AuroraX, aurorax-api (or pyaurorax?), and aurorax-asilib: a user-friendly auroral all-sky imager analysis framework

M. Shumko[1,2], B. Gallardo-Lacourt[1,2], A.J. Halford[1], E. Donovan[3], E.L. Spanswick[3], D. Chaddock[3], I. Thompson, and K.R. Murphy

[1]NASA's Goddard Space Flight Center, Greenbelt, Maryland, USA
[2]Universities Space Research Association, Columbia, Maryland, USA
[3]University of Calgary, Calgary, Alberta, Canada

Key Points:

- AuroraX is an online interface to visualize the aurora and calculate conjunctions
- aurorax-asilib is a companion Python package for detailed analysis of auroral all-sky imager data
- Together, these tools enable effortless end-to-end discovery and analysis of the aurora

Corresponding author: M. Shumko, `msshumko@gmail.com`

Abstract

Abstract

Plain Language Summary

1 Introduction

OUTLINE

- Brief history of ASIs and ASI arrays. Talk about why THEMIS ASI exists. Discuss CANOPUS?
- Breadth of possible science questions that can be answered with aurora image data.
- Problem: modern ASI arrays produce an immense volume of data.
- Why this software? Aurora ASI data formats very greatly, each with their own caveats. This centralized software package is maintained by the AuroraX team.
- Benefits: Maintained by the AuroraX team so it's usability is of paramount importance
- Reduce the barrier to entry into auroral physics. Reduce the technical requirements and enable rapid discovery of new science.
- Instead of case study results, larger statistical behavior will likely appear.
- remove the need for scientists needing to write duplicate code to use these popular missions. AS a result, this will enable scientists to dive right into the science and not need to know the details of data management (downloading and loading data, as well as applying routine data processing steps

The rapidly increasing amount of imager data, together with unique data formats, significantly burdens space physicists with mondale and duplicated software engineering tasks—download data, load and parse the data correctly, etc. This unnecessary burden can also lead to mistakes in analysis software that may require unnecessary troubleshooting time from the ASI team. The goal of AuroraX is to overcome these drawbacks by providing a set of robust tools that most researchers need to analyze all-sky images.

We describe our progress towards that goal in this article. First, we showcase the main features of the online AuroraX interface (https://aurorax.space/) such as the conjunction finder and the keogram finder. Second, we describe the aurorax-api, a Python library containing the interface to the AuroraX server to automatically download keograms and identify conjunctions. Third, we describe aurorax-asilib, the Python all-sky imager library. aurorax-asilib provides functions to the download, load, process data, and visualize THEMIS and REGO ASI data.

2 Design Philosophy (Principals?)

OUTLINE

- The primary design philosophy is to offer a robust set of functions that are useful for most researchers studying the aurora. We strived to strike a balance between complicated and user-friendly tools.
- Online keogram and conjunction interface accessible anywhere with internet connection.
- Comprehensive ASI data analysis functionality on a PC.
- Abstract away data management steps: downloading data, loading data, applying routine data processing steps, and common visualizations.

3 AuroraX

OUTLINE

- What is it?
- A highly optimized conjunction search
- On-demand keograms
- Virtual Observatory
- pyaurorax (aurorax-api) to directly access AuroraX services.
- Figure 1: a) a screenshot of the nightly keograms, b) screenshot of the conjunction search tool.

4 aurorax-asilib

OUTLINE

- What is it? A Python library that helps researchers analyze THEMIS and REGO ASI images.The main functions are summarized in Table 1. It is designed to be simple and runnable on personal machines (relatively low memory usage). We strived to strike a balance between complicated and user-friendly tools.
- A table of function names and one sentence to describe their functions.
- The large file sizes lead to relatively long processing time. This is a fact that can be partly mitigated by an SSD.
- Plug-in based architecture that allows new ASI arrays to be added and called by the core aurorax-asilib software.

aurorax-asilib allows researchers to analyze ASI data on a PC. It provides a set of functions for common data analysis tasks using ASI data. Here we overview the functions and the online documentation has more examples, a tutorial, and a thorough API reference `https://aurora-asi-lib.readthedocs.io/`

The asilib functions are designed to help the user with the lower-level function calls. For example, if you want to load imager data, asilib will try to download it if it is not already saved on the PC. Likewise, if you call `asilib.plot_keogram()`, it will automatically load (and optionally download) the ASI data for you.

4.1 Download and load ASI image and skymap data

OUTLINE

- Handles the downloading and loading of ASI images. Main design principle: Ultimately, ASI image files consists of time stamps and images, so the asilib functions really only need to return that data
- Similarly with skymap calibration files
- If a file is already downloaded, you do not need an internet connection to work with the data

The four functions that download and load ASI image and skymap data are:

- `asilib.download_image()`,
- `asilib.download_skymap()`,
- `asilib.load_image()`, and
- `asilib.load_skymap()`

that are described below.

The `asilib.download_image()` function downloads the level 1 hourly Common Data Format (CDF) files from `http://themis.ssl.berkeley.edu/data/themis/thg/l1/asi/` for THEMIS, and `http://themis.ssl.berkeley.edu/data/themis/thg/l1/reg/` for REGO. The files are saved in the `asilib.config['ASI_DATA_DIR']` directory that at `~/asilib-data/` be default, and is customizable.

The `asilib.download_skymap()` function downloads all of the skymap files from `https://data.phys.ucalgary.ca/sort_by_project/THEMIS/asi/skymaps/` and `https://data.phys.ucalgary.ca/sort_by_project/GO-Canada/REGO/skymap/` for a given set of `asi_array_code` and `location_code`. The skymap files are in the Interactive Data Language (IDL) `.sav` file format. Noteworthy is that `asilib` downloads all of the skymap files for an imager because the skymaps are only valid for a set time period (typically a year; see `asilib.load_skymap()`below).

As the name implies, `asilib.load_image()` loads into memory and returns the ASI time stamps and images for a specified imager. This function loads both single and multiple images: a single time stamp and image if `time` is provided, and an array of time stamps and images if `time_range` is provided. As previously mentioned,`asilib.load_image()` will try to download an hourly CDF file if it does not exist locally.

`asilib.load_skymap()` is the last noteworthy input function; it loads the relevant skymap file into memory and returns the data in a dictionary. A relevant skymap file is the latest one before the specified `time`. As with `asilib.load_image()`, `asilib.load_skymap()` will attempt to download the skymap functions if they are not already downloaded.

4.2 Plotting single images

OUTLINE

- Fisheye lens view
- Project onto a geographic map
- Figure 2: a) fisheye view, b) that fisheye view mapped to 110 km. (THEMIS)

4.3 Keograms

OUTLINE

- Keograms along the meridian
- Uses generators to minimize the RAM usage
- Figure 3: A keogram for a full night. (REGO)

4.4 Creating ASI movies

OUTLINE

- Basic fisheye animation
- Basic map animation (need to add functionality)
- Using co-routines to superpose data onto images (extensively used for conjunctions).
- Reference SI movie 1.

4.5 ASI analysis tools

OUTLINE

- lla2azel
- lla2footprint (requires IRBEM)
- area2pixels

4.6 An example: a satellite-ASI conjunction

OUTLINE

- Combine everything above into an example showing where the footprint of a LEO satellite is, and what
- Figure 4: A conjunction montage and a time series
- (Implement an Imager.conjunction function)
- Reference movie S2

5 Quality Assurance

OUTLINE

- asilib on GitHub. unit and integration tests run automatically before every release.
- THEMIS and REGO data formats are set and won't change.

6 Future Development

OUTLINE

- Switch from CDF to pgm files.
- Support more ASI arrays such as TREx
- Combine into a class architecture to share data.

While aurorax-asilib is feature complete, we plan to keep developing it.

7 Conclusion

OUTLINE

- AuroraX, aurorax-api, and aurorax-asilib tools provide the science community with a simple and a robust set of analysis tools
- Enable system-level science to be easily done
- Quickly sift through an immense volume of data to uncover new physics
- This is an end-to-end solution
- Plan to add support for other ASI arrays and satellites
- Help promote a uniform ASI data format for future cameras

Acknowledgments

We are thankful for the engineers and scientists who made AuroraX, THEMIS ASI, and REGO ASI projects possible. M. Shumko and B. Gallardo-Lacourt acknowledge the support provided by the NASA Postdoctoral Program at the NASA's Goddard Space Flight Center, administered by Universities Space Research Association under contract with

NASA. The THEMIS and REGO ASI data is available from `https://data.phys.ucalgary.ca/`.