Project Proposal

Spectral timing in Julia

Mentors: fjebaker, matteobachetti , stefanocovino

Personal Information

Name: KASHISH SHRIVASTAV
emails: Kashish Shrivastav Kashish Shrivastav
Github: Kashish2210
Slack: Kashish Shrivastav
Matrix:@kashish2210:matrix.org
Time Zone: IST (GMT + 5:30)

TABLE OF CONTENT:

PR links: open source contributions

Contributions to the Stingray software

1. Implementing and Testing Jacobian Functions in Stingray:
   Jacobian functions for GeneralizedLorentzianJacobian and SmoothBrokenPowerLawJacobian in stingray/simulator/models.py.
2. Small addition of Docs: I found a complication while testing Stingray, so I thought to add a fix for that in the docs.
3. bugfix-for-issue#851 : To ensure that at least one bin is always created, we modify the h_ones calculation to:

Pull requests in Stingray.jl

4. RECIPIES: recipes for creating Lightcurve assigned to me in this issue
5. Fourier transforms and spectra: building a powerspectrum.jl for this issue

6. Events are typically only in one FITS extension: fixing one of my merged PR for this issue

7. porting power_colors to Stingray.jl: I have implemented the PowerColours core functions from Stingray to Stingray.jl

8. Update Dependencies & GitHub Workflow for Newer Julia Versions: I have largely updated the previous library by removing metadata.
9. Readevent: this pr is used for reading the events files referencing this issue

10. Adding Sample_data_test in julia: this PR has been under maintenance for a while, will open after careful instruction from maintainers:)
11. Docs: this PR majorly includes basic documentation for the repo Stingray.jl as mentioned by fergus@cosroe.com to add some docs to test the workflow of docs

Some of my pull requests in other julia open source projects<JuliaAstro>

12. add "Proper memory safety with GC.@preserve:Added new safe_table_string_convert function for safer string handling in table operations:
13. In this PR, I have added SkyCoords.jl in gcirc.jl to make it more compatible

My Background:

I'm exploring augmented reality, AI, and scientific computing. I've been working with Python, JavaScript, and Django, and have some experience in computer vision using tools like OpenCV, PyVista, and CVZone. I also use Blender for basic 3D modeling and recently started learning Julia for scientific tasks.

Beyond technology, I have a deep fascination with **astrophysics, black holes, and relativity**. I have explored these topics through extensive reading, including works such as *The Theory of Relativity, The Theory of Everything*, and *The Mysteries of Black Holes*.

Interest in Open Astronomy:

I am excited about Open Astronomy's mission to advance open-source tools for astrophysical research. The opportunity to contribute to Stingray.jl matches my growing interest in scientific computing.

I deeply value Open Astronomy's commitment to **open science, reproducibility, and collaboration**. This project presents an opportunity to apply my skills, learn from experts, and contribute to tools that push the boundaries of **high-energy astrophysics**. I look forward to being part of a community driving innovation in astronomical data analysis.

1.0 Project Overview

This project aims to develop a comprehensive suite of high-performance tools for analyzing time-series data from astronomical X-ray observations, with a particular focus on signals originating from accreting black holes and neutron stars.

X-ray timing analysis plays a critical role in testing advanced physical theories related to compact objects. By studying periodicities in X-ray emissions from black hole surroundings or tracking the evolution of neutron star rotation periods, researchers can gain profound insights into extreme physical conditions that cannot be replicated on Earth.

To facilitate such research, this project will develop **Stingray.jl**, a Julia-based alternative to existing Python tools, offering superior performance for processing large-scale X-ray datasets. The core functionalities of **Stingray.jl** will include:

- **Periodogram Implementations** – Efficient algorithms for detecting periodic signals in time-series data.
- **Cross-Spectral Analysis** – Advanced techniques for studying correlations between different time-series datasets.
- **Energy-Dependent Variability Spectra** – Tools for analyzing how X-ray variability changes across different energy bands.

By leveraging Julia's speed and just-in-time (JIT) compilation, **Stingray.jl** aims to outperform existing Python-based tools, particularly in computationally intensive tasks such as power spectrum fitting using automatic differentiation (AD) compiled to machine code as fergus@cosroe.com had explained earlier in this discussion. This approach will enable more efficient and scalable research in high-energy astrophysics, providing astronomers with a robust and optimized framework for X-ray timing analysis.

Anticipated Scientific Impact

This project has the potential to drive significant advances in high-energy astrophysics by enabling faster and more robust analysis of X-ray data from accreting black holes and neutron stars. The improved performance of the Julia-based tools may help answer key scientific questions such as:

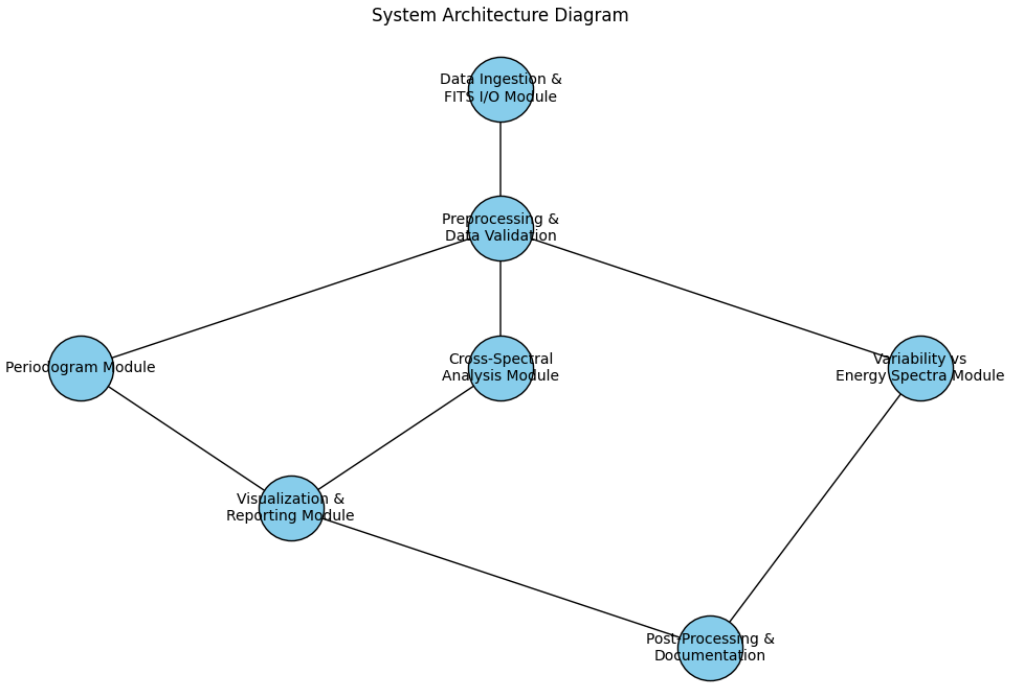
- **Detecting Subtle Periodicities:** How do small-scale periodic signals in X-ray emissions influence our understanding of black hole accretion processes?

- **Time-Dependent Phenomena:** Can high-resolution cross-spectral analysis reveal new insights into the time lags between different energy bands that may signal novel physical processes near compact objects?

- **Variability Analysis Across Energy Bands:** How does the variability in X-ray emissions change as a function of energy, and what does that imply for the structure of the emitting region?

By offering a computational platform with improved efficiency and scalability compared to existing Python tools, the project is expected to facilitate rapid prototyping, testing, and validation of astrophysical theories, ultimately accelerating discoveries in the field.

System Architecture Overview (Visual Aid)



#The following picture is genrated using matplotlib

Summary

The primary goal of **Stingray.jl** is to explore and exploit Julia's unique capabilities to enhance X-ray timing analysis, offering features that are difficult to achieve with Python-based implementations. By integrating cutting-edge computational techniques, this project seeks to push the boundaries of astronomical data analysis and facilitate groundbreaking discoveries in astrophysics.

Technical Implementation Roadmap:

A. Core Data Structures

- EventList type for handling time-tagged photon data
- LightCurve type with support for:
  * Good Time Intervals (GTI) management
  * Automated gap handling
  * Flexible binning options

B. Spectral Timing Analysis

- Power Spectral Density (PSD) calculations with:
  * Multiple normalization options (Leahy, rms, etc.)
  * Geometric frequency rebinning
  * Proper error propagation
- Cross-spectral analysis featuring:
  * Time lag calculations
  * Coherence estimation
  * Phase relationship studies

C. Advanced Statistical Tools

- Maximum Likelihood Estimation (MLE) for:
  * PSD modeling using multiple components
  * Automatic differentiation integration
- Power Colors Analysis:
  * Variance ratio calculations across frequency bands
  * Hue angle computations for state classification

D. Performance Optimizations

- Parallel processing for large datasets
- SIMD optimizations for core computations
- GPU acceleration where applicable

2.0 Milestones{as mentioned in project Idea}

- ☑ **Coding starts:**
- ☑ Gain familiarity with the codebase
  I have gained much familiarity with the code base by implementing operations and making contributions to it [my_prs]
- ☑ Apply existing analysis to simulated datasets
- ☑ Implement I/O operation on FITS files
  I have applied this operation on the previous existing codebase, Stingray in Python, and Julia, my notebook carries info of how I have read "ni1200120104_0mpu7_cl.evt.gzt
  " NICER files and plotted specific graphs to gain knowledge of how things are working, and fergus@cosroe.com also helped me a lot by suggesting necessary changes in my PRs

- ☐ 1st evaluation
- ☑ Implement a series of tests in Julia that the new code will have to pass:
  A series of tests is going to be implemented as I move forward towards our final evaluation
- ☑ Extend basic operations (periodograms and cross spectra) to event lists and light curves
  Peridogram milestone is ongoing, and currently I have been working on cross spectra, and I have already added read event function :)
- ☑ Time lags and coherence spectra
- ☑ Final evaluation
- ☑ A working framework for variability vs energy spectra (covariance spectra, time lags)

3.0 Deliverables

1. Core Analysis Modules:

- Implement periodogram and cross-spectral analysis routines.
- Extend functionality to accommodate event lists and light curves.

2. I/O and Testing Infrastructure:[as I will develop and extend functionalities, I will implement edge cases in Tests.]

- Develop robust I/O operations for FITS files.
- This is the FTP site for nicer data. I will upload the event files here and use this data for testing.

- Create comprehensive test suites ensuring code reliability and performance.

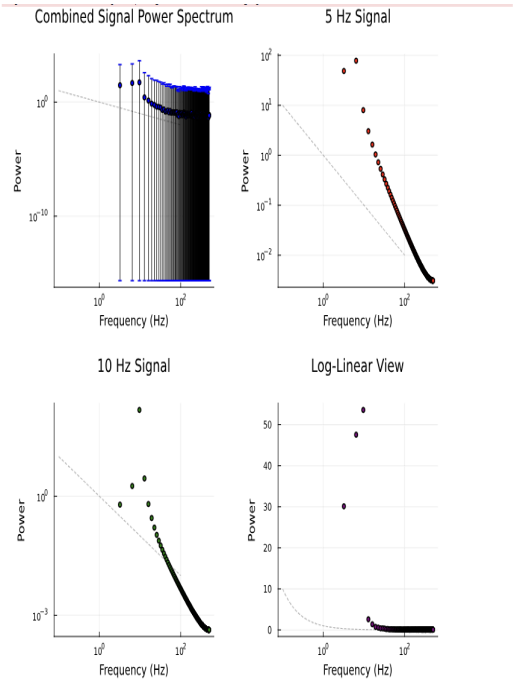
3. Advanced Analysis Features:

- Integrate advanced features such as time lag and coherence spectra.
- Finalize a working framework for variability vs energy spectra (including covariance spectra and time lags).

3.1.1 Implement periodogram and cross-spectral analysis routines.:

```
+
+ struct PowerSpectrum{T} <: AbstractPowerSpectrum{T}
+     freqs::Vector{T}
+     power::Vector{T}
+     power_errors::Vector{T}
+     dt::T
+     n::Int
+ end
```

```
+ struct AveragedPowerspectrum{T} <: AbstractPowerSpectrum{T}
+     freqs::Vector{T}
+     power::Vector{T}
+     power_errors::Vector{T}
+     m::Int
+     n::Int
+     dt::T
+ end
+
```



For this, I have decided to use this type of structure, and mostly I was planning to use an abstract type of structure, as you can see in the above images, this type of structure I have used as discussed in this discussion with fergus@cosroe.com
For a complete analysis of the periodogram, u may consider my notebook: click_here
I have plotted some graphs too by using these functions:

Note: cross_spectra adjustment in Fourier.jl remains the same although my plan is to make t like this it will be updated after updating the compare_table function in test_fourier.jl

```
struct CrossSpectrum
    freq::Vector{Float64}
    cross_spectrum::Vector{Complex{Float64}}
    coherence::Vector{Float64}
    phase_lag::Vector{Float64}
    time_lag::Vector{Float64}
end
```

3.1.2 Extend functionality to accommodate event lists and light curves.

1. Implementation of basic functionalities of eventlist with this type of structure has already been implemented and merged :)

```
struct DictMetadata
    headers::Vector{Dict{String,Any}}
end
```

Now the addition of the abstract structure is going on in my recipes PR.

```
+ """
+ struct EventList{T}
+     filename::String
+     times::Vector{T}
+     energies::Vector{T}
+     metadata::DictMetadata
+ end
```

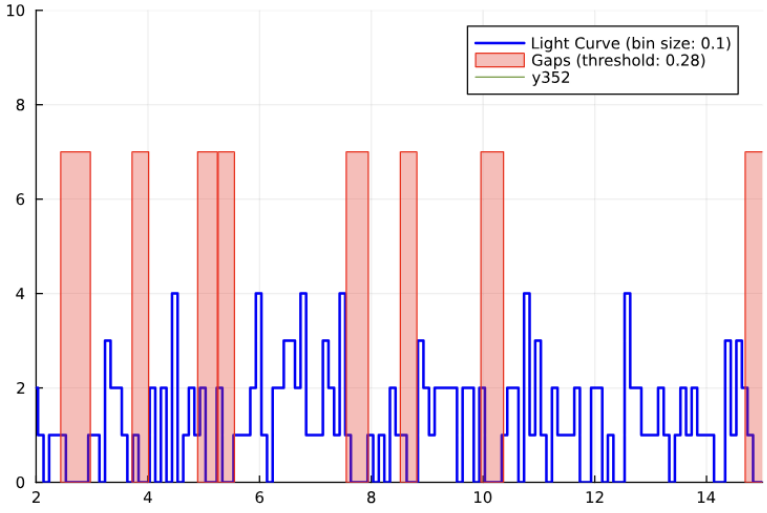
   The function readevent data will be extended with abstract, where we will add a mutable EVENTLIST:: Or raw EVENTLIST:: with more data

   Final goal: we will extend this function so that it will be able to read NICER event files and metadata about the dataset (e.g., telescope, object ID, etc.) are going to be initiated.

2. Implementation of basic functionalities of lightcurve with this type of structure

```
+ """
+ struct LightCurve{T} <: AbstractLightCurve{T}
+     timebins::Vector{T}
+     counts::Vector{Int}
+     count_error::Vector{T}
+     err_method::Symbol
+ end
```

3. All of up will going to be used in the @rescpies function to implement a plot I have implemented most of the functions in my PR, like plot(event_list,0.1,show_gaps=true,gap_threshold=5.0,axis_limits=[2, 15, 0, 10])



3.1.3 Integrate advanced features such as time lag and coherence spectra.

The basic steps are:

1. **Compute the Fourier Transforms:** Convert your two light curves (or time series) into the frequency domain.

2. **Calculate the Cross Spectrum:** Multiply one Fourier transform by the complex conjugate of the other.

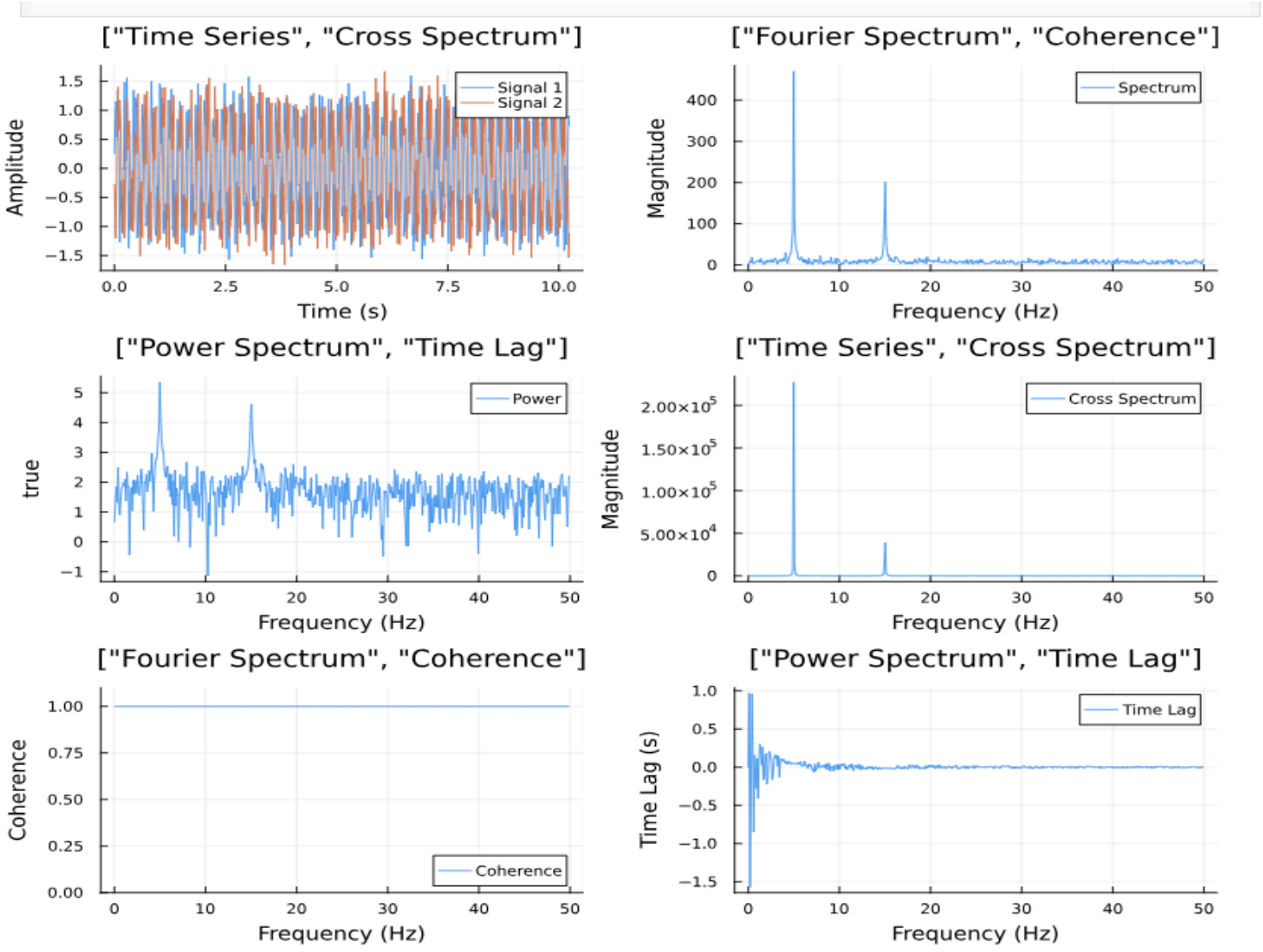
3. **Derive the Time Lag:** Compute the phase difference (angle) of the cross spectrum and convert it into a time delay by dividing by 2πf2\pi f2πf.

4. **Compute the Coherence:** Measure the linear correlation between the two signals as a function of frequency by comparing the cross-spectrum power to the product of the individual power spectra.

For the implementation of these, please review my notebook :)

Note: I am using an inner constructor, which is not good. I found in this discussion that I will change it to outer in the actual implementation

While testing, I was thinking of using Gaussian signal and its windows from this reference



3.3.2 Finalize a working framework for variability vs energy spectra (including covariance spectra and time lags).

These are some basic implementations I have figured out, yet I need guidance from mentors to implement them and create an efficient code base.
Note: I have included some extra recipes, like heatmaps, that can be removed. I was curious to see how that worked.

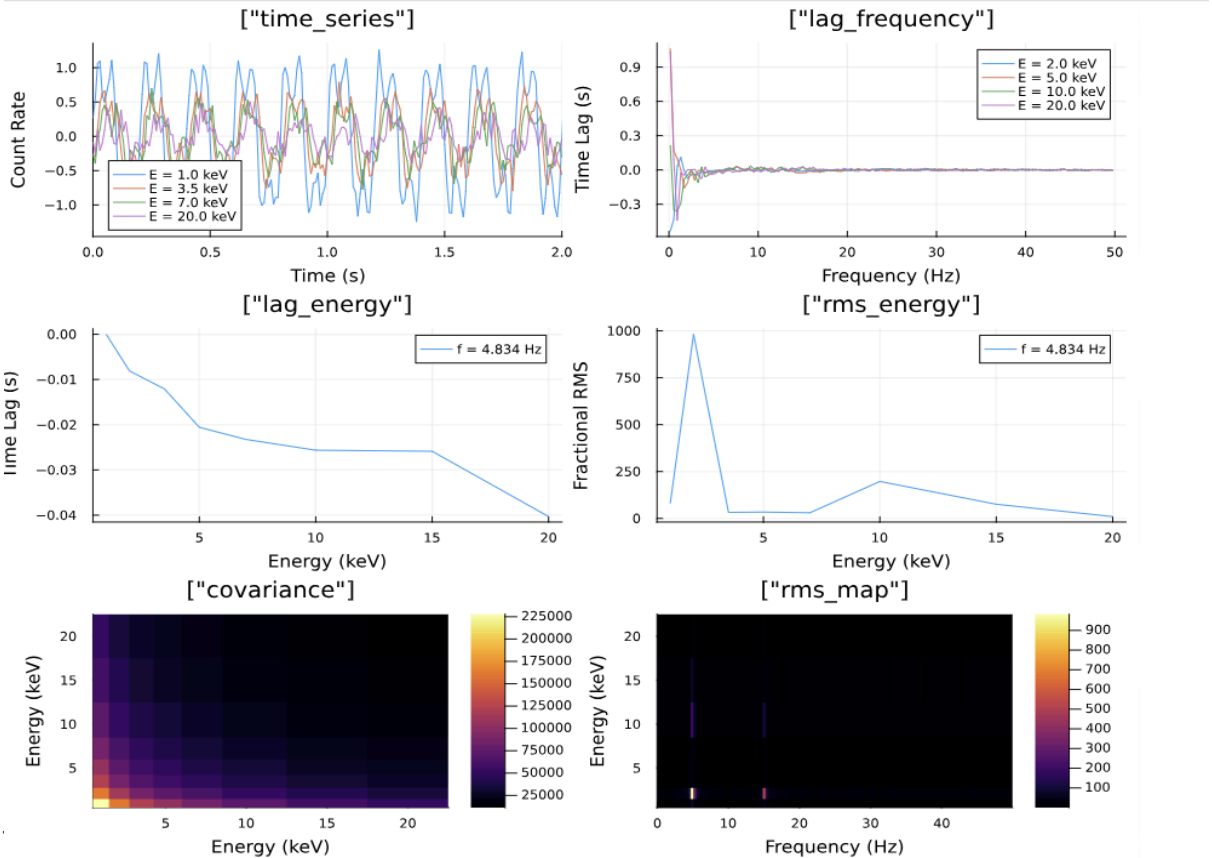
New Data Structures(also curious for time series)

1. **EnergyTimeSeries** - Extends time series to include multiple energy channels
2. **EnergyFourierSignal** - Fourier transforms for multiple energy channels
3. **CovarianceSpectrum** - Stores energy-energy covariance at each frequency
4. **EnergyLagSpectrum** - For analyzing time lags between different energy bands
5. **RMSEnergySpectrum** - For calculating fractional RMS variability per energy band

Key Analysis Functions

1. **Fourier transforms** - Extended to handle multiple energy channels simultaneously
2. **Covariance spectrum** - Correlations between variability in different energy bands
3. **Time lag spectrum** - Energy-dependent time lags relative to a reference band
4. **RMS spectrum** - Fractional variability as a function of energy and frequency
5. **Frequency rebinning** - For improving signal-to-noise in spectral products

I have implemented basic structures of code, you may refer to this notebook :)



Risk Management and Mitigation Strategies

As with any ambitious software project, several risks may arise. Below are identified risks along with proposed strategies to mitigate them

Evolving Julia Releases:

Risk: Breaking changes or instability in Julia nightly builds may affect module compatibility.

Mitigation: Maintain a dedicated branch that tracks the stable release while testing new features on nightly builds. Implement continuous integration tests focused on key functionalities.

Integration with Existing Tools:

Risk: Difficulties integrating the new modules with the current Stingray and Astro-related libraries might delay progress.

Mitigation: Engage early with the community and mentor feedback. Schedule initial integration tests as part of the first evaluation phase and gradually iterate based on feedback.

Performance Bottlenecks:

Risk: Unforeseen performance issues, particularly with large datasets, might compromise the project's objectives.

Mitigation: Set up performance benchmarks early in development and integrate profiling tools to constantly monitor and optimize code performance.

4.0 Pre-community bonding and current working:

I have joined the Slack channels for Julia, Stingray, and the Matrix channel for OpenAstronomy, where I introduced myself and began engaging with the community. While exploring the project and tools, I encountered some basic issues, which I discussed in the Julia helpdesk. I truly appreciate the community's responsiveness and support in addressing these questions.

During this process, I discovered that Julia's nightly build currently exhibits instability with certain features such as resumable functions and DataFrames. As a result, I am planning a future contribution aimed at improving compatibility with Julia nightly by refactoring parts of the codebase—replacing unstable patterns with more robust structures, such as structs.

This enhancement remains part of my long-term plan to support the maintainability and performance of the project.
Anyways, this plan is an afterword plan,

Initially, what I am doing now:
Learning about the stingray, inspecting the codebase, and knowing how it can be implemented,
In deliverables, I haven't mentioned GTI and bad time intervals, etc, but I am studying that part, initiated the basic idea of that, also you can see in this discussion:)

5.0 GSOC timeline and my availability

Period	Tasks
May 8 - May 20 (Community Bonding)	1. Connect with mentors and understand expectations for the project. 2. Revisit Stingray.jl and its Python counterpart to understand the scope of porting. 3. Familiarize with core concepts: X-ray timing analysis, light curves, power spectra, periodograms, cross spectra, and FITS file format. 4. Discuss and finalize milestones, Julia conventions, testing strategies, and code structure.
May 20 - June 20 (Coding Period)	1. Implement basic I/O operations in Julia for reading and writing FITS files using `ReadEvent.jl`.(extending functionalities to read NICER data and adding more Metadata)

	2. Build core periodogram functionality using recipes and benchmarking against Stingray Python. 3. Create reusable modules for working with light curves and event lists. (improvement in Fourier.jl and its testing) 4. Document code and get early feedback from mentors via notebooks. 5. Overall, completing this milestone
June 20 - July 2 (Midterm Evaluation)	1. Finalize and polish basic periodogram and I/O tools. 2. Implement cross-spectrum analysis with appropriate recipes/functions and test on synthetic datasets. 3. Ensure consistency with astrophysical conventions and Stingray standards. 4. New milestone for PowerColors and I will add @recpies and working functions, which I have already initialized but need improvements. 5. Prepare midterm documentation and blog updates. 6. Submit progress report and respond to mentor feedback.
July 2 - August 2 (Coding Phase 2)	1. Implement time lag and coherence spectra modules. 2. Integrate the ability to analyze variability vs energy using covariance spectra. 3. Improve modularity, performance, and error handling for large datasets. 4. Optimize recipes for visualizing energy-dependent results. 5. Continue writing unit tests and update documentation.
August 2 – August 25 (Final Evaluation Week)	1. Final testing and performance benchmarking. 2. Refactor code to replace deprecated or unstable patterns with `struct`-based implementations. 3. Submit final code with documentation, examples, and tutorials. 4. Write the final blog post and usage guide. 5. Complete final mentor evaluations.
Post-GSoC (If Timeline Allows)	1. Collaborate on integration with Astro-related Julia packages. 2. Extend the project to support GPU-based computations using `CUDA.jl` (optional).

	3. Write an academic-style report or whitepaper summarizing the tool's performance and astrophysical applications. 4. - Contribute to long-term roadmap planning for Stingray.jl. 5. I will work for Julia versions (especially stable vs nightly)

NOTE: Additionally, more work will be added, and it will be flexible dates. These are just for context :)

5.2 Availability

Since this is a 350-hour project, I plan to dedicate around 3–4 hours each weekday and 6–8 hours on weekends and holidays. As a student, I'll make sure to manage my time well to maintain a healthy balance between academics and GSoC. I'm genuinely excited about the project and fully committed to giving it my best.

6.0 Some of my projects:

6.6.1 Real-time Einstein's field equation- space fabric generator:

This project uses Einstein's field equation to generate space-time fabric distortion. This project is based on "**Minkowski Space**" built in Julia

6.6.2WEB-TOOLKIT:

This project aims to provide HTML codes to specified users, like SVG_SCROLL and Shader_scroll. It is deployed here, u can try to use it :) It is built on Python [Flask]

6.6.3Health_app:

This project built on training image datasets to detect the emotion of a human, and also uses intermediate blockchain and new securityware like geocite at time stamp 6:15 :) built on DJANGO

6.6.4NASA_CLIMATE:

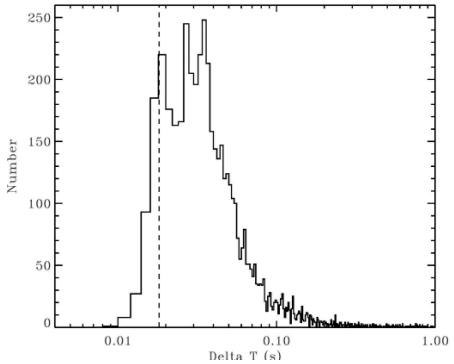
This project was initiated during the NASA_web_app_challenge. This project gathers data from NASA's climate data and generates the visual structure of Earth, with rendering different textures of Earth showing the effect of climate change based on the data provided, and many more :)

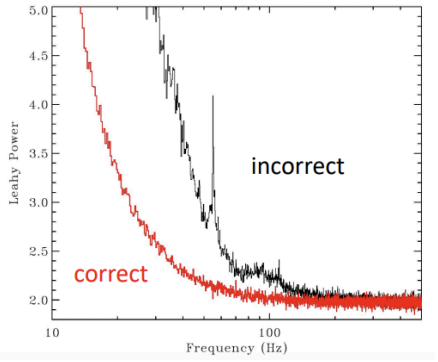
And many more projects on my GitHub and Glitch

7.0 What I am studying and for what

[1]https://heasarc.gsfc.nasa.gov/docs/nicer/data_analysis/workshops/nicer_wkshp_timing_5_4_21.pdf

From this PDF, I have studied the correct way of implementing GTI
Like “Fragmented GTIs and Timing Analysis”





[2] https://juliaastro.org/BackgroundMeshes.jl/dev/1.%20Basics/#RMS-Estimators

I will be learning RMS implementation in our @recpies functions.

[3] https://github.com/JaimeRZP/LimberJack.jl/blob/main/src/spectra.jl
https://github.com/JaimeRZP/LimberJack.jl

I am learning the spectra implementation and various cosmology functions.

[4]https://nbodykit.readthedocs.io/en/latest/getting-started/cosmology.html#Theoretical-Power-Spectra

Deep learning of powerspectrum.

[5] https://github.com/astropy/astropy/issues/17920

This issue is being in FITS, I will make a separate function like rm(temp_fits)
This stage is very crucial in managing memory efficiently while testing our files

[6]https://stackoverflow.com/questions/40423658/how-to-plot-statsbase-histogram-object-in-julia

This discussion is very useful while using a histogram, since I found an error I
Some of my implementations

[7] https://www.bromberger.com/LightGraphs.jl/dev/developing/ like lightkurve in Python
This was one favorite packages for lightcurve, sadly, it's been shut down, but we can use its implementation for testing and general purposes

[8]https://optiwave.com/opti_product/how-to-setup-the-gaussian-pulse-generator-in-optisystem/

We will use various signals to maintain simplicity, we can use these stable signals to meet our output in testing

[9]https://heasarc.gsfc.nasa.gov/docs/nicer/data_analysis/nicer_analysis_tips.html#lightleakincrease

For NICER data

[10]https://juliahub.com/ui/Notebooks/chris-rackauckas2/ASKEM-Scenarios/Scenario1(7).jl

We will deeply study these cases and make sure to meet these types of conditions or format while making functions :)

[11]https://docs.stingray.science/en/stable/notebooks/Spectral%20Timing/Spectral%20Timing%20Exploration.html:Mandatory:)

8.0 Appreciation:

First of all, I would like to express my sincere appreciation for the OpenAstronomy community. It truly feels right to be here—welcoming, insightful, and inspiring. I've already learned a great deal just by engaging with the community for a month and exploring its resources.

A special thank you to fergus@cosroe.com for his invaluable guidance on this project. His insights helped me understand the project direction clearly and provided a well-structured foundation for my preparation.

9.0 Gsoc:

Have you participated previously in GSoC? When? With which project?

Nope, this is my first GSOC

Are you also applying to other projects?

Yes, one small project: geomscale if I got selected:)