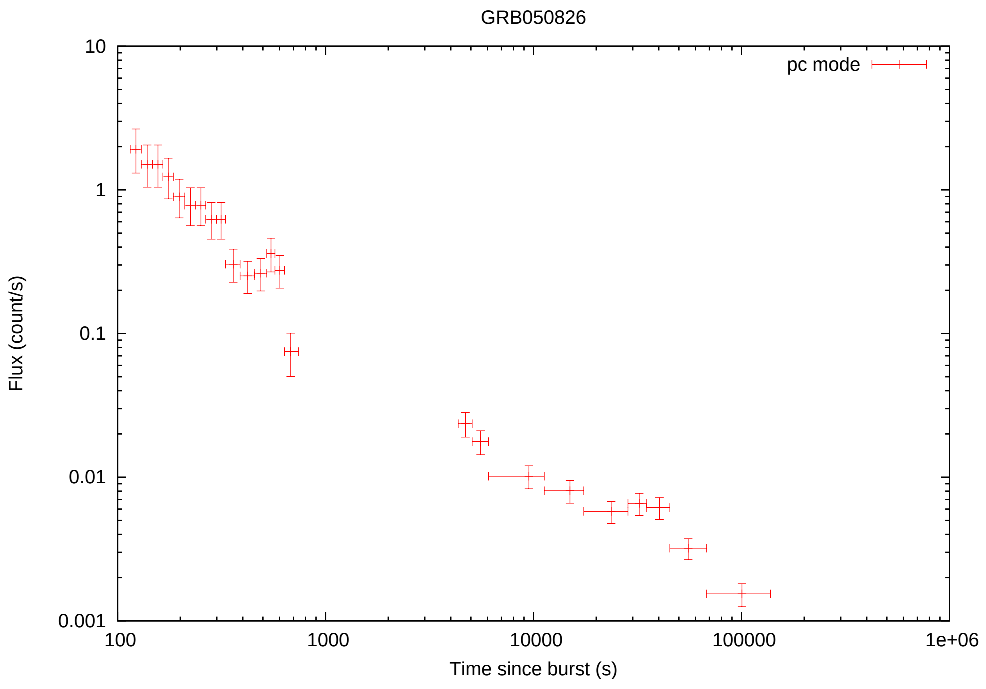

GRB050815
pc mode
Flux (count/s)
1
0.1
0.01
0.001
10
100
1000
10000
100000
Time since burst (s)

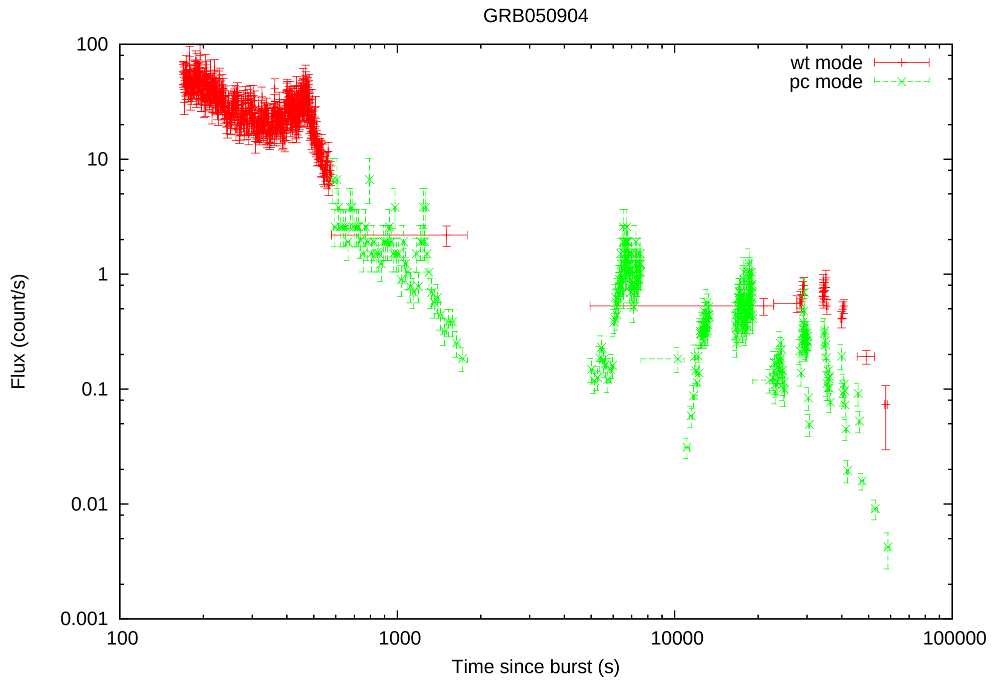

GRB050819
pc mode
Flux (count/s)
10
1
0.1
0.01
0.001
100
1000
10000
100000
Time since burst (s)

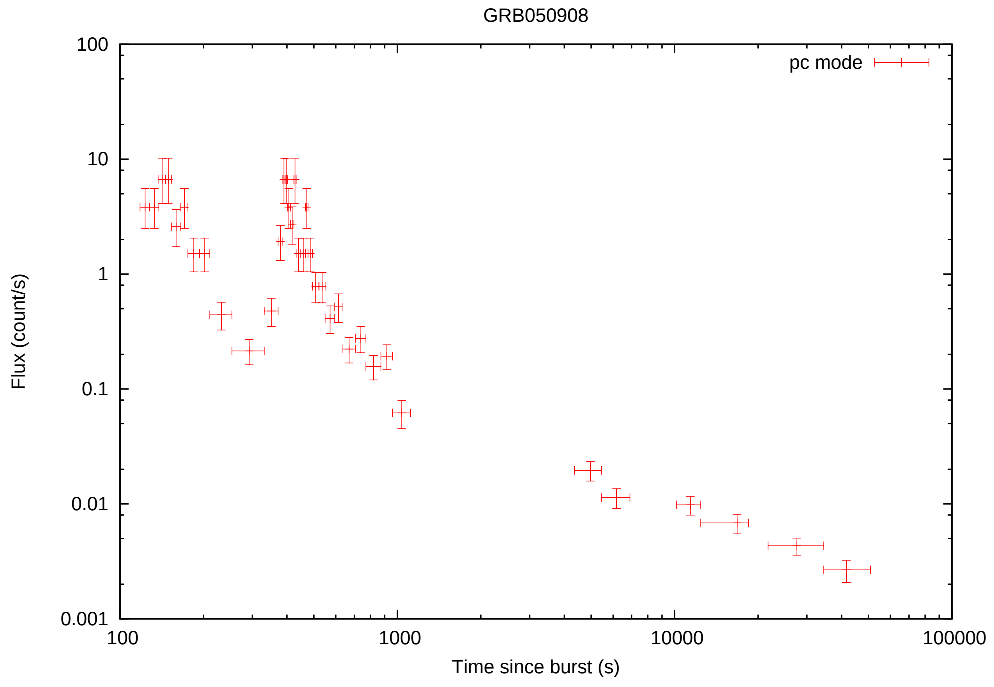

GRB050820A
Flux (count/s)
Time since burst (s)
wt mode
pc mode
1000
100
10
1
0.1
0.01
10
100
1000
10000
100000

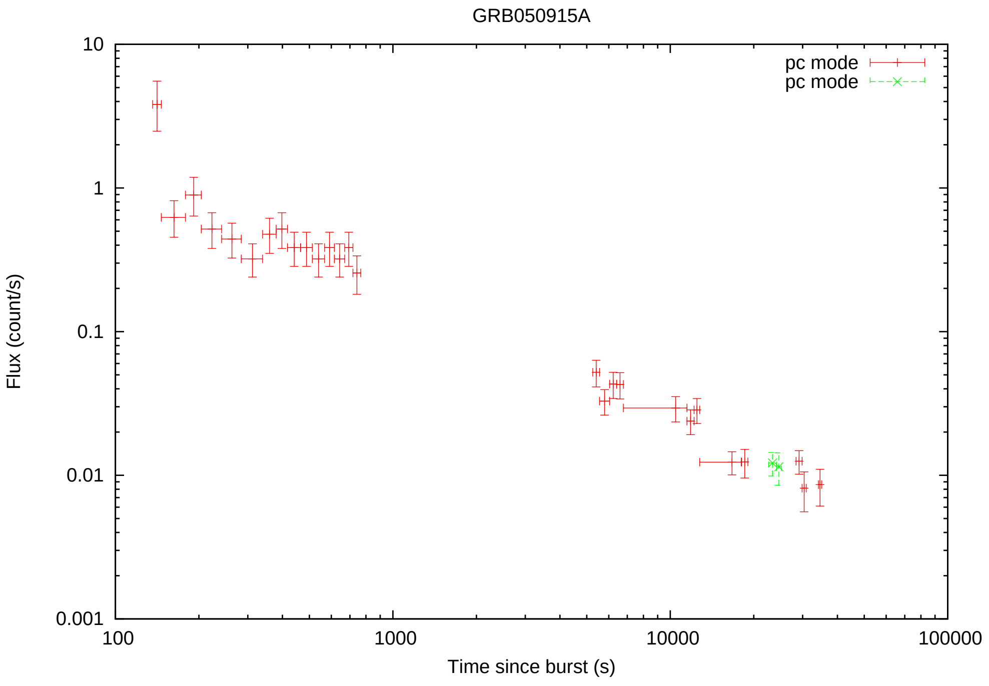

GRB050820B
Flux (count/s)
Time since burst (s)
pc mode
0.1
0.01
0.001
0.0001
1000
10000
100000

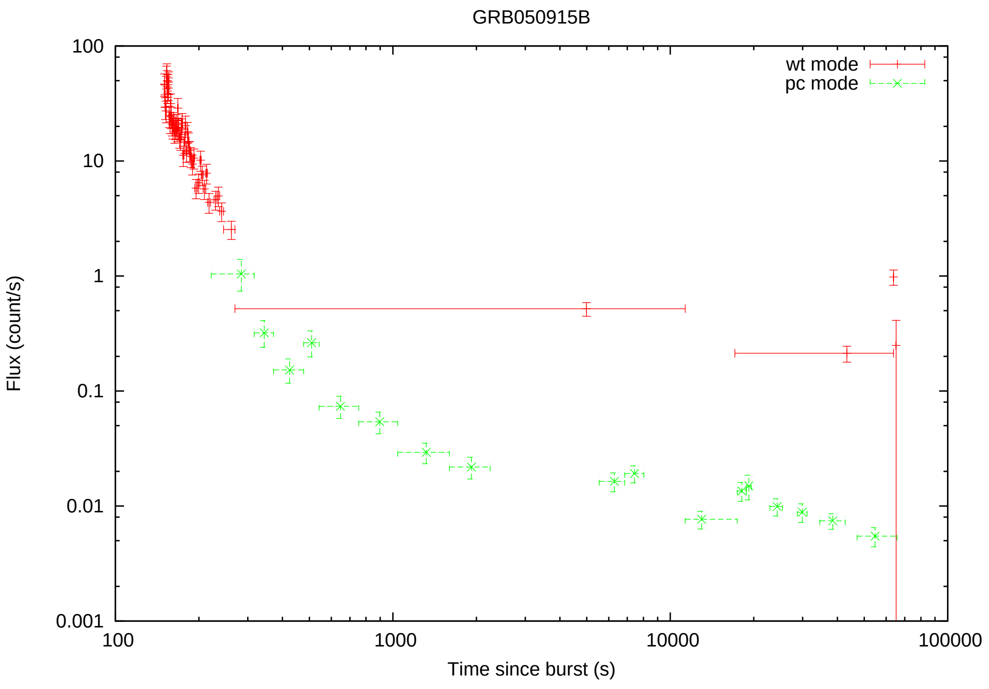

GRB050822
wt mode
pc mode
Flux (count/s)
1000
100
10
1
0.1
0.01
100
1000
10000
100000
Time since burst (s)

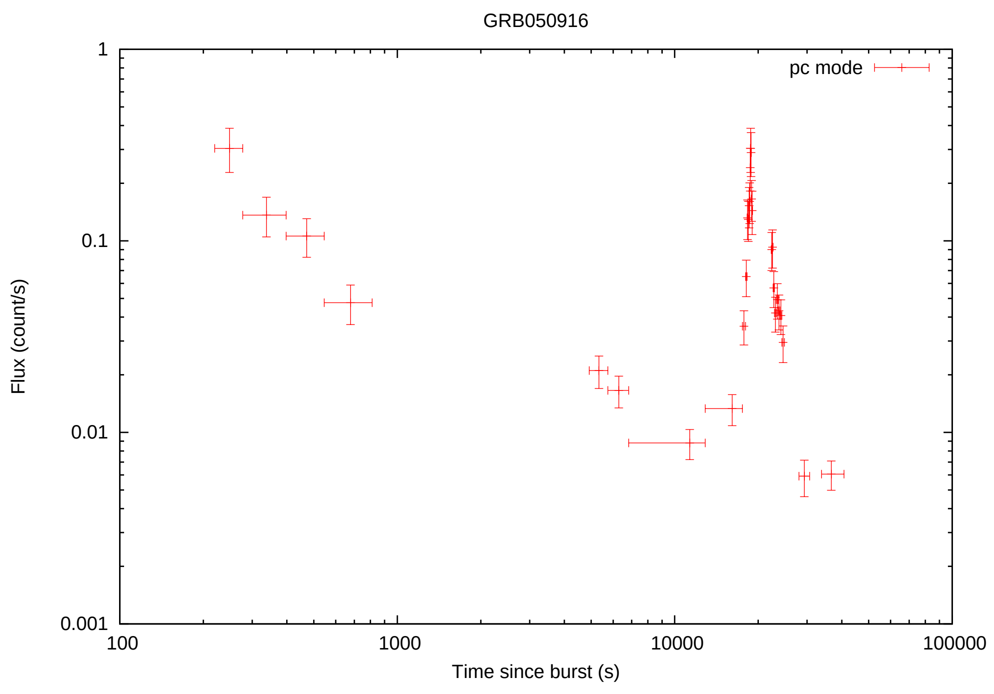

GRB050826
pc mode
Flux (count/s)
10
1
0.1
0.01
0.001
100
1000
10000
100000
1e+06
Time since burst (s)

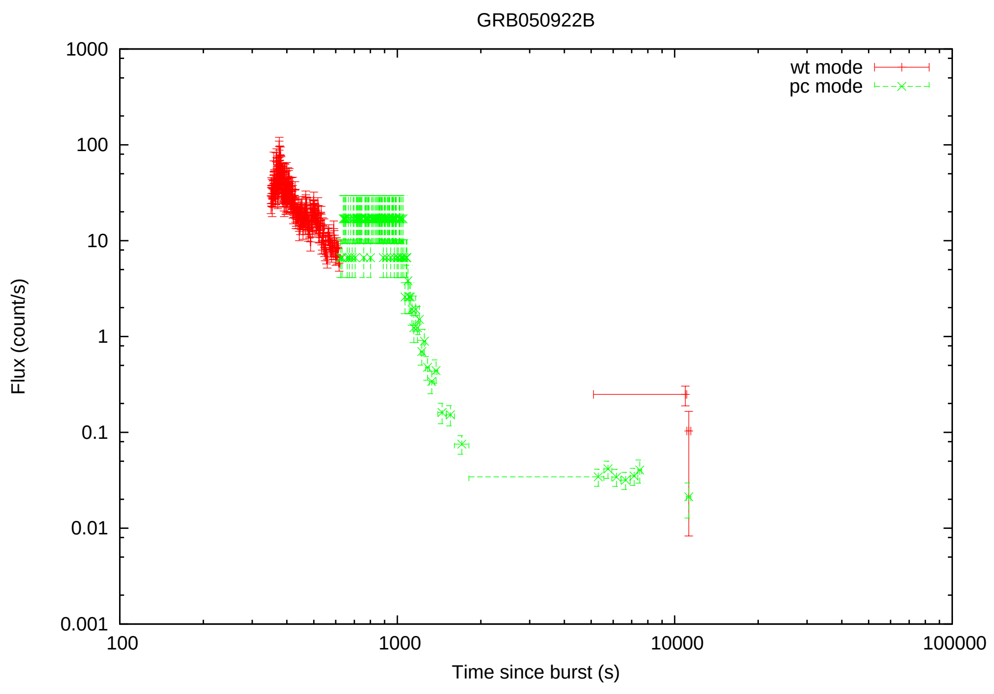

GRB050904
Flux (count/s)
Time since burst (s)
wt mode
pc mode
100
10
1
0.1
0.01
0.001
100
1000
10000
100000

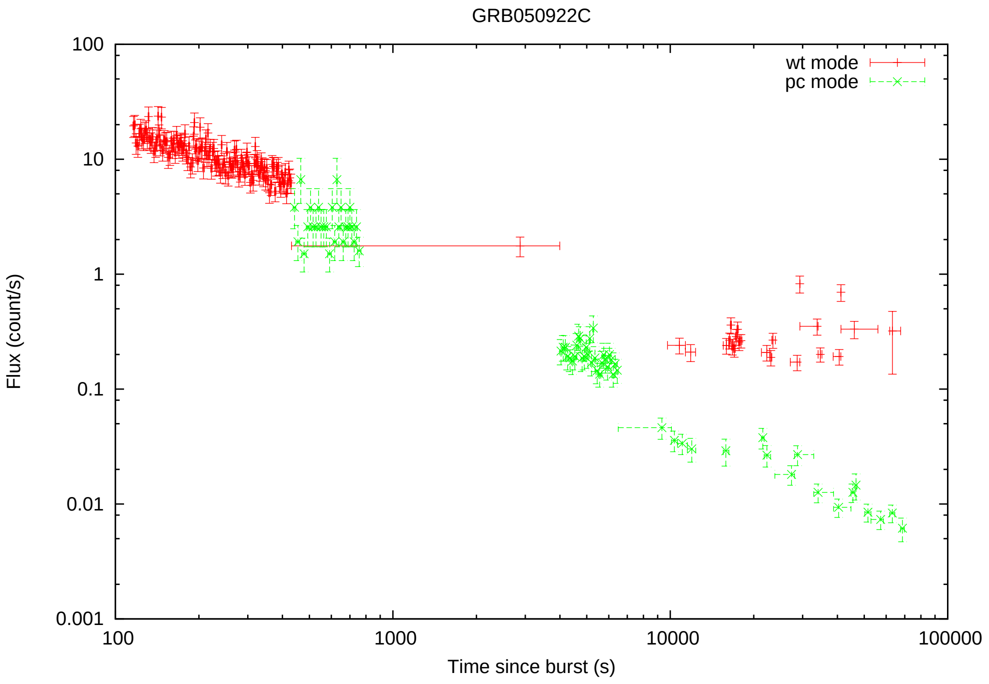

GRB050908
Flux (count/s)
Time since burst (s)
pc mode
100
10
1
0.1
0.01
0.001
100
1000
10000
100000

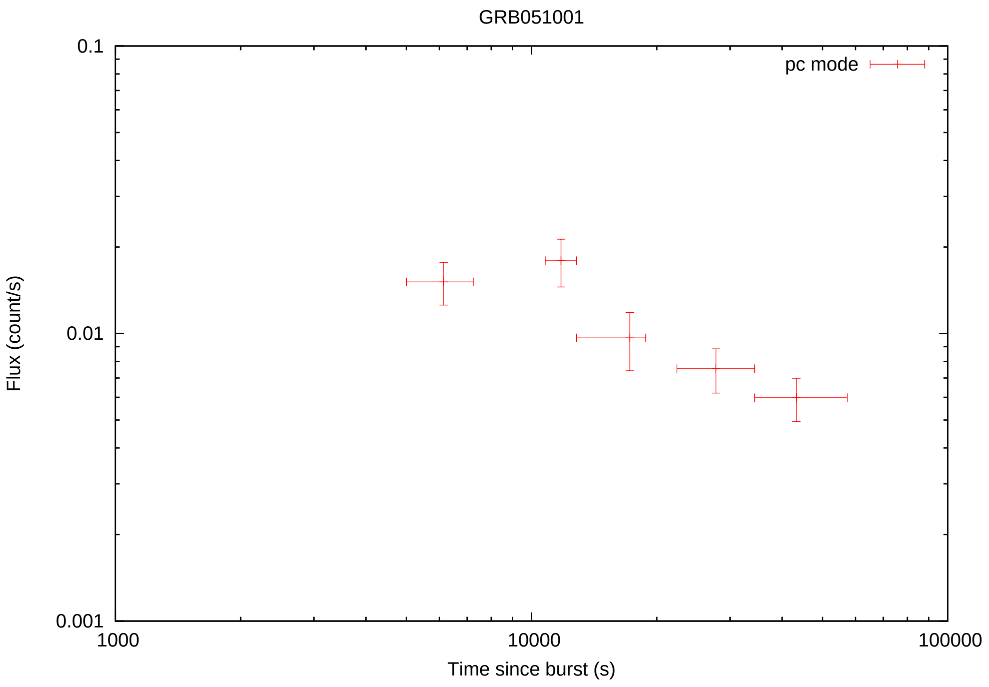

GRB050915A
pc mode
pc mode
Flux (count/s)
Time since burst (s)
10
1
0.1
0.01
0.001
100
1000
10000
100000

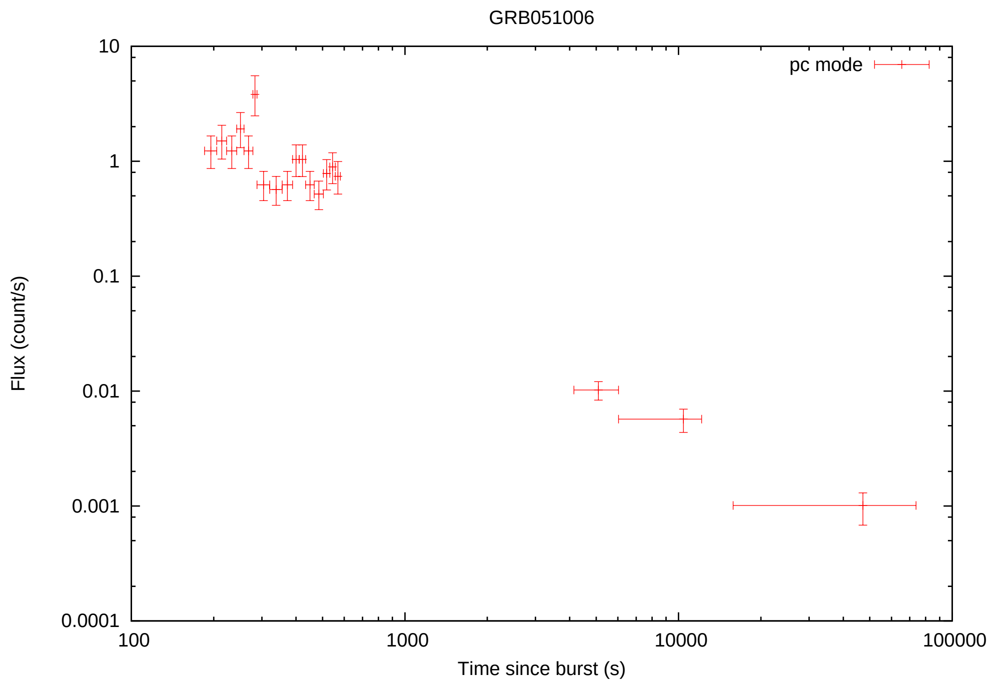

GRB050915B
wt mode
pc mode
Flux (count/s)
Time since burst (s)
100
10
1
0.1
0.01
0.001
100
1000
10000
100000

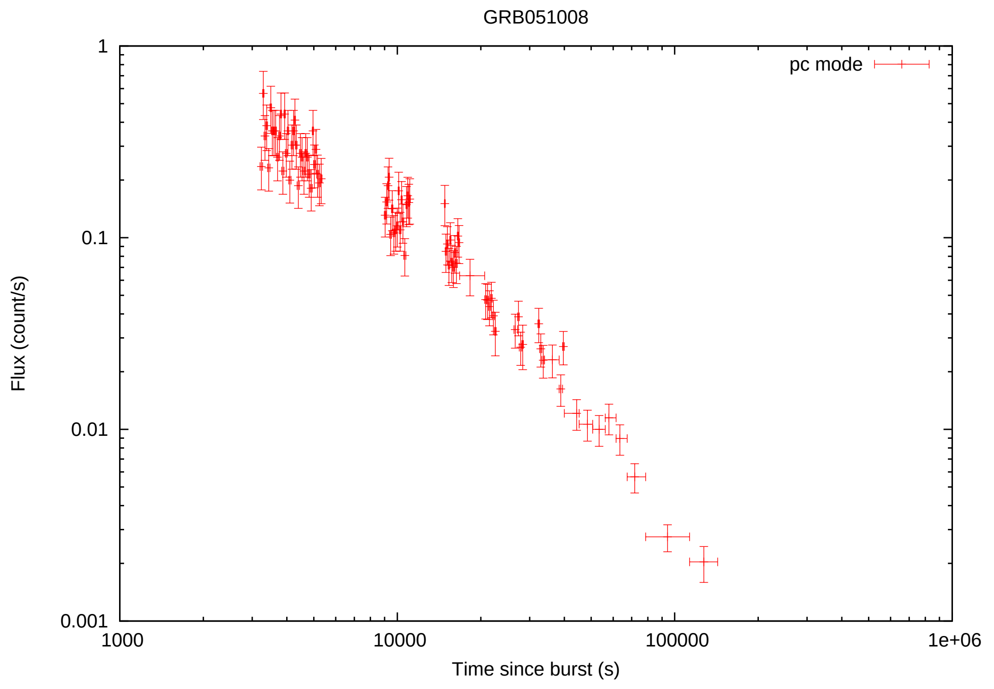

GRB050916
pc mode
Flux (count/s)
Time since burst (s)
1
0.1
0.01
0.001
100
1000
10000
100000

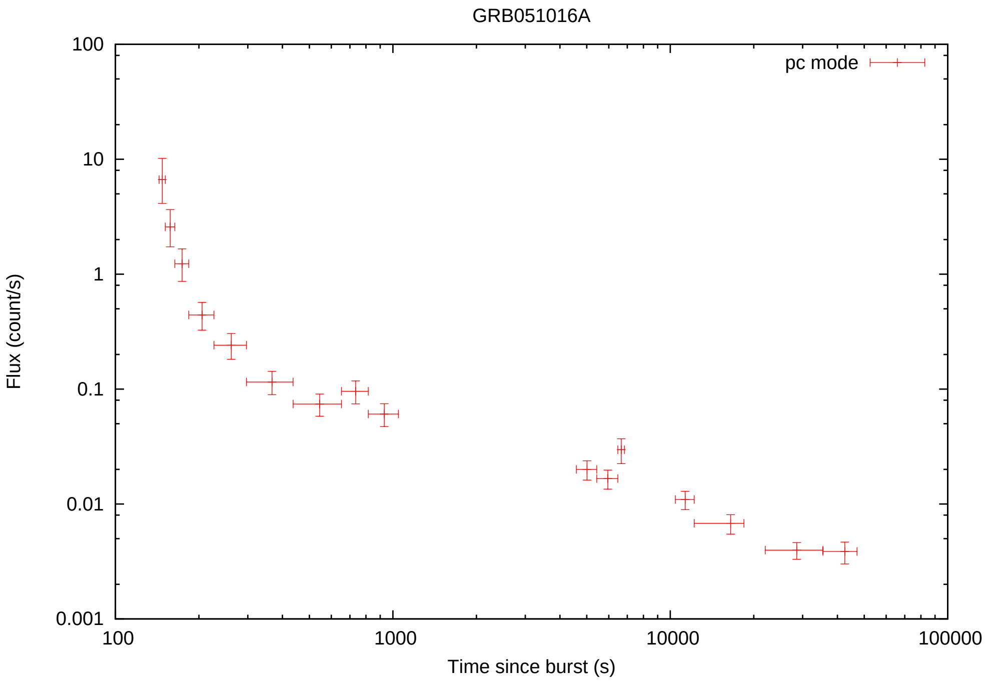

GRB050922B
wt mode
pc mode
Flux (count/s)
Time since burst (s)
1000
100
10
1
0.1
0.01
0.001
100
1000
10000
100000

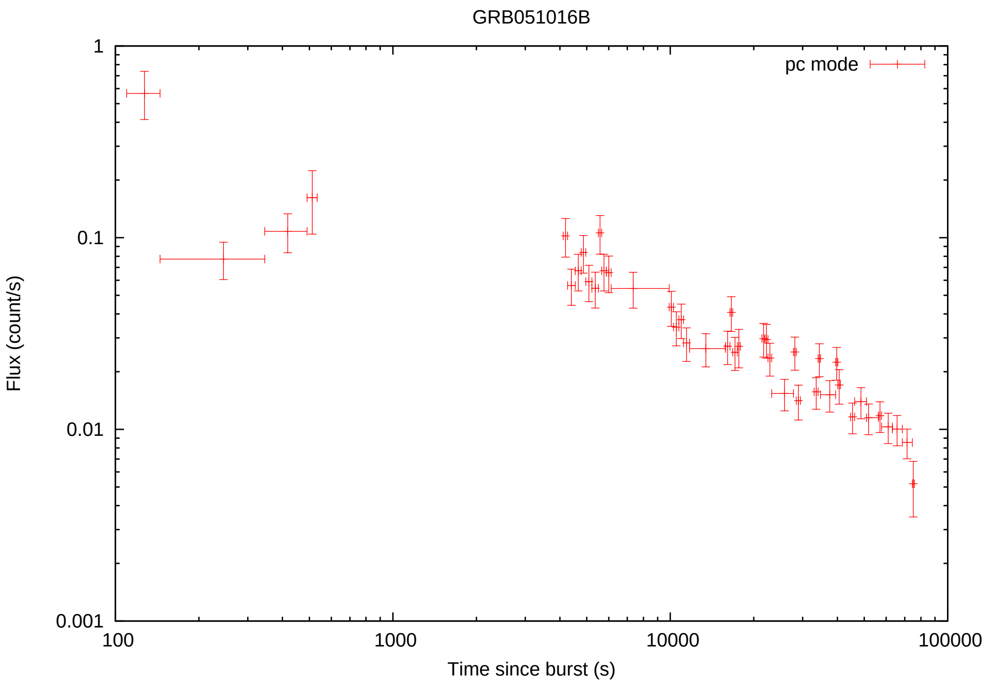

GRB050922C
wt mode
pc mode
Flux (count/s)
Time since burst (s)
100
10
1
0.1
0.01
0.001
100
1000
10000
100000

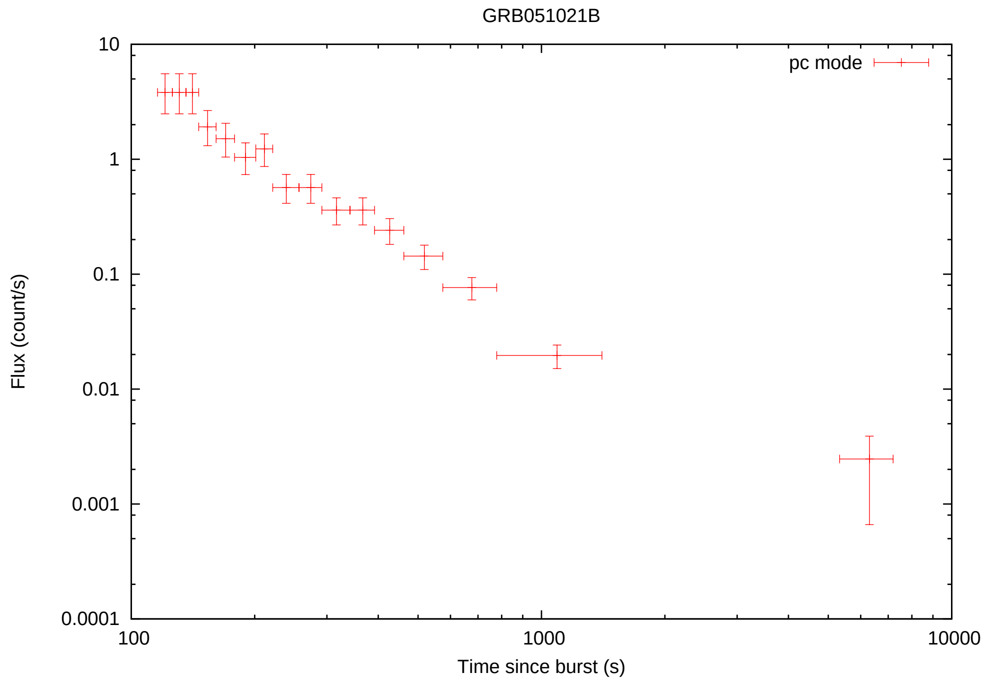

GRB051001
pc mode
Flux (count/s)
Time since burst (s)
0.1
0.01
0.001
1000
10000
100000

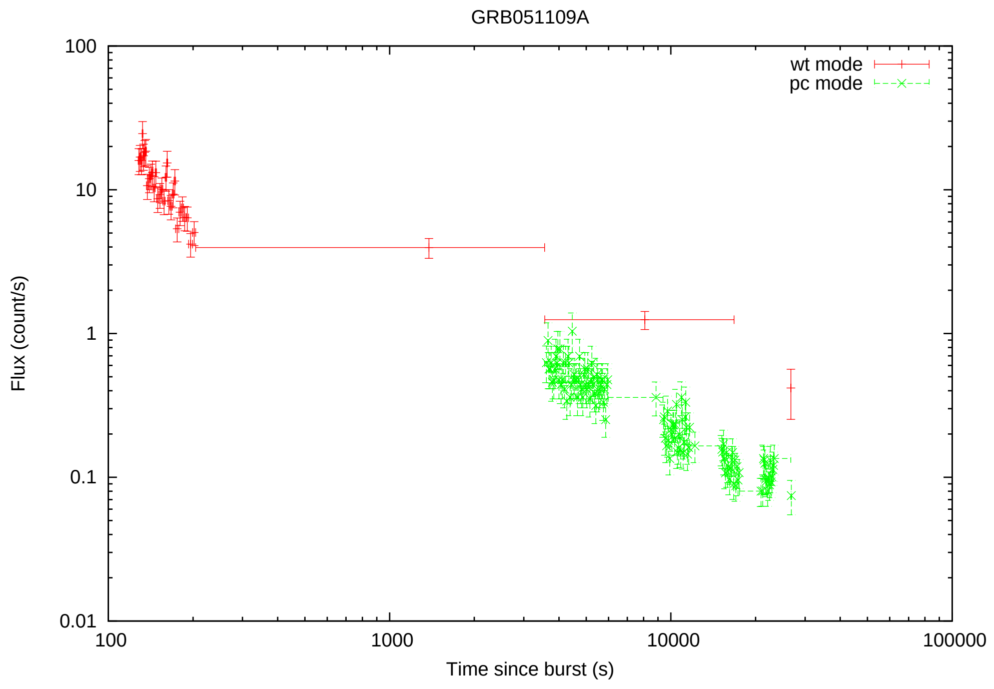

GRB051006
pc mode
Flux (count/s)
Time since burst (s)
10
1
0.1
0.01
0.001
0.0001
100
1000
10000
100000

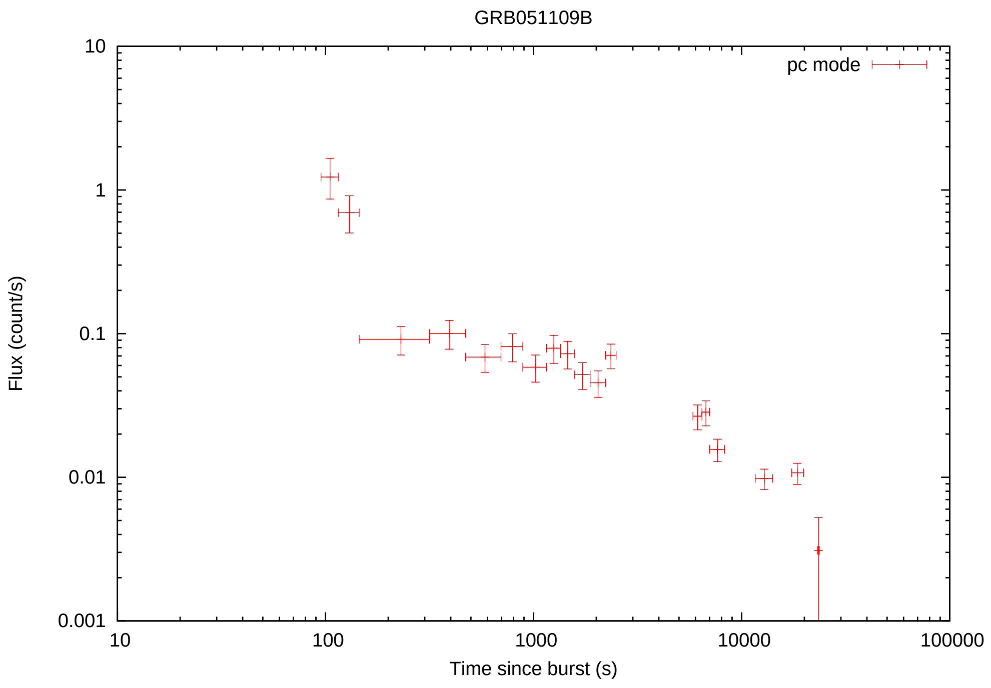

GRB051008
pc mode
Flux (count/s)
Time since burst (s)
1
0.1
0.01
0.001
1000
10000
100000
1e+06

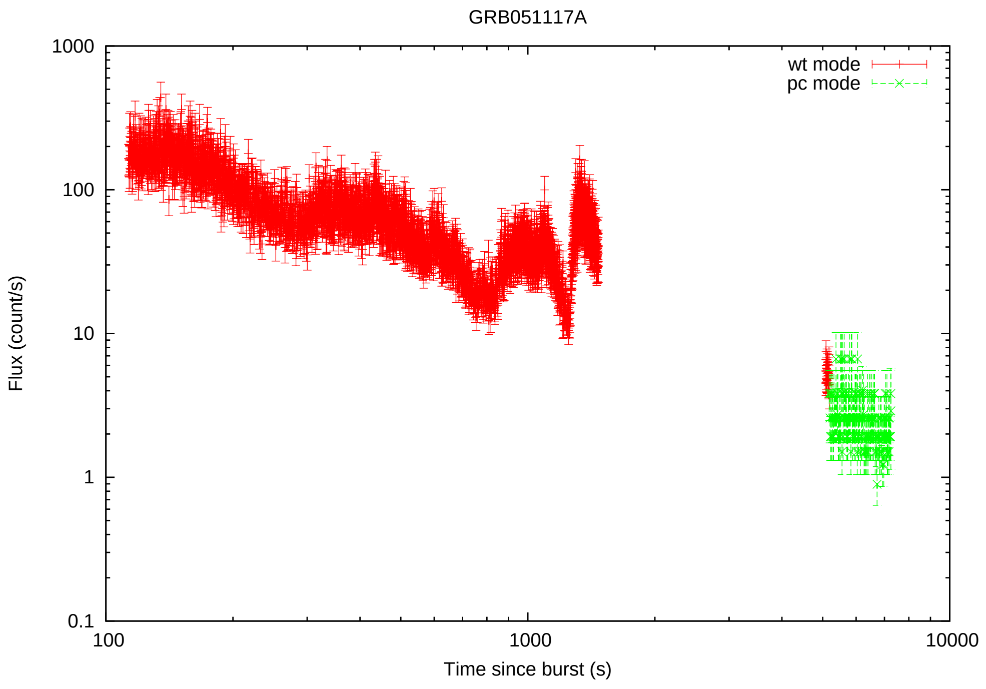

GRB051016A
pc mode
Flux (count/s)
Time since burst (s)
100
10
1
0.1
0.01
0.001
100
1000
10000
100000

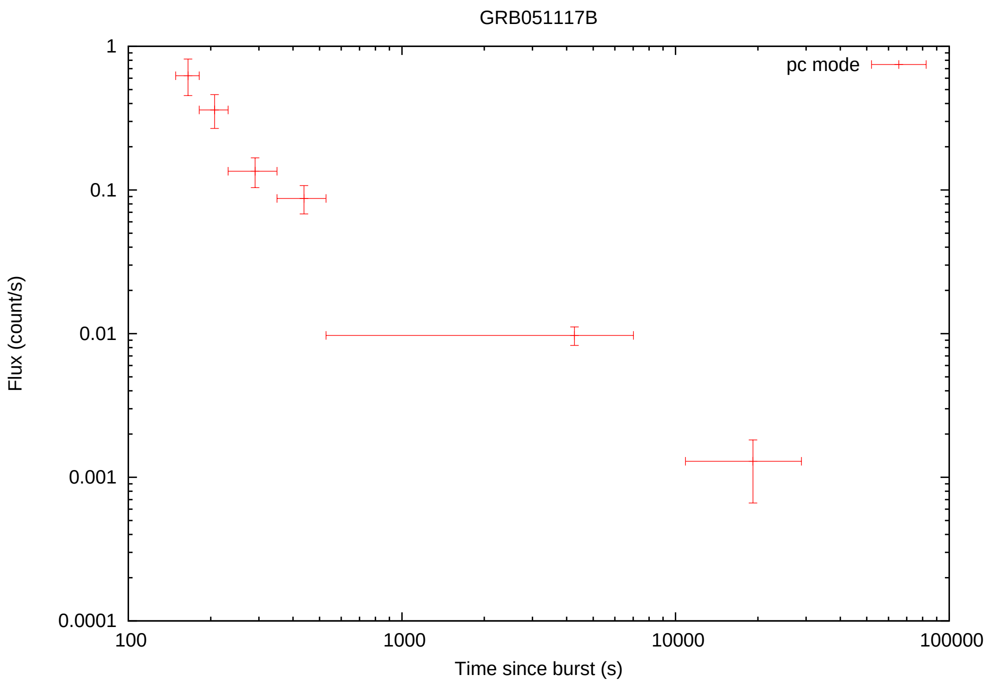

GRB051016B
pc mode
Flux (count/s)
Time since burst (s)
1
0.1
0.01
0.001
100
1000
10000
100000

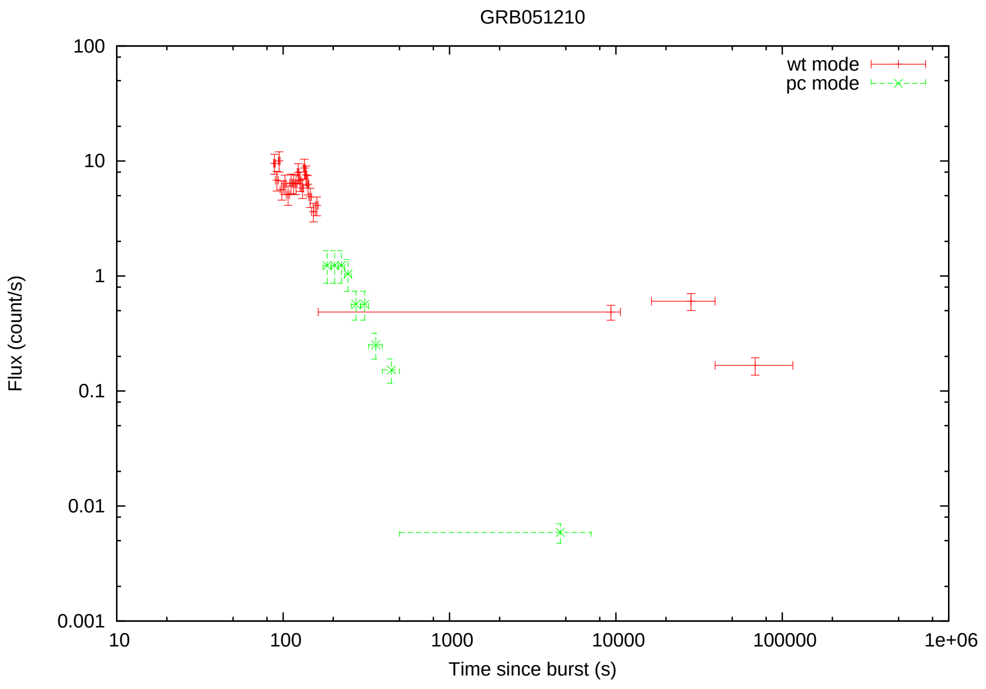

GRB051021B
pc mode
Flux (count/s)
10
1
0.1
0.01
0.001
0.0001
100
1000
10000
Time since burst (s)

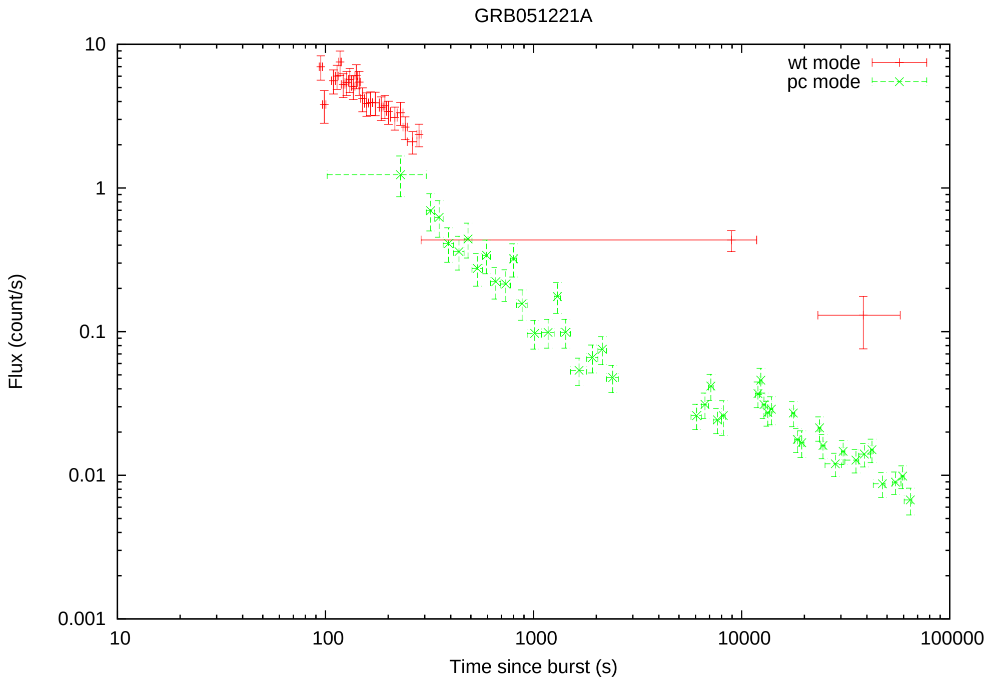

GRB051109A
wt mode
pc mode
Flux (count/s)
100
10
1
0.1
0.01
100
1000
10000
100000
Time since burst (s)


GRB051109B
pc mode
Flux (count/s)
10
1
0.1
0.01
0.001
10
100
1000
10000
100000
Time since burst (s)

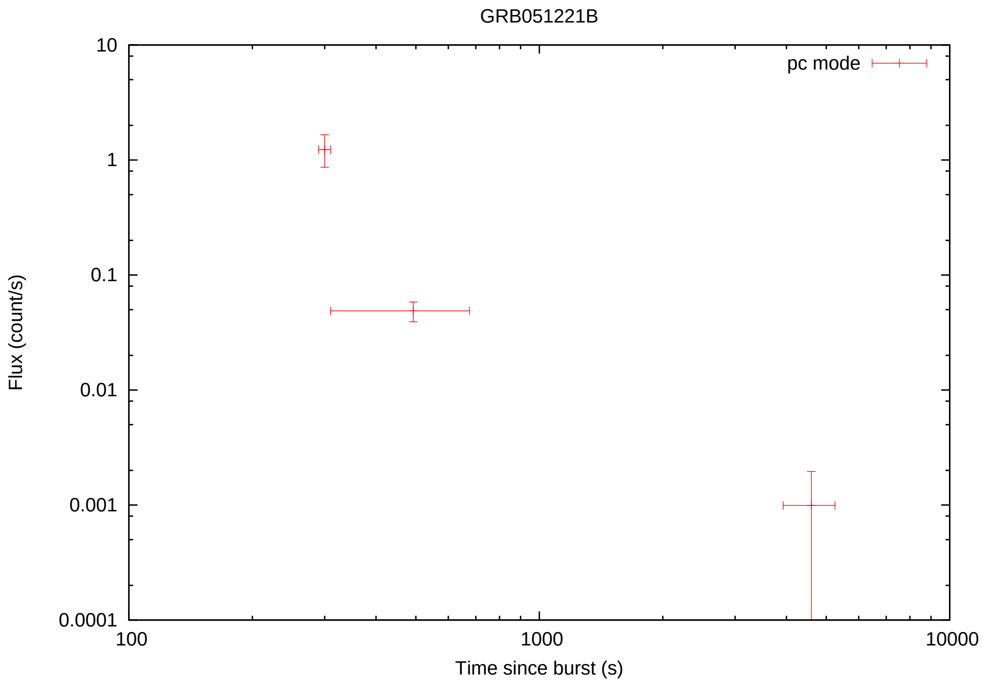

GRB051117A
wt mode
pc mode
Flux (count/s)
1000
100
10
1
0.1
100
1000
10000
Time since burst (s)


GRB051117B
pc mode
Flux (count/s)
1
0.1
0.01
0.001
0.0001
100
1000
10000
100000
Time since burst (s)

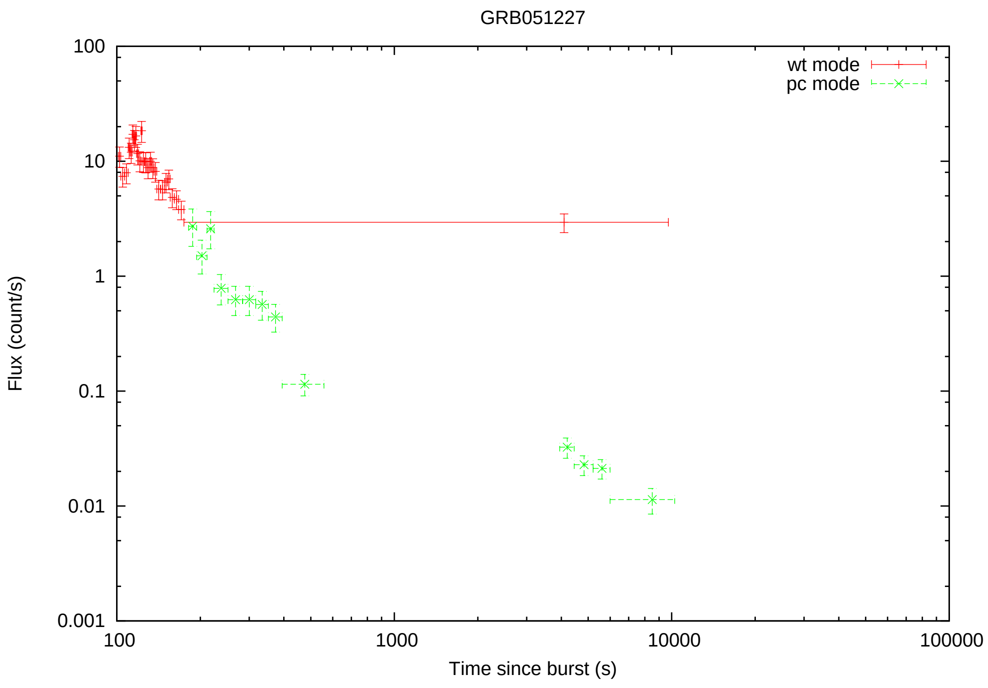

GRB051210
wt mode
pc mode
Flux (count/s)
100
10
1
0.1
0.01
0.001
10
100
1000
10000
100000
1e+06
Time since burst (s)

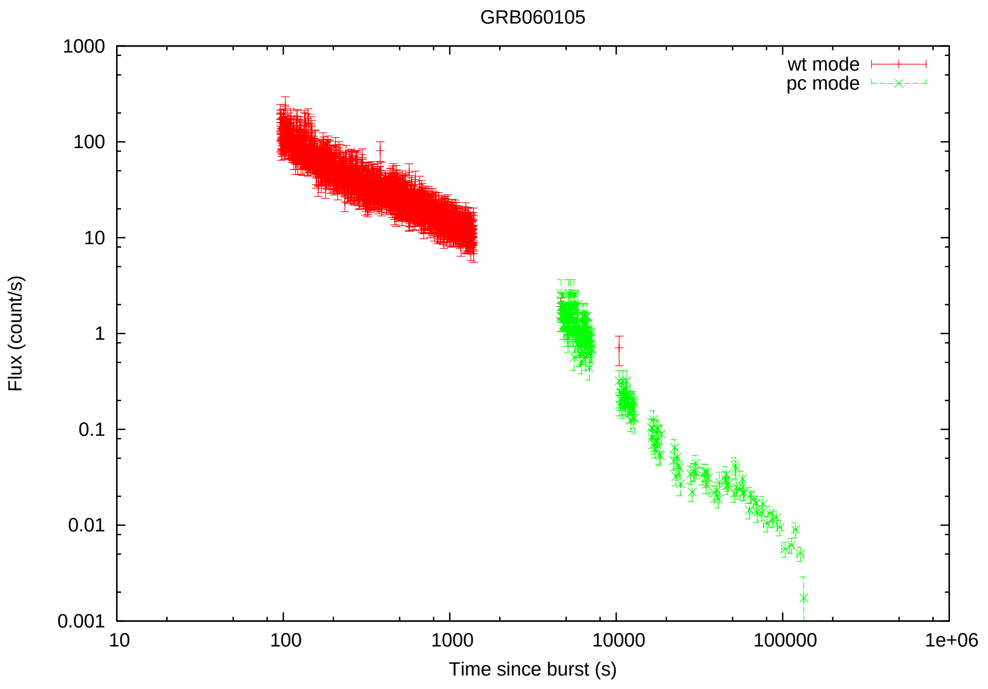


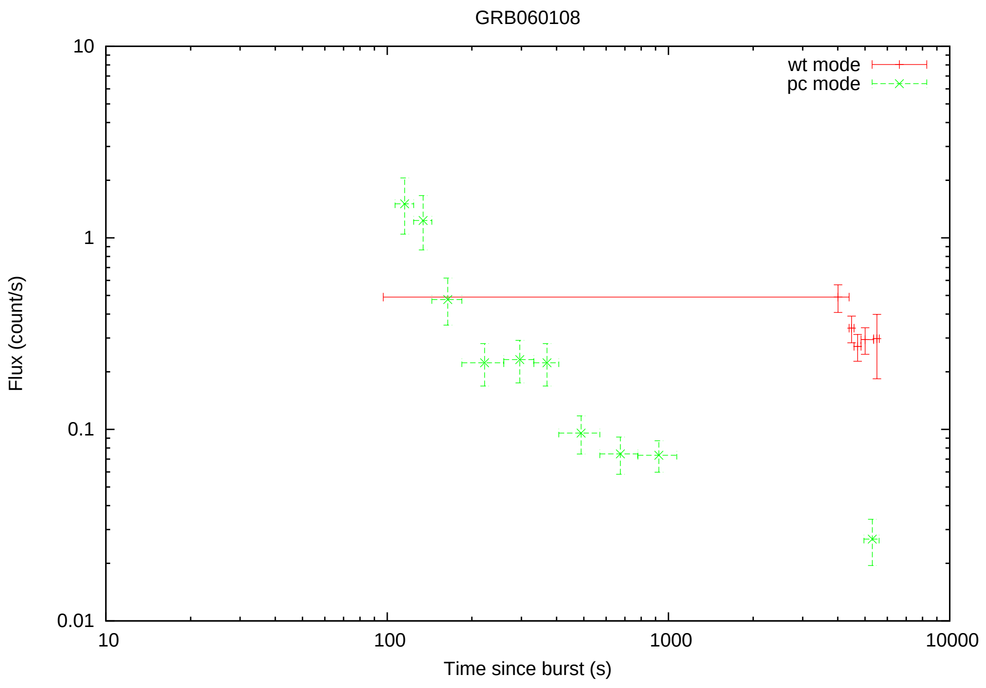
GRB051221A
wt mode
pc mode
Flux (count/s)
Time since burst (s)
10
1
0.1
0.01
0.001
10
100
1000
10000
100000

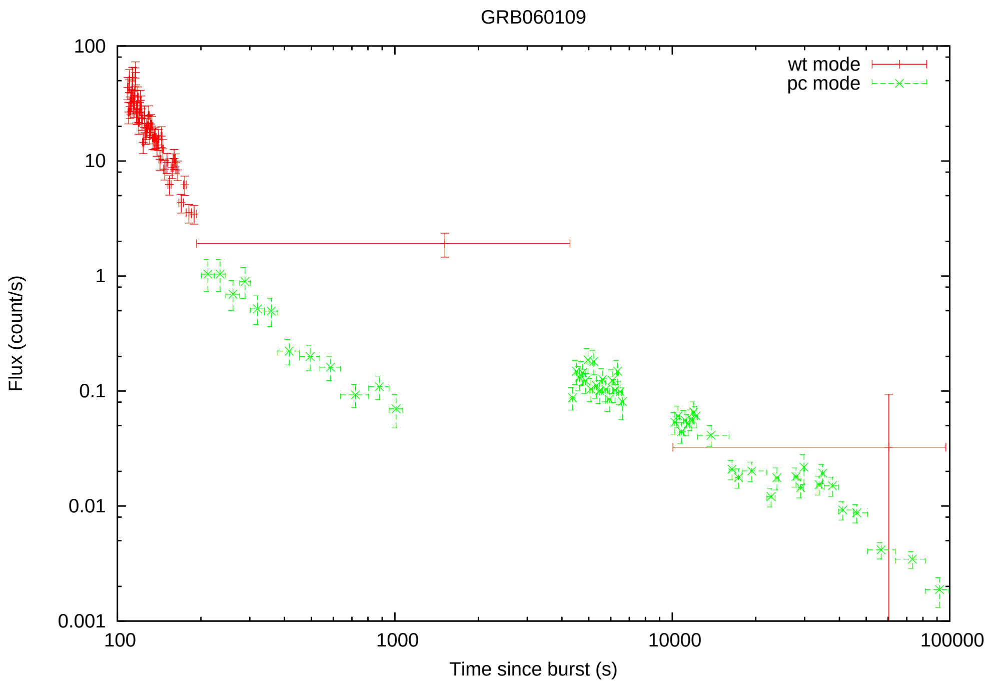


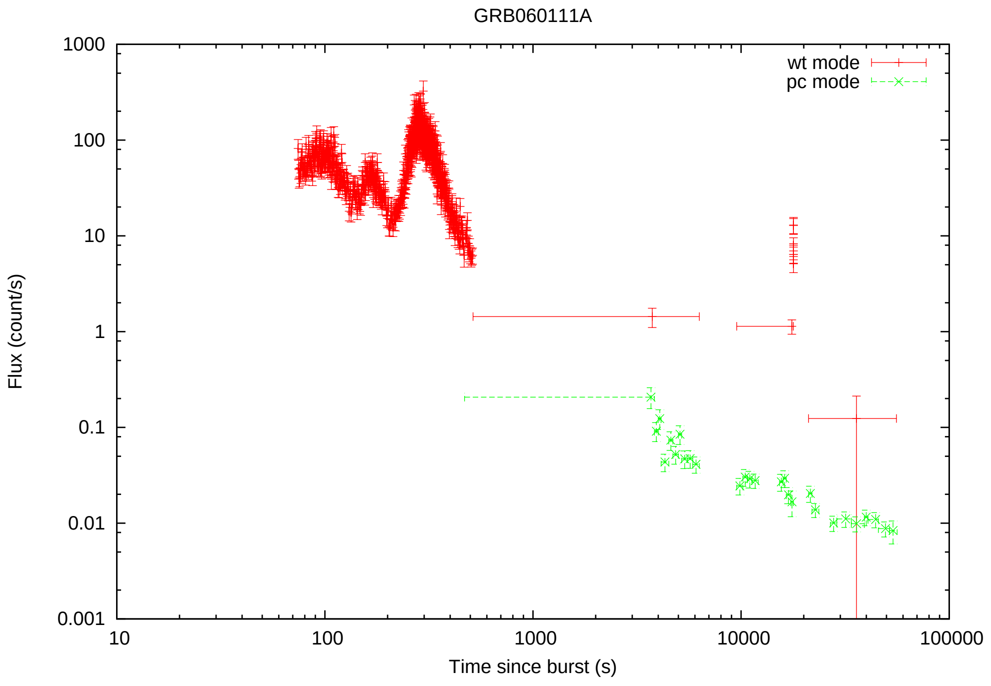
GRB051221B
pc mode
Flux (count/s)
Time since burst (s)
10
1
0.1
0.01
0.001
0.0001
100
1000
10000

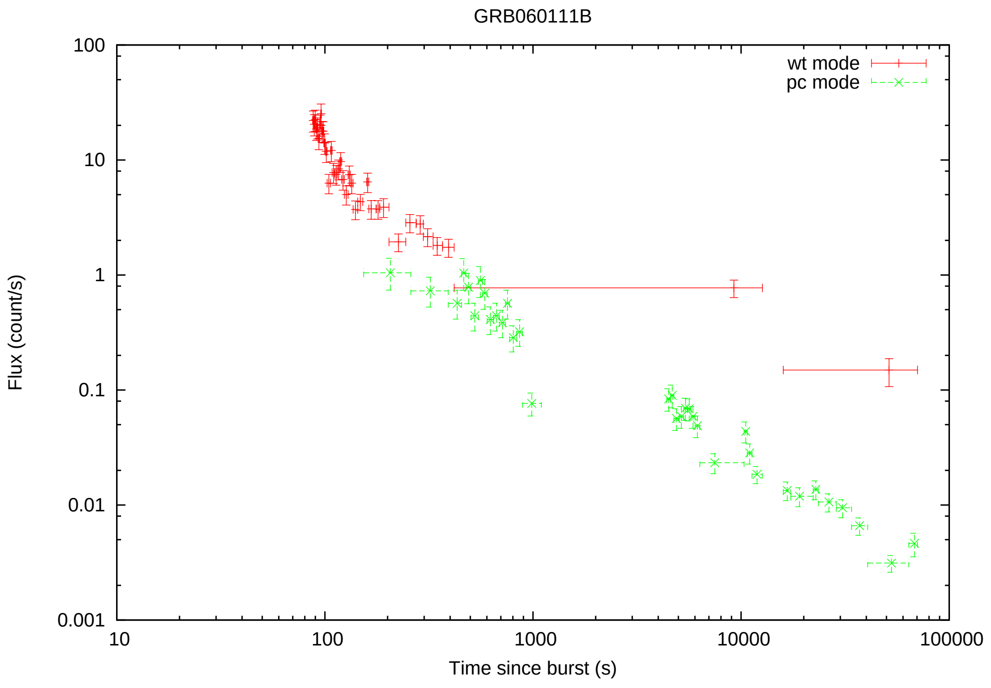

GRB051227
wt mode
pc mode
Flux (count/s)
100
10
1
0.1
0.01
0.001
100
1000
10000
100000
Time since burst (s)

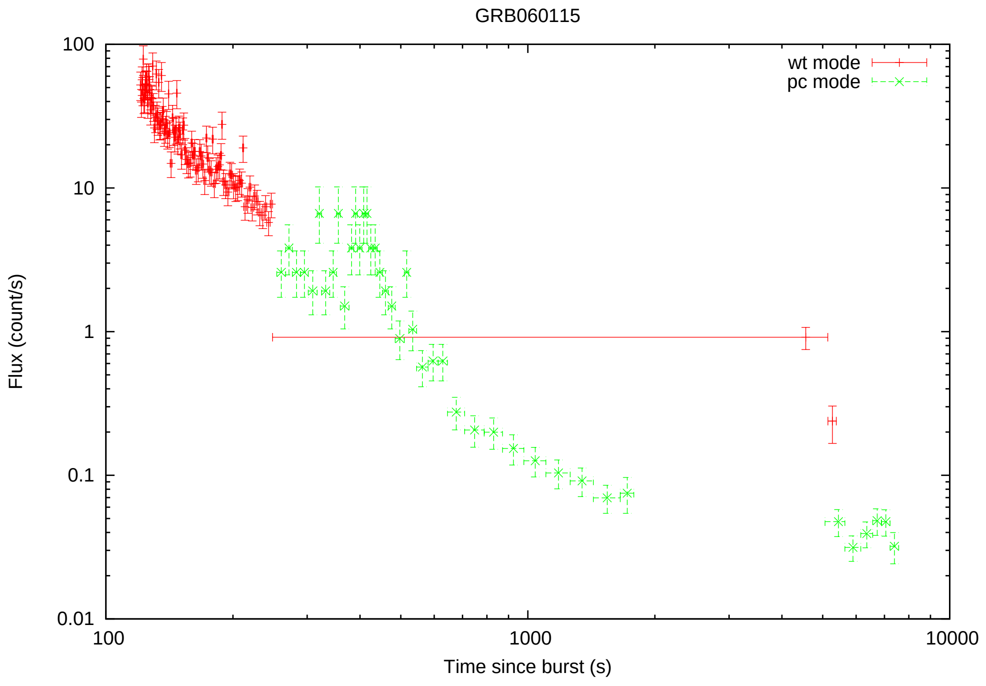

GRB060105
wt mode
pc mode
Flux (count/s)
1000
100
10
1
0.1
0.01
0.001
10
100
1000
10000
100000
1e+06
Time since burst (s)

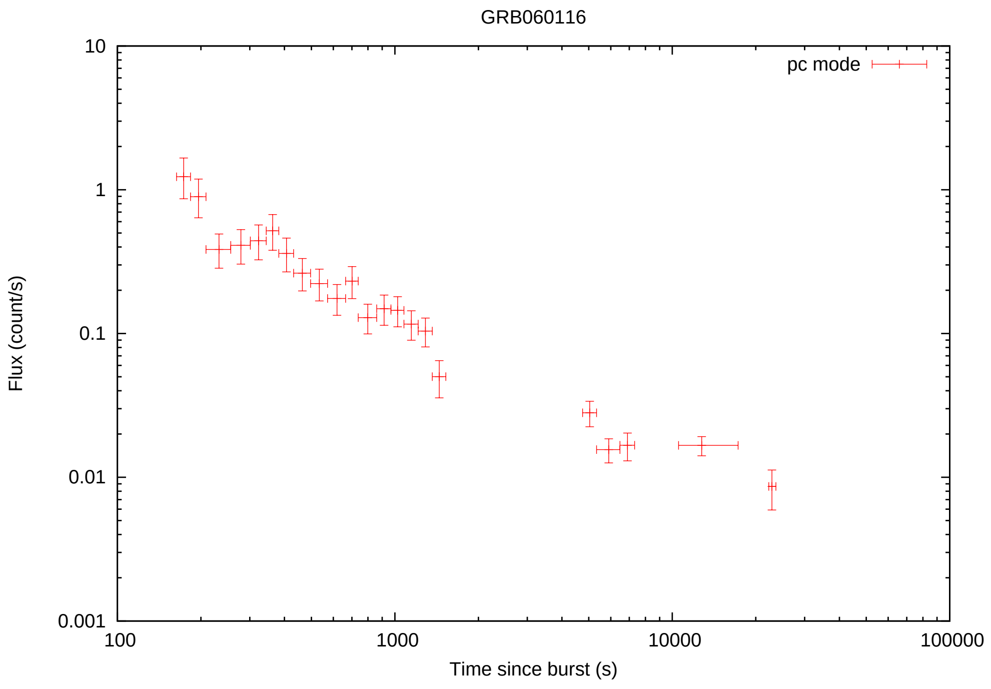

GRB060108
Flux (count/s)
Time since burst (s)
wt mode
pc mode
10
1
0.1
0.01
10
100
1000
10000

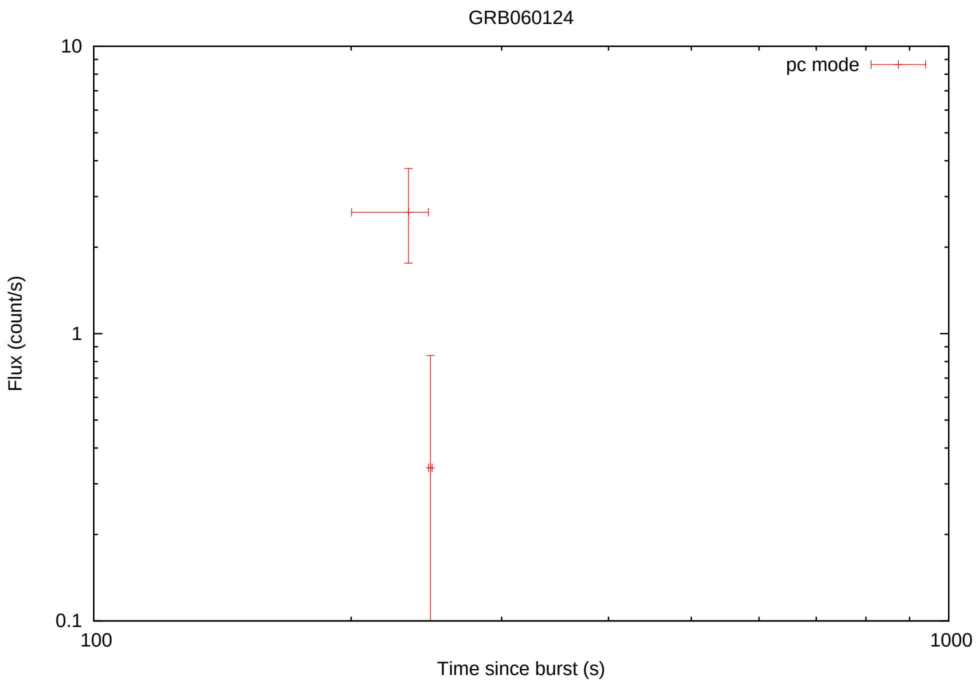

GRB060109
Flux (count/s)
Time since burst (s)
wt mode
pc mode
100
10
1
0.1
0.01
0.001
100
1000
10000
100000

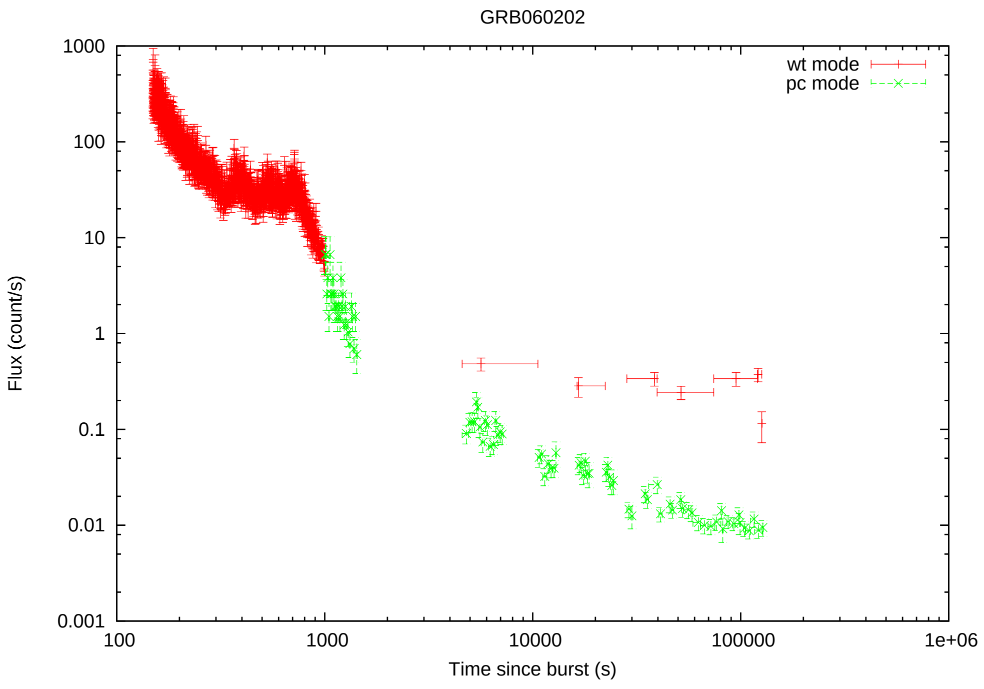

GRB060111A
wt mode
pc mode
Flux (count/s)
1000
100
10
1
0.1
0.01
0.001
10
100
1000
10000
100000
Time since burst (s)

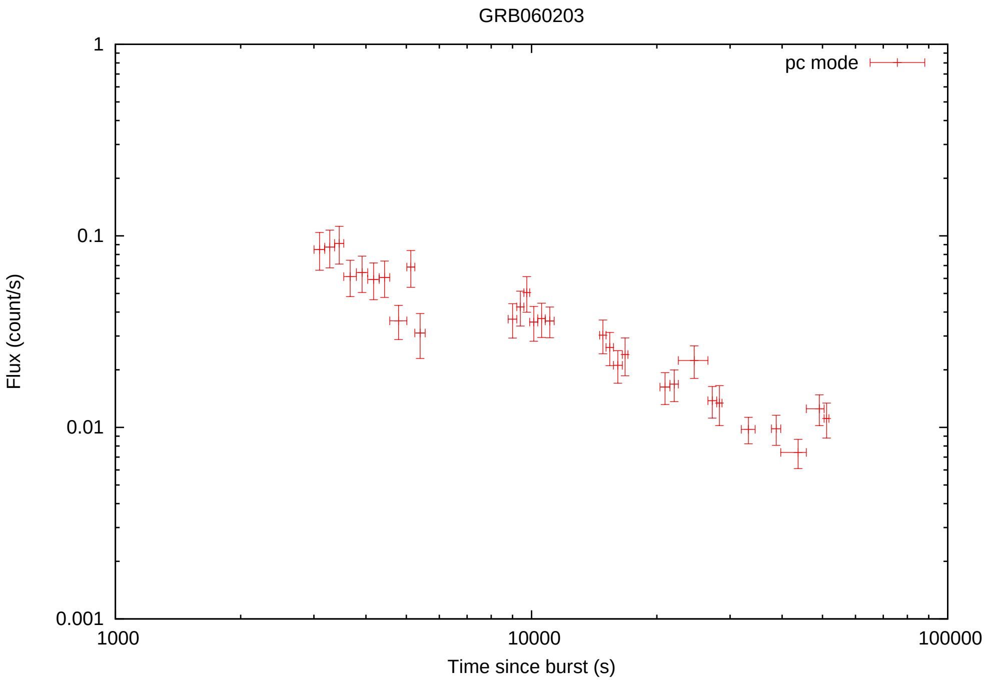

GRB060111B
wt mode
pc mode
Flux (count/s)
100
10
1
0.1
0.01
0.001
10
100
1000
10000
100000
Time since burst (s)

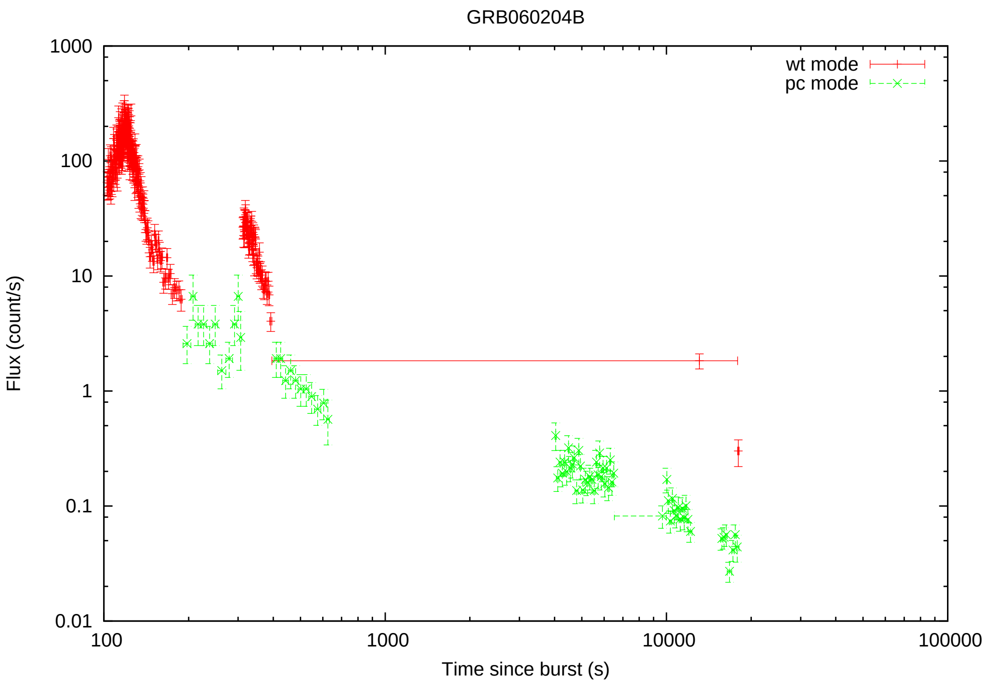

GRB060115
wt mode
pc mode
Flux (count/s)
Time since burst (s)
100
10
1
0.1
0.01
100
1000
10000

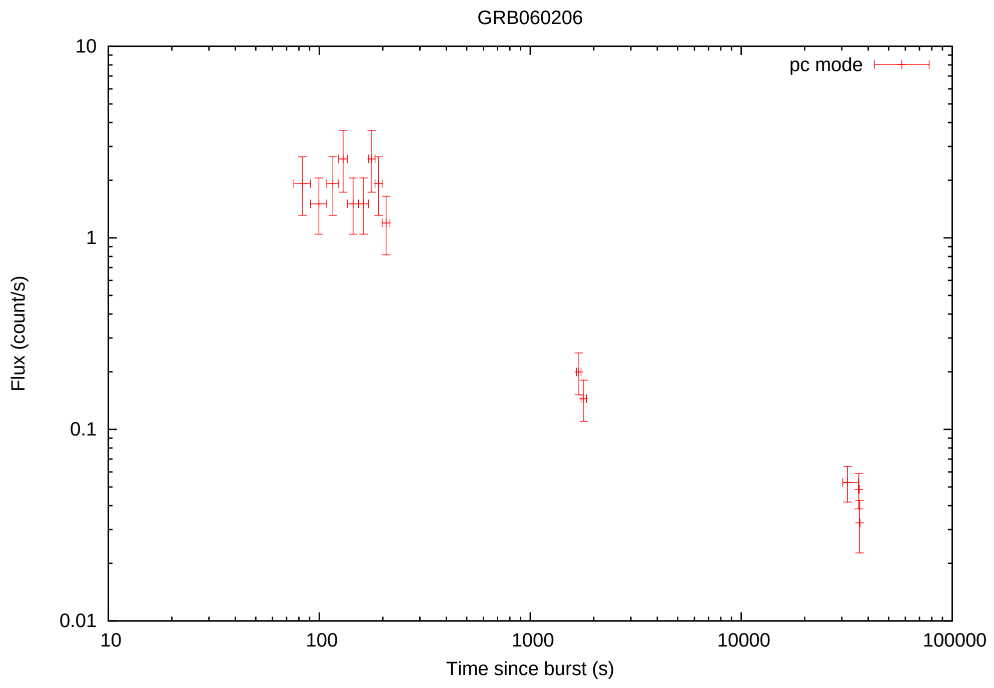

GRB060116
pc mode
Flux (count/s)
Time since burst (s)
10
1
0.1
0.01
0.001
100
1000
10000
100000

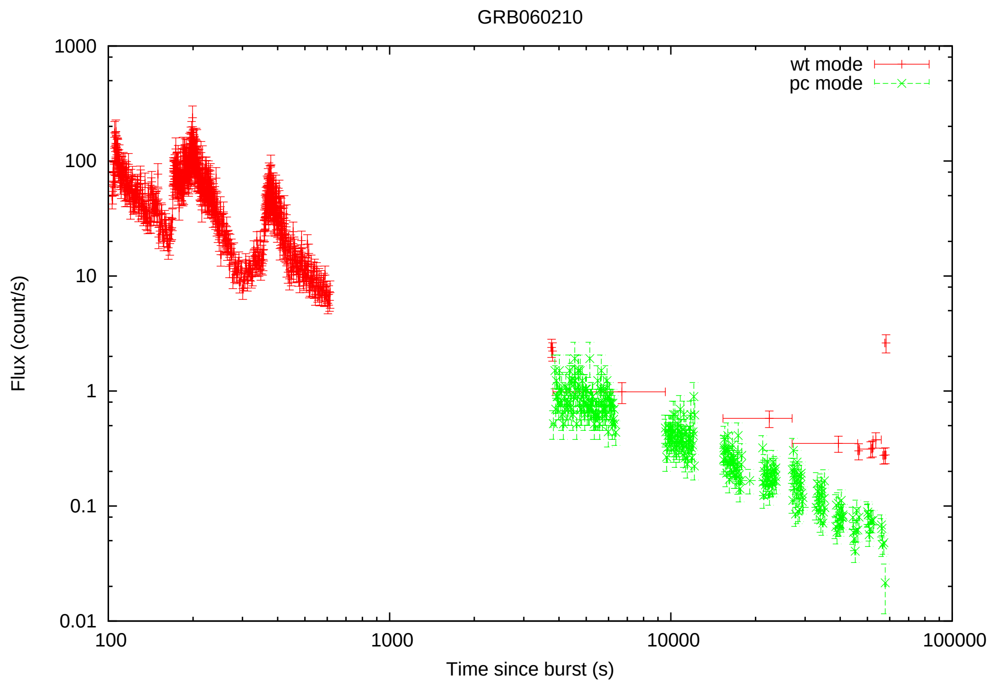

GRB060124
pc mode
10
1
0.1
100
1000
Flux (count/s)
Time since burst (s)

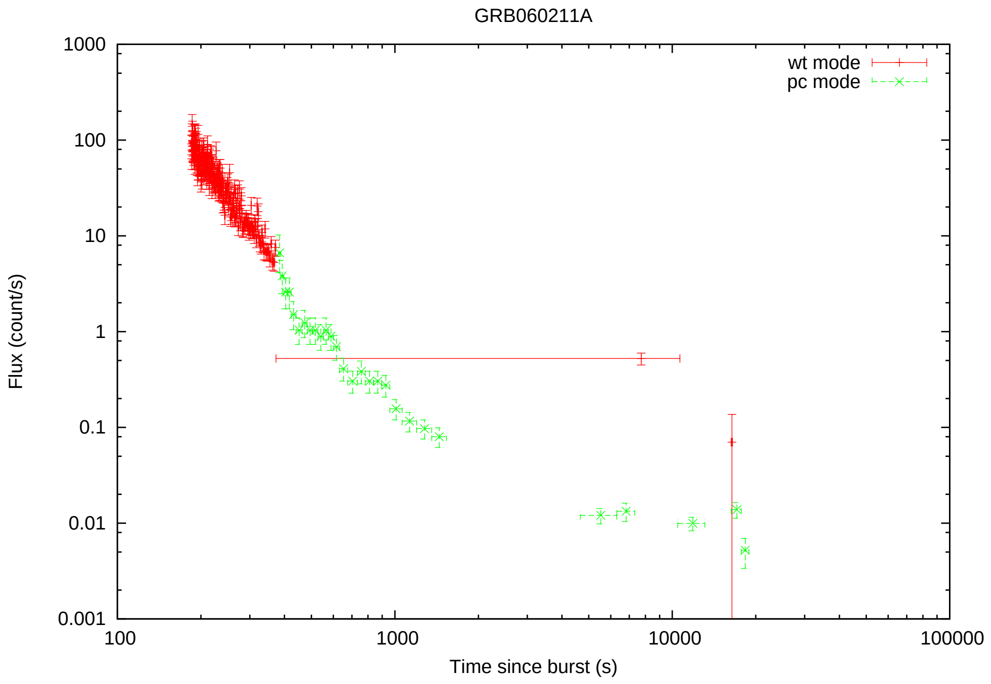

GRB060202
wt mode
pc mode
1000
100
10
1
0.1
0.01
0.001
100
1000
10000
100000
1e+06
Flux (count/s)
Time since burst (s)

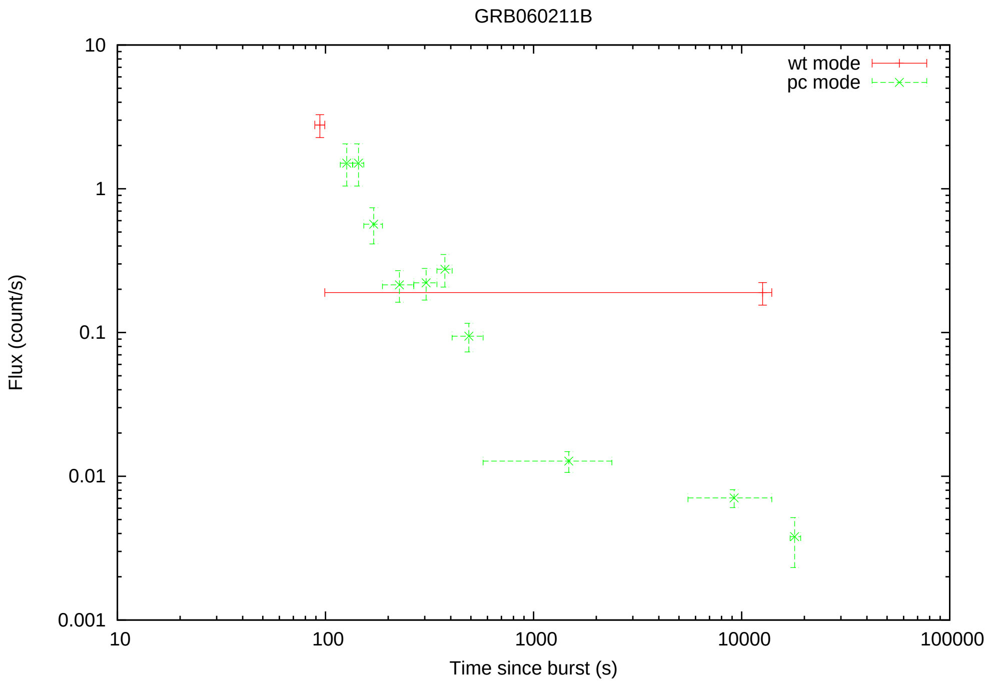

GRB060203
pc mode
Flux (count/s)
Time since burst (s)
1
0.1
0.01
0.001
1000
10000
100000

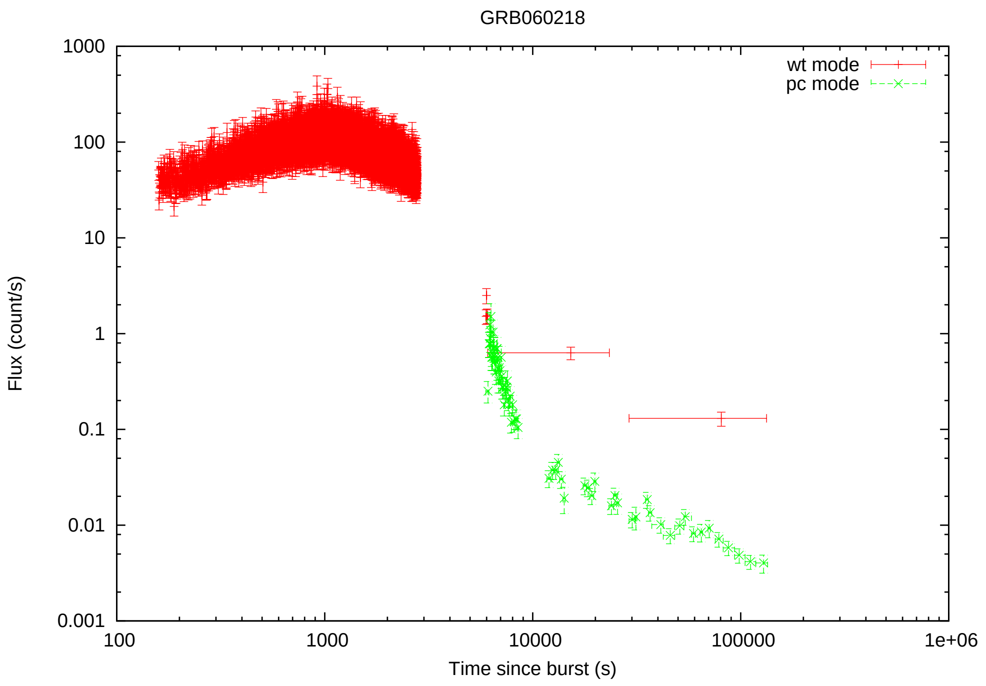

GRB060204B
wt mode
pc mode
Flux (count/s)
Time since burst (s)
1000
100
10
1
0.1
0.01
100
1000
10000
100000

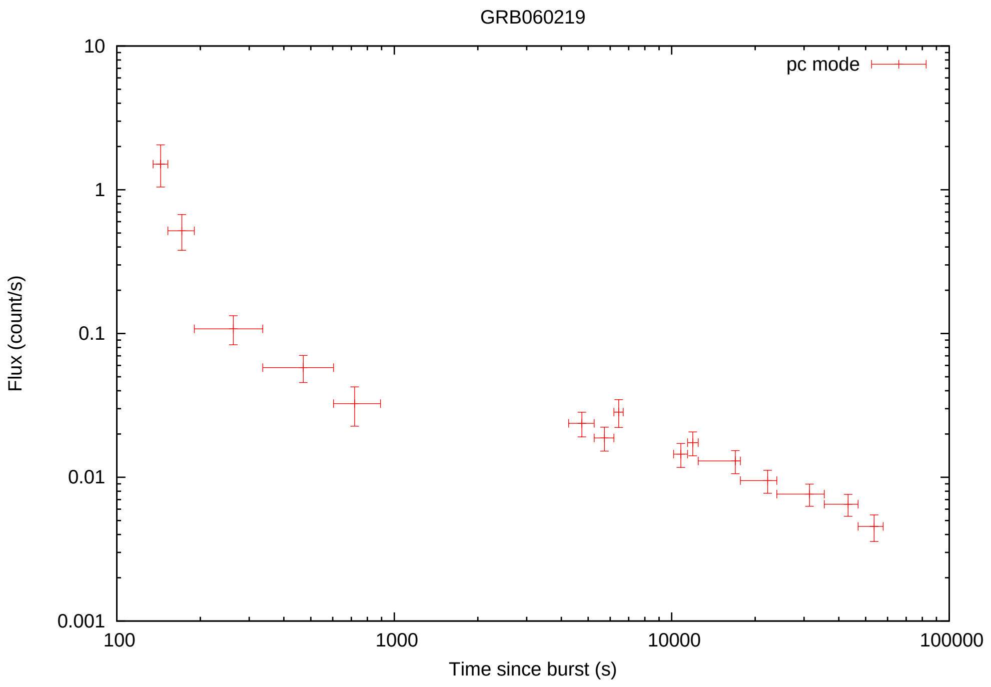

GRB060206
pc mode
Flux (count/s)
10
1
0.1
0.01
10
100
1000
10000
100000
Time since burst (s)

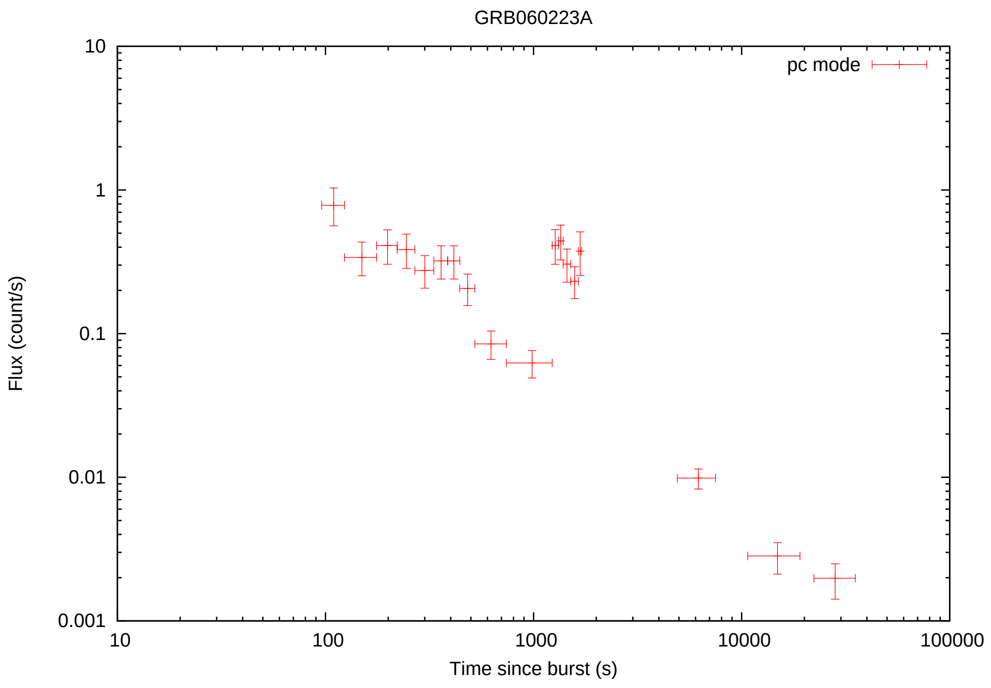

GRB060210
wt mode
pc mode
Flux (count/s)
1000
100
10
1
0.1
0.01
100
1000
10000
100000
Time since burst (s)

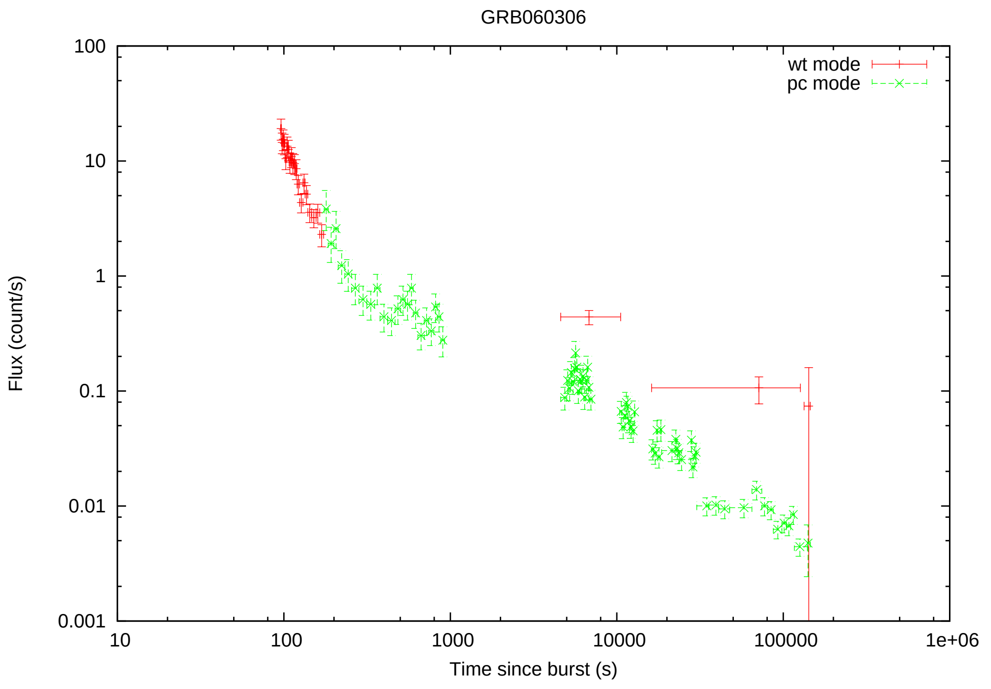

GRB060211A
wt mode
pc mode
Flux (count/s)
1000
100
10
1
0.1
0.01
0.001
100
1000
10000
100000
Time since burst (s)

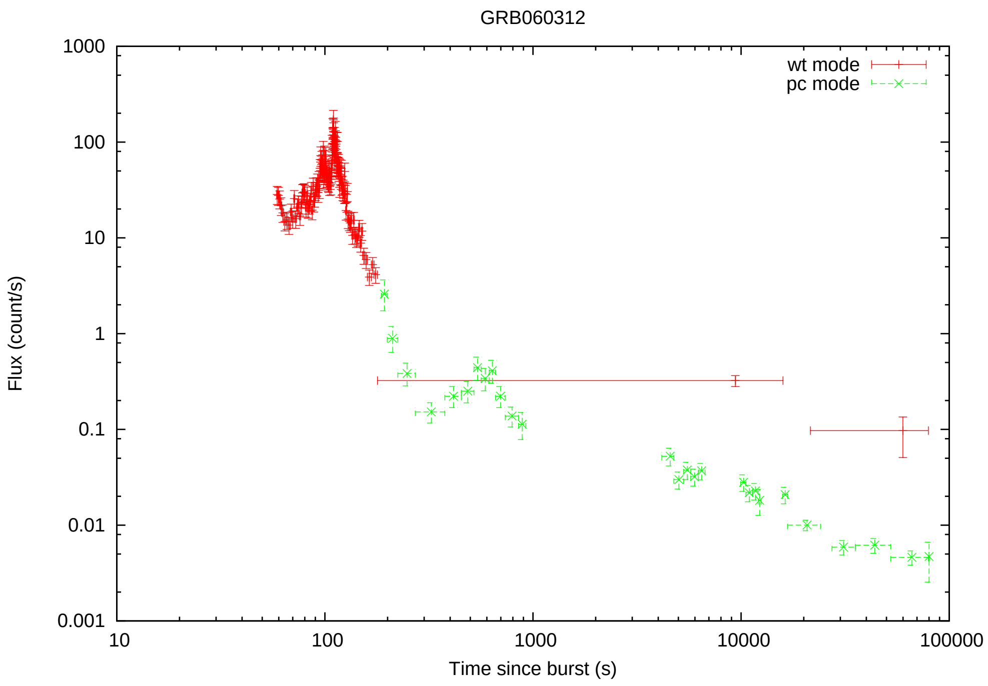

GRB060211B
wt mode
pc mode
Flux (count/s)
10
1
0.1
0.01
0.001
10
100
1000
10000
100000
Time since burst (s)

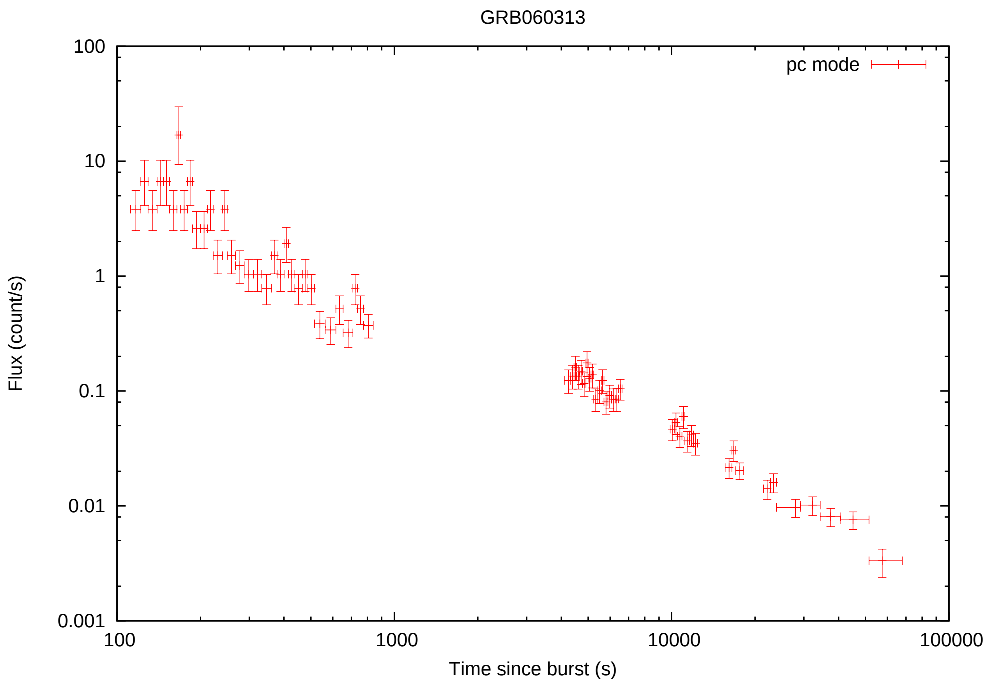

GRB060218
wt mode
pc mode
Flux (count/s)
Time since burst (s)
1000
100
10
1
0.1
0.01
0.001
100
1000
10000
100000
1e+06

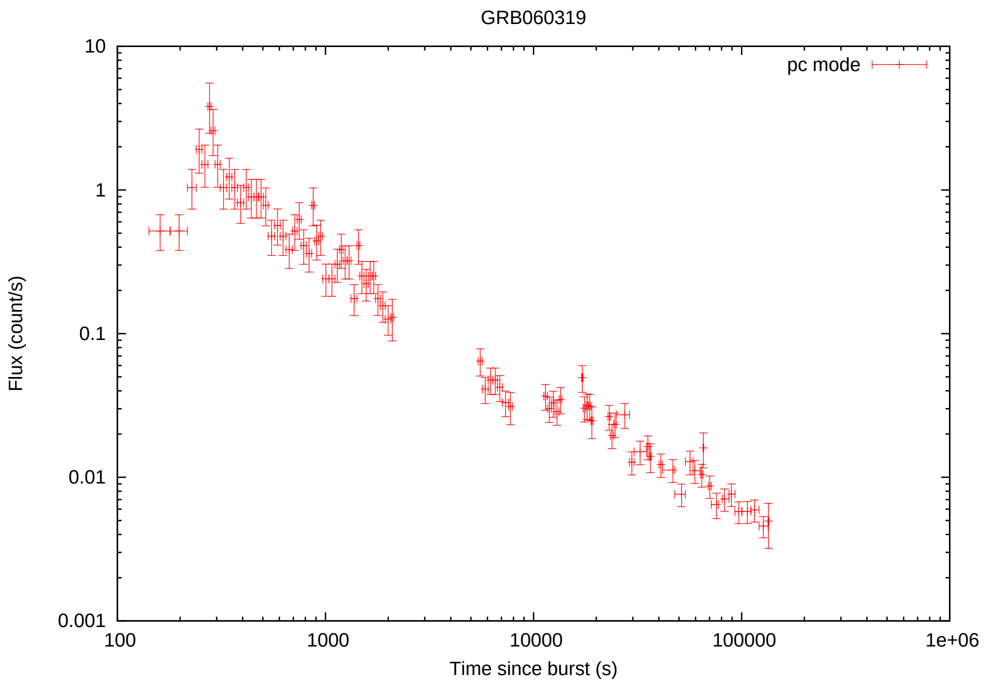

GRB060219
pc mode
Flux (count/s)
Time since burst (s)
10
1
0.1
0.01
0.001
100
1000
10000
100000

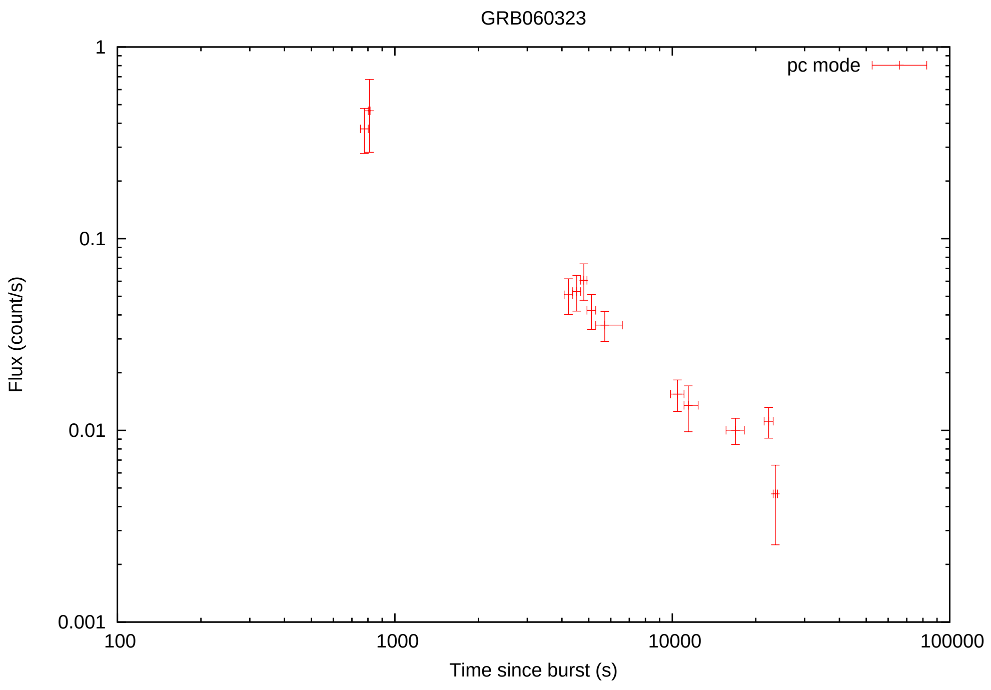

GRB060223A
pc mode
Flux (count/s)
10
1
0.1
0.01
0.001
10
100
1000
10000
100000
Time since burst (s)

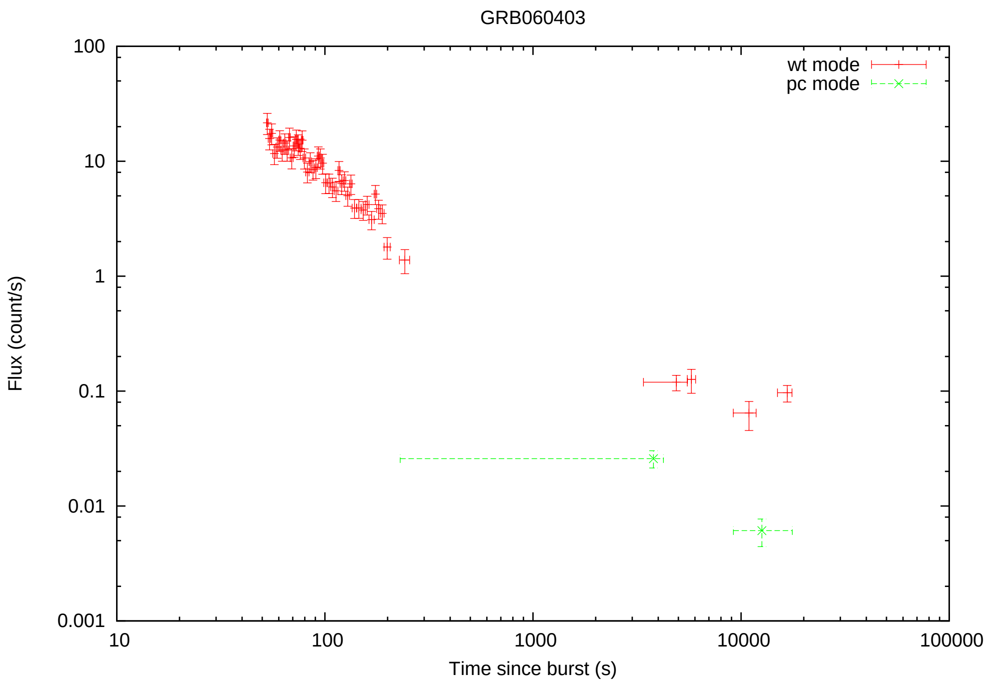

GRB060306
wt mode
pc mode
Flux (count/s)
100
10
1
0.1
0.01
0.001
10
100
1000
10000
100000
1e+06
Time since burst (s)

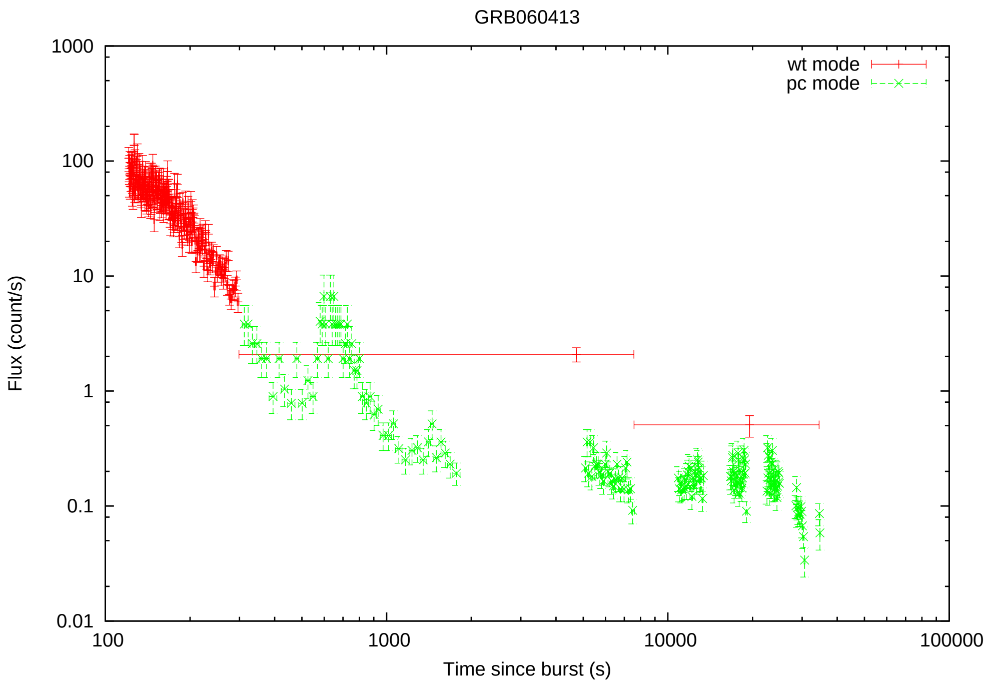

GRB060312
wt mode
pc mode
Flux (count/s)
Time since burst (s)
1000
100
10
1
0.1
0.01
0.001
10
100
1000
10000
100000

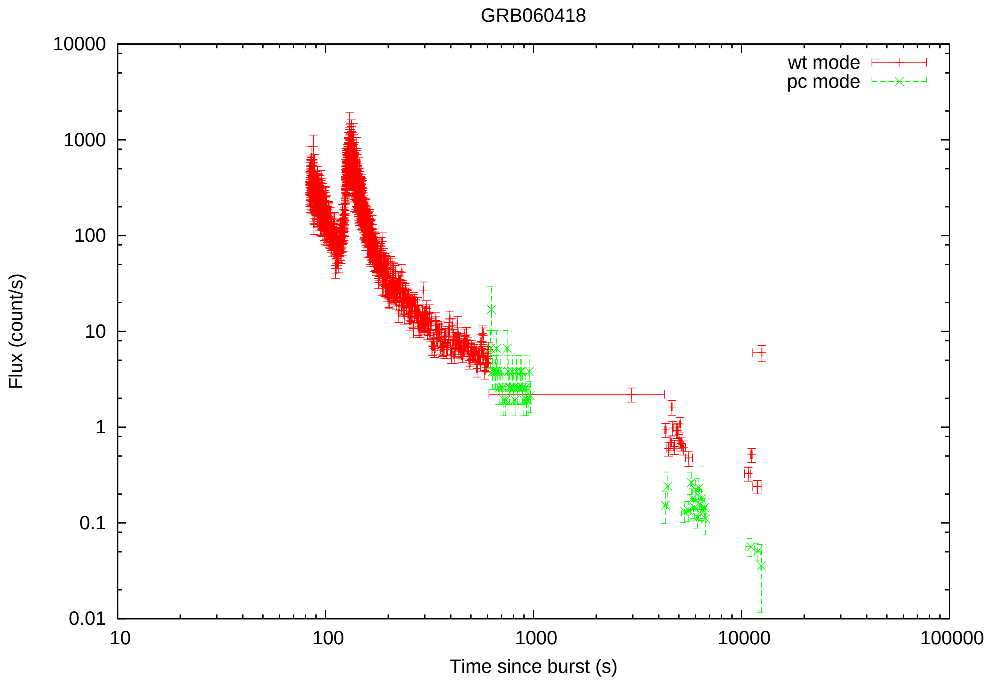

GRB060313
pc mode
Flux (count/s)
Time since burst (s)
100
10
1
0.1
0.01
0.001
100
1000
10000
100000

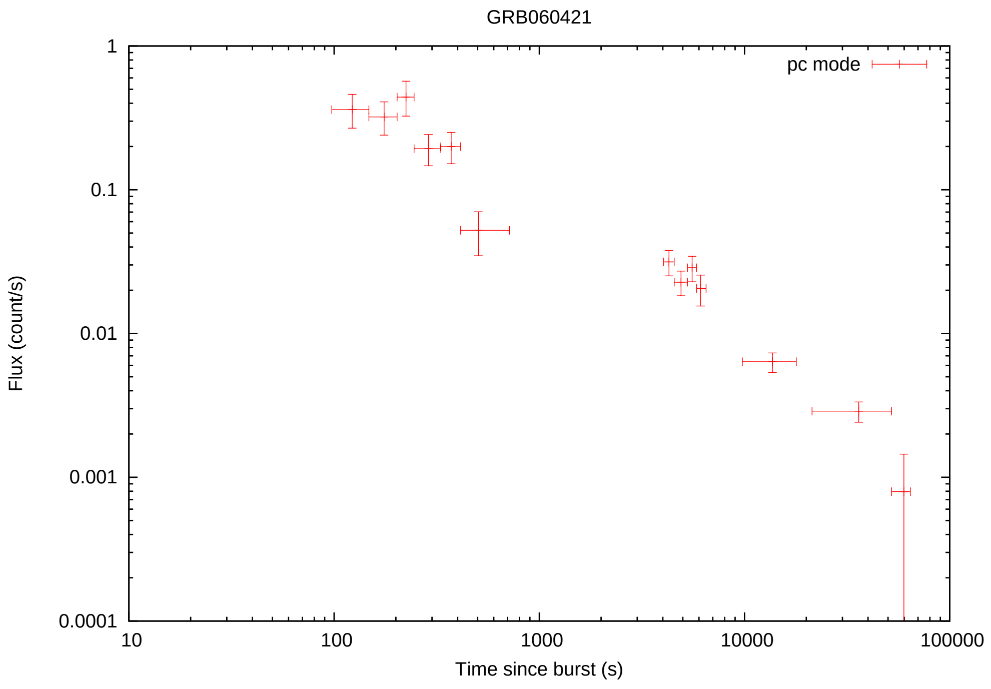

GRB060319
pc mode
Flux (count/s)
Time since burst (s)
10
1
0.1
0.01
0.001
100
1000
10000
100000
1e+06

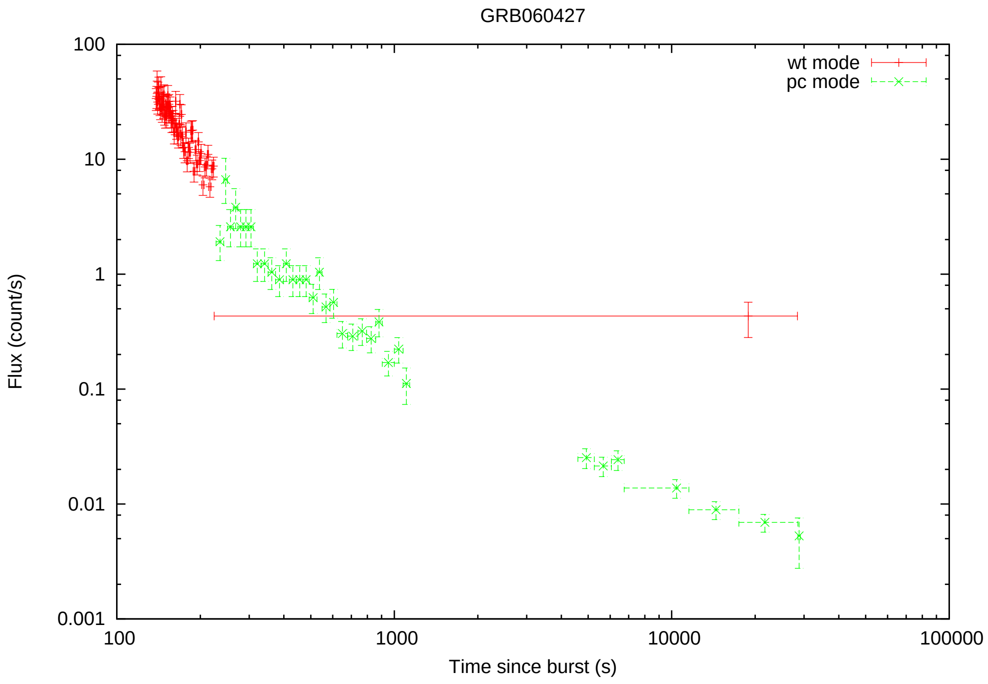

GRB060323
pc mode
Flux (count/s)
Time since burst (s)
1
0.1
0.01
0.001
100
1000
10000
100000

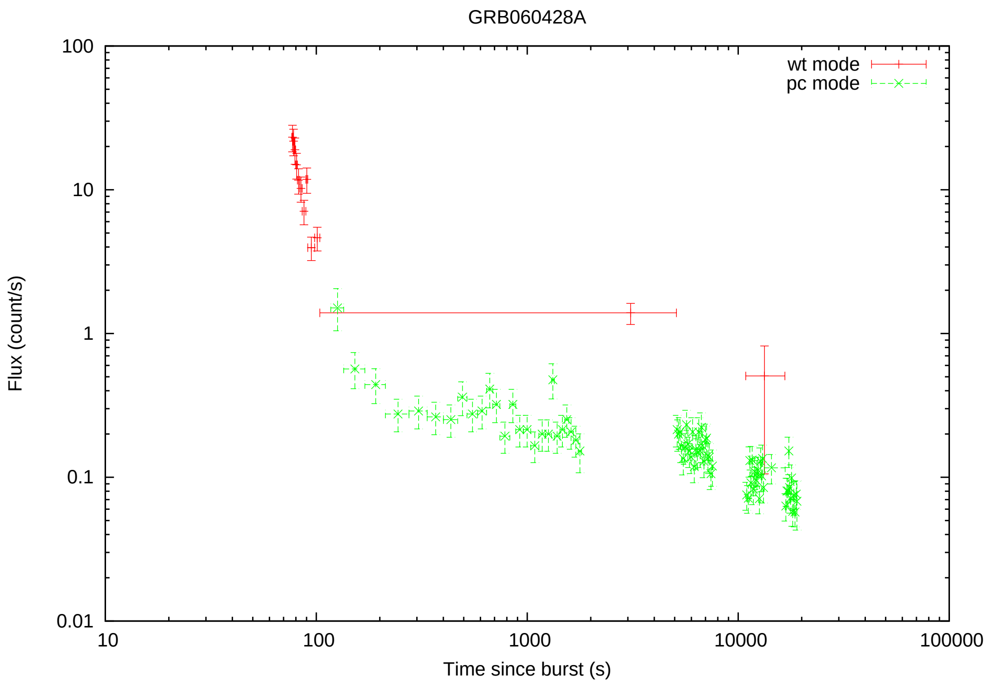

GRB060403
Flux (count/s)
Time since burst (s)
wt mode
pc mode
100
10
1
0.1
0.01
0.001
10
100
1000
10000
100000

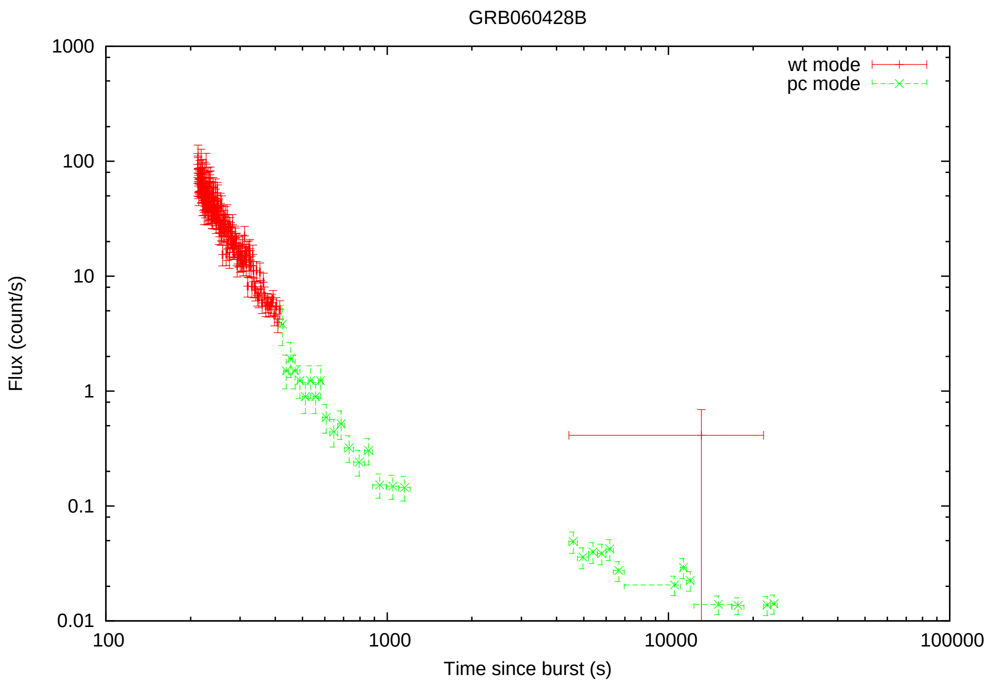

GRB060413
Flux (count/s)
Time since burst (s)
wt mode
pc mode
1000
100
10
1
0.1
0.01
100
1000
10000
100000

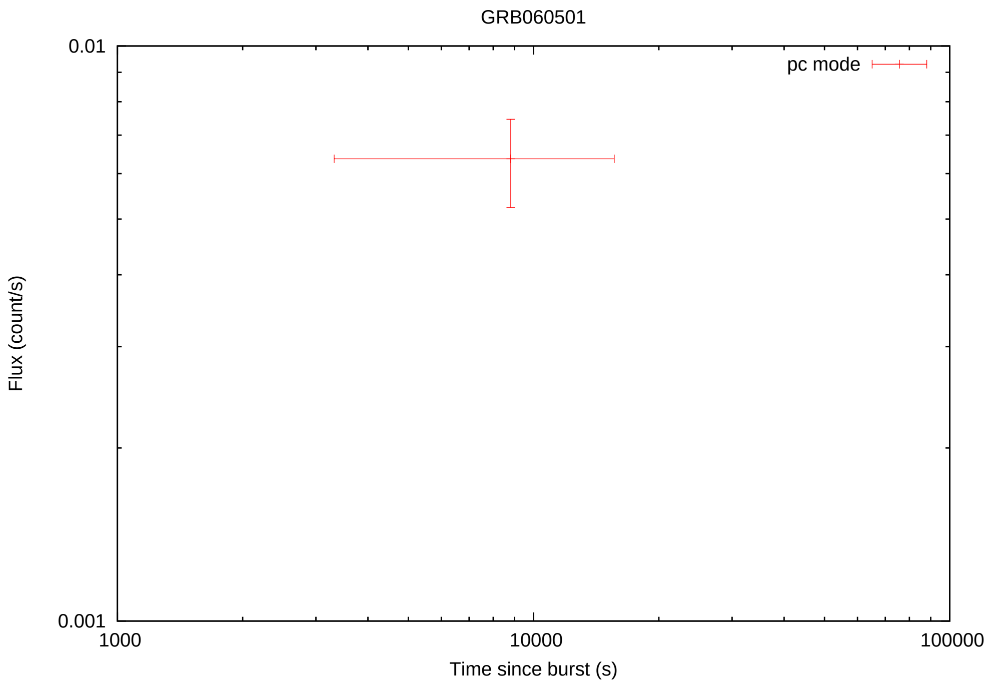

GRB060418
wt mode
pc mode
Flux (count/s)
Time since burst (s)
10000
1000
100
10
1
0.1
0.01
10
100
1000
10000
100000

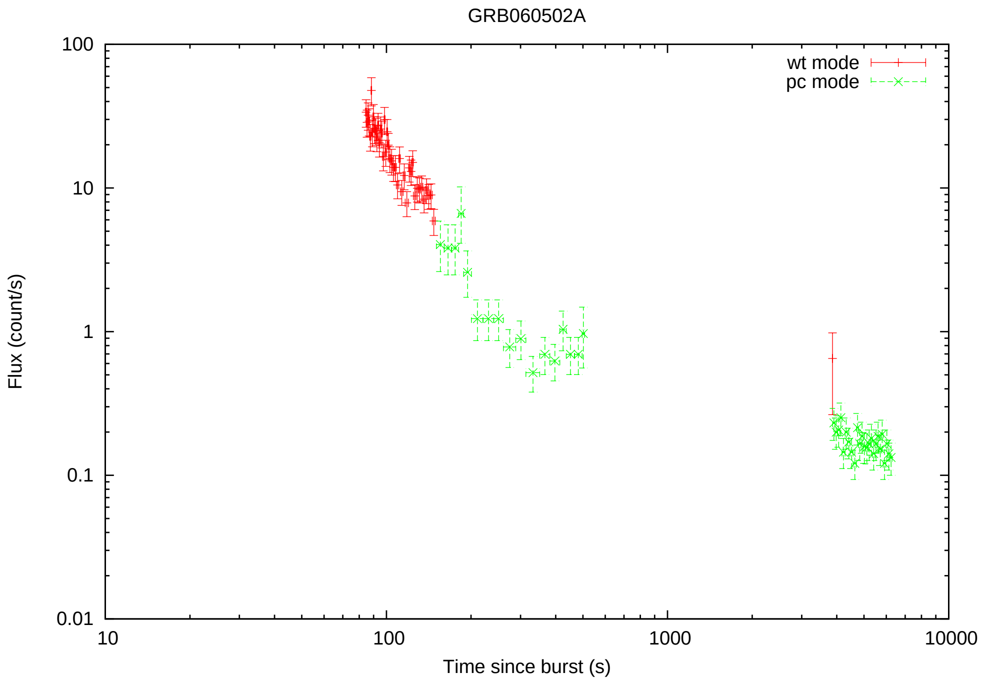

GRB060421
pc mode
Flux (count/s)
Time since burst (s)
1
0.1
0.01
0.001
0.0001
10
100
1000
10000
100000


GRB060427
wt mode
pc mode
Flux (count/s)
Time since burst (s)
100
10
1
0.1
0.01
0.001
100
1000
10000
100000

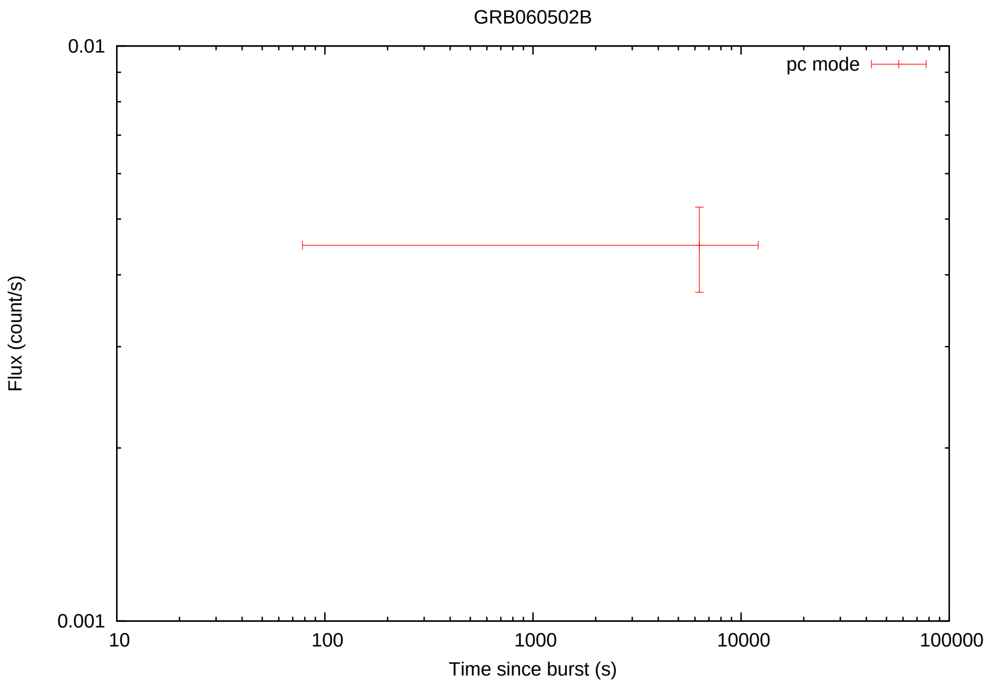

GRB060428A
wt mode
pc mode
Flux (count/s)
Time since burst (s)
100
10
1
0.1
0.01
10
100
1000
10000
100000


GRB060428B
wt mode
pc mode
Flux (count/s)
1000
100
10
1
0.1
0.01
100
1000
10000
100000
Time since burst (s)

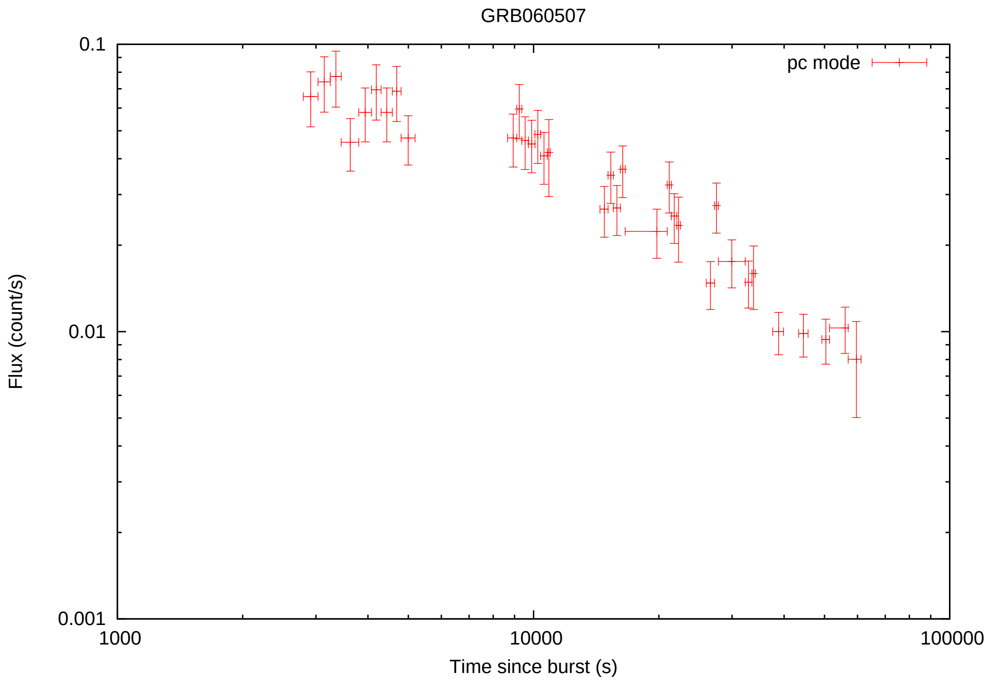

GRB060501
pc mode
Flux (count/s)
0.01
0.001
1000
10000
100000
Time since burst (s)

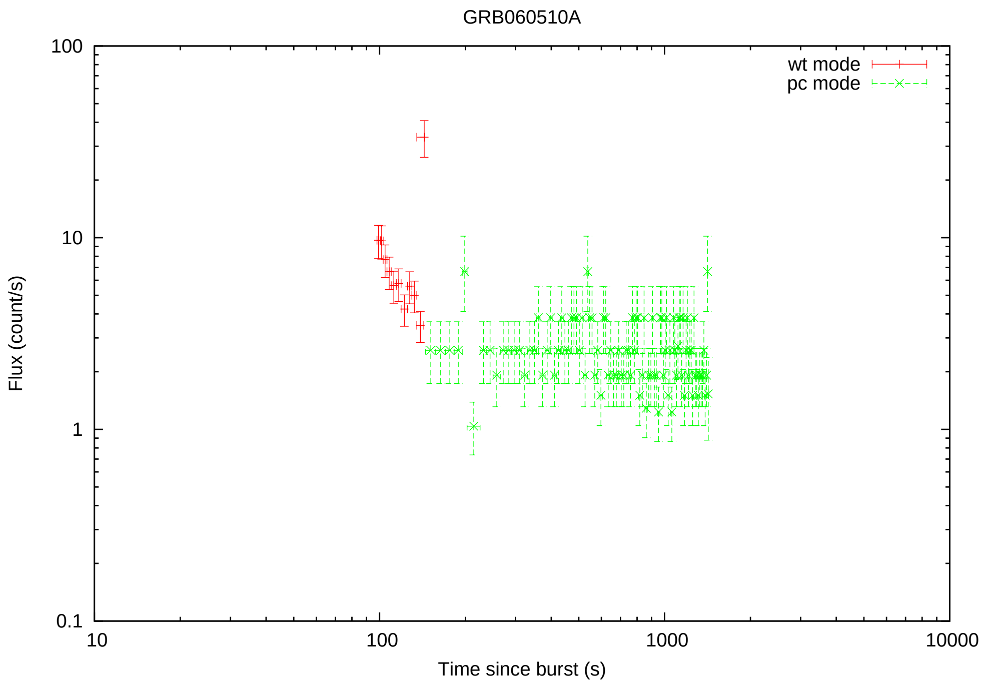


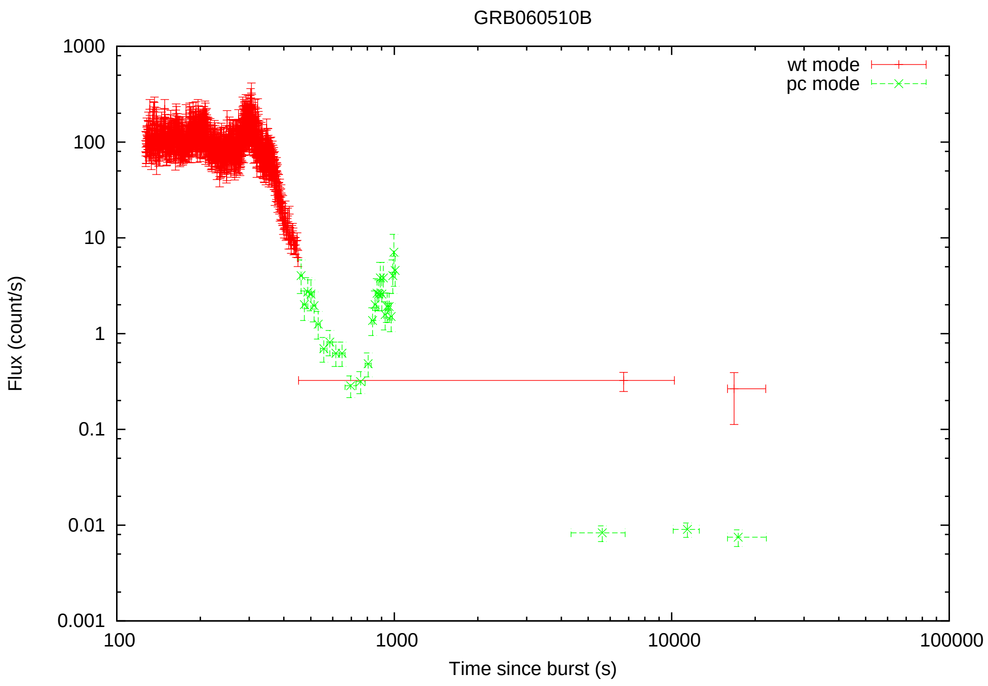
GRB060502A
wt mode
pc mode
Flux (count/s)
Time since burst (s)
100
10
1
0.1
0.01
10
100
1000
10000

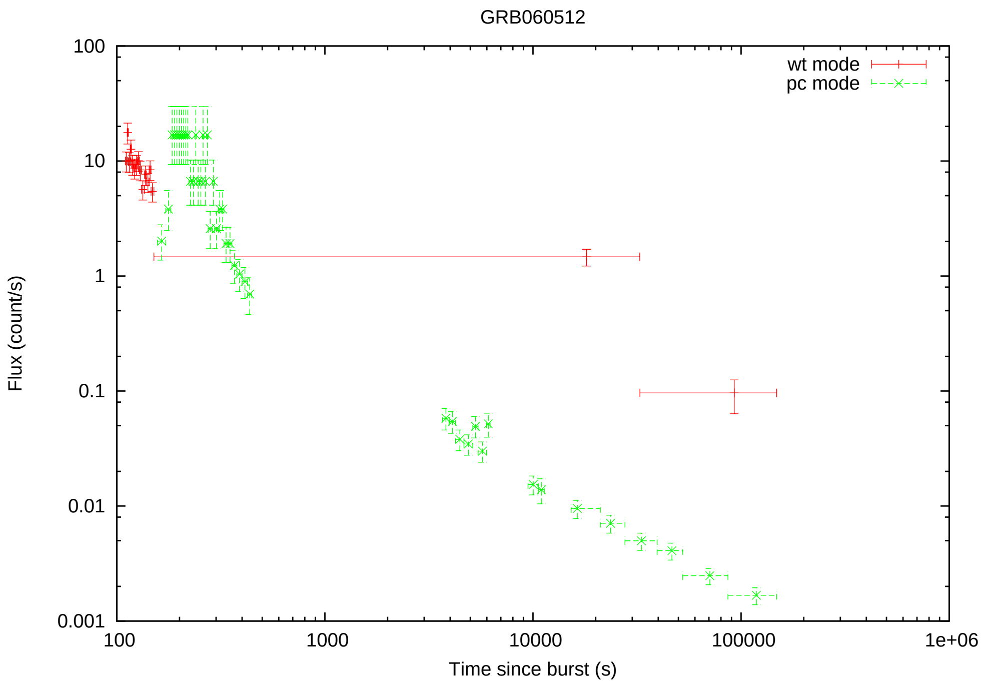


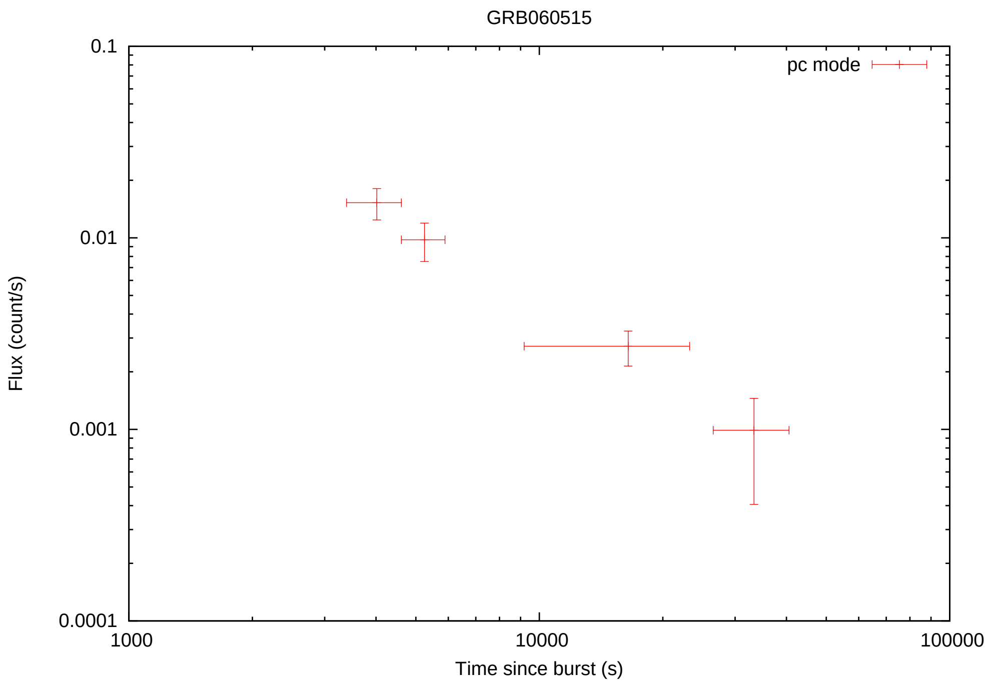
GRB060502B
pc mode
Flux (count/s)
Time since burst (s)
0.01
0.001
10
100
1000
10000
100000

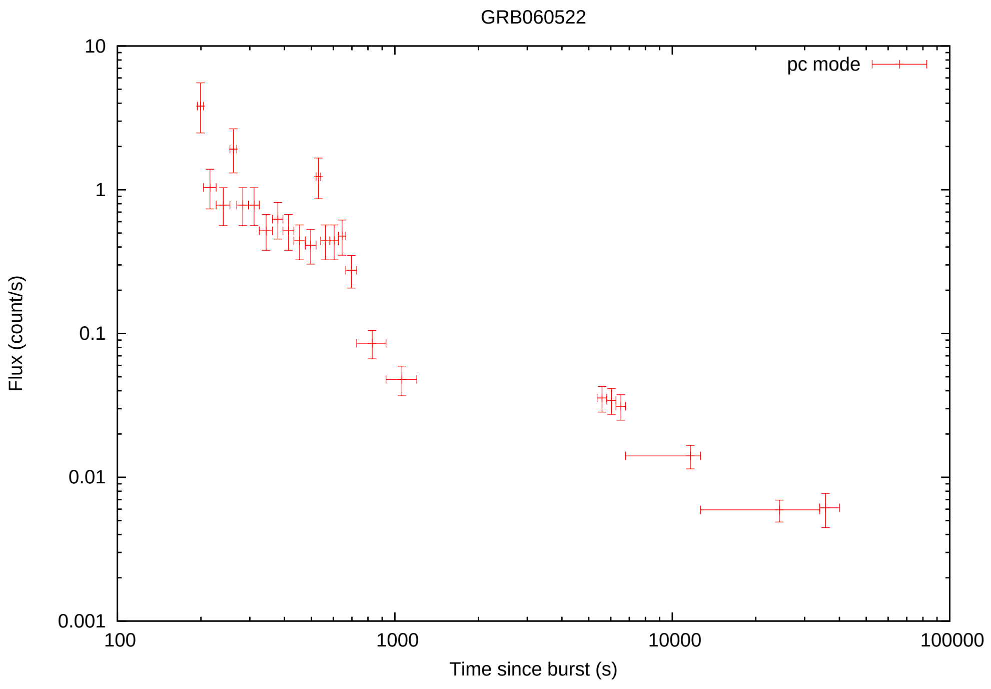

GRB060507
pc mode
Flux (count/s)
0.1
0.01
0.001
1000
10000
100000
Time since burst (s)

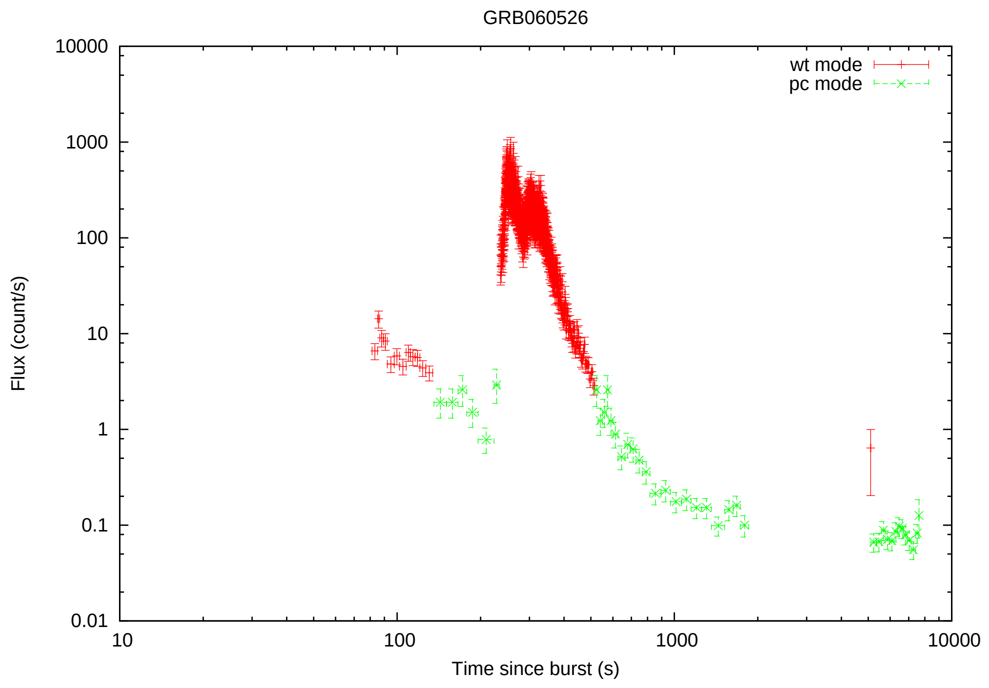

GRB060510A
wt mode
pc mode
Flux (count/s)
100
10
1
0.1
10
100
1000
10000
Time since burst (s)

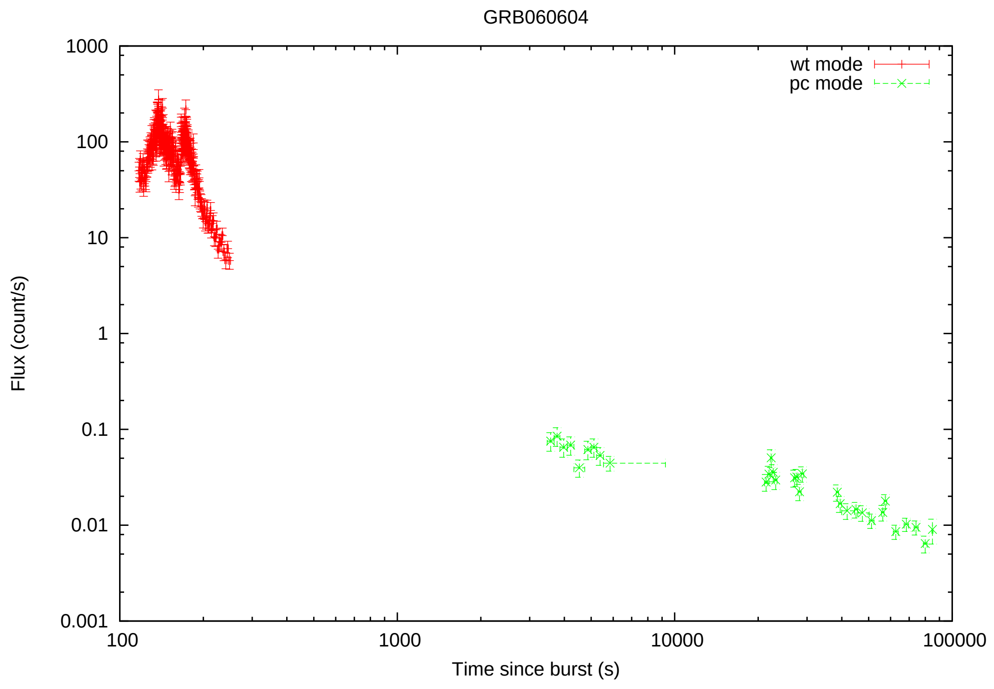

GRB060510B
Flux (count/s)
Time since burst (s)
wt mode
pc mode
1000
100
10
1
0.1
0.01
0.001
100
1000
10000
100000

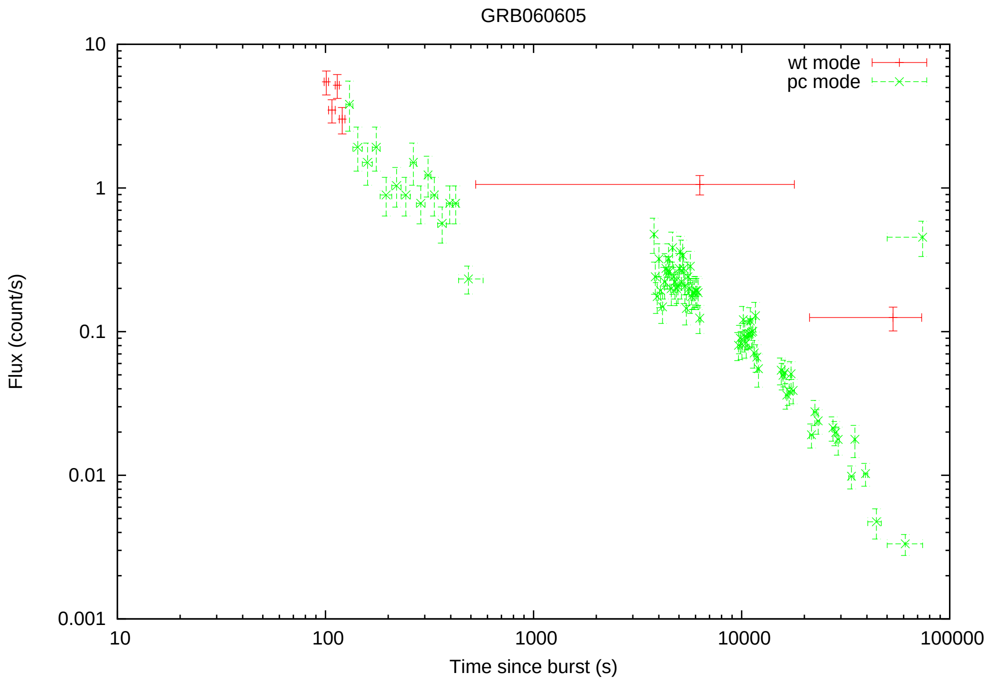

GRB060512
Flux (count/s)
Time since burst (s)
wt mode
pc mode
100
10
1
0.1
0.01
0.001
100
1000
10000
100000
1e+06

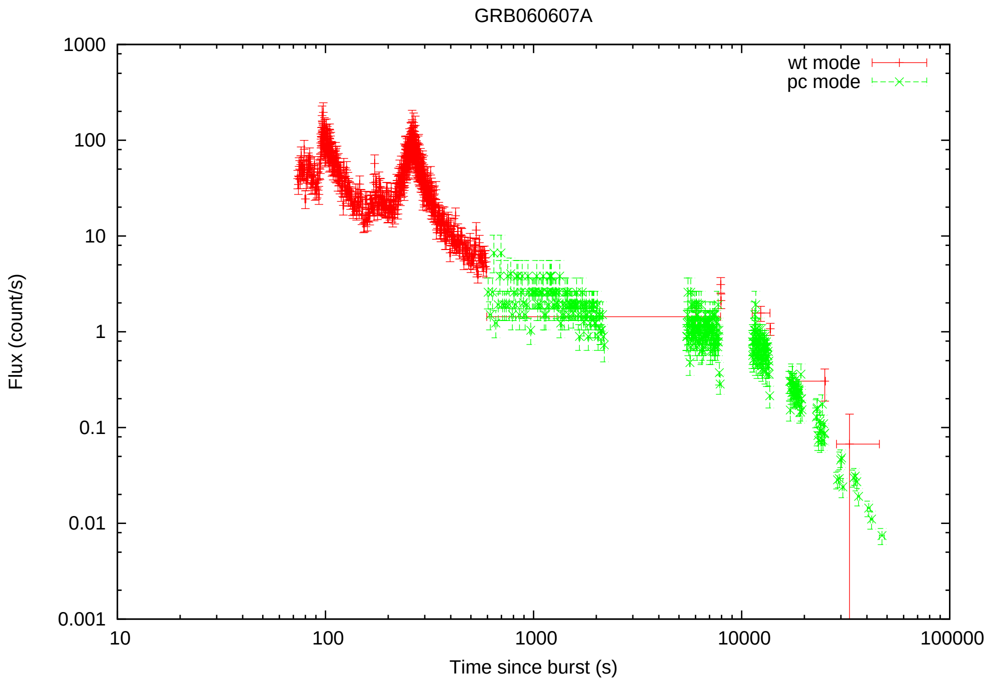

GRB060515
pc mode
Flux (count/s)
0.1
0.01
0.001
0.0001
1000
10000
100000
Time since burst (s)

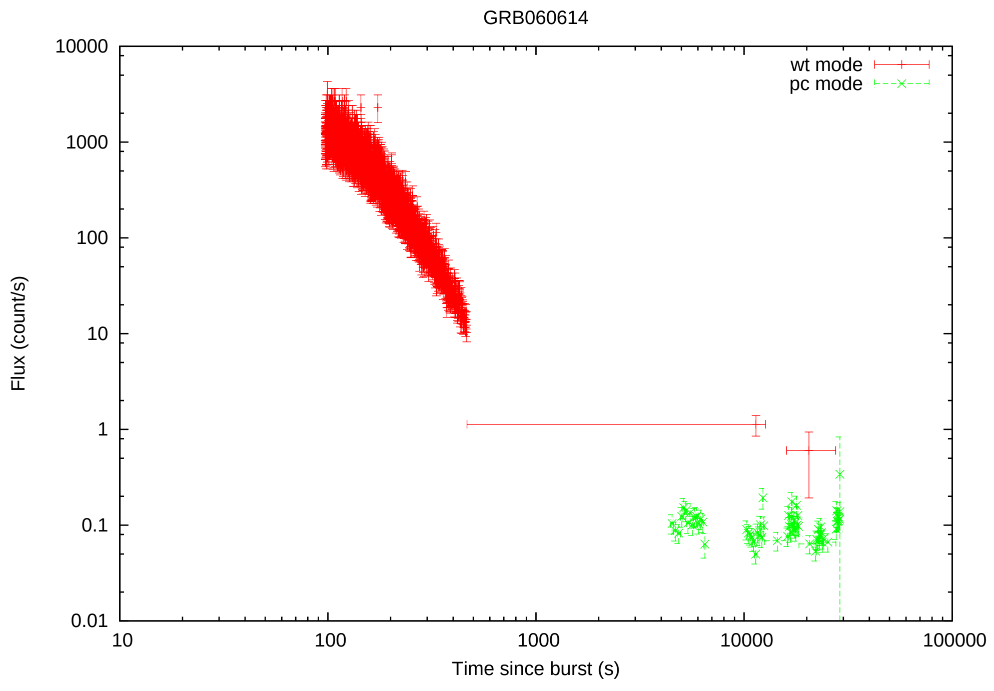

GRB060522
pc mode
Flux (count/s)
10
1
0.1
0.01
0.001
100
1000
10000
100000
Time since burst (s)

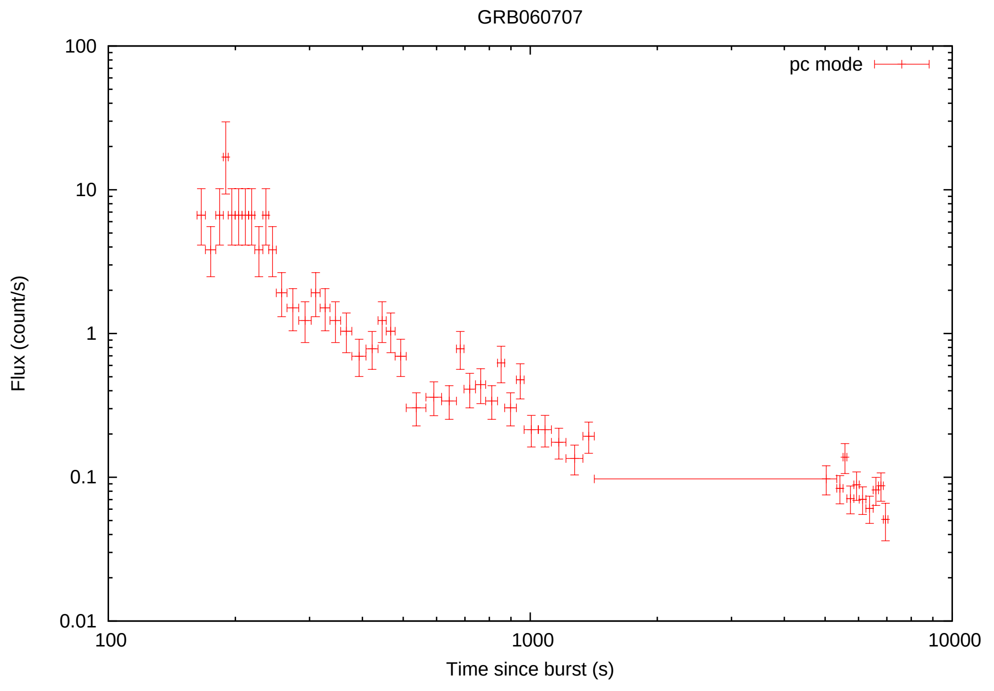

GRB060526
wt mode
pc mode
Flux (count/s)
10000
1000
100
10
1
0.1
0.01
10
100
1000
10000
Time since burst (s)

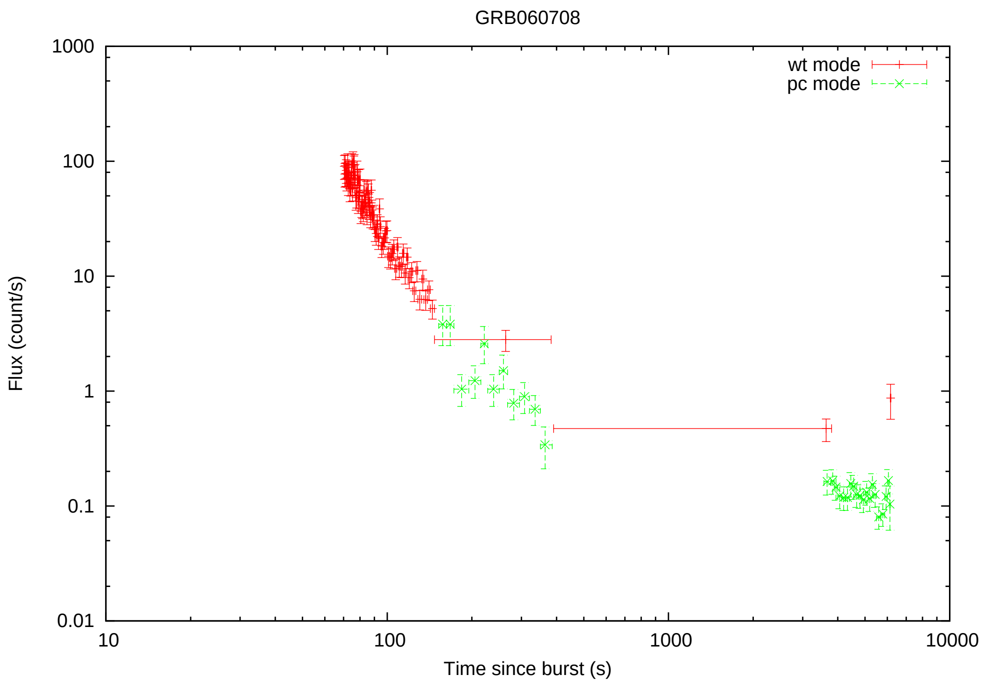

GRB060604
wt mode
pc mode
Flux (count/s)
1000
100
10
1
0.1
0.01
0.001
100
1000
10000
100000
Time since burst (s)

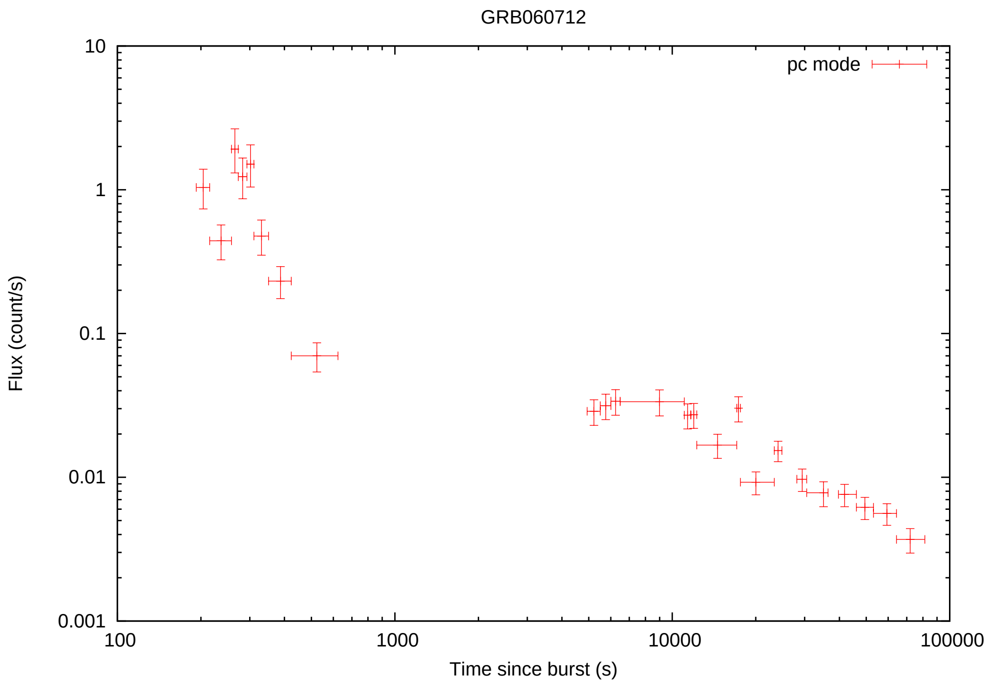

GRB060605
Flux (count/s)
Time since burst (s)
wt mode
pc mode
10
1
0.1
0.01
0.001
10
100
1000
10000
100000

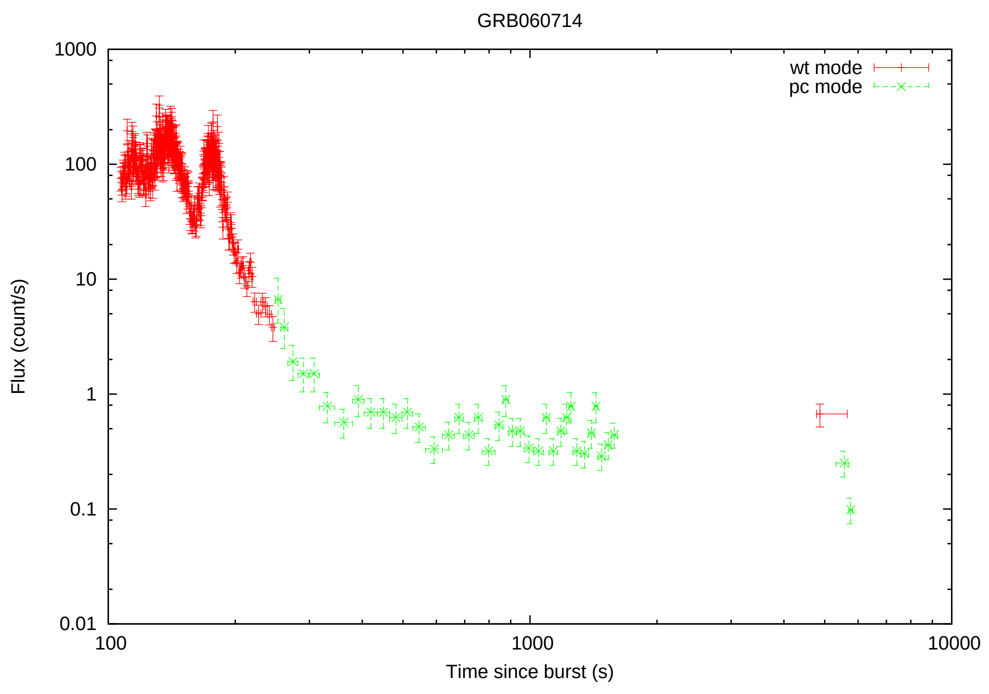

GRB060607A
Flux (count/s)
Time since burst (s)
wt mode
pc mode
1000
100
10
1
0.1
0.01
0.001
10
100
1000
10000
100000

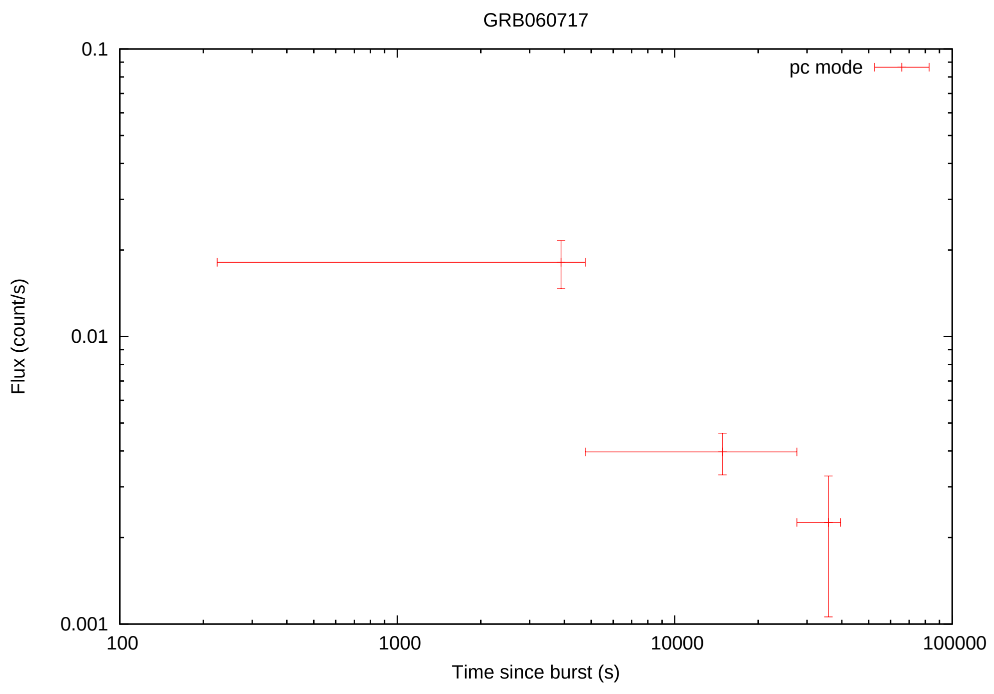

GRB060614
wt mode
pc mode
Flux (count/s)
Time since burst (s)
10000
1000
100
10
1
0.1
0.01
10
100
1000
10000
100000

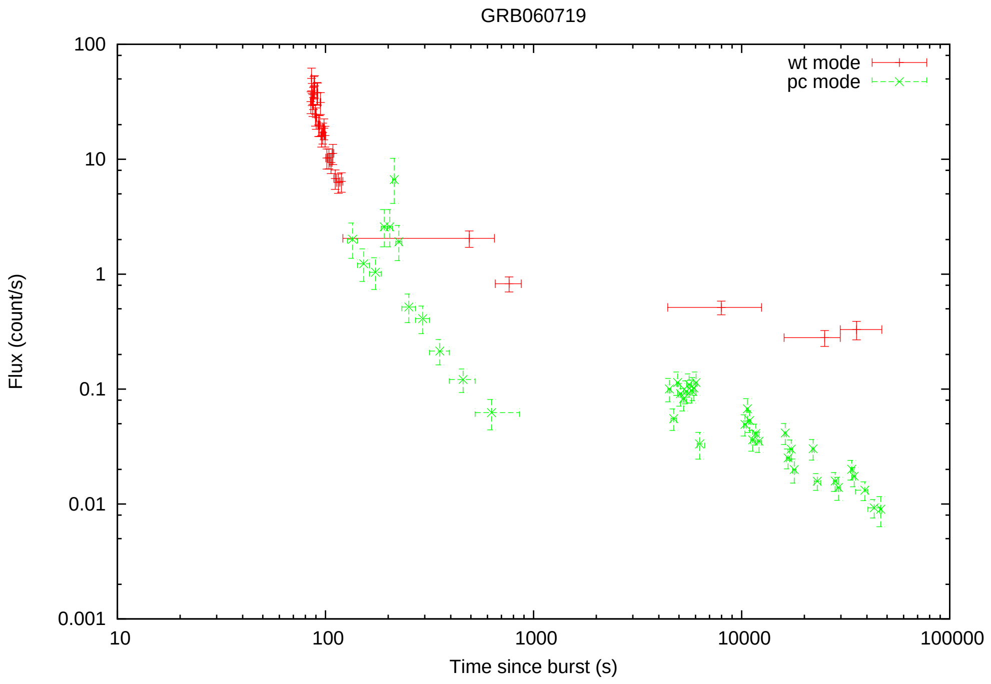

GRB060707
pc mode
Flux (count/s)
Time since burst (s)
100
10
1
0.1
0.01
100
1000
10000

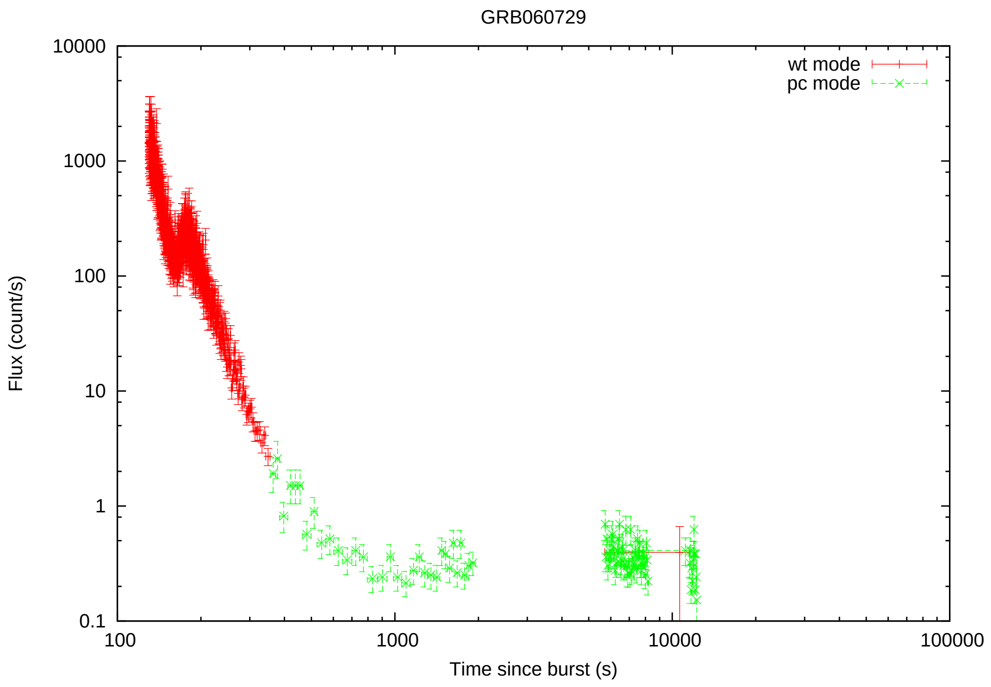

GRB060708
wt mode
pc mode
Flux (count/s)
1000
100
10
1
0.1
0.01
10
100
1000
10000
Time since burst (s)

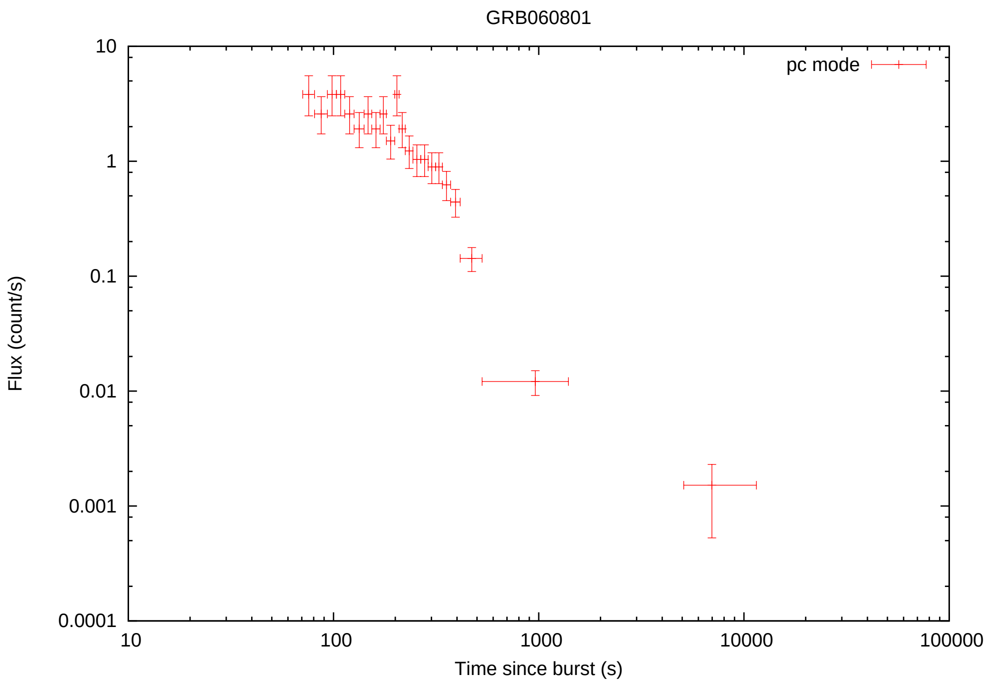

GRB060712
pc mode
Flux (count/s)
10
1
0.1
0.01
0.001
100
1000
10000
100000
Time since burst (s)

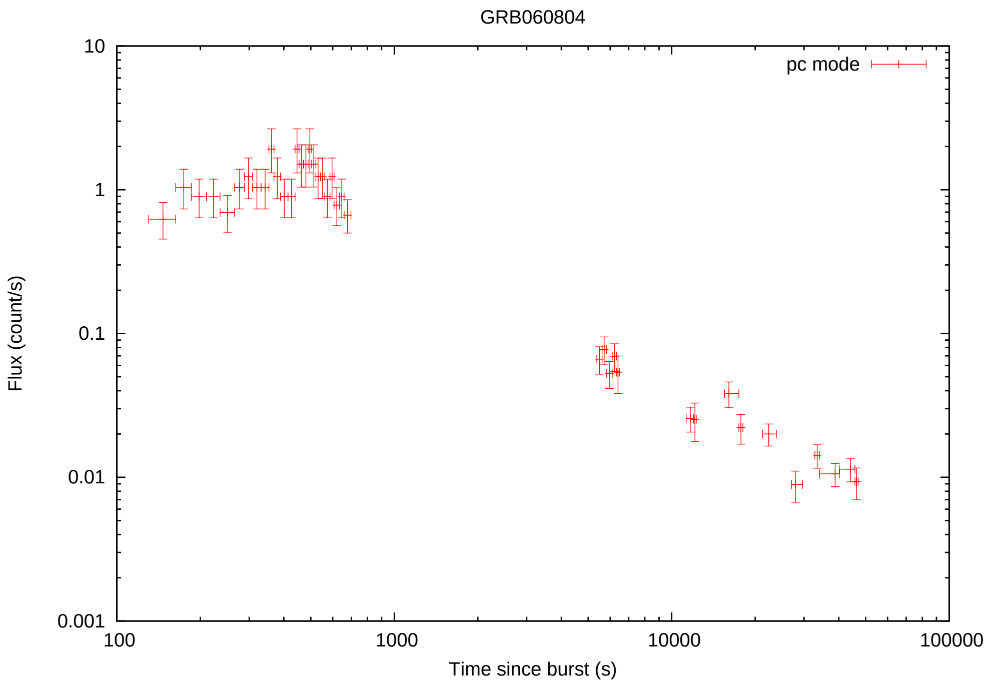

GRB060714
Flux (count/s)
Time since burst (s)
wt mode
pc mode
1000
100
10
1
0.1
0.01
100
1000
10000

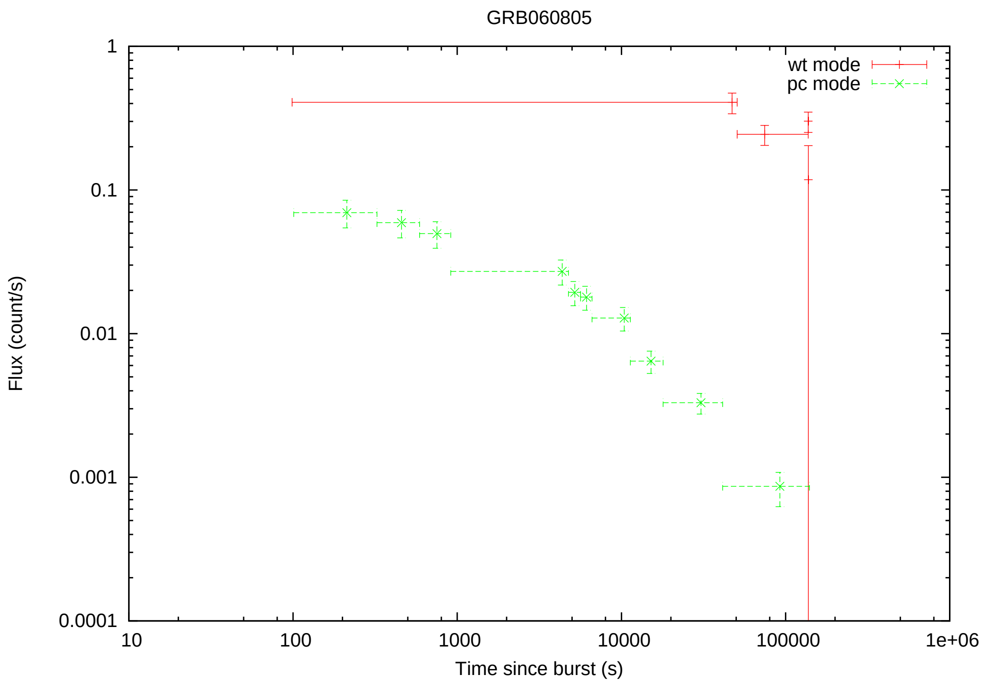

GRB060717
Flux (count/s)
Time since burst (s)
pc mode
0.1
0.01
0.001
100
1000
10000
100000

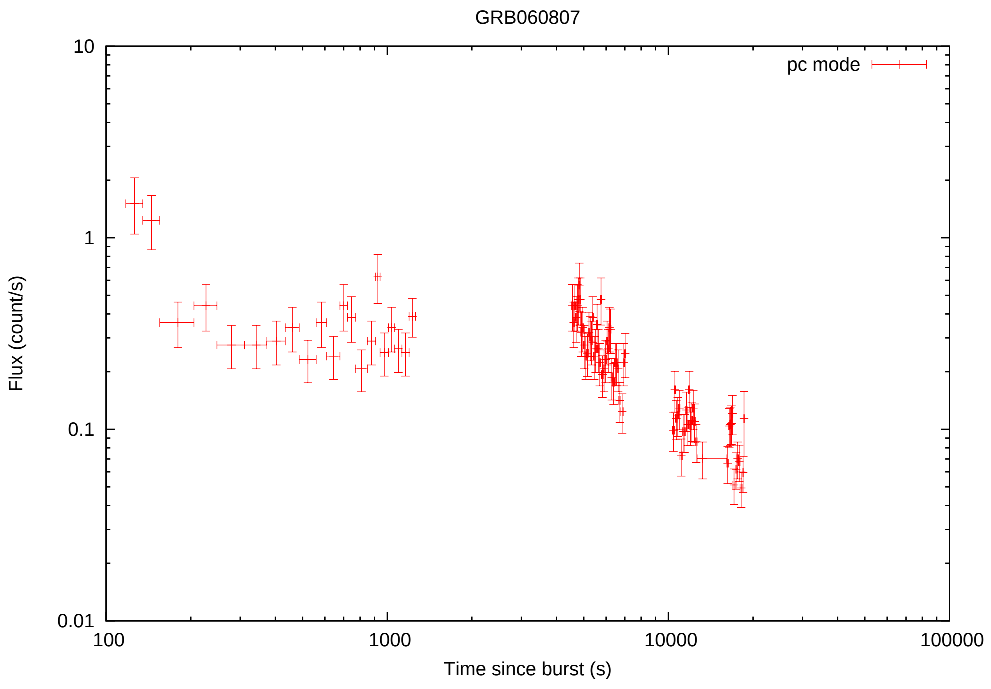

GRB060719
wt mode
pc mode
Flux (count/s)
Time since burst (s)
100
10
1
0.1
0.01
0.001
10
100
1000
10000
100000

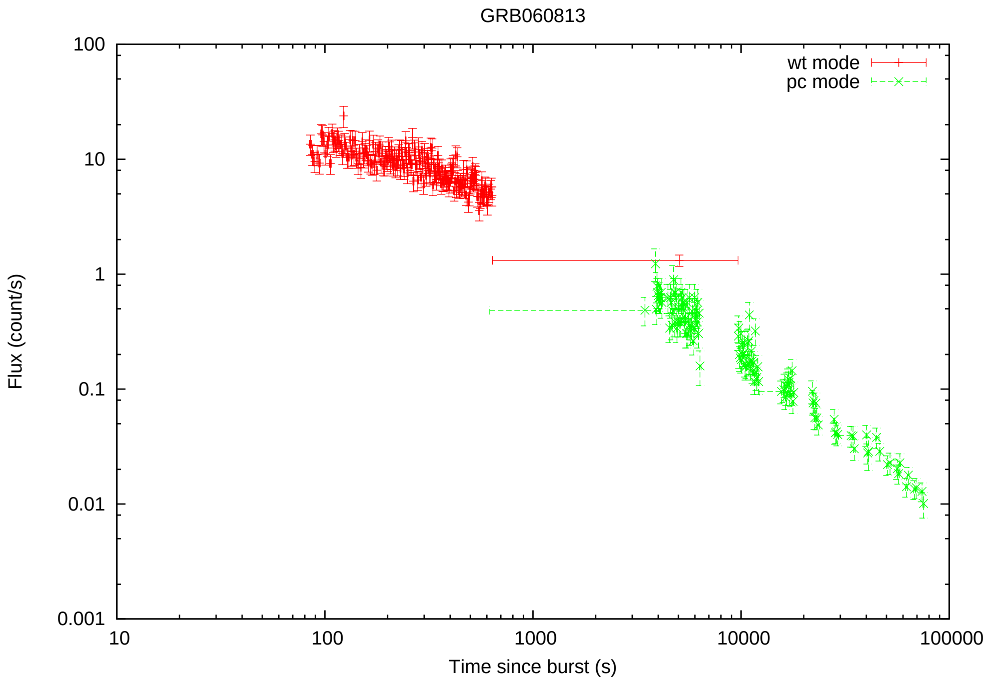

GRB060729
wt mode
pc mode
Flux (count/s)
Time since burst (s)
10000
1000
100
10
1
0.1
100
1000
10000
100000

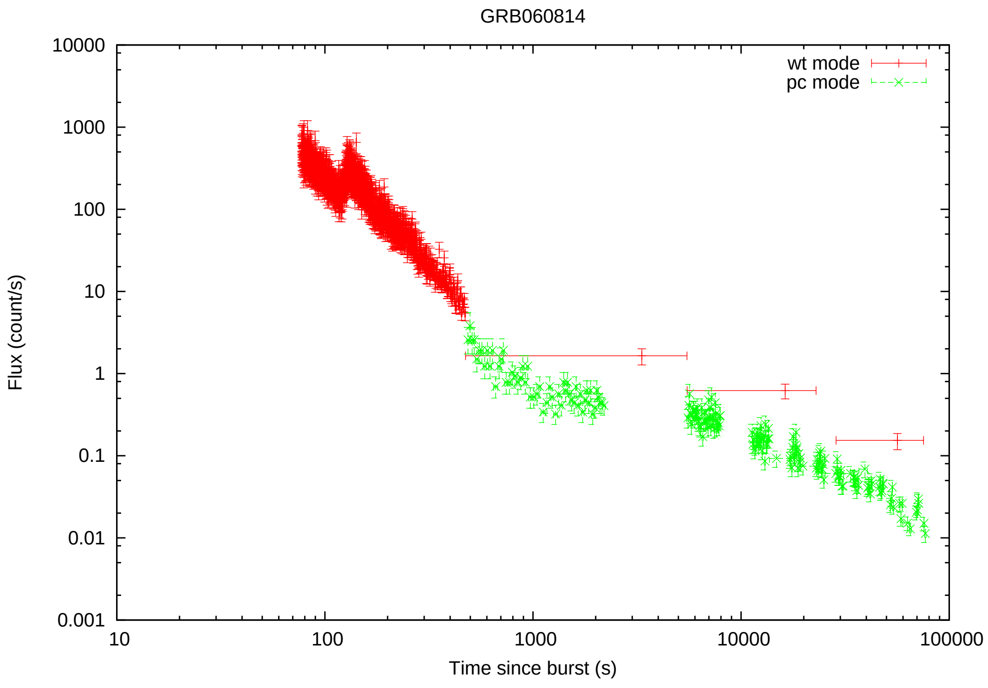

GRB060801
pc mode
Flux (count/s)
Time since burst (s)
10
1
0.1
0.01
0.001
0.0001
10
100
1000
10000
100000

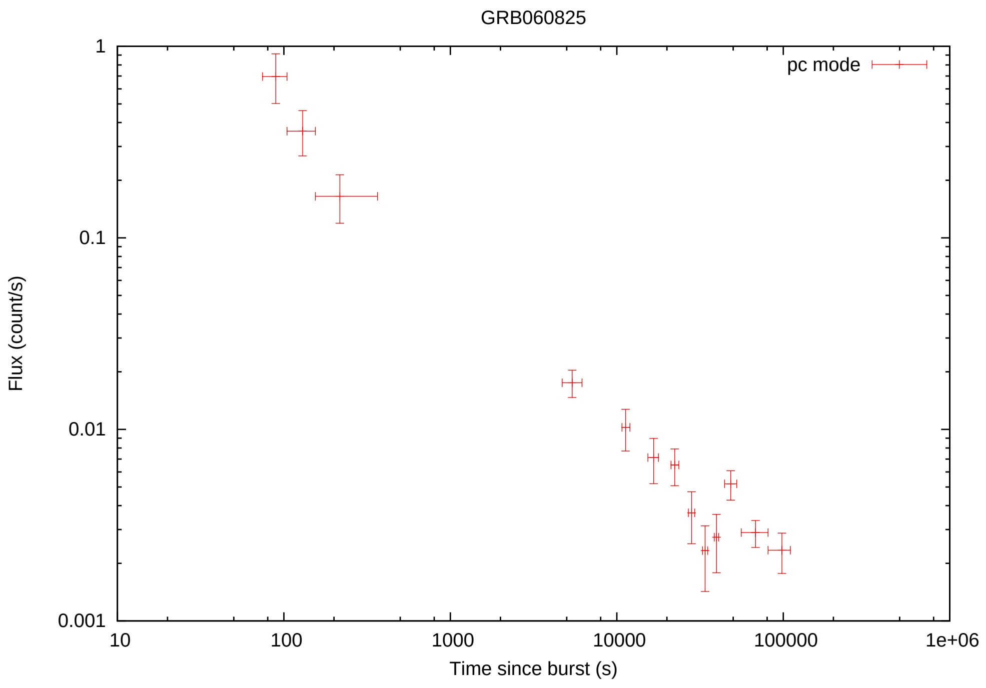

GRB060804
pc mode
Flux (count/s)
Time since burst (s)
10
1
0.1
0.01
0.001
100
1000
10000
100000

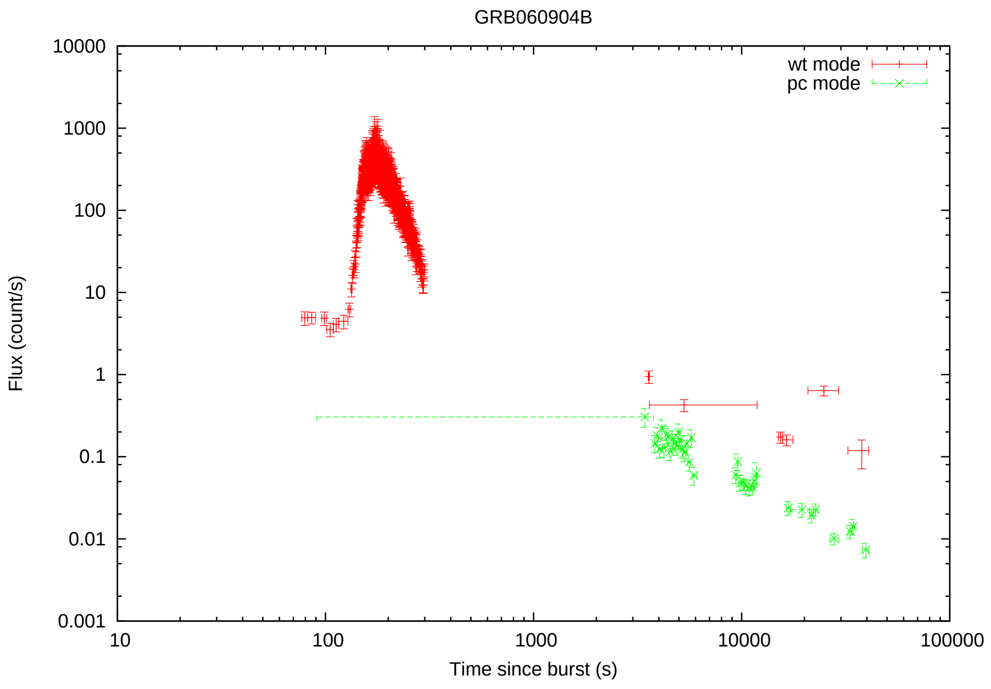

GRB060805
wt mode
pc mode
Flux (count/s)
Time since burst (s)
1
0.1
0.01
0.001
0.0001
10
100
1000
10000
100000
1e+06

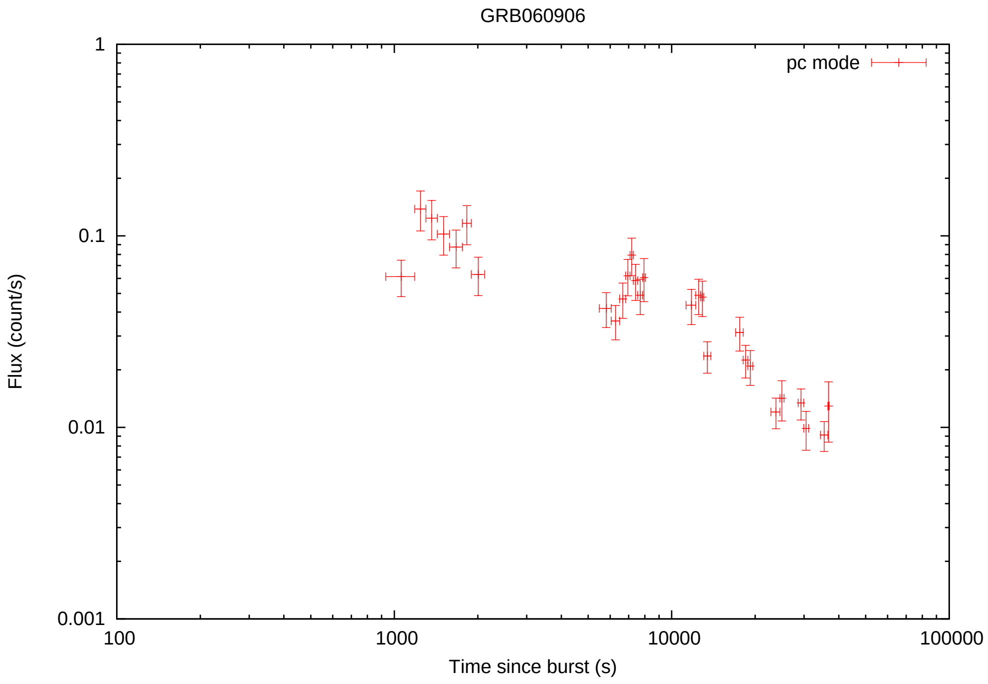

GRB060807
pc mode
Flux (count/s)
Time since burst (s)
10
1
0.1
0.01
100
1000
10000
100000

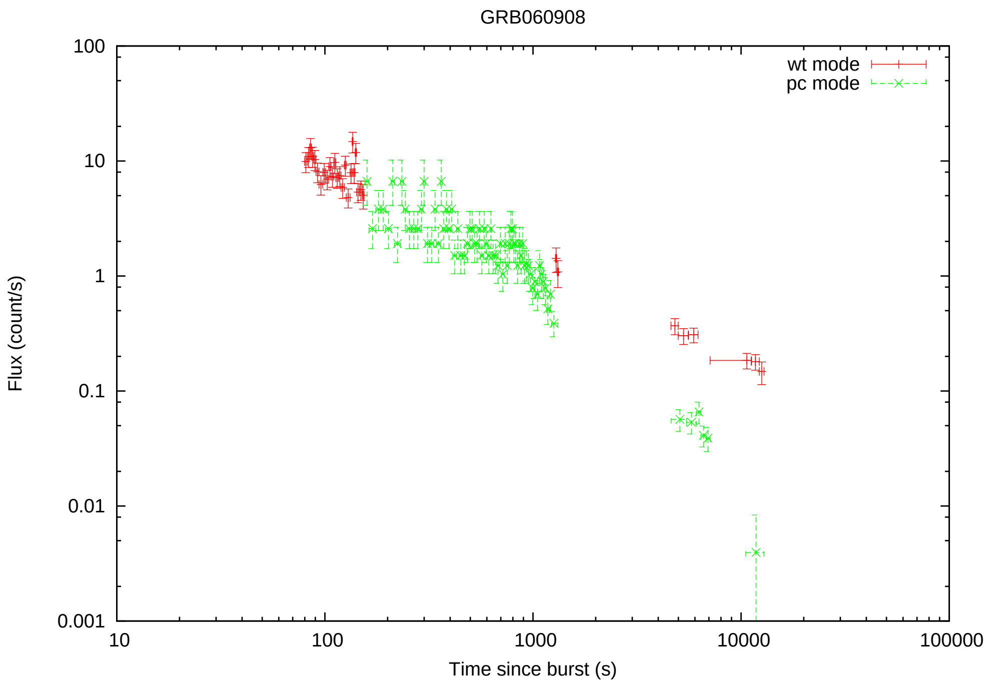

GRB060813
wt mode
pc mode
Flux (count/s)
Time since burst (s)
100
10
1
0.1
0.01
0.001
10
100
1000
10000
100000

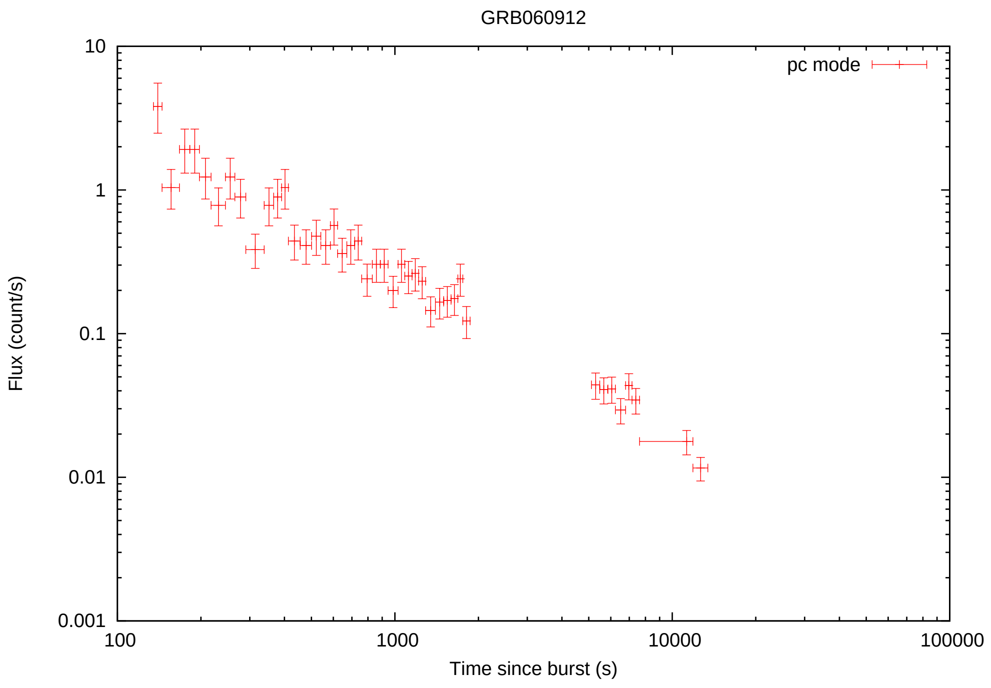

GRB060814
wt mode
pc mode
Flux (count/s)
Time since burst (s)
10000
1000
100
10
1
0.1
0.01
0.001
10
100
1000
10000
100000

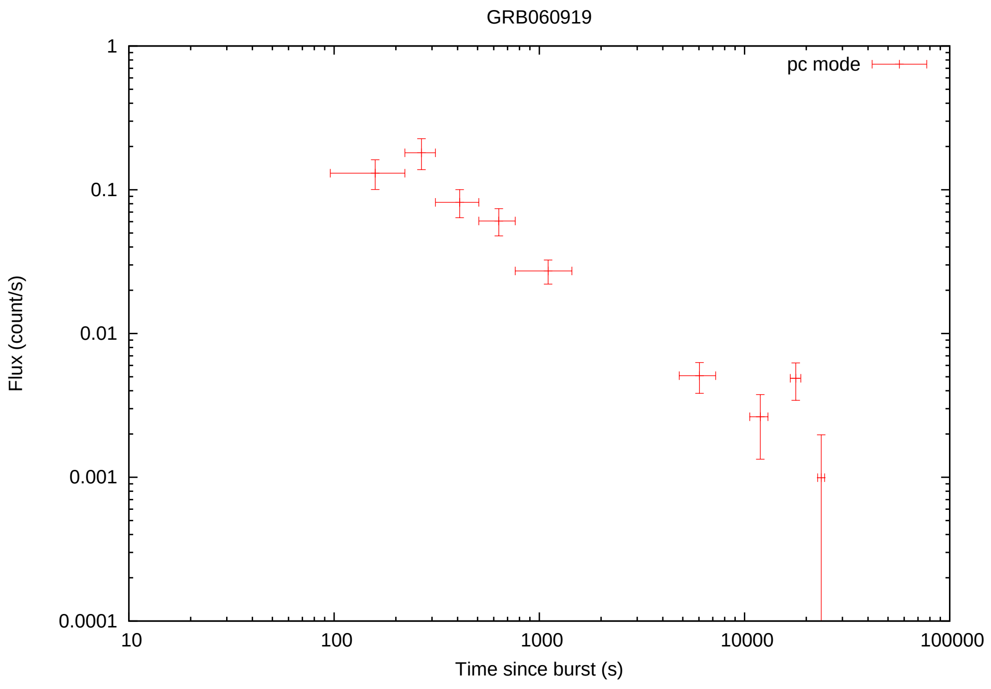

GRB060825
pc mode
Flux (count/s)
1
0.1
0.01
0.001
10
100
1000
10000
100000
1e+06
Time since burst (s)

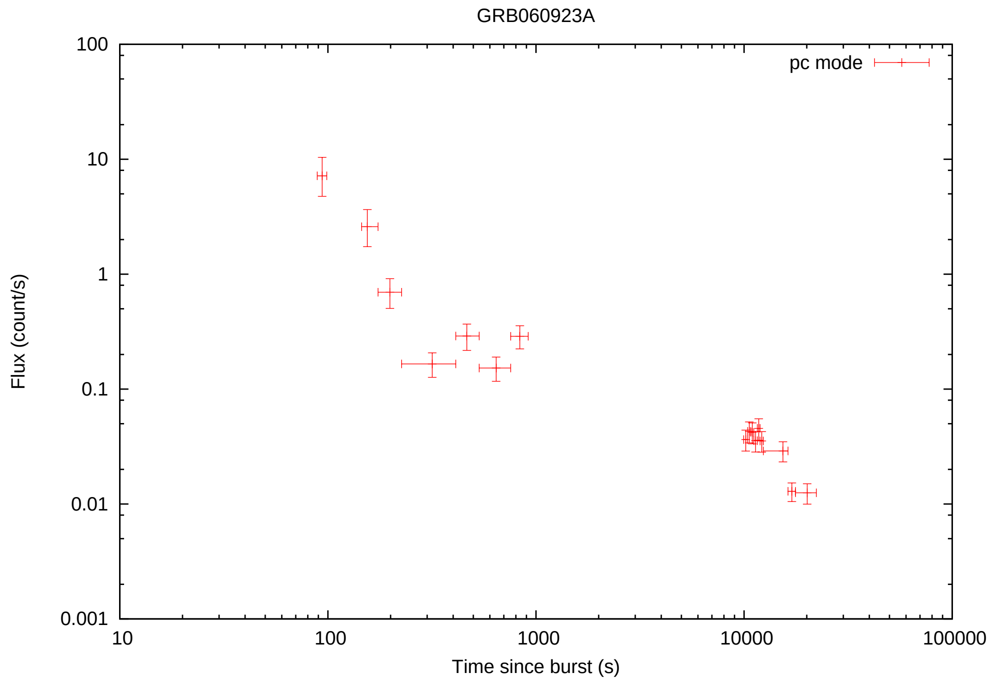

GRB060904B
wt mode
pc mode
Flux (count/s)
10000
1000
100
10
1
0.1
0.01
0.001
10
100
1000
10000
100000
Time since burst (s)

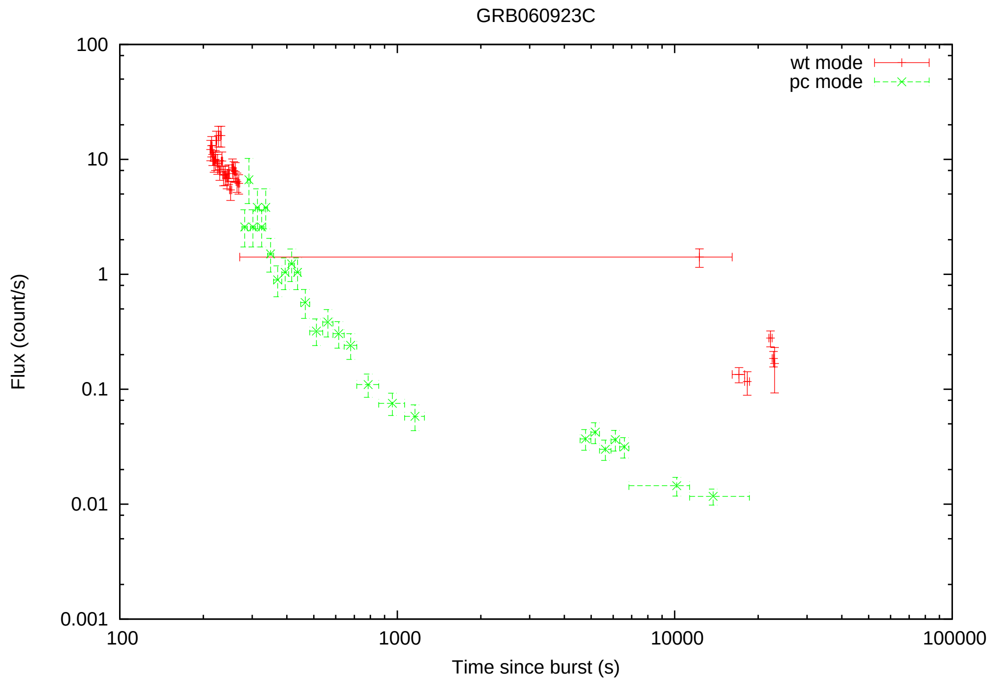

GRB060906
pc mode
Flux (count/s)
Time since burst (s)
1
0.1
0.01
0.001
100
1000
10000
100000

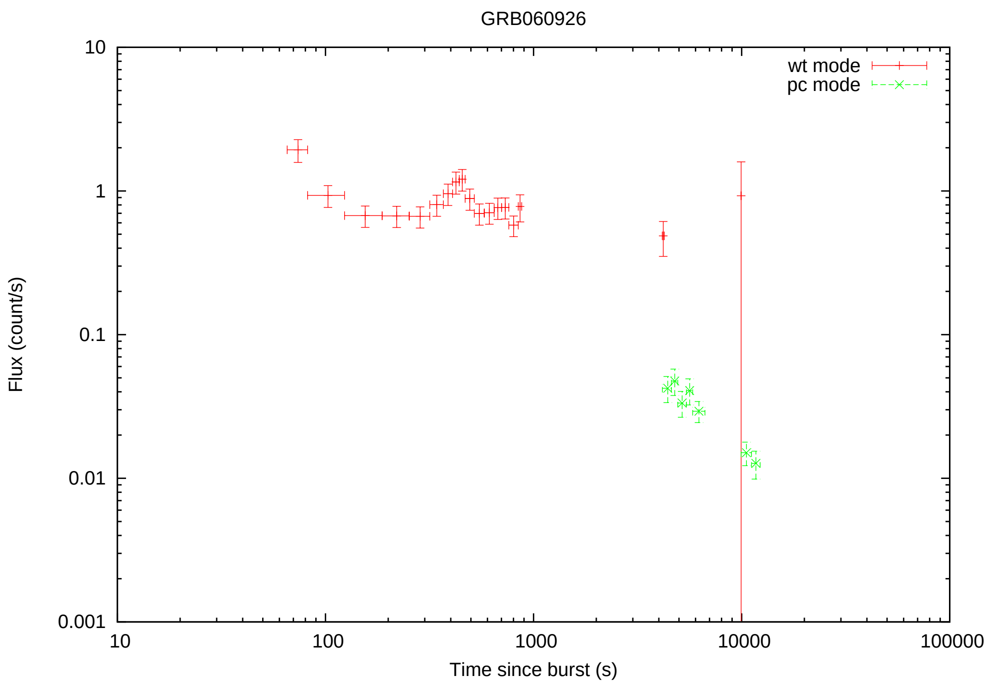

GRB060908
wt mode
pc mode
Flux (count/s)
Time since burst (s)
100
10
1
0.1
0.01
0.001
10
100
1000
10000
100000

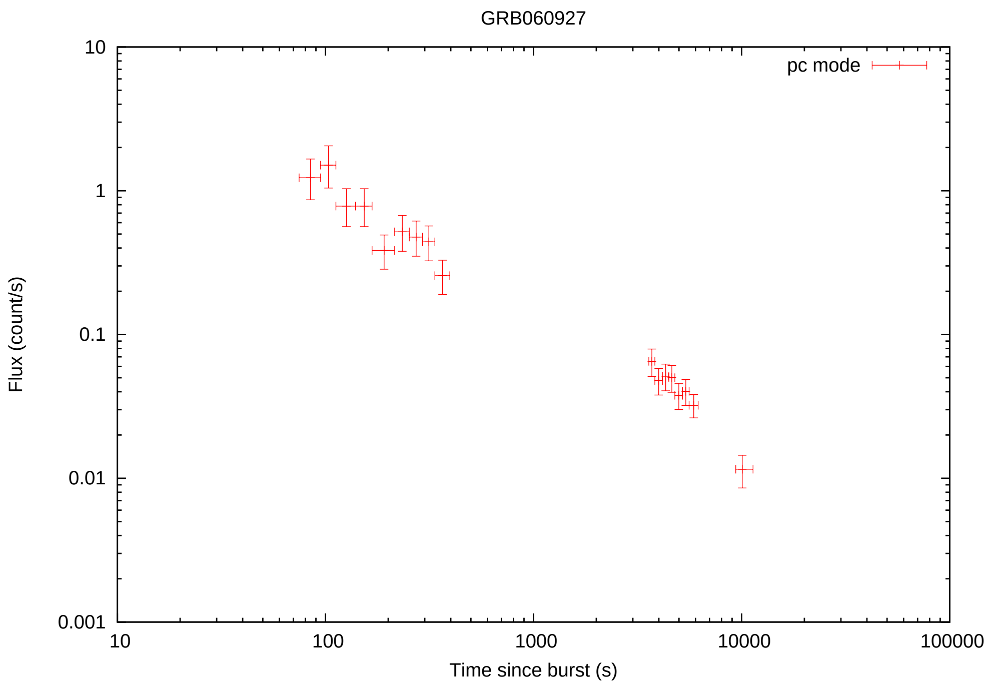

GRB060912
pc mode
Flux (count/s)
Time since burst (s)
10
1
0.1
0.01
0.001
100
1000
10000
100000

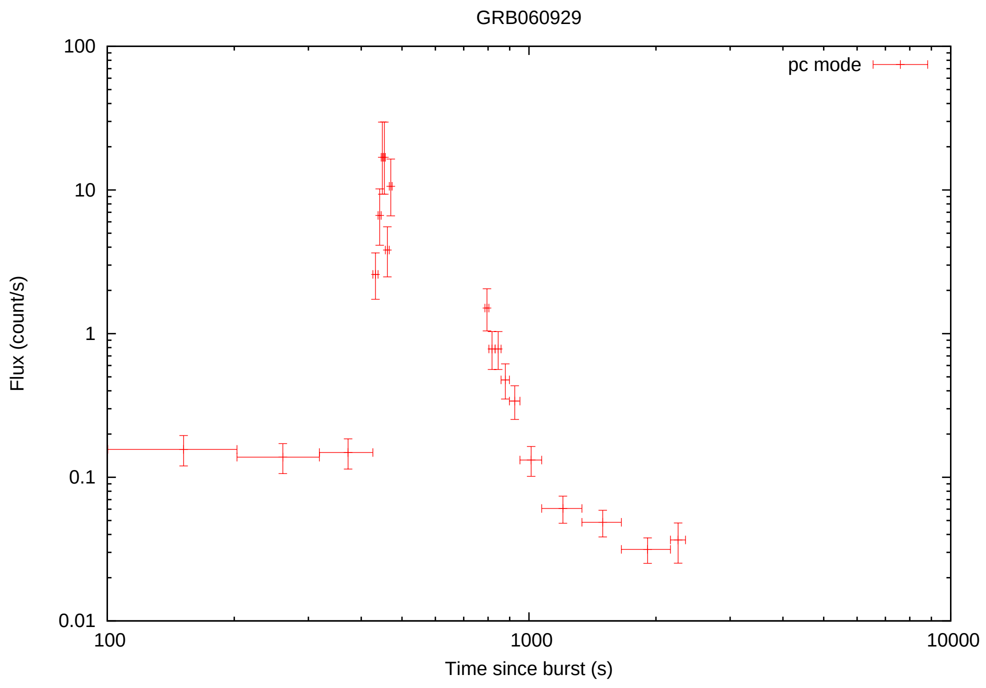

GRB060919
pc mode
Flux (count/s)
Time since burst (s)
1
0.1
0.01
0.001
0.0001
10
100
1000
10000
100000

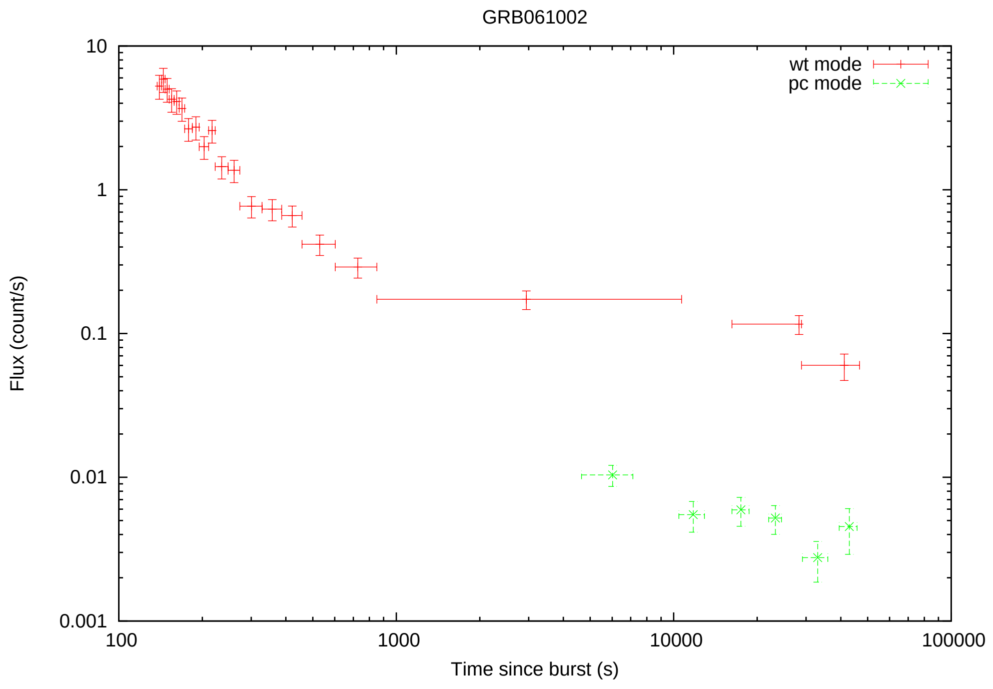

GRB060923A
pc mode
Flux (count/s)
Time since burst (s)
100
10
1
0.1
0.01
0.001
10
100
1000
10000
100000

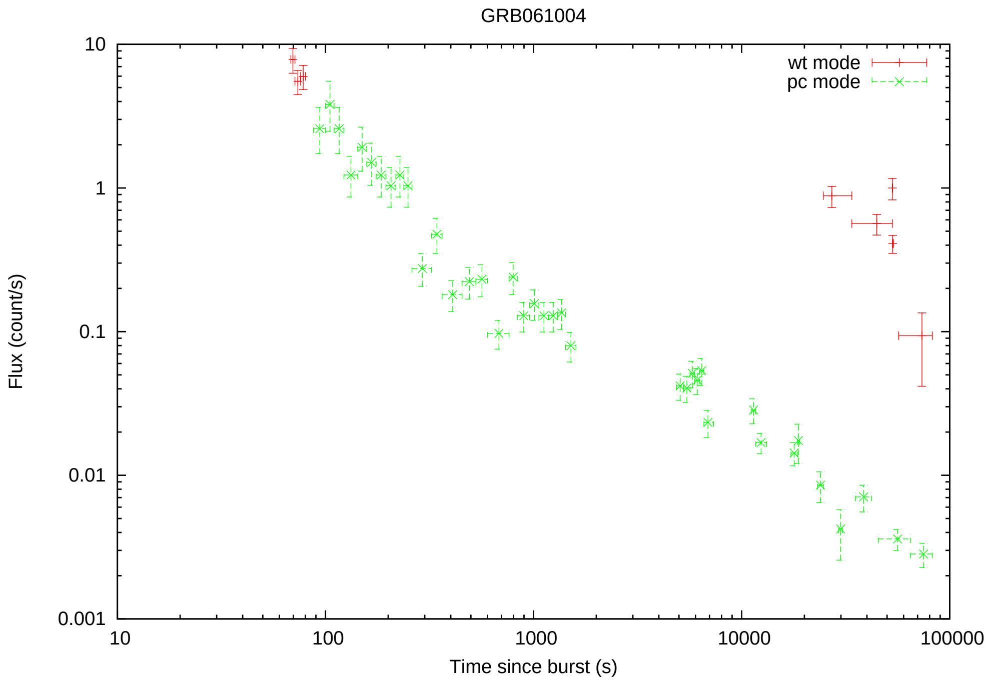

GRB060923C
wt mode
pc mode
Flux (count/s)
Time since burst (s)
100
10
1
0.1
0.01
0.001
100
1000
10000
100000

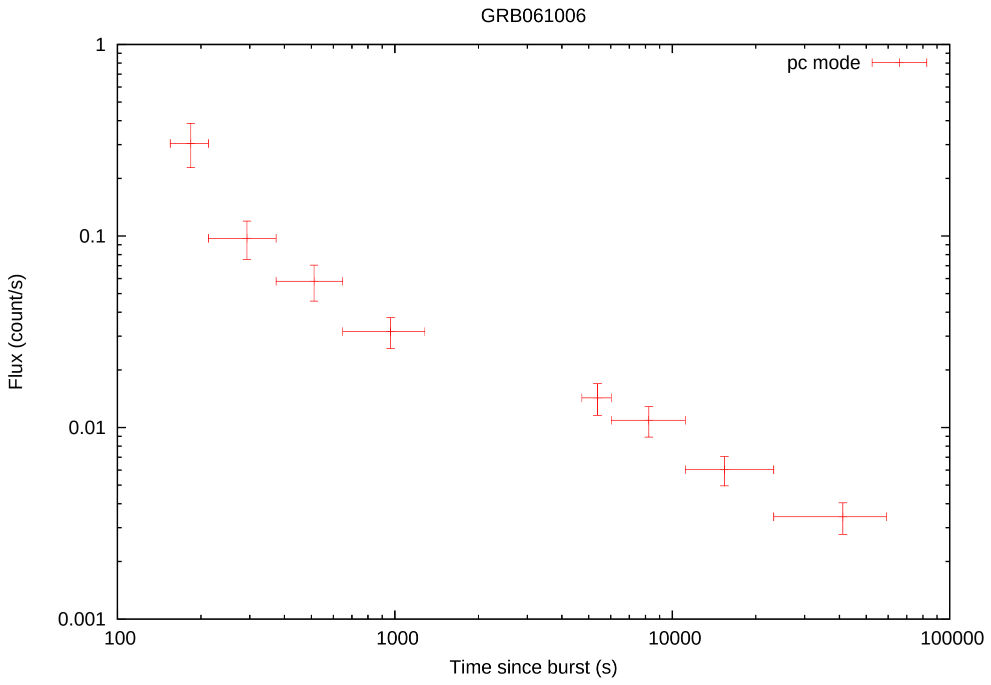

GRB060926
wt mode
pc mode
Flux (count/s)
Time since burst (s)
10
1
0.1
0.01
0.001
10
100
1000
10000
100000

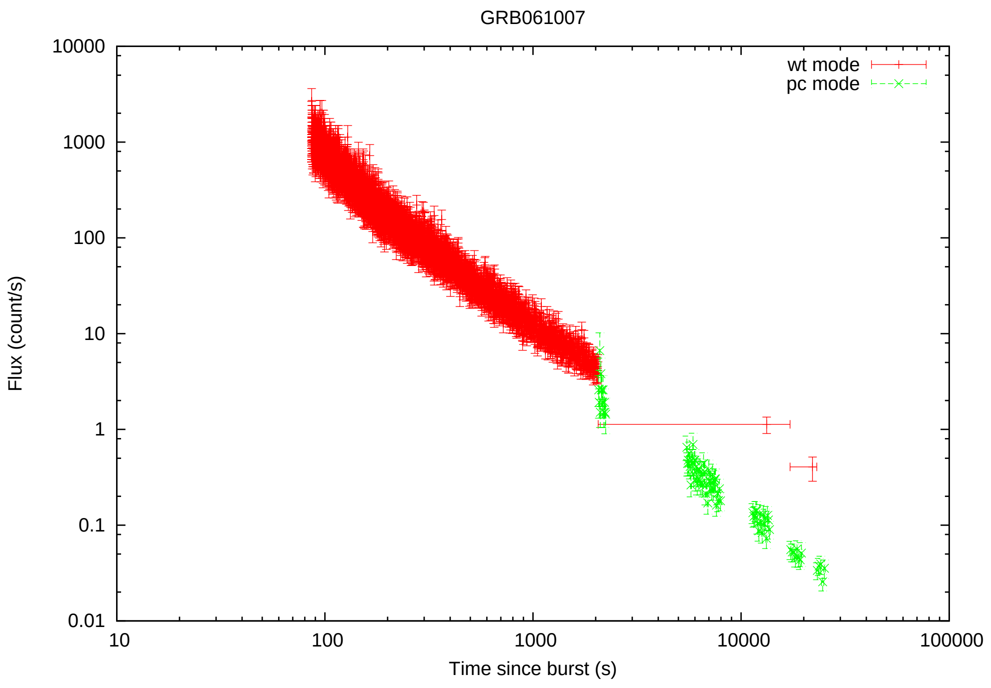

GRB060927
pc mode
Flux (count/s)
Time since burst (s)
10
1
0.1
0.01
0.001
10
100
1000
10000
100000


GRB060929
pc mode
Flux (count/s)
Time since burst (s)
100
10
1
0.1
0.01
100
1000
10000

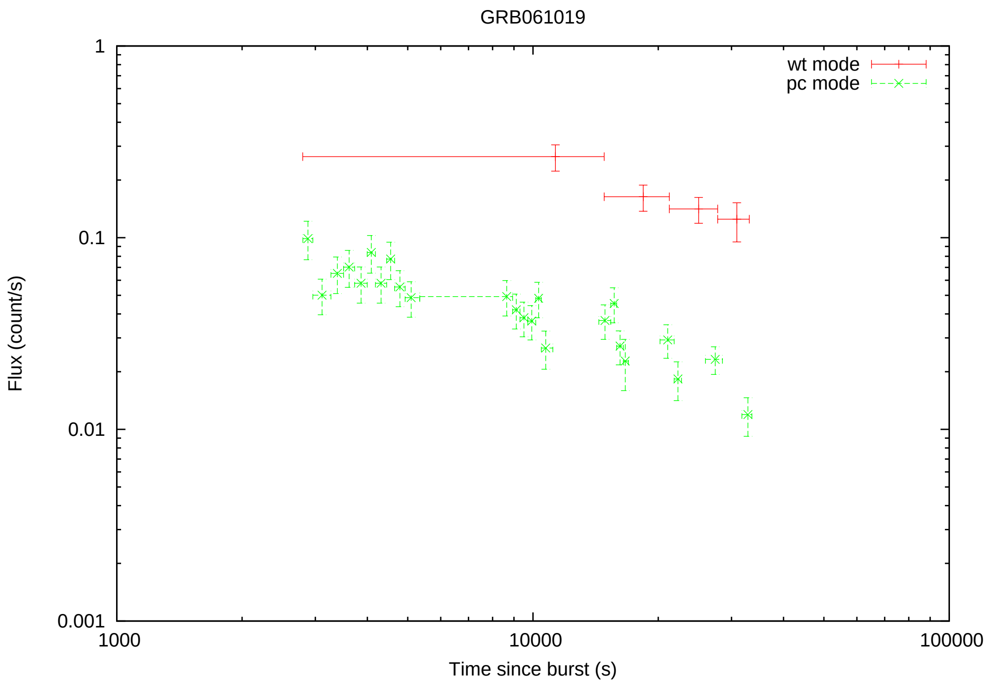

GRB061002
wt mode
pc mode
Flux (count/s)
Time since burst (s)
10
1
0.1
0.01
0.001
100
1000
10000
100000


GRB061004
wt mode
pc mode
Flux (count/s)
10
1
0.1
0.01
0.001
10
100
1000
10000
100000
Time since burst (s)

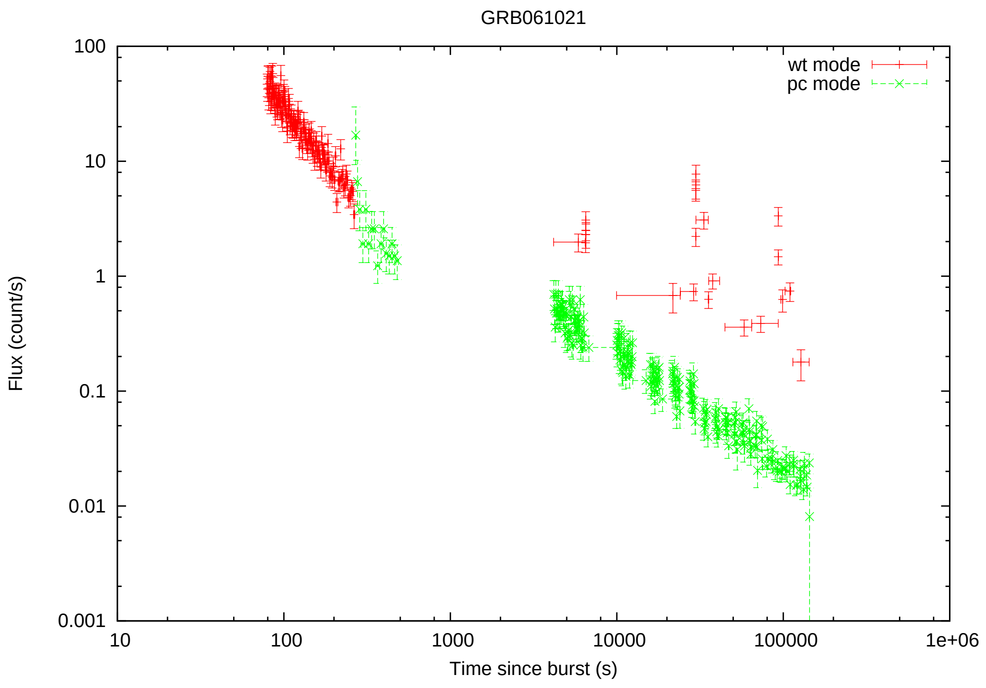

GRB061006
pc mode
Flux (count/s)
1
0.1
0.01
0.001
100
1000
10000
100000
Time since burst (s)

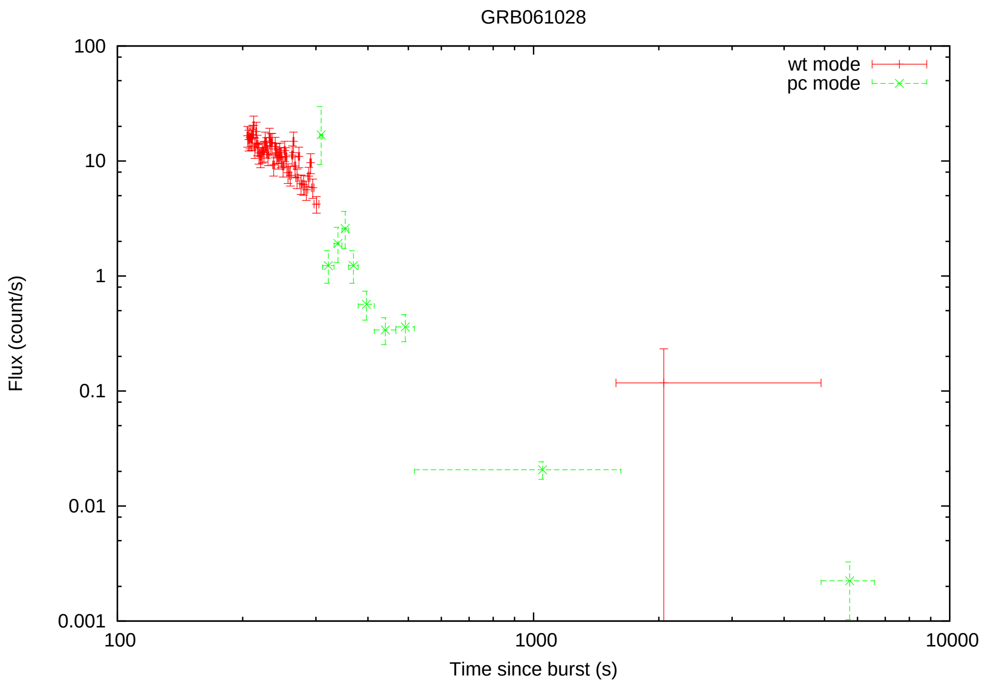


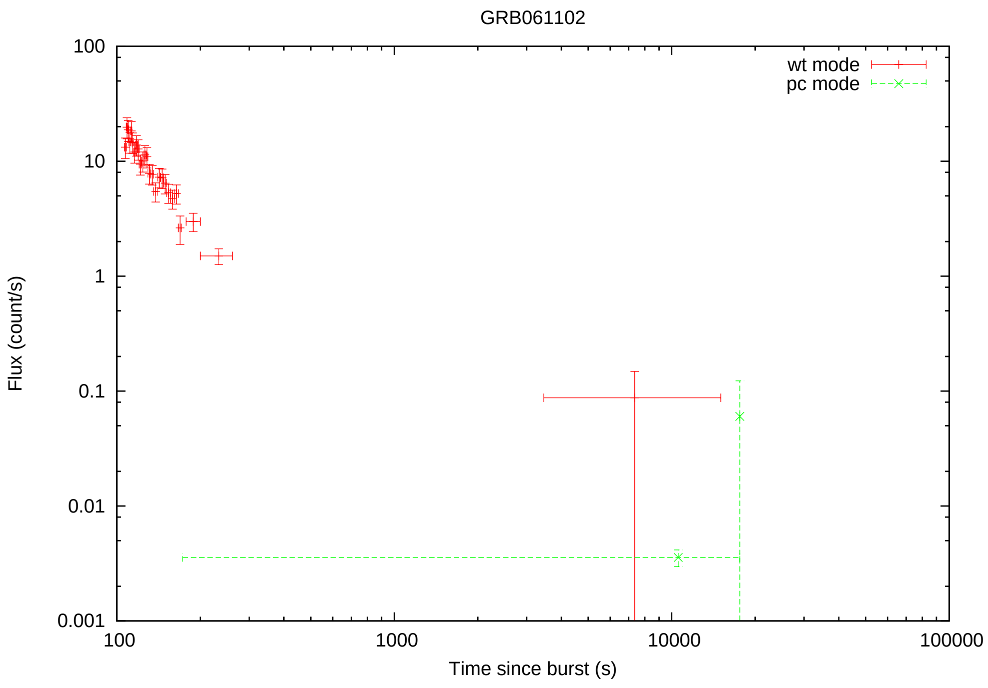
GRB061007
wt mode
pc mode
Flux (count/s)
Time since burst (s)
10000
1000
100
10
1
0.1
0.01
10
100
1000
10000
100000

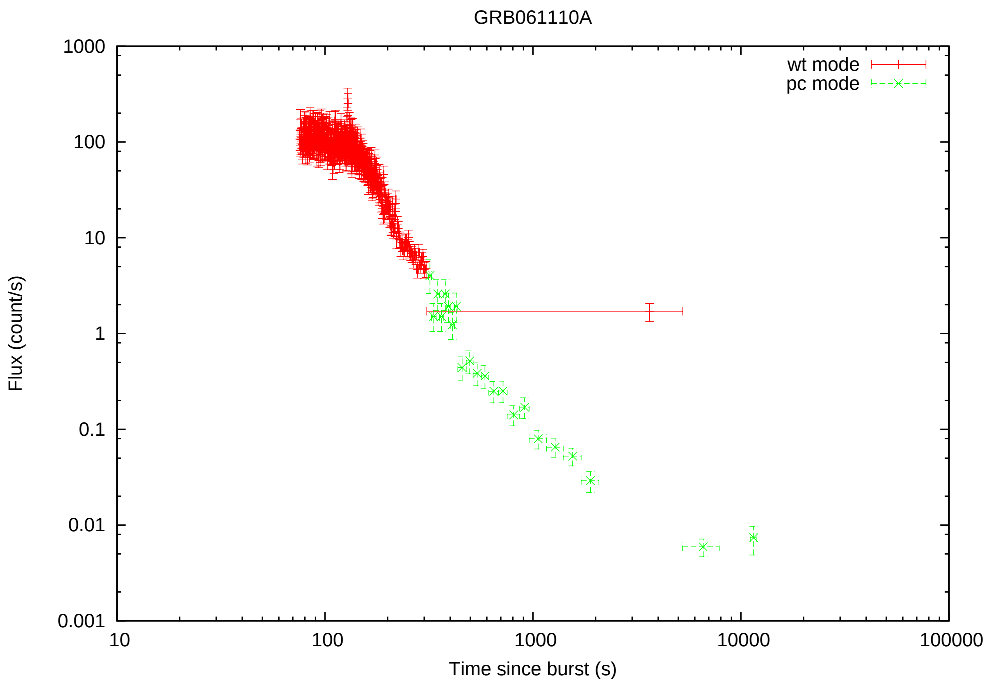


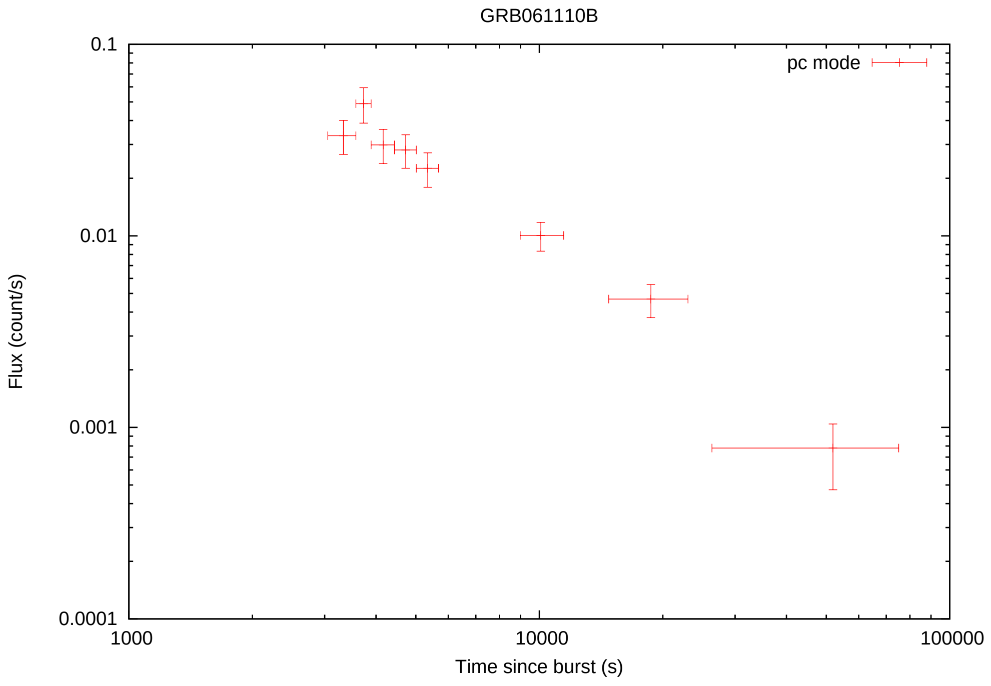
GRB061019
wt mode
pc mode
Flux (count/s)
Time since burst (s)
1
0.1
0.01
0.001
1000
10000
100000

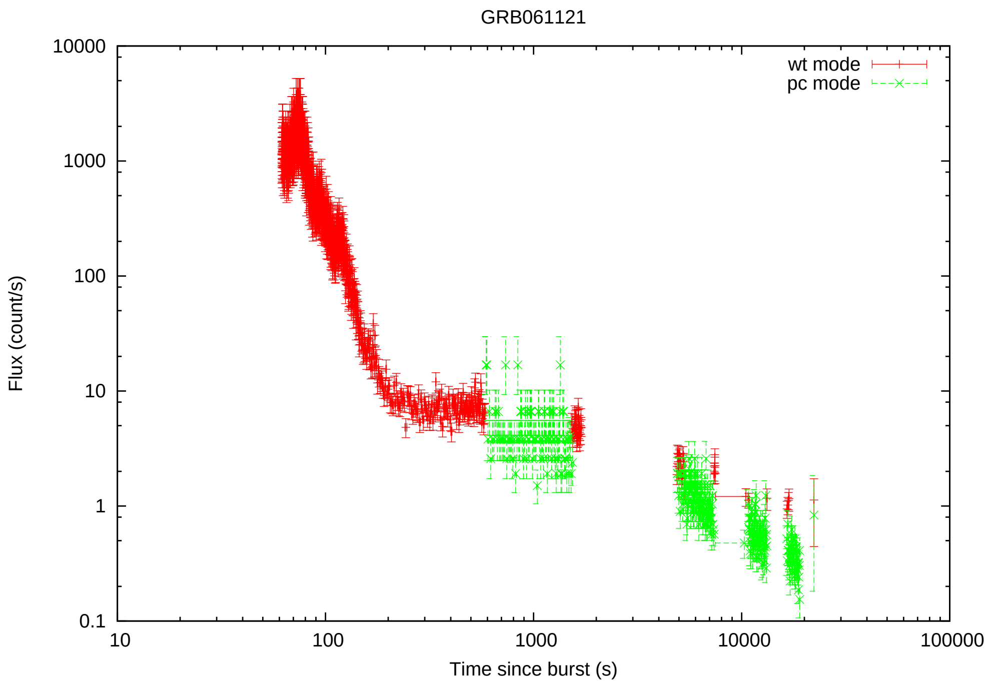

GRB061021
wt mode
pc mode
Flux (count/s)
Time since burst (s)
100
10
1
0.1
0.01
0.001
10
100
1000
10000
100000
1e+06

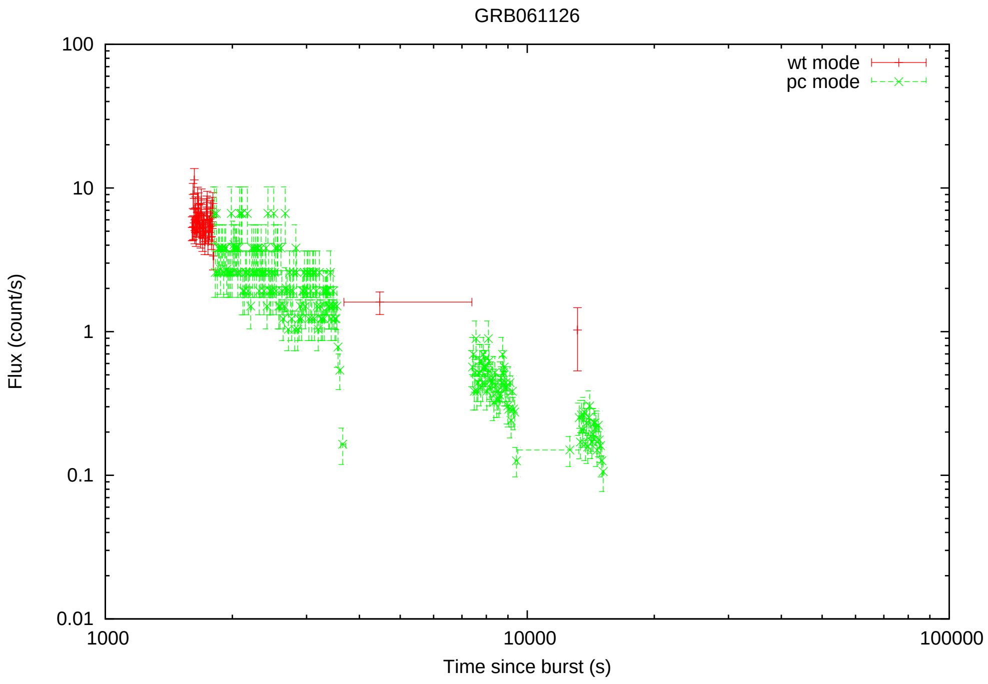

GRB061028
wt mode
pc mode
Flux (count/s)
Time since burst (s)
100
10
1
0.1
0.01
0.001
100
1000
10000

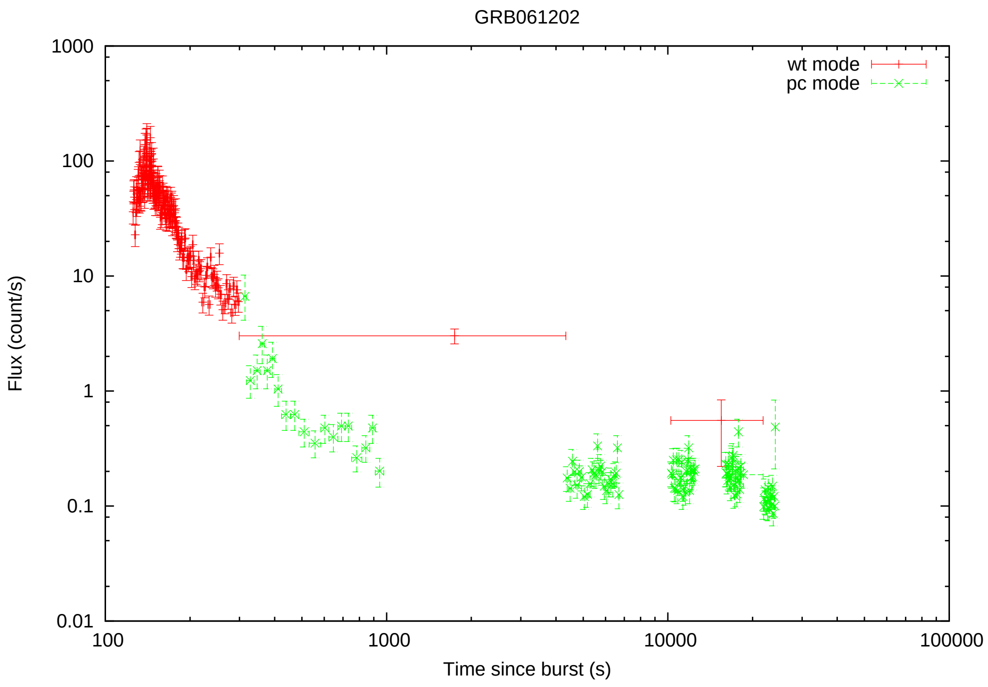

GRB061102
Flux (count/s)
Time since burst (s)
wt mode
pc mode
100
10
1
0.1
0.01
0.001
100
1000
10000
100000

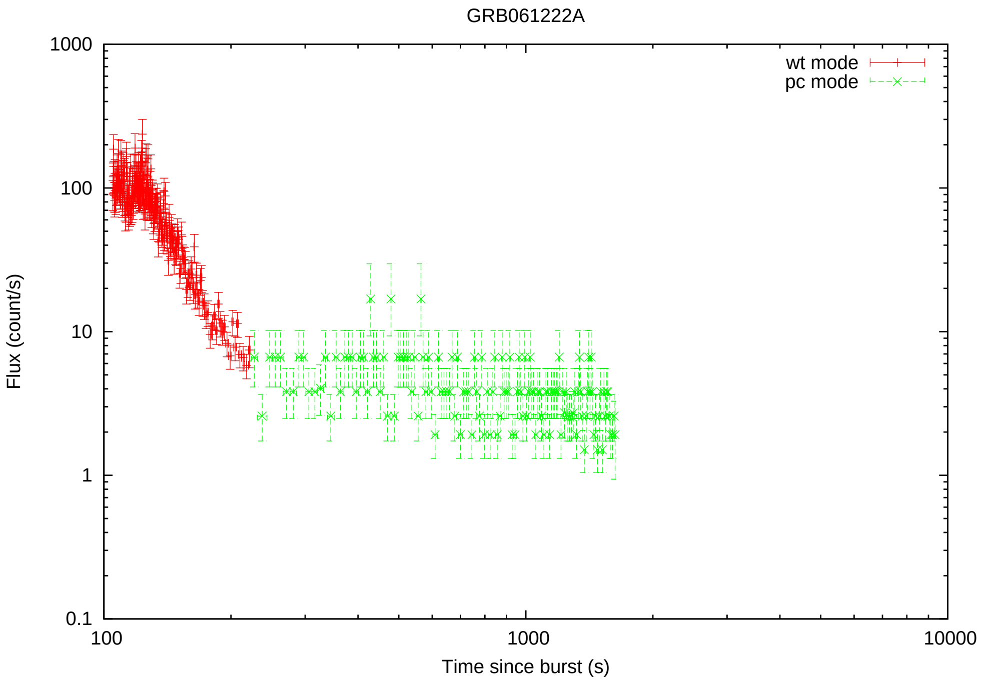

GRB061110A
Flux (count/s)
Time since burst (s)
wt mode
pc mode
1000
100
10
1
0.1
0.01
0.001
10
100
1000
10000
100000

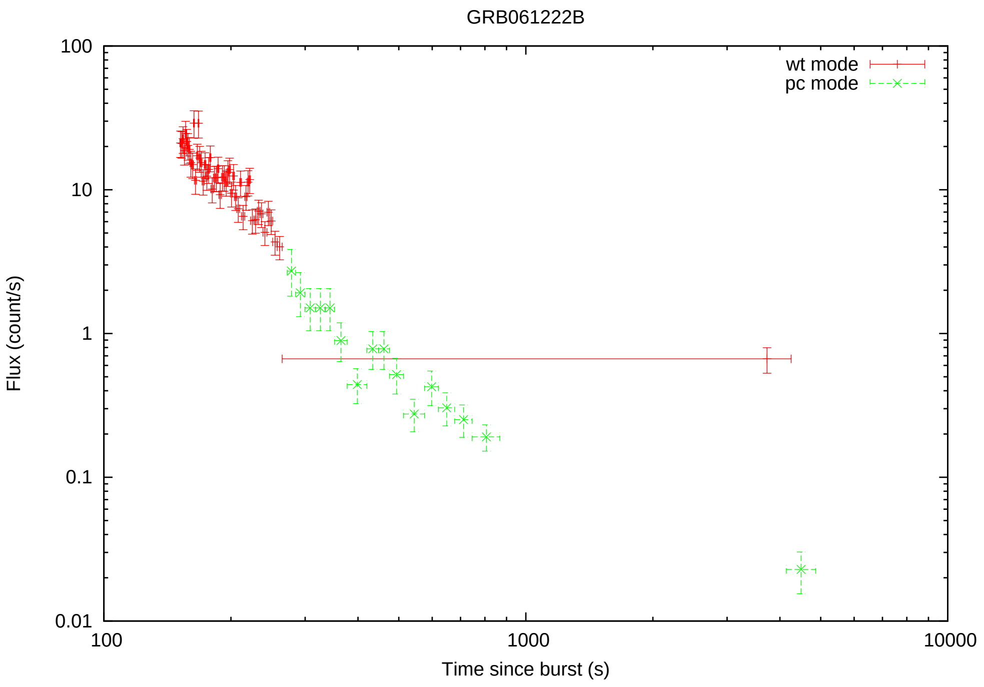

GRB061110B
pc mode
Flux (count/s)
0.1
0.01
0.001
0.0001
1000
10000
100000
Time since burst (s)

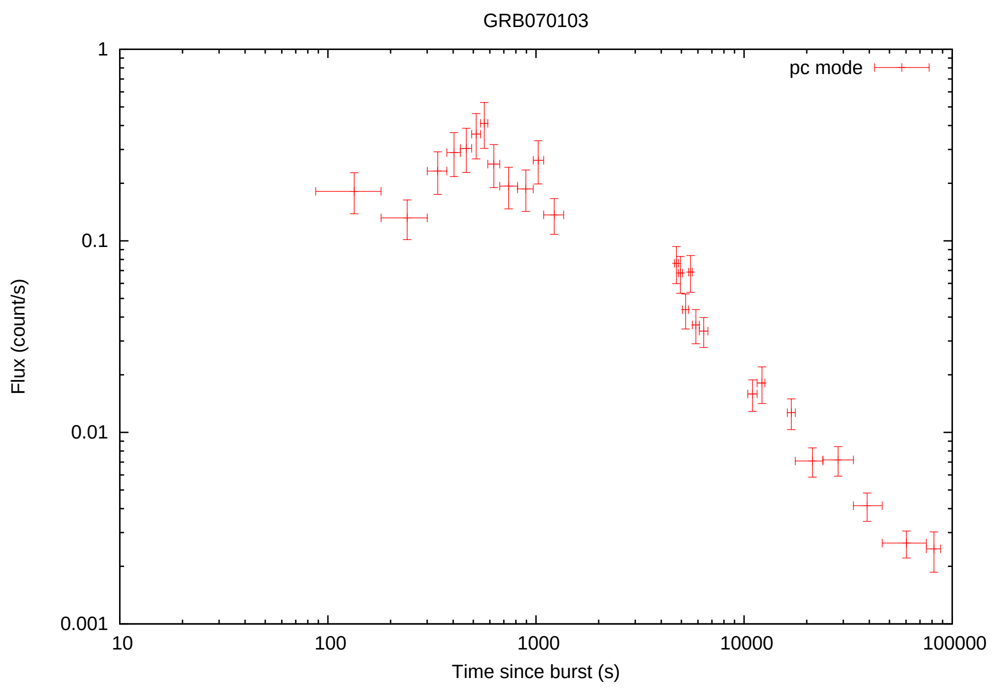

GRB061121
wt mode
pc mode
Flux (count/s)
10000
1000
100
10
1
0.1
10
100
1000
10000
100000
Time since burst (s)

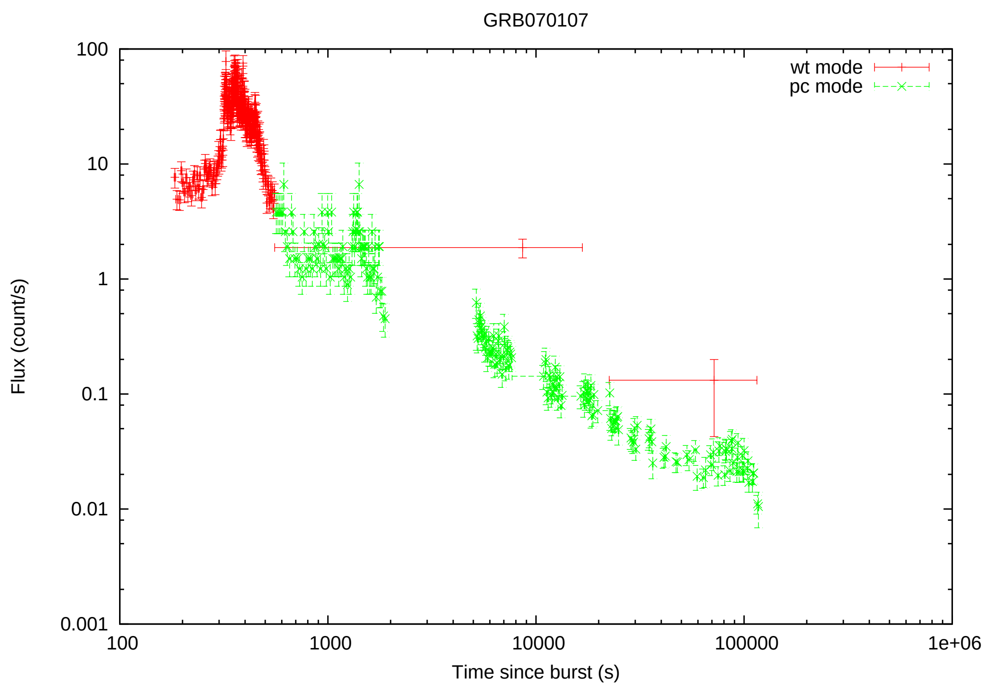

GRB061126
wt mode
pc mode
Flux (count/s)
Time since burst (s)
100
10
1
0.1
0.01
1000
10000
100000

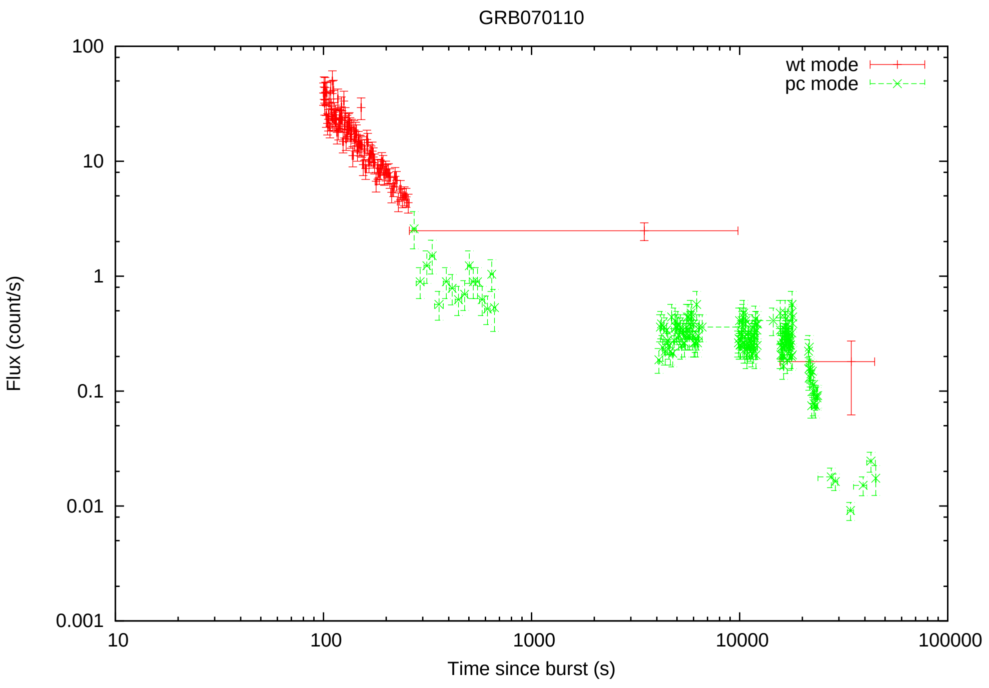

GRB061202
wt mode
pc mode
Flux (count/s)
Time since burst (s)
1000
100
10
1
0.1
0.01
100
1000
10000
100000

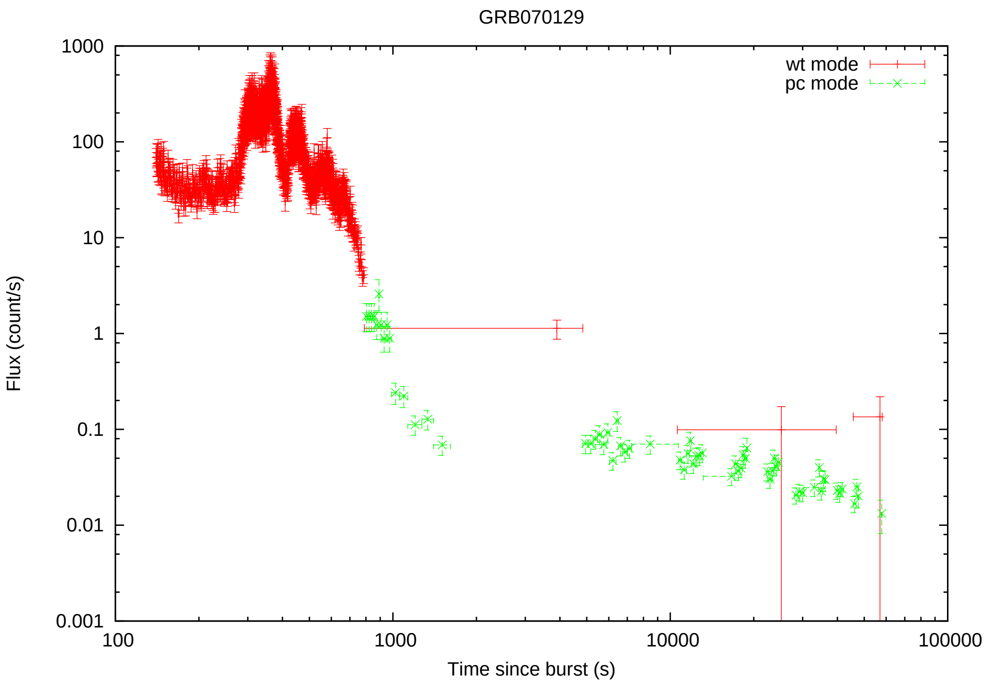

GRB061222A
Flux (count/s)
Time since burst (s)
wt mode
pc mode
1000
100
10
1
0.1
100
1000
10000

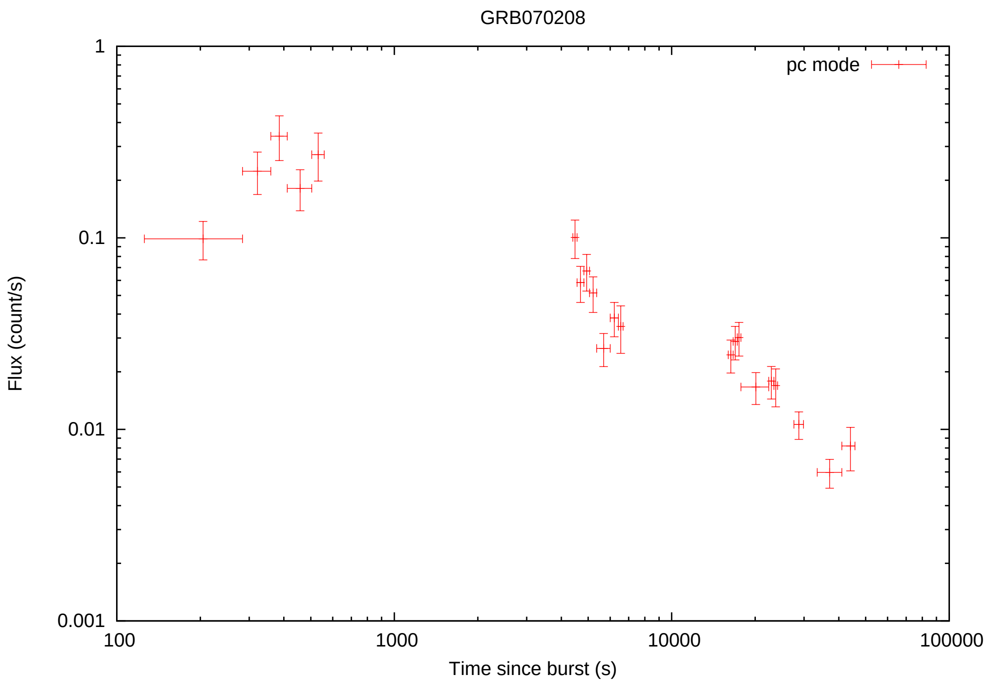

GRB061222B
Flux (count/s)
Time since burst (s)
wt mode
pc mode
100
10
1
0.1
0.01
100
1000
10000

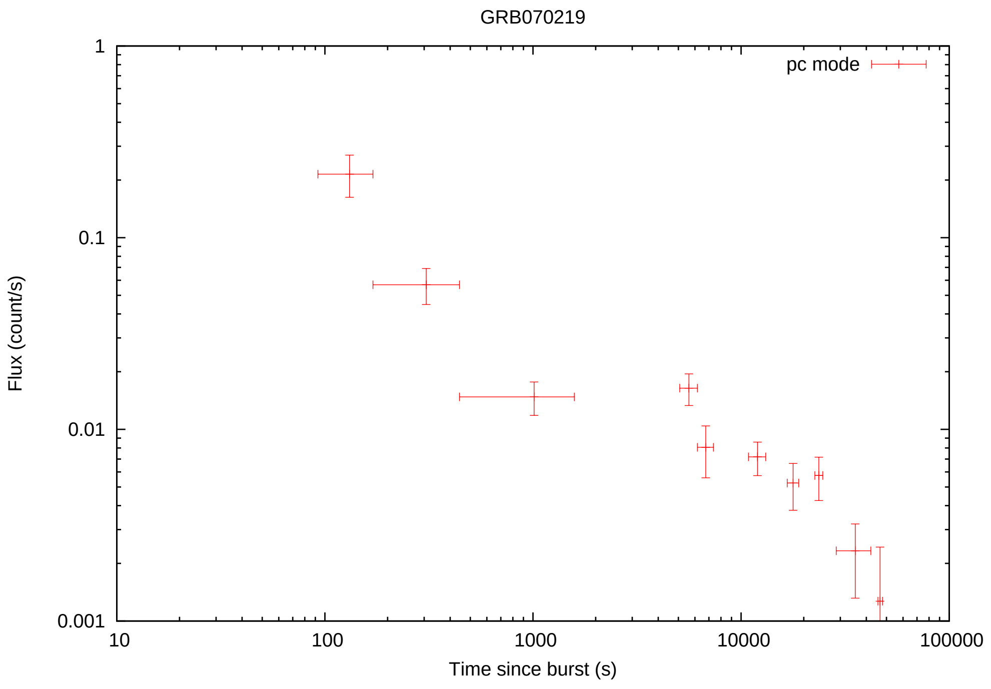

GRB070103
pc mode
Flux (count/s)
1
0.1
0.01
0.001
10
100
1000
10000
100000
Time since burst (s)

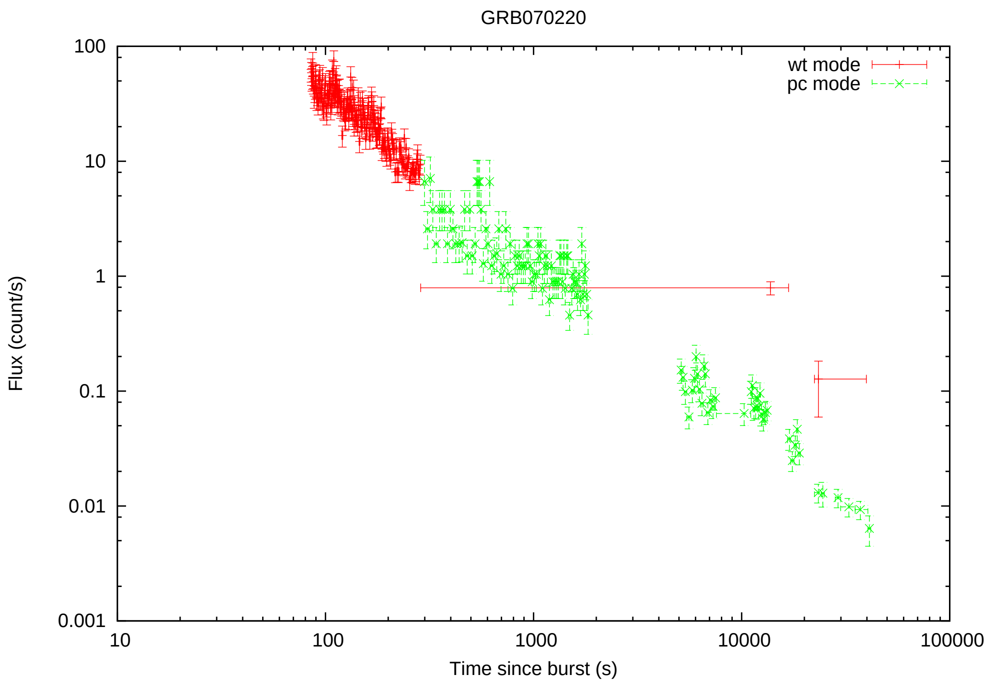

GRB070107
wt mode
pc mode
Flux (count/s)
100
10
1
0.1
0.01
0.001
100
1000
10000
100000
1e+06
Time since burst (s)

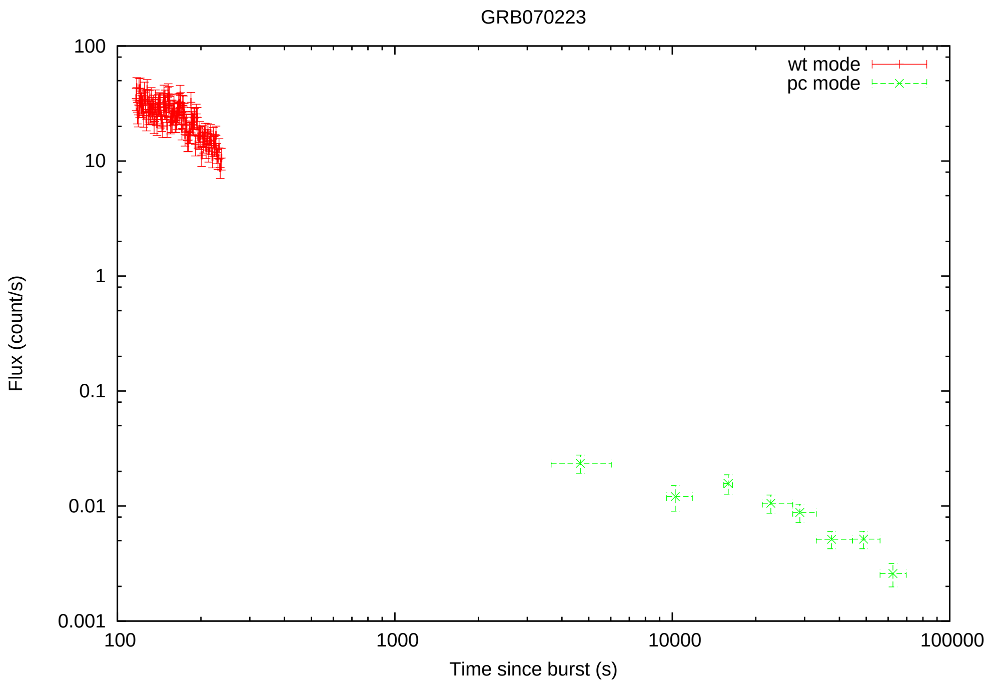

GRB070110
wt mode
pc mode
Flux (count/s)
100
10
1
0.1
0.01
0.001
10
100
1000
10000
100000
Time since burst (s)

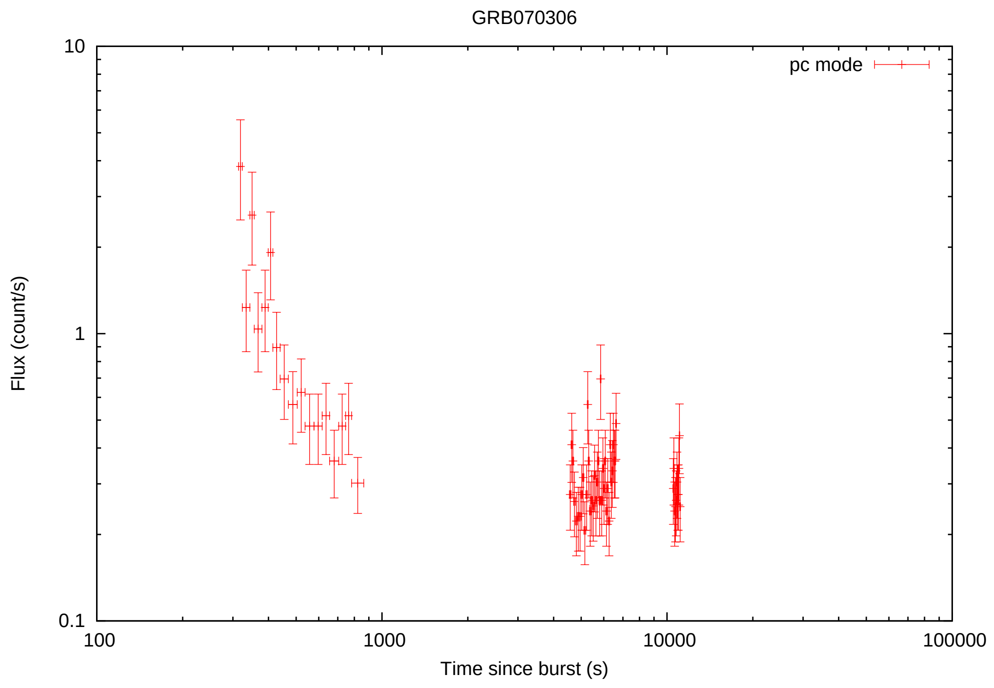

GRB070129
wt mode
pc mode
Flux (count/s)
1000
100
10
1
0.1
0.01
0.001
100
1000
10000
100000
Time since burst (s)

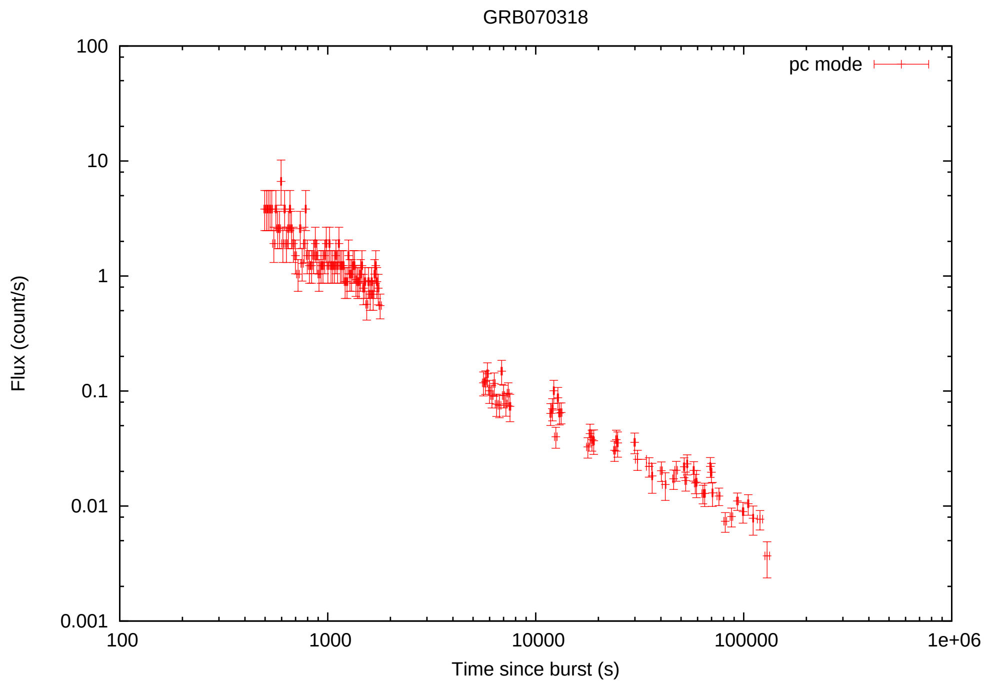

GRB070208
pc mode
Flux (count/s)
1
0.1
0.01
0.001
100
1000
10000
100000
Time since burst (s)

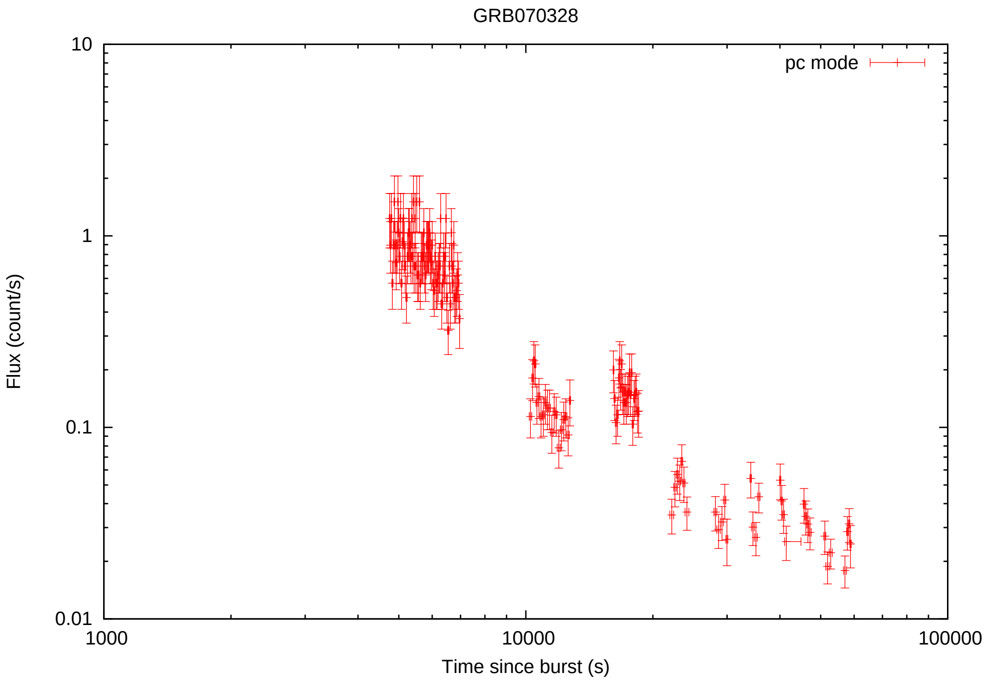

GRB070219
pc mode
Flux (count/s)
1
0.1
0.01
0.001
10
100
1000
10000
100000
Time since burst (s)

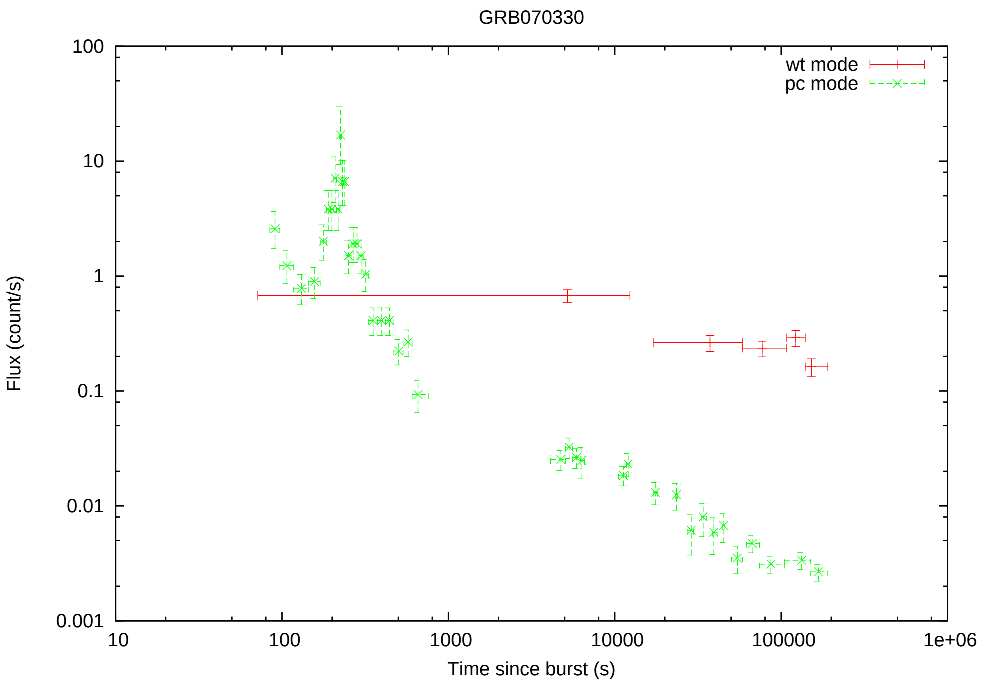

GRB070220
wt mode
pc mode
Flux (count/s)
100
10
1
0.1
0.01
0.001
10
100
1000
10000
100000
Time since burst (s)

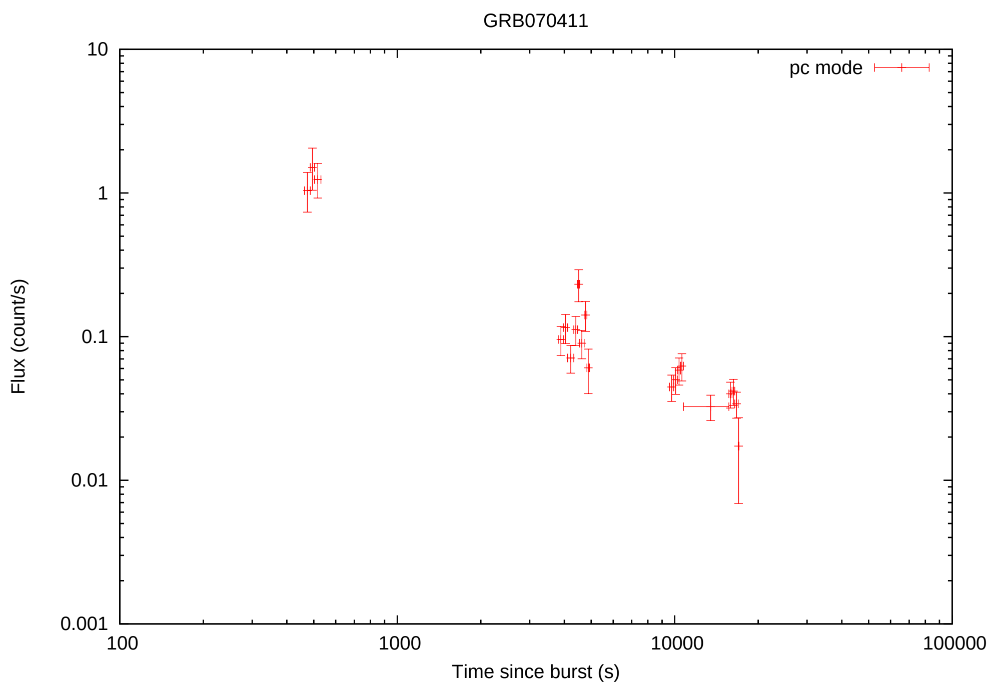

GRB070223
wt mode
pc mode
Flux (count/s)
100
10
1
0.1
0.01
0.001
100
1000
10000
100000
Time since burst (s)

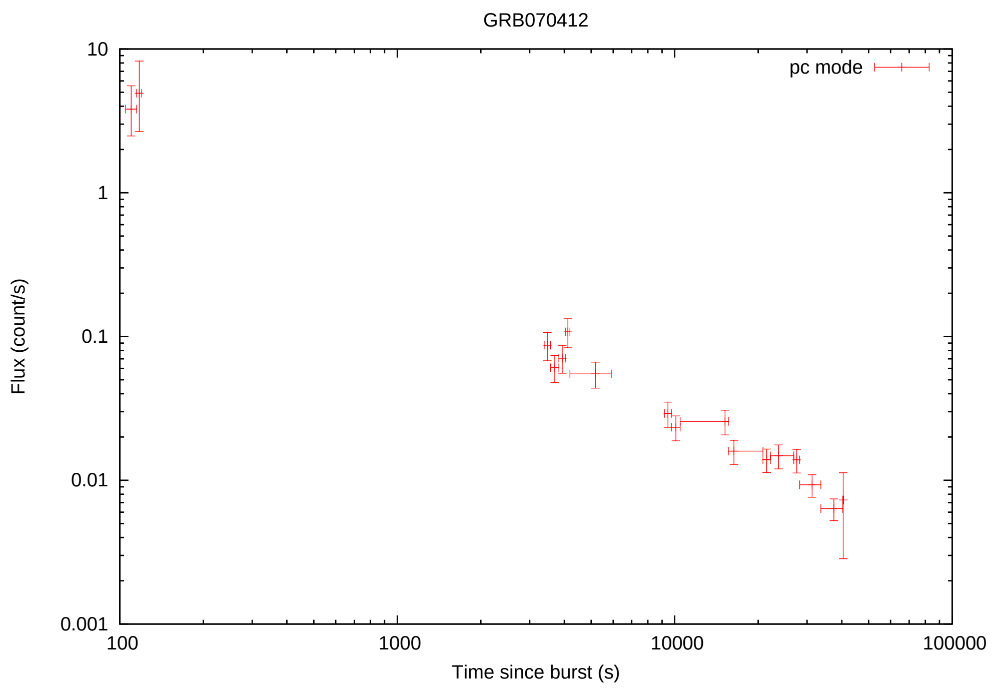

GRB070306
Flux (count/s)
10
1
0.1
100
1000
10000
100000
Time since burst (s)
pc mode

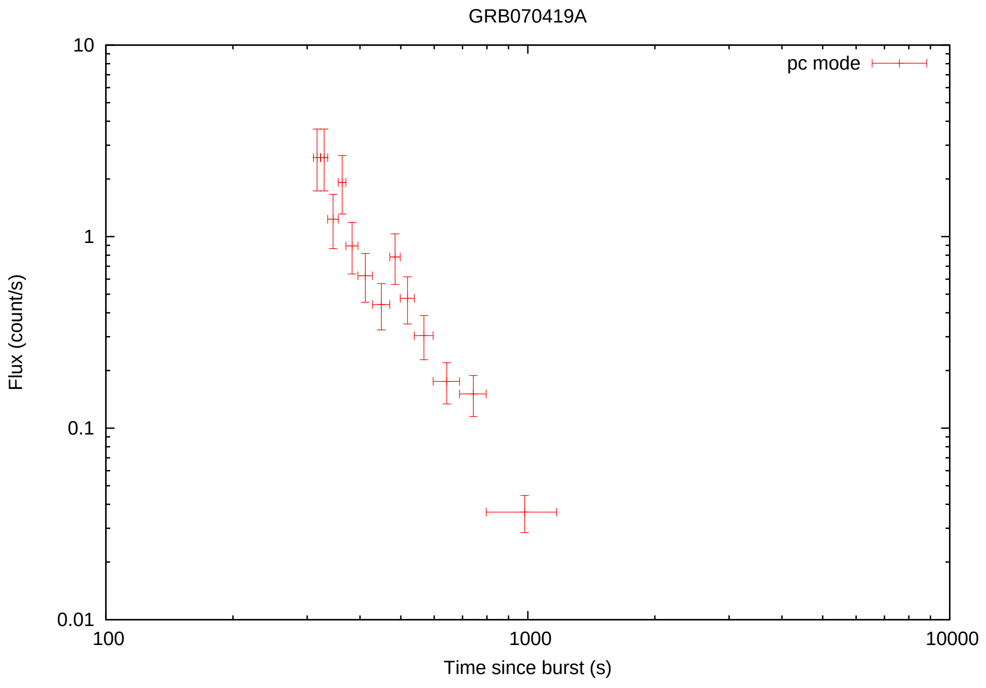

GRB070318
Flux (count/s)
100
10
1
0.1
0.01
0.001
100
1000
10000
100000
1e+06
Time since burst (s)
pc mode


GRB070328
pc mode
Flux (count/s)
10
1
0.1
0.01
1000
10000
100000
Time since burst (s)


GRB070330
wt mode
pc mode
Flux (count/s)
100
10
1
0.1
0.01
0.001
10
100
1000
10000
100000
1e+06
Time since burst (s)


GRB070411
pc mode
Flux (count/s)
Time since burst (s)
10
1
0.1
0.01
0.001
100
1000
10000
100000


GRB070412
pc mode
Flux (count/s)
Time since burst (s)
10
1
0.1
0.01
0.001
100
1000
10000
100000


GRB070419A
pc mode
Flux (count/s)
Time since burst (s)
10
1
0.1
0.01
100
1000
10000

Appendix B

The lightcurve extraction code

This program was written by my supervisor during this project, Dr. Darach Watson. Its function is to extract and background correct Swift GRB lightcurves (using *heasoft* software) and plotting them. When the pile-up correction was ready I wrote it into the code of this program near the end (clearly marked by double lines and statements above and below that part). That short section is *the only part* of this program that I wrote or had any part in writing. The reason for including the entire code of the program here despite it not being my own work is to make the method used to produce the lightcurve plots as clear and transparent as possible. It should be mentioned that this is the windowed timing mode version of the pile-up correction, the photon counting mode version only differs from this one in a few constants which can be found in the text.

```
#include <string.h>
#include <stdio.h>
#include <math.h>
#include "fitsio.h"

/* I still want to add

a) command line switches and defaults for input parameters
1) BACKSCAL determination (call an ftool)
2) pile-up correction
3) flux calibration

Perhaps add a wrapper (perl?) around this programme to do these things */
```

```
/* A subroutine to write the binned data out */
int write_out(long srcbin, long bgdbin, double binstart, double binstop,
double intervals[][2], int nintervals, float backscale, double grbtime);

int main(int argc, char *argv[])
{
    fitsfile *srcfptr;       /* Source FITS file pointer, defined in
fitsio.h */
    fitsfile *backfptr;      /* Background FITS file pointer, defined
in fitsio.h */
    char   nullstr[]="*";   // The Null string used in fits reading
    char   keyword[FLEN_KEYWORD], colname[FLEN_VALUE]; // Holders of
keywords and column names for CFITSIO
    int    status = 0;   /*  CFITSIO status value MUST be initialized
to zero!  */
    int    anynul; // Used in fitsreading commands
    int    previous_nintervals = 0; // Number of intervals in the
previous bin
    int    all_nintervals=0; // Sum of number of intervals in current
and previous bin
    int    close_bin = 0; // Flag to tell the writing routine to write
out the previous bin
    int    previous_exists = 0; // Flag to say if there is a valid previous
bin so that the current one may be written out
    int    srctimecol = 0, backtimecol = 0; // Number of the TIME column
in the source and background EVENTS extensions
    int    gtistartcol = 0, gtistopcol = 0; // Number of the TSTART and
TSTOP columns in the source GTI extension
    long   srcnrows = 0, backnrows = 0, gtinrows = 0; // Number of rows in
the source and background EVENTS extension and the source GTI extension
    long   total_src_counts = 0, total_bgd_counts = 0; // Final number of
source and bgd counts found in the data, used for a sanity check
    long   gti_ctr = 0; // counter for the current GTI (array)
    long   interval_ctr = 0; // counter for number of intervals in a bin
(array)
    long   ii = 0; // iterator for reading intervals (array)
    long   jj = 1; // iterator for reading gti rows into array (file)
    long   ss = 1; // counter for the current source event (file)
    long   bb = 1; // counter for the current background event (file)
    long   srcfinalbin = 0, backfinalbin = 0; // co-added data for the
final bin before a big gap
    long bin_src_counts = 0, bin_bgd_counts = 0; // source and background
```

```
counts in the current bin
    long previous_bin_src_counts = 0, previous_bin_bgd_counts = 0;
// source and background counts in the previous bin
    double bin_start_time = 0, bin_end_time = 0; // start and end times of
the current bin
    double previous_bin_start_time = 0, previous_bin_end_time = 0;
// start and end times of the previous bin
    double SNratio = 0; // S/N ratio of the current bin
    double SNmin = 5; // Required S/N ratio per bin. An input parameter
    double backscale = 1; // Scaling factor for the background. An input
parameter
    double grbtime = 0; // Time of the GRB in units of the input file
(e.g. Swift spacecraft seconds). An input parameter
    double gtis[1000][2]; // Array of start and stop times of each GTI
(the source GTI extension data)
    double time_interval[1000][2], previous_time_interval[1000][2];
// Start and stop times of intervals in the current and previous (before
a big gap) bin
    double srctimes = 0, backtimes = 0; // Event times read from the src
and background EVENTS extensions.
    int open_bin = 0;
    int open_interval = 0; // The start time for the next bin, cannot set
the next bin's start time until we close the current bin
    double tolerance_time = 1e-7; // An epsilon time added to the end of
each bin set using an event arrival time
(must be smaller than the frame time!)
    double input_total_exposure_time = 0; // The total exposure time in
the events file, use for a sanity check
    double final_total_exposure_time = 0; // The total exposure time found
from the bintimes, use for a sanity check

/*  Set up. Checking and setting input arguments */
    if (argc != 6) {
      printf("Usage:  snbinevts srcfilename[ext][col filter][row filter]
[regfilter(\"src_sky.reg\")] bgdfilename[ext][col filter][row filter]
[regfilter(\"bgd_sky.reg\")] backscale snratio grbtime\n");
      printf("\n");
      printf("Make a signal-to-noise ratio binned lightcurve from events
data \n");
      printf("\n");
      printf("Output is an ascii data file\n");
      return(0);
    }
    backscale = atof(argv[3]);
    SNmin = atof(argv[4]);
    grbtime = atof(argv[5]);


    fits_open_file(&srcfptr, argv[1], READONLY, &status);
// Open source file
    fits_open_file(&backfptr, argv[2], READONLY, &status);
// Open background file


/*    Read the source GTI extension into an array */
    if (fits_movnam_hdu(srcfptr, BINARY_TBL, "GTI", 0, &status))
        printf("Error: this program only works with event files with GTI
extensions\n");
    else
    {
      fits_get_colnum(srcfptr, CASEINSEN, "START", &gtistartcol, &status);
      fits_get_colnum(srcfptr, CASEINSEN, "STOP", &gtistopcol, &status);
      fits_get_num_rows(srcfptr, &gtinrows, &status);

      for (jj = 1; jj <= gtinrows && !status; jj++) {
        if (fits_read_col (srcfptr,TDOUBLE,gtistartcol,jj,1,1,nullstr,
&srctimes,&anynul,&status) )
          break;  /* jump out of loop on error */
        if (srctimes == gtis[jj-2][1]) srctimes += tolerance_time;
// We should really do error-checking here as well.
        gtis[jj-1][0] = srctimes;
        if (fits_read_col (srcfptr,TDOUBLE,gtistopcol,jj,1,1,nullstr,
&srctimes,&anynul,&status) )
          break;  /* jump out of loop on error */
        gtis[jj-1][1] = srctimes;
        input_total_exposure_time += gtis[jj-1][1]-gtis[jj-1][0];
      }
    }



/*    Move to the EVENTS extension for both src and bgd files */
    if (fits_movnam_hdu(srcfptr, BINARY_TBL, "EVENTS", 0, &status)
|| fits_movnam_hdu(backfptr, BINARY_TBL, "EVENTS", 0, &status))
        printf("Error: this program only works with fits files with EVENTS
extensions\n");
```

```
    else
    {
       fits_get_colnum(srcfptr, CASEINSEN, "TIME", &srctimecol, &status);
       fits_get_num_rows(srcfptr, &srcnrows, &status);

       fits_get_colnum(backfptr, CASEINSEN, "TIME", &backtimecol, &status);
       fits_get_num_rows(backfptr, &backnrows, &status);

      /* Need to do error handling here where no TIME column is found*/

     printf("## Lightcurve created by snbinevts from the input command:\n");
     printf("## \'%s %s %s %s %s %s\'\n",argv[0],argv[1],argv[2],argv[3],
argv[4],argv[5]);
     printf("## \n");
     printf("#Time (s) \t Rate (c/s)\t Time_low\t Time_high\t Rate_low\t
Rate_high\t src_cts\t bgd_cts\t Exposure\n");

//   CREATING THE BINS
//   =================

    fits_read_col (backfptr,TDOUBLE,backtimecol,bb,1,1,nullstr,&backtimes,
&anynul,&status);  // Read the 1st bgd event arrival time

    time_interval[0][0] = gtis[0][0];
// Set the start time of the first bin
    time_interval[0][1] = time_interval[0][0];
// Set the end of the first time interval for the first bin
    open_bin = 1;
    interval_ctr = 0;

    /* We want to read all source events and all GTIs, but we do not use a
for loop because we do not always want to increment */
    while (gti_ctr < gtinrows)
    {
      if (ss <= srcnrows)
      {
        if (open_bin){
          /* Start a new bin */
          bin_src_counts = 0; // Initialise the bin src counts
          bin_bgd_counts = 0; // Initialise the bin bgd counts
          bin_start_time = time_interval[interval_ctr][0];
// Set the start time of the new bin
          interval_ctr = 0; // Reset the time interval counter for this bin
          time_interval[interval_ctr][0] = bin_start_time;
          open_bin = 0;
        }


        fits_read_col(srcfptr,TDOUBLE,srctimecol,ss,1,1,nullstr,&srctimes,
&anynul,&status); // Read the ss'th source event arrival time




        /* Source event inside the current GTI */
        if (srctimes <= gtis[gti_ctr][1])
// If the next photon is inside the current GTI,
        {
          bin_src_counts++;
// increment the current bin source count
          time_interval[interval_ctr][1] = srctimes;
// Increase the current time interval stop time to this arrival_time
          time_interval[interval_ctr+1][0] = srctimes + tolerance_time;
// Start time of the next time interval set to the end of the current one
plus epsilon
          ss++;  // Prepare to read a new photon arrival time
        } // End of photon inside the current GTI loop





        /* Source event outside the current GTI */
        else
        {
          time_interval[interval_ctr][1] = gtis[gti_ctr][1];
// Fix the end of the current interval to the end of the current GTI
          gti_ctr++;
// Move to the next GTI
          time_interval[interval_ctr+1][0] = gtis[gti_ctr][0];
// Start time of the next time interval set to the start of the next GTI

          /* We do not want to sum over time periods where there is no
data if we can avoid it, so we... */
          /* Check if the next GTI is closer than the width of the
previous bin */
```

```
          /* If it is too far away, we need to tie off the old bins and
start a new one */

          /* Next GTI is close enough to span. This loop is where we plop
into a new GTI in the same bin */
          if ( !(previous_exists) || ((gtis[gti_ctr][0]-gtis[gti_ctr-1][1])
< (previous_bin_end_time-previous_bin_start_time)) )
          {
            open_interval = 1;
          }

          else if ( (bin_src_counts <= 1) || (gti_ctr >= 2) &&
((gtis[gti_ctr][0]-gtis[gti_ctr-1][1]) < (gtis[gti_ctr-1][0]-gtis[gti_ctr-2][1])) )
          {
            open_interval = 1;
          }

          /* Next GTI is too far away to add to this bin */
          else
          {
            /* We have to stop a bin here whether it reaches SNmin or not, */
            /* So we want to even out the last two bins so that the last   */
            /* bin does not have a tragically low S/N                      */

            /* First check if there are enough counts in the current bin to stand
alone  */
            /* If so, we will need to write out the previous bin, set the current
bin to */
            /* the previous and start a new bin                            */
            if ( (bin_src_counts-(backscale*bin_bgd_counts)) >
(0.4*(previous_bin_src_counts-(backscale*previous_bin_bgd_counts))) )
            {
              close_bin = 1;
// Ideally I want to divide the last bins into two even ones but it's too hard
            }

            /* If there are too few counts in the current bin to stand alone, */
            /* add the current bin to the previous bin and start a new bin    */
            else
            {
              /* what we do is rewind by one bin and set previous_exists = 0 */
              /* add the two bins together */
              bin_src_counts += previous_bin_src_counts;
              bin_bgd_counts += previous_bin_bgd_counts;
              bin_start_time = previous_bin_start_time;
// set the start of the current bin to the start of the previous bin

              /* add current intervals to previous */
              for (ii = 0; ii <= (interval_ctr + 1); ii++)
// The +1 is important because we have defined time_interval[interval_ctr+1][0]
above
              {
                previous_time_interval[previous_nintervals+1+ii][0]
= time_interval[ii][0];
                previous_time_interval[previous_nintervals+1+ii][1]
= time_interval[ii][1];
              }

              interval_ctr += (previous_nintervals + 1);
// Add the current number of intervals to the previous

              for (ii = 0; ii <= (interval_ctr + 1); ii++)
              {
                time_interval[ii][0] = previous_time_interval[ii][0];
                time_interval[ii][1] = previous_time_interval[ii][1];
              }

              for (ii = 0; ii <= (previous_nintervals + 1); ii++) {
// Clear the previous time intervals
                previous_time_interval[ii][0] = 0;
                previous_time_interval[ii][1] = 0;
              }

              close_bin = 1;
              previous_exists = 0;
            } // End of can it stand alone loop
          } // End of is it close enough loop
        } // End of photon outside current GTI loop



        bin_end_time = time_interval[interval_ctr][1];
// Set the end of the time bin to be the current interval stop time

        if (backtimes < time_interval[interval_ctr][0])
printf("Warning: There is a problem with the background.\n
Background event with time %8.7lf is earlier than the current time interval
(before %8.7lf)\n", backtimes, time_interval[interval_ctr][0]);
        /* Work out the background counts up to this point */
        while ((backtimes >= time_interval[interval_ctr][0]) &&
(backtimes <= time_interval[interval_ctr][1]))
// increase background bin count by background photons in newly added
time section
        {
           bin_bgd_counts++;
           bb++;
           if (bb > backnrows) backtimes = 1e300;
// Don't want to read beyond last row, but still need to exit
           else  fits_read_col (backfptr,TDOUBLE,backtimecol,bb,1,1,
nullstr,&backtimes,&anynul,&status);
// Read the bb'th arrival time (increment if inside current interval)
        }

        /* Get the SN ratio and see if we can close the bin yet */
        if (bin_src_counts) SNratio = (bin_src_counts -
backscale*bin_bgd_counts)/sqrt(bin_src_counts +
backscale*backscale*bin_bgd_counts);
        else SNratio = 0;
        if (SNratio > SNmin) close_bin = 1; // Set the write flag




        /* Write out the previous bin, swap the current bin to previous
and start a new bin */
        if (close_bin)
        {
          /* Finalise the background counts for the interval */

          /* We can now safely write the previous bin out if it exists */
          if (previous_exists) {
            write_out(previous_bin_src_counts, previous_bin_bgd_counts,
previous_bin_start_time, previous_bin_end_time, previous_time_interval,
previous_nintervals, backscale, grbtime);
            for (ii = 0; ii <= previous_nintervals; ii++) {
              final_total_exposure_time += (previous_time_interval[ii][1] -
previous_time_interval[ii][0]);
            }
            total_src_counts += previous_bin_src_counts;
            total_bgd_counts += previous_bin_bgd_counts;
          }
          for (ii = 0; ii <= previous_nintervals; ii++) {
// Clear the previous time intervals
            previous_time_interval[ii][0] = 0;
            previous_time_interval[ii][1] = 0;
          }

          /* Swap the current bin to previous */
          for (ii = 0; ii <= interval_ctr; ii++) {
// Set the previous time intervals to the current values
            previous_time_interval[ii][0] = time_interval[ii][0];
            previous_time_interval[ii][1] = time_interval[ii][1];
          }
          for (ii = 0; ii <= interval_ctr; ii++) {
// Clear the current time interval
            time_interval[ii][0] = 0;
            time_interval[ii][1] = 0;
          }
          previous_bin_src_counts = bin_src_counts;
          previous_bin_bgd_counts = bin_bgd_counts;
          previous_bin_start_time = bin_start_time;
          previous_bin_end_time = bin_end_time;
          previous_nintervals = interval_ctr;
          close_bin = 0; // Reset the write flag
          open_bin = 1;
          open_interval = 1;
          previous_exists = 1;
        } // End of bin closing loop

        if (open_interval) {
          interval_ctr++;
// Make a new interval
          time_interval[interval_ctr][1] = time_interval[interval_ctr][0];
// Set the end of the new interval to the start of the new interval to
start with (should be overwritten by the next event)
          open_interval = 0;
        }

      } // End of reading source counts loop
      /* Where there are no counts left, but some GTI, we need to write out */
      /* the remaining GTIs to the intervals and then write out the last point */
      else
      {
         time_interval[interval_ctr][1] = gtis[gti_ctr][1];
         bin_end_time = time_interval[interval_ctr][1];

         while ( (backtimes >= time_interval[interval_ctr][0]) &&
(backtimes <= time_interval[interval_ctr][1]) )
// increase background bin count by background photons in newly added
time section
         {
            bin_bgd_counts++;
            bb++;
            if (bb > backnrows) backtimes = 1e300;
// Don't want to read beyond last row, but still need to exit
            else  fits_read_col (backfptr,TDOUBLE,backtimecol,bb,1,1,nullstr,
&backtimes,&anynul,&status);
// Read the bb'th arrival time (increment if inside current interval)
         }

         gti_ctr++;
         interval_ctr++;
         time_interval[interval_ctr][0] = gtis[gti_ctr][0];
         time_interval[interval_ctr][1] = time_interval[interval_ctr][0];

      }




    }

  write_out(previous_bin_src_counts, previous_bin_bgd_counts,
previous_bin_start_time, previous_bin_end_time, previous_time_interval,
previous_nintervals, backscale, grbtime);
// Write the previous bin to output
  for (ii = 0; ii <= previous_nintervals; ii++) {
    final_total_exposure_time += (previous_time_interval[ii][1] -
previous_time_interval[ii][0]);
```

```
  }
  total_src_counts += previous_bin_src_counts;
  total_bgd_counts += previous_bin_bgd_counts;
  write_out(bin_src_counts, bin_bgd_counts, bin_start_time, bin_end_time,
time_interval, interval_ctr, backscale, grbtime);
// Write the final bin to output
  for (ii = 0; ii <= interval_ctr; ii++) {
    final_total_exposure_time += (time_interval[ii][1] - time_interval[ii][0]);
  }
  total_src_counts += bin_src_counts;
  total_bgd_counts += bin_bgd_counts;

  }

  fits_close_file(srcfptr, &status);
  fits_close_file(backfptr, &status);

  printf("\n\n");
  printf("#Number of source counts input = %4d\t\t",srcnrows);
  printf("%4d = Number of source counts found\n",total_src_counts);
  printf("#Number of background counts input = %4d\t\t",backnrows);
  printf("%4d = Number of background counts found\n",total_bgd_counts);
  printf("#Total exposure time input: %8.7lf\t\t",input_total_exposure_time);
  printf("%8.7lf = Total exposure time found\n",final_total_exposure_time);

  if (status) fits_report_error(stderr, status); /* print any error message */
  return(status);

}
```

```
/*
  In the last bin, we must either divide the counts evenly between the last
  two bins or just drop them all into the penultimate bin and scratch the
  last one. This division is at ~30% (lots of work on this number... is this
  just sqrt(2)?)
*/


/*
   The mean bintime (bin_mean_time) is defined as the weighted mean of the
   exposures. So the bin will span from the start of the first relevant
   exposure time to the end of the relevant exposure time (bin_start_time
   and bin_end_time respectively, but the time of the bin will be the
   weighted mean.
*/
```

```
int write_out(long srcbin, long bgdbin, double binstart, double binstop,
double intervals[][2], int nintervals, float backscale, double grbtime)
{
  int i = 0; // iterator over the intervals in the bin
  double exposure = 0; // Exposure value for the bin
  double bintime = 0; // Mean weighted bin time
  double count_rate = 0; // The count rate
  double count_rate_err = 0; // The error on the count rate
  double ratelow = 0;
  double ratehigh = 0;
  double lograte = 0;
  double logratelow = 0;
  double logratehigh = 0;
  double ratesqrt = 0;
  double ratelowsqrt = 0;
  double ratehighsqrt = 0;
  double ratecorr = 0;
  double ratelowcorr = 0;
  double ratehighcorr = 0;
  double ten = 10;
  double ratepower = 0;
  double ratelowpower = 0;
  double ratehighpower = 0;

  binstart -= grbtime;
  binstop -= grbtime;
  for (i = 0; i <= nintervals; i++) {
    intervals[i][0] -= grbtime;
    intervals[i][1] -= grbtime;
    exposure += intervals[i][1] - intervals[i][0];
//    printf("%8.7lf %8.7lf\n",intervals[i][1],intervals[i][0]);
  }
  for (i = 0; i <= nintervals; i++) {
    bintime += 0.5*(intervals[i][1] + intervals[i][0])*((intervals[i][1] -
intervals[i][0])/exposure);  // This is the weighted mean bin time
  }
  count_rate = (srcbin - backscale*bgdbin)/exposure;
  count_rate_err = sqrt(srcbin + backscale*backscale*bgdbin)/exposure;
  ratelow = count_rate - count_rate_err;
  ratehigh = count_rate + count_rate_err;

/* ------------------------------------------------------------------ */
/* pile-up correction for windowed timing mode                        */
/* by Jón Hafsteinn Guðmundsson follows:                              */
/* ------------------------------------------------------------------ */

  lograte = log10(count_rate);
  logratelow = log10(ratelow);
  logratehigh = log10(ratehigh);
  ratesqrt = sqrt(1.56986 - (0.373140 * lograte));
  ratelowsqrt = sqrt(1.56986 - (0.373140 * logratelow));
  ratehighsqrt = sqrt(1.56986 - (0.373140 * logratehigh));
  ratepower = 6.91301 - (ratesqrt / 0.186570);
  ratelowpower = 6.91301 - (ratelowsqrt / 0.186570);
  ratehighpower = 6.91301 - (ratehighsqrt / 0.186570);
  ratecorr = pow(ten, ratepower);
  ratelowcorr = pow(ten, ratelowpower);
  ratehighcorr = pow(ten, ratehighpower);
```

```
/* ------------------------------------------------------------------ */
/* end of pile-up correction                                          */
/* ------------------------------------------------------------------ */

  printf(" %8.7lf\t", bintime);
  printf(" %8.7lf\t", ratecorr);
  printf(" %8.7lf\t", binstart);
  printf(" %8.7lf\t", binstop);
  printf(" %8.7lf\t", ratelowcorr);
  printf(" %8.7lf\t", ratehighcorr);
  printf(" %4d\t", srcbin);
  printf(" %4d\t", bgdbin);
  printf(" %8.7lf\n", exposure);
  return 0;
}
```

Bibliography

[1] -Butler, N.; Kocevski, D. *X-ray Hardness Evolution in GRB Afterglows and Flares: Late Time GRB Activity Without N_H Variations* The Astrophysical Journal, 663:407, 2007

[2] -Davis, John E. *Event Pileup in Charge Coupled Devices* The Astrophysical Journal, 562:575-582, 20. Nov. 2001

[3] -Evans, P. A. et al. *Automatic analysis of Swift-XRT data* arXiv:0711.0433v1, astro-ph, 3. Nov. 2007

[4] -Evans, P. A. et al. *An online repository of Swift/XRT light curves of γ-ray bursts* Astronomy and Astrophysics, 469:379-385, Jul. 2007

[5] -Jóhannesson, G. *Numerical Simulations of Gamma-Ray Burst Afterglows* University of Iceland, Faculty of Science, Department of Physics, 2006

[6] -Moretti, A. et al. *In-flight calibration of the Swift XRT Point Spread Function* UV, X-Ray, and Gamma Ray Space Instrumentation for Astronomy, Proceedings of the SPIE, 5898:348-356, Jan. 2005

[7] -Nousek, J. A. et al. *Evidence for a Canonical Gamma-Ray Burst Afterglow Light Curve in the Swift XRT Data* The Astrophysical Journal, 642:389-400, 1. May 2006