Recipe: X-ray Analysis (NICER)

Tejas Sewak[1]
[1]Department of Physics and Astronomy, Texas Tech University

February 2025

Abstract

A detailed overview of steps (and files to use from this repository) for spectral and timing analysis of X-ray data from NICER. The scripts were made to analyse AGN data but can also be used for other systems with minor changes to filtering (and other required) processes.

Step - 0: Setting up environment

HEAsoft (High Energy Astrophysics Software) is the main software to use for all kinds of X-ray Analysis. The tar file [1] for the software can be downloaded from the HEAsoft website. The user must ensure that mission specific tools (say for NICER) are selected along with other required packages (XANADU, General-Use FTOOLS and XSTAR is the usual order of requirement) before creating the tar file.

The user is advised to create and commit to a clean directory (say `softwares`) to download and extract HEAsoft (or any other astrophysics software). Once configured, the relocation might lead to failure of software running properly. The installation process is explained in detail for Mac and Linux users. For Mac users, make sure the required system-level packages are present before the configuration step.

Step - 1: Downloading the Data

All NICER data is publicly available and can be accessed through NICER Master Catalog. Use the Browse interface to search for your data. The most straightforward way is to type the source name and look for all available data for that particular source. In case the user requires data from certain observations, more information such as pi_lname (personal investigator last name) or obsid (observation ID) can be provided to filter the search.

[1]A `.tar` file (short for tape archive) is used to bundle multiple files and directories into a single file. Once downloaded, the user can extract the contents by running `tar -xvzf heasoft-6.34src.tar.gz` (make sure the tar file name and version matches your file if you are using this code line).

Once the required data is selected, the user can choose the option to retrieve the data (and download the `.tar` file [2]) or create a download script (network utility `wget` is required). Again, the user is advised to create a clean directory (say by source name: NGC_4051) and store the data.

Further analysis steps are followed according to the tasks designed by the NICER team. There are two levels of analysis: Level 2 for calibration and screening, and Level 3 for generating spectral products and light curves. Please refer to the NICER Data Analysis threads before using scripts from the next sections.

Step - 2: Calibration and screening using `nicerl2`

2.1 Setting up Calibration environment

1. Setup a calibration directory `caldb` or `nicer_caldb`. I prefer it setting up inside the directory of HEAsoft.

2. The easiest way to access calibration data for NICER is via Remote Access. Download the two required files: `caldb.config` and `alias_config.fits` in the `caldb` directory.

3. Set the environment variable for the two files and caldb remote access link in your `.bashrc` file:

   ```
   CALDBCONFIG=/.../heasoft-6.34/caldb/caldb.config; export CALDBCONFIG
   CALDBALIAS=/.../heasoft-6.34/caldb/alias_config.fits; export CALDBALIAS
   CALDB=https://heasarc.gsfc.nasa.gov/FTP/caldb; export CALDB
   ```

   where `...` represents the path where your HEAsoft directory is located.

4. Once the remote access is setup, the `nicerl2` task will automatically use the latest calibration files for data processing. There is no need to worry about using the latest calibration files (or downloading them).

2.2 Downloading Geomagnetic Data

Several background models such as SCORPEON require geomagnetic data (read more in the last section of this webpage). To download the required data:

1. Setup a geomagnetic data directory `geomag`. I prefer setting it up inside the directory of HEAsoft.

2. Set the environment variable for this directory path in your `.bashrc` file:

[2] extract using `tar -xvf downloadedfilename.tar`

`export GEOMAG_PATH=/.../heasoft-6.34/geomag` where again `...` represents the path where your HEAsoft directory is located.

3. Download geomagnetic data and install by running the following command anywhere:

   `nigeodown`

   Before beginning any new project, just run the `nigeodown` to ensure the geomagnetic data is up to date.

2.3 Running `nicerl2`

1. Copy the script `nicerl2-all.sh` to the directory in which data is stored and run it (read further before running). The following command will run for each observation:

   `nicerl2 indir=$obsid clobber=YES`

2. The above code runs the `nicerl2` with the standard settings where the screening tasks are set to their default values. The following are default values (taken from here) for some important screening conditions and are not required to be specified when running the above code:

   - `detlist=launch` : Choose all detectors active at the time of launch (equivalent of writing all). The user can then look for detectors with noisy behavior and use different `detlist` (read more here on how `nifpmsel` which is part of `nicerl2` works to select/exclude certain detectors).
   - `overonly_range = 0-30` : (purpose: to mitigate background flares) Excludes times when overshoot count rate is out of range A-B counts per second per FPM, as NICER's background level is correlated with "overshoots", which are saturating particle detections. The default range of 0-30 will exclude most strong background flare-ups. This can be changed depending on count rate of source, say faint source, where the user might want strict filtering.
   - `cor_range = *-*` : (purpose: to mitigate background flares) Cutoff rigidity used for filtering of high background regions. A cutoff rigidity of `1.5-**` can be used along with lower `overonly_range` (human judgment required) for faint sources or sources where flares are expected (read more here on how to mitigate electron precipitation flares). If the source is bright, then the default setting for both `cor_range` and `overonly_range` can be used and the user can later look for flares in the plots as explained in next section. If no strong flares are seen, the background models will do the job of accounting for flares. Another method to do this is to exclude epochs characterized by a total source-plus-background exceeding some value (say much above median) in the 8-10 keV band.

- `underonly_range = 0-500` : NICER's performance, especially the energy gain scale, is affected by optical loading. Reduce optical loading effects by reducing the default range 0-500 (this default value was taken from here and not sure if this default was changed compared to here) to a smaller range (0-50 is optimal).
- `mingti=5.0` : Minimum size of a GTI file is 5 seconds. Can be changed if the user wants minimum exposure for all GTIs.

3. A revised filter file (`.mkf`), calibrated but unscreened event files per-MPU (`ufa.evt`), and cleaned and screened full array event file (`cl.evt`) will be produced after running `nicerl2` .
4. After an initial run for all data, further screening can be made based on the background behavior (most importantly checking for background flares).

2.4 Checking for Background Flares

One good way (there are other ways too) to check if electron flares have made their way to the data is to make plots in the form of Figure 1. However, to make that kind of plot we need light curves in different energy bands for each single spectrum. We will revisit this topic but first let us see how to generate initial spectra and light curves.

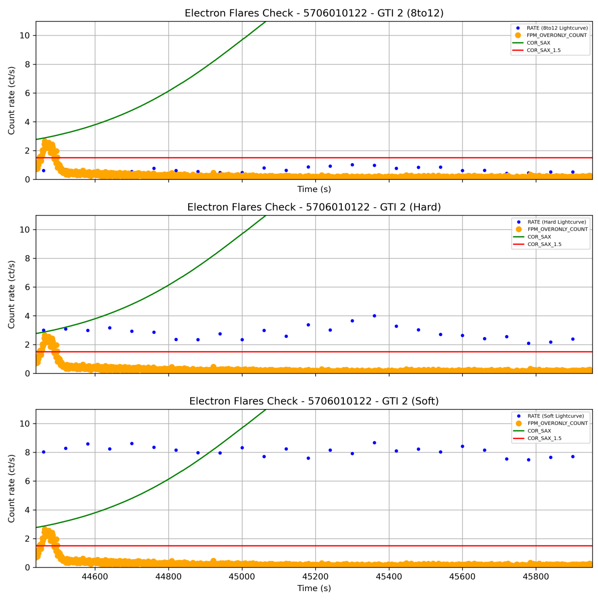

Figure 1: To check background flares. Orange data points are overshoot count rates, green line is cutoff rigidity values, red line is cutoff rigidity value of 1.5 (below which precipitating electron flares are found most of the time), and in blue points are count rates in 8-12 keV, 2-8 keV and 0.3-2 keV from top to bottom respectively.

Step - 3: Generating Spectra and Light Curves

3.1 Running `nicerl3-lc`

To create individual spectral products, we need to create custom gti files. One easy way to do is to first generate light curves for each complete observation. Using the timestamps for the light curves, we can create custom GTI files. These GTI files can then be used to create individual spectral products. So first, let us create lightcurves.

1. Copy the script `nicerl3-lc-all.sh` to the directory in which data is stored and run it. This will run the following command for each complete observation:

   ```
   nicerl3-lc $obsid pirange=30-200 timebin=60.0 suffix=_soft clobber=YES
   nicerl3-lc $obsid pirange=200-800 timebin=60.0 suffix=_hard clobber=YES
   nicerl3-lc $obsid pirange=800-1200 timebin=60.0 suffix=_8to12 clobber=YES
   ```

2. This will create three light curves: 0.3-2.0 keV, 2.0-8.0 keV and 8.0-12.0 keV and store in each of the directory. The user has the choice to edit `nicerl3-lc-all.sh` file and add commands to create more light curves in their choice of energy bands.

3.2 Creating custom GTI files

Depending on the observations and analysis goals, the user may want to create custom GTI files to create individual spectra. For example, the NICER observations taken during October 2022 for NLS1 MRK-359 had a cadence of 0.5 a day on average. So for each day, there were two Good Time Intervals (two segments) each with a good exposure (say >500 seconds after filtering). If the user wants to do timing analysis, i.e., to obtain light curves for each of the two segments, they can create two gti files, and then create subsequent spectral products using the custom gti files.

To do this task, the user can copy the python script `sc_gti_creator.py` to the directory where data is stored and run it. Here is a summary of what the script does:

1. For all observations, use the light curve (`.lc`) file to get the `gti_start` and `gti_stop` timestamps.

2. Create multiple gti files based on the timestamps and save them in the format of `gtin.fits` where n is the gti row number.

   For example, if the master gti looked something like this:

GTI-Start	GTI-Stop
278525000.0	278526000.0
278526500.0	278527500.0

GTI-Start	GTI-Stop
278525000.0	278526000.0

GTI-Start	GTI-Stop
278526500.0	278527500.0

then the two created gti files (`gti1.fits` and `gti2.fits`) will look like this:

Now that the individual gti files are created, the user can make plots as described in Figure 1 to identify background flares.

Note: We will later come back to combining certain gti files in order to increase exposure of our observations. For example, if the two gti files were only separated by a few seconds (or timescales $\ll$ expected exposure), then it is preferred to combine the two gti files and create a new spectral product to increase quality of the data. More on that later, after we have figured out which good time intervals are affected by background flares so we can ignore any such observations.

3.3 (Revisit) - Checking for Background Flares

1. Copy the script `sc_electron_flares.py` to the directory in which data is stored. Edit the script according to the file format of your observation id and number of observations present (more information available in the script itself). After customizing the script according to your needs, run it.

2. A new directory `pl_electron_flares` will be created in which plots as shown in Figure 1 will be present. From such plots, verify the presence of background flares (read more here).

3.4 Recreate custom GTI files

1. Along with showing information regarding flares, the plots will also tell the gaps between different gti files. If the two (or more) gti files were only separated by a few seconds (or timescales $\ll$ expected exposure), the user might want to combine them and increase exposure. To do so, the user can make use of the script `sc_gti_combiner.py`. The script asks for the observation id and the gti files to combine. It is not automated so as to allow user to combine only GTIs of their choice. Follow the instructions in `sc_gti_combiner.py` file to make a combined gti files.

2. After you have created the combined gti files from nearby gti files, the script will delete the original gti files. This will ensure that no obsolete gti files will be present that might affect further analysis.

3. Sometimes, a very small duration of the good time interval would be exhibiting flare like feature. In this case, the user can use `sc_gti_trimmerc.py` to trim gti files

and replace the original files. The user should make sure that this file is adjusted according to your requirement and format of observation ids.

4. It is time to remove gti files which exhibited flares. The plots will tell which Good Time Intervals show flare and the user would want to exclude them. The user can choose to manually delete the gti files or use the method in the next section.

5. Now that we have ensured that our gti files are up-to-date with only best and combined observations, we can proceed with creating final dataset for our analysis, by running `nicerl3-spect`.

3.5 Running `nicerl3-spect`

1. Copy the script `sc_list_gti.sh` to the directory in which data is stored and run it. The script will look for all gti files in the observation directories (including combined gti files), and copy their path (with respect to the directory where data is stored) to a file called `list_gti.txt`. The file will look something like this:

   ```
   ./5706010101/xti/event_cl/gti1.fits
   ./5706010101/xti/event_cl/gti2.fits
   ./5706010102/xti/event_cl/gti1.fits
   ...
   ```

   This helps us in many ways. If we were to create some new gti files in future (say a combination of two gti files - `gti12.fits`), we can simply run `sc_list_gti.sh` again to update our `list_gti.txt`.

   Or, if the user did not want to manually delete gti files with flares, they can let the file paths come to `list_gti.txt` and then simply comment out files which they do not want. As long as we maintain our `list_gti.txt` file correctly, we can choose which observations to consider or not.

2. Now, with an updated gti list, we can run `nicerl3-spect` to generate spectral products.

3. Copy the script `nicerl3-all-eachgti.sh` to the directory in which data is stored and run it. The following command will run for each observation with the specific gti file:

   ```
   nicerl3-spect $obsid gtifile=$gtifilepath grouptype=optmin
   groupscale=25 suffix=_$gtinumber clobber=YES
   ```

4. The above code runs the `nicerl3-spect` for each observation segment (for each gti). The `grouptype=optmin` is the default way of grouping data for `nicerl3-spect` (read more here). The `groupscale=25` was chosen so that adjacent bins are

combined to ensure a minimum of 25 counts per bin. The following argument for minimum counts per bin was taken from here: The minimum number of counts is typically 10-25. A value of 10 is the bare minimum for approaching Gaussian statistics (~3 σ), while a value of 25 leaves a more comfortable margin. However, this value may need to be adjusted based upon feedback or experience.

5. By default, `nicerl3-spect` choose SCORPEON as the background model which can be simultaneously fitted with the source spectrum. The case with using the other model 3C50 is little different if we want to do time resolved spectroscopy (i.e., create spectral products for individual gti files). This is because 3C50 background model is currently not supported when using nicerl3-spect with a GTI file. A workaround to this is to run nicerl2 with the custom GTI file, and then run nicerl3-spect on that output from nicerl2. In this case no GTI file needs to be passed to nicerl3-spect. to use model 3c50 for custom gtis. This can be done once we know which all observations are to be considered final (i.e., after excluding flares).

6. The following products were generated (taken from here):

   - Extracted spectrum from cleaned event file (format `sr.pha`).
   - Background spectrum file (format `bkgfile`) or script (format `bkgscript`) depending on the background model setting.
   - ARF response file (format `.arf`) - effective area for target.
   - Sky ARF response file (format `sk.arf`) - effective area for diffuse sky.
   - RMF response file (format `.rmf`) - redistribution matrix for target.
   - Background response file (format `bg.rmf`) - redistribution matrix for particle background.
   - XSPEC "load" file (format `load.xcm`) - script to load files into XSPEC. The "load" file will help get the user started quickly by giving an example script that will load all the products into `XSPEC` (including the background).

7. Now in each of the observation directory, there will be load files like:

   ```
   ./5706010101/xti/event_cl/ni5706010101mpu7_load_gti1.xcm
   ./5706010101/xti/event_cl/ni5706010101mpu7_load_gti2.xcm
   ./5706010102/xti/event_cl/ni5706010102mpu7_load_gti1.xcm
   ...
   ```

   which can be loaded from xspec. Remember to open `xspec` from the main directory where all observations are present and use the above path to load. This is because the path of all files is hard coded in the `load.xcm` file with respect to the main directory.

3.6 (Optional:) Rerun `nicerl3-lc` for individual light curves

If the user wish to create light curves for individual GTIs, they can copy the script `nicerl3-lc-all-eachgti.sh` and run it from the main directory. This will create $3 \times \#$ of GTIs light curves, i.e., for 0.3-2.0 keV, 2.0-8.0 keV and 8.0-12.0 keV energy bands.

As this will create multiple files which the user may not require (say in case they want to obtain light curves through spectral modelling), it is optional to run this script.

The next section to generate plots to identify background flares will still work even when individual light curves were not created. This is because in that script, the master light curve for each observation is used to take data for good time interval regions.