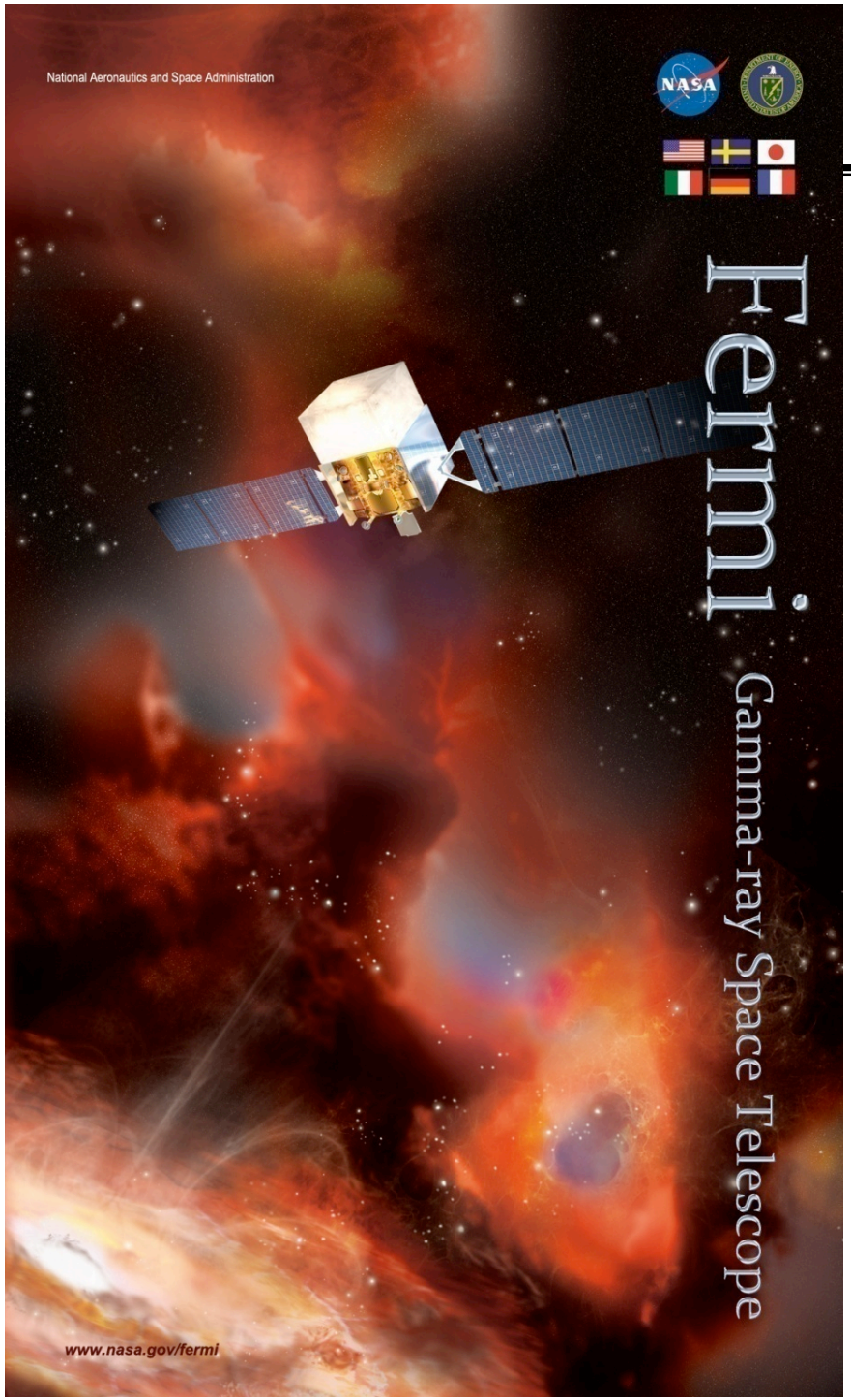


Fermi Gamma-ray Space Telescope

gtgrb

Status and prospects

intro

- Gtgrb was developed starting from Nukri's scripts
- Johann made a package to be used within ipython framework
- Many scripts collected in a common framework
  - GRB, LAT (Detector), GRB (Detector) encapsulate all the needed quantities, and share the various parameters
    - File paths
    - Ra, Dec, ROI, Tstart, Tstops,
  - A series of utilities is also available
    - Latutils, genutils, plotter (pylab and/or ROOT)
  - Examples: To convert MET in date and vice versa
  - Conversions from root to fits and vice versa
  - Merging of Fits files, FT1 and FT2
  - The common frameword help the development of new scripts and ensure the trasparency (the developer takes care of interfacing the script with gtgrb, and provide a method to call…)
- Code maintained under cvs

Interactive analysis

- Gtgrb is not an automated analysis!
- GRB analysis is unique for each burst, and the goal is provide here a tool for helping BA in gong through the analysis, interactively…
- Once all the classes are loaded into the ipython interface they can be called via the command line
- A tutorial of how to use gtgrb interactively is here (some procedure might be old and need to be reviewd…)
  - https://confluence.slac.stanford.edu/display/SCIGRPS/Scripting+and+Interactive+analysis+of+GLAST+GRBs+within+a+single+framework

Installation

- Gtgrb works everywhere works ST
- (I use at SLAC, on noric, in bash)
- cvs co -d GRBanalysis users/cohen/GRBanalysis
- cd GRBanalysis
- Edit gtgrb_slac indicating the location of the directories containing the files

```
#!/usr/bin/env bash
#####################################################
export BASEDIR=/nfs/farm/g/glast/u33/omodei/GRBanalysis
export INDIR=/nfs/farm/g/glast/u33/omodei/DATA/FITS
export OUTDIR=/nfs/farm/g/glast/u33/omodei/DATA/GRBOUT
#####################################################
# Science Tools installation
export DIFFUSEMODEL='/nfs/farm/g/glast/u33/diffuse/MapCubes/gll_iem_v01.fit'
export CMTCONFIG=rh9_gcc32opt
export STVER=v9r15p2
#####################################################
```

Input and Output

The gtgrb_slac script define the following variables
DATA IN: /nfs/farm/g/glast/u33/omodei/DATA/FITS
DATA OUT: /nfs/farm/g/glast/u33/omodei/DATA/GRBOUT

Expects all the LAT files (ft1 and ft2 needed here):
/nfs/farm/g/glast/u33/omodei/DATA/FITS/LAT
And the GBM file here:
/nfs/farm/g/glast/u33/omodei/DATA/GBM/GRBYYMMDDFFF

./gtgrb_slac
… long text, paths,

```
**************************************************
* WELCOME TO THE GLAST GRB QUICK ANALYSIS FRAMEWORK *
* BASED ON IPYTHON ICL and the GLAST ScienceTools   *
**************************************************
Activating auto-logging. Current session state plus future input saved.
Filename       : /nfs/farm/g/glast/u33/omodei/GRBanalysis/logfiles/gtgrb-2009-06-22.log
Mode           : rotate
Output logging : False
Raw input log  : False
Timestamping   : False
State          : active
```

Run iteratively

An example in scripts/GRB080910.py
(in ipython, if you execfile('scripts/GRB080910.py') it will run all the commands

./gtgrb_slac

```
TTRIGGER=263607781.0
RA=333.55
DEC=-26.61
ROI=15
EMIN=100
EMAX=100000
EBINS=10
TSTART=-10
TSTOP=60
DT=0.1
FT1='/nfs/farm/g/glast/u33/omodei/DATA/FITS/LAT/gll_ph_r0263605997_v001.fit'
FT2='/nfs/farm/g/glast/u33/omodei/DATA/FITS/LAT/gll_pt_r0263605997_v001.fit'

tstart=TTRIGGER+TSTART
tstop=TTRIGGER+TSTOP

IRFS='P6_V3_TRANSIENT'
```

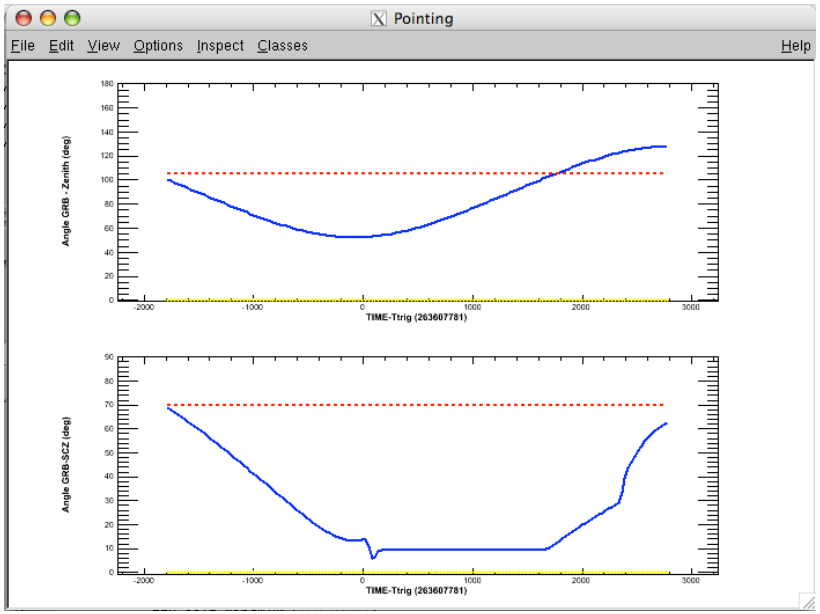
One can define the needed variables…

Various constructors

```
grb=GRB.GRB('GRB090510')
grb.Ttrigger=TTRIGGER
lat=LAT.LAT(grb=grb,ft1=FT1,ft2=FT2)
lat.Ebins=EBINS
lat.setEmin(EMIN)
lat.setEmax(EMAX)

lat.setTmin(tstart)
lat.setTmax(tstop)

lat.setRa(RA)
lat.setDec(DEC)
lat.setROI(ROI)

lat.ResponseFunction=IRFS

lat.print_parameters()
lat.plotAngSeparation(BEFORE=1000,AFTER=1000)
```

This print the current confuguration:

And plot the “angular separation”



```
lat.make_select(zmax=105)
#lat.make_gtmktime()
## Fixed ROI:
lat.make_pha2(tstart=lat.GRB.TStart,  tstop=lat.GRB.TStop,dtime=DT)
lat.make_pha1(tstart=lat.GRB.TStart,  tstop=lat.GRB.TStop)
lat.make_rsp()
## Energy dependent ROI:
#lat.make_rsp2('PHA2')
#lat.make_rsp2('FIT')

lat.saveEvents2Root()
lat.make_LightCurve(DT)
lat.plotLC()

binsz=.1
lat.make_skymap(nxpix=int(ROI/binsz), nypix=int(ROI/binsz), binsz=binsz)
lat.plotCMAP(drawopt="colz")
```

<- “Aurelien scrip” is called for a ROI(E)

<- This save the events in a root file

The pha2 file are ready for rmfit

likelihood

```
DIFFUSECOMPONENT=True

filePath='source_model.xml'
latutils.CreateSource_XML(filePath)

if DIFFUSECOMPONENT:
    latutils.Add_DiffuseComponents_XML(filePath)

    latutils.Add_PointSource_XML(xmlFileName=filePath,
                                 name='GRB',
                                 ra=RA,
                                 dec=DEC,
                                 flux=1.e-5,  index=-2.0)
    pass
latutils.Close_XML(filePath)

if DIFFUSECOMPONENT:
    lat.make_expCube()
    lat.make_expMap()
    lat.make_gtdiffrsp()
else:
    lat.expMap='none'
    lat.expCube='none'
pass
like      = lat.pyLike(model=filePath)
TS        = like.Ts('GRB')
like.plot()
like.plotSource('GRB','red')
like.model
print TS
```

This is just a switch in case you don't want
the diffuse component added in the fit

Like is returned, so, all the methods from likelihood are accessible

xspec

- **Since xspec cannot be directly interface, gtgrb write a script containing all the xspec commend, and then execute this with xspec**

```
import GTGRB.makeXSPEC as makeXSPEC
makeXSPEC.LAT_spectrum(grb=grb, lat=lat, x='yes')
```

- **Similar interface for making LAT and GBM spectral fit in XSPEC**

Script/GRBREPORT.py

- Executing the script/GRBREPORT.py one is “driven” through the various part of the analysis…
- The code ask input values and save the answer…
- The idea is to develop something that can be used from BA, without any skill of python, avoiding automatic procedures… some BA have found this useful.

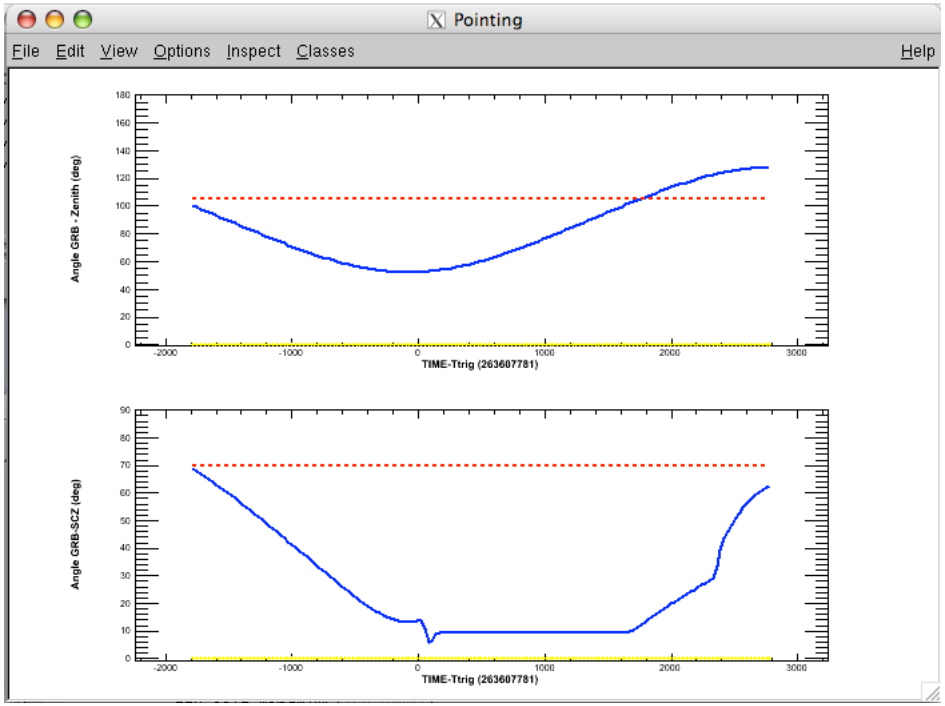
How to

- **execfile('scripts/GRBREPORT.py')**

Enter the GRBTRIGGERDATE (MET or YYYY-MM-DD HH:MM:SS [2008-08-03 18:31:23]263607781.0
Enter the TSTART with respect the trigget time (s)   [-10.0]
Enter the TSTOP with respect the trigget time (s)   [60.0]
Enter the RA of the GRB (deg) [281.833]333.55
Enter the DEC of the GRB (deg) [81.55]-26.61
Enter the R.O.I.   (deg) [15.0]
Enter the Minimum Energy (MeV) [10.0]80
Enter the Maximum Energy (MeV) [600000.0]80000
Enter the Number of Log bins   [10]
Enter the IRF   [P6_V1_TRANSIENT]P6_V3_TRANSIENT
Optional: FT1 file (or look in $INDIR):[]
Optional: FT2 file (or look in $INDIR):[]
Enter the Localization Error at 1-sigma (deg) [0.1]0.01
%%

[…]

plot angular separation? [y/N]



- **Plot the angular separation (it calls the script that Aurelien did time ago, and uses root for displaying the results)**

GRB Ra, dec= (333.6,-26.6)
SC Ra, dec= (337.1,-13.5)
Zenith Ra, dec= (338.3,25.5)
Lat in SAA? False
GRB THETA : 13.4957334623
FROM ZENIT : 52.2664880184
press enter to continue

make select? [y/N]y
ZMAX: [180]105
time -p gtselect infile=/nfs/farm/g/glast/u33/omodei/DATA/FITS/LAT/gll_ph_r0263605997_v001.fit outfile=/nfs/farm/g/glast/u33/omodei/DATA/GRBOUT/090510016/090510016_LAT_ROI.fits ra=333.55 dec=-26.61 rad=15.0 tmin=263607771.0 tmax=263607841.0 emin=80.0 emax=80000.0 zmax=105.0 evclsmin=0 evclsmax=10 convtype=-1 phasemin=0.0 phasemax=1.0 evtable="EVENTS" chatter=2 clobber=yes debug=no gui=no mode="ql"
Done.
real 2.46
user 0.60
sys 0.09
Selected : 243 Events...

Apply GTMKTTIME? [y/N]:y
time -p gtmktime scfile=/nfs/farm/g/glast/u33/omodei/DATA/FITS/LAT/gll_pt_r0263605997_v001.fit sctable="SC_DATA" filter="IN_SAA!=T && LIVETIME>0 && (ANGSEP(RA_ZENITH,DEC_ZENITH,333.55,-26.61) + 15.0 < 105.0)" roicut=no evfile=/nfs/farm/g/glast/u33/omodei/DATA/GRBOUT/090510016/090510016_LAT_ROI.fits evtable="EVENTS" outfile="tmp.fits" apply_filter=yes overwrite=no header_obstimes=yes tstart=0.0 tstop=0.0 gtifile="default" chatter=2 clobber=yes debug=no gui=no mode="ql"
real 1.10
user 0.38
sys 0.09

Apply an energy dependent selection? [y/N]:y
Make PHA2 and RSP [y/N]:y
Theta GRB = 13.4957334623 at time: -10.400000006
Tmin=[-10.0]
Tmax=[60.0]
Localization Error in deg [0.01]=

If N selected => Standard ROI analysis (fixed radius)
If y => call rspgen_v4 (“Aurelien script”) and make a ROI(E)

An simple example to interface

- Basically gtgrb uses the stored data to dump the txt file that rspgen_v4 takes as input.
- Only 1 change to the original script:
  - rspgen contains duplicate code (pha2rmfit) that, in this framework, could be inherited from latutils module.
-

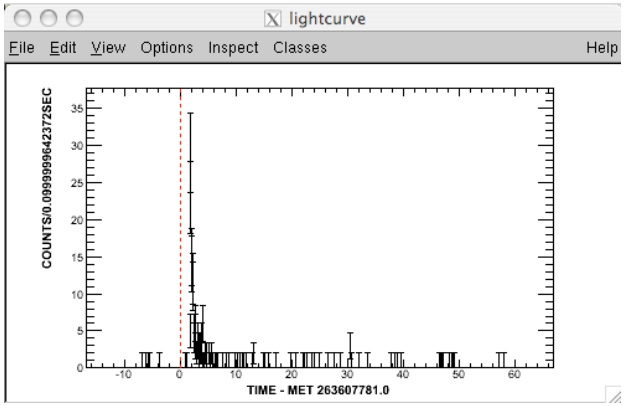
```
#
# Parameter file
#

PHAtype= PHA2
### Output files ###
FIT = /nfs/farm/g/glast/u33/omodei/DATA/GRBOUT/090510016/090510016_LAT_ROI.fits
PHA = /nfs/farm/g/glast/u33/omodei/DATA/GRBOUT/090510016/090510016_LAT.PHA2
RSP = /nfs/farm/g/glast/u33/omodei/DATA/GRBOUT/090510016/090510016_LAT.RSP
### Input parameters ###
FT1 = /nfs/farm/g/glast/u33/omodei/DATA/GRBOUT/090510016/090510016_LAT_ROI.fits
FT2 = /nfs/farm/g/glast/u33/omodei/DATA/FITS/LAT/gll_pt_r0263605997_v001.fit
Tstart = -10.0
Tstop = 60.0
Ttrig  = 263607781.0
RA = 333.55
DEC = -26.61
ERROR_RADIUS = 0.01
THETA = 13.5
EMIN = 80.0
EMAX = 80000.0
Ebins = 10
#dTime = 10
tbInputFile = './timebins_pha2.txt'
ResponseFunction = P6_V3_TRANSIENT
```

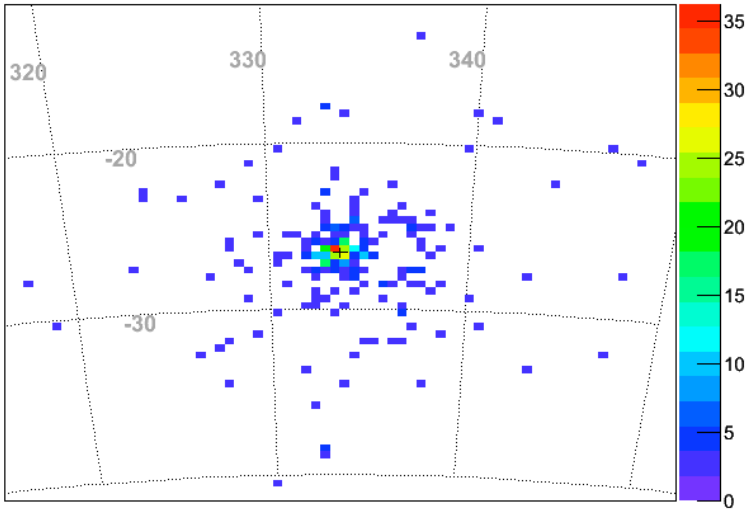
After selecting the events…

- **It create a root file with a simple tree containing all the events selected (useful for inspection)**

LIGHT CURVE BIN SIZE (seconds):.1

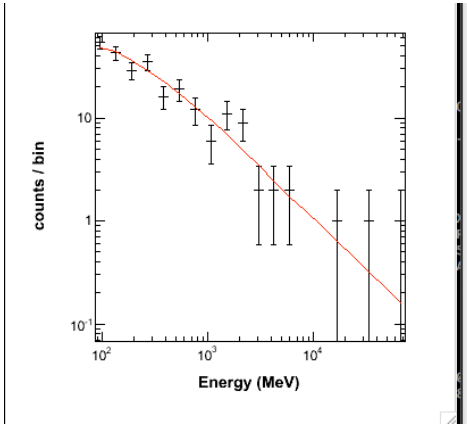
SKY MAP Bin size (degrees):.1

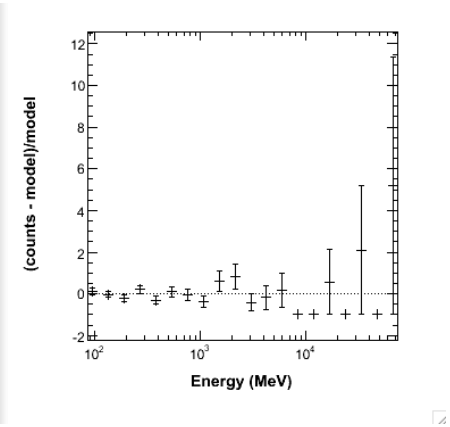




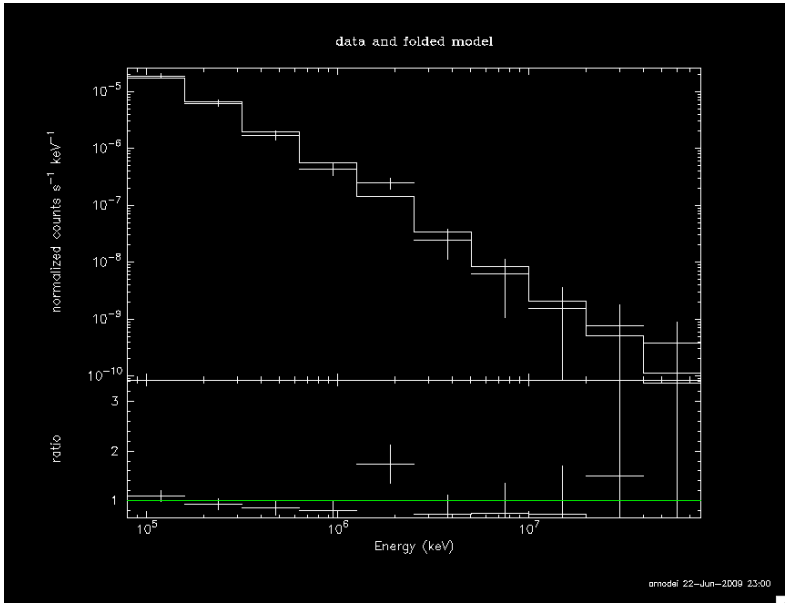
Spectral analysis

- **Once gtgrb makes pha2 files it saves them in the correct format to be used in rmfit… (and you will do the spectral analysis is rmfit)**
- **There are also implemented the pylike and xspec intefaces for making spectral fitting**






```
log(like)=  838.987207843
Warning: divide by zero encountered in double_scalars
 Flux = 0.389892 +/- 0.059617 erg/cm^2/s
 Flux = 0.000594 +/- 0.000038  ph/cm^2/s
```



Make use the upperLimits module (from Jim) that compute the UL using the likelihood profile method.

Conclusions

- Gtgrb is a framework optimal for importing new scripts from the GRB group based on ST
- It could be used interactively, or using scripts for facilitateing the analysis
  - (scripts/GRBREPORT.py, scripts/GRB090510.py)
- It is really useful if more than one people adopt this framework (easy to develop in parallel)
- We could distribute with ST within the collaboration

- Coming soon…
  - Make nice LC
  - Making UL using Fred and Veronique method (fit the background to constrain the parameter of the diffuse emission model)