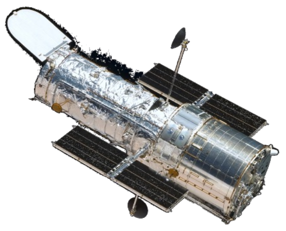
November 2019



Introduction to the Hubble Space Telescope Data Handbooks

Version 9.0

Also available online at

https://hst-docs.stsci.edu/hstdhb

3700 San Martin Drive
Baltimore, MD 21218
https://hsthelp.stsci.edu

Introduction to the Hubble Space Telescope Data Handbooks

Version 9.0 – November 2019
PDF version

Table of Contents

Expand all   Collapse all

User Support

Please contact the *HST* Help Desk for assistance. We encourage users to access the new web portal where you can submit your questions directly to the appropriate team of experts.

- Web: https://hsthelp.stsci.edu/
- E-mail: help@stsci.edu

Additional Resources

Information and other resources are available:

- http://www.stsci.edu/hst
- http://www.stsci.edu/hst/documentation

Revision History

Version	Date	Editors
9.0	November 2019	Tyler Desjardins & Ray Lucas Technical Editor: Susan Rose
8.0	May 2011	Ed Smith Technical Editors: Susan Rose & Jim Younger
7.0	January 2009	Brittany Shaw, Jessica Kim Quijano, & Mary Elizbeth Kaiser Technical Editors: Susan Rose & Jim Younger
6.0	January 2006	Diane Karakla Technical Editors: Susan Rose & Jim Younger
5.0	July 2004	Diane Karakla & Jennifer Mack Technical Editor: Susan Rose
4.0	January 2002	Bahram Mobasher, Michael Corbin, & Jin-chung Hsu
3.1	March 1998	Tony Keyes

3.0, Vol. II	October 1997	Tony Keyes
3.0, Vol. I	October 1997	Mark Voit
2.0	December 1995	Claus Leitherer
1.0	February 1994	Stefi Baum

Contributors

For version 9.0, the editors are grateful for the advice and information provided by Matthew Burger, Ivelina Momcheva, Doug Branton, Allyssa Riley, Elaine Frazer, Benjamin Kuhn, and Jenna Ryon.

Citation

In publications, refer to this document as:

- Desjardins & Lucas 2019, "Introduction to the Hubble Space Telescope Data Handbooks," Version 9.0, (Baltimore: STScI)

Change Log

Changes made to this document after version 9.0 was published are logged here.

November 9, 2022:

- References to Astroconda are replaced with stenv in Sections 4.2 and 4.3.

Preface

This *Introduction to the Hubble Space Telescope Data Handbooks* is intended to provide an introductory-level understanding of topics general to the data from all of the *Hubble Space Telescope* (*HST*) instruments. It introduces methods for: retrieving *HST* observational data from data archives; understanding the data file structure including how to find important information stored within the data files and observation logs; and software tools and methods for working with data from different types of *HST* instruments.

Users new to working with *HST* data are encouraged to review this document before proceeding to the data handbook for the relevant instrument(s). In addition, users already familiar with *HST* data may find it useful to consult the chapters about data retrieval and analysis for current best practices recommended by STScI.

This version (9.0) of the *Introduction to the Hubble Space Telescope Data Handbooks* contains numerous updates from the previous version, and it is the first version available via the *HST* User Documentation (HDox) website. Updated information in this version includes data archive access, file format descriptions, data analysis software, and additional information regarding observation log files. Beginning in late 2019, STScI no longer supports the use of IRAF/PyRAF for *HST* analysis. In light of this change, descriptions of IRAF/PyRAF tools have been removed. Users requiring information about IRAF/PyRAF or GEIS file formats should consult version 8.0 of this document.

Typographic Conventions

To help you understand the material in this Data Handbook, we will use a few consistent typographic conventions.

Visual Cues

The following typographic cues are used:

- **bold** words identify the name of task of package name from **Python** (and historical **IRAF**/**PyRAF** when necessary).
- `typewriter-like` words identify a file name, system command, or response that is typed or displayed.
- *italic* type indicates a new term, an important point, a mathematical variable, a task parameter, or a satellite name.
- SMALL CAPS identifies a header keyword.
- ALL CAPS identifies a table column.

Comments

Occasional side comments point out four types of information, each identified with a different border color and symbol on the left-hand side of the box.

Warning: You could corrupt data, produce incorrect results, or create some other kind of severe problem.

Heads up: Here is something that is often done incorrectly of that is not obvious.

Tip: No problems...just another way to do something or a suggestion that might make your life easier.

Information likely to be updated on a specific website is indicated by this symbol.

1. Obtaining HST Data

Chapter Contents

1.1 Archive Overview

All *HST* data files are stored in the Barbara A. Mikulski Archive for Space Telescopes (MAST; https://archive.stsci.edu/). *HST* Guaranteed Time Observers (GTOs), Guest Observers (GOs), and Archival Researchers can retrieve data in one of the following ways:

- Via gzipped tar file from the MAST Portal.
- Via ftp from the MAST staging disk.
- Via USB drives that are shipped to the user.
- Via the Python **astroquery** MAST API (https://astroquery.readthedocs.io/en/latest/mast/mast.html).
- Via Amazon Web Services (public data only; https://registry.opendata.aws/hst/).

Note that users may need to disable pop-up blockers when downloading data from the MAST Portal.

Users requiring alternative means of data retrieval or access to large volumes of data from MAST are encouraged to contact the MAST Help Desk (web: https://masthelp.stsci.edu/; e-mail: archive@stsci.edu) for assistance.

All science data retrieved from MAST, regardless of the method used, will be in FITS (Flexible Image Transport System) format. Further information on *HST* file formats is presented in Chapter 3.

HST data in MAST may be either available to the public or have exclusive access (formerly called "proprietary") privileges. Users wishing to retrieve exclusive access data must have the appropriate My Space Telescope (MyST; https://profile.stsci.edu/) account permissions granted by the Principal Investigator (PI) of the program. As of Cycle 25, exclusive access privileges are automatically granted to PIs.

Non-exclusive access data in MAST can be retrieved electronically either by registered MyST account users or anonymously. All calibration observations as well as observations made as part of GO parallel programs are immediately public. All observations made as part of a Treasury Program will either be immediately public or have only a brief proprietary period. The High-Level Science Products (HLSPs) in MAST also contain several sets of publicly available and fully reduced *HST* data such as the Ultra Deep Field (UDF) and the GEMS survey data (see Section 1.2.2 for more information about HLSPs).

When accessing data via the MAST Portal or the staging disk, the archive recommends retrieving compressed data to shorten the retrieval time without any significant information loss.

1.1.1 Archive Processing

MAST contains all observations ever made by *HST* and a catalog that describes the data. Each time a user makes a data request, MAST delivers data that have been processed with the best available reference files from the Calibration Reference Data System (CRDS; http://hst-crds.stsci.edu/). Calibrated data from all *HST* instruments are stored in a static form for retrieval. When changes are made to active instrument (ACS[1], COS, STIS, or WFC3) data pipelines and/or reference files, all affected observations are reprocessed. User requests for data from observations in the reprocessing queue will increase the reprocessing priority of the observations requested.

Requests for data in the reprocessing queue increase the reprocessing priority, but do not guarantee that the reprocessed data will be delivered to the user. If in doubt, users can check the DATE keyword in the FITS header to find when the file was last updated. Users can also reprocess data themselves using the latest versions of the relevant instrument calibration pipeline and reference files. For more information on how to reprocess data, please see the appropriate chapter of the data handbook for your instrument.

Table 1.1 Active *HST* Instruments

Instrument	Detectors
Advanced Camera for Surveys (ACS[1])	Wide Field Channel (WFC), Solar Blind Channel (SBC)
Cosmic Origins Spectrograph (COS)	FUV Channel, NUV Channel
Space Telescope Imaging Spectrograph (STIS)	CCD, NUV-MAMA, FUV-MAMA
Wide Field Camera 3 (WFC3)	UV/Visible Channel (UVIS), Infrared Channel (IR)

Data from *HST* legacy instruments are preserved in a static form, and there are no plans to regularly reprocess these data. In this context, legacy instruments include the ACS High Resolution Channel (ACS /HRC), Faint Object Camera (FOC), Faint Object Spectrograph (FOS), Goddard High Resolution Spectrograph (GHRS), High Speed Photometer (HSP), Near Infrared Camera and Multi-Object Spectrometer (NICMOS), Wide-Field Planetary Camera 1 (WF/PC-1), and Wide-Field Planetary Camera 2 (WFPC2). For ACS/HRC, FOC, FOS, GHRS, NICMOS, and WFPC2, no further improvements in the calibration for these instruments are expected. The user is provided with a copy of the raw and final calibrated data from the archive once a request is made. For HSP and WF/PC-1, no calibration has been done nor is any planned. Once raw data from these instruments are retrieved from MAST, they need to be calibrated locally by users.

1.1.2 Archive Registration (MyST)

MAST authentication is performed using a MyST account. From their MyST account, observing program PIs can view a list of accepted observing proposals and control access rights to exclusive access data for other users. Users accessing public data do not need to register for a MyST account. To make a new MyST account, or to update information in an already existing account, please visit https://profile.stsci.edu/. Users requiring a password reset can contact the HST Help Desk for assistance.

1.1.3 Archive Documentation and Help

The MAST Portal provides a wealth of useful information for data retrieval. Investigators expecting to work regularly with *HST* and other datasets supported by MAST should ensure they are subscribed to the MAST electronic newsletter for information regarding updates to the data archive. Users with a MyST account are registered for the MAST newsletter by default, and can verify their subscription status in the "Manage Subscriptions" menu in their MyST account. Questions about MAST should be directed to the MAST Help Desk either via the web portal or via e-mail to archive@stsci.edu.

[1]The ACS High Resolution Channel (ACS/HRC) is no longer operational after Servicing Mission 4. The Wide-Field Channel (WFC) and Solar Blind Channel (SBC) are available for observing.

1.2 Enhanced Data Products

1.2.1 Hubble Legacy Archive And Hubble Source Catalog

The Hubble Legacy Archive (HLA) is designed to facilitate access to *Hubble Space Telescope* data in a form that is suitable for use by a broad variety of science projects. The HLA is a joint project of the STScI, the European Coordinating Facility (ST-ECF; operations ceased on Jan. 1, 2011 and their archives are now served out of ESO/ST-ECF), and the Canadian Astronomy Data Centre (CADC). HLA data are available from the HLA website (http://hla.stsci.edu/) or via the MAST Portal by selecting the "HLA" mission.

The HLA is periodically updated to include new products, and users should consult the release notes (http://hla.stsci.edu/hla_release.html) for the most up-to-date information. The vast majority of public ACS, WFPC2, WFC3, and NICMOS imaging data, as well as some ACS and NICMOS grism data, are available through the HLA. Source lists for ACS, WFPC2, and WFC3 are also available. Statistics regarding the coverage of the HLA can be found on the HLA homepage.

The Hubble Source Catalog (HSC) is constructed by cross-matching overlapping HLA source lists from different visits to the same sky area. The resulting catalog has excellent astrometry (based on *Gaia*) along with accurate multi-color and multi-epoch photometry for more than 100 million objects. The HSC database can be used for large-scale science projects that would require much more effort if started from scratch using the standard *HST* pipeline products. For further information on the catalog properties and data access, see the HSC home page (https://archive.stsci.edu/hst/hsc).

The HLA and the MAST Portal provide a sky view allowing users to discover which datasets are available at a given position on the sky. Using the HLA interactive display (also accessible via the Portal), *Hubble* imaging data may be displayed with catalog overlays from SDSS, 2MASS, GSC2, FIRST, *GALEX*, HSC, *Gaia*, or PanSTARRS. Source lists from the HLA may also be plotted in the HLA interface.

1.2.2 High-Level Science Products

High Level Science Products (HLSPs) are observations, catalogs, or models that complement, or are derived from, MAST-supported missions. These include *Hubble* (*HST*), *James Webb* (*JWST*), *TESS*, PanSTARRS, *Kepler* /K2, *GALEX*, *Swift*, *XMM*, and others. HLSPs can include images, spectra, light curves, maps, source catalogs, or simulations. They can include observations from other telescopes, or data that have been processed in a way that differs from what is available in the originating archive.

Some HLSPs are available through the MAST Portal (http://mast.stsci.edu), and more will be added in the future. For access to the complete collection of HLSPs, and instructions for users to contribute their own HLSPs to the collection, please visit http://archive.stsci.edu/hlsp/.

1.3 Calibration Reference Data

Reference files used by the various *HST* data calibration pipelines are managed by CRDS. The used reference files are retrievable via the MAST portal while retrieving observational data. Alternatively, users may browse all active reference files in a given reference file context via the *HST* CRDS website at https://hst-crds.stsci.edu/. When a new reference file is delivered, or old reference files are updated for any reason, a new context is generated and the reason for delivery is logged. Historical reference file contexts may also be explored on the CRDS website.

Users who wish to determine if there have been updates to reference files since their data files were downloaded can use the **crds** Python package. Instructions for best reference file matching may be found at https://hst-crds.stsci.edu/static/users_guide/basic_use.html. These tools can update FITS files to use the new calibration data and also download all of the required reference files to a local cache.

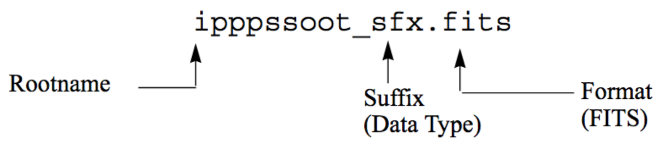
2. HST File Names

Chapter Contents

2.1 File Name Format

HST data file names encode a large amount of information about the files themselves. Datasets retrieved from MAST consist of multiple files in FITS format, each with a name that looks like this:

Figure 2.1: *HST* file name structure.



- **Rootname**: The first part of the file name (`ipppssoot`) is the *rootname* of the dataset to which the file belongs. All files belonging to a given dataset share the same rootname. For more information on the rootname composition, see Section 2.2.
- **Suffix**: The part of the filename between the "_" and the ".fits" is called the *suffix* (*sfx*), and it indicates the type of data the file contains. All science instruments except for COS have data file suffixes with three characters. COS data file suffixes are between three and seven characters long.
- **Format**: The identifier ".fits" indicates that this file is in FITS format.

For example, a STIS data file named `o8v502010_x1d.fits` is a FITS file that belongs to the dataset with rootname "`o8v502010`", and its "`_x1d`" suffix indicates that it contains calibrated science spectra. The suffixes for *HST* files vary between instruments, and the data handbooks for each *HST* instrument list the suffix types and their meanings.

⚠ The identifier referred to here as a "suffix" has often been called an "extension" in the past. However, the individual pieces of multi-extension FITS files are also known as "extensions" (see Section 3.2.1). For clarity, this handbook will use the term "extension" when referring to a component of file and the term "suffix" when referring to the three character identifier in a filename.

2.2 Rootnames

Rootnames of *HST* data files follow the naming convention defined in Table 2.1.

Table 2.1: IPPPSSOOT Root Filenames

Character	Meaning
`I`	Instrument used, will be one of: ***e*** - Engineering data ***f*** - Fine Guidance Sensors ***i*** - Wide Field Camera 3 ***j*** - Advanced Camera for Surveys ***l*** - Cosmic Origins Spectrograph ***n*** - Near Infrared Camera and Multi-Object Spectrograph ***o*** - Space Telescope Imaging Spectrograph ***s*** - Engineering subset data ***t*** - Guide star position data ***u*** - Wide Field/Planetary Camera-2 ***v*** - High Speed Photometer ***w*** - Wide Field/Planetary Camera ***x*** - Faint Object Camera ***y*** - Faint Object Spectrograph ***z*** - Goddard High Resolution Spectrograph
`PPP`	Program ID; can be any combination of letters or numbers (46,656 possible combinations). There is a unique association between program ID and proposal ID.
`SS`	Observation set ID; any combination of letters or numbers (1,296 possible combinations).
`OO`	Observation ID; any combination of letters or numbers (1,296 possible combinations).
`T`	Source of transmission or association product number ***m*** - Merged real time and tape recorded ***n*** - Retransmitted merged real time and tape recorded ***o*** - Retransmitted real time (letter "O') ***p*** - Retransmitted tape recorded ***q*** - Solid-state recorder ***r*** - Real time (not recorded) ***s*** - Retransmitted solid-state recorder ***t*** - Tape recorded ***0*** - Primary association product (number zero) ***1-8*** - ACS, COS, NICMOS, or WFC3 association sub-product

2.3 Suffixes of Files Common to All Instruments

The suffix of an *HST* file identifies the type of data that it contains. Several types of file suffixes are common to all instruments. These common suffixes are briefly described below. Please refer to the appropriate chapter concerning data structure in the instrument-specific data handbooks for the definitions of the instrument-specific suffixes.

- *Trailer files* (suffix "trl") are FITS files containing an ASCII table that logs the processing of a dataset through the data pipeline.
- *Support files* (suffix "spt") are FITS files that contain information about the observation and engineering data from the instrument and spacecraft recorded at the time of the observation.
- *Observation logs* (suffix ji* or cm*, aka "obslogs") are FITS files that contain information describing how the *HST* spacecraft behaved during a given observation. More information on these files is given in Chapter 5.

2.3.1 Historical File Structures

Old Version Observation Logs

The engineering telemetry, which is the basis of the observation logs, does not get reprocessed. Thus any significant change in its processing results in a different type of observation log. Such a change occurred in February 2003, when the existing Observation Monitoring System (OMS) was streamlined to reduce the software maintenance costs. Observation logs produced prior to this time have cm* suffixes and contain science specific information, in addition to the pointing information. Observation logs produced after this date have ji* suffixes and do not contain any instrument-specific information.

Observation log headers are divided into groups of keywords that deal with particular topics such as spacecraft data, background light, pointing control data, and line-of-sight jitter summary. The headers themselves provide short descriptions of each keyword. Observation log tables and images record spacecraft pointing information as a function of time. For more information on these files, consult Chapter 5.

PDQ Files

The suffix "pdq" denotes Post Observation Summary and Data Quality files which contain predicted as well as actual observation parameters extracted from the standard header and science headers. These files may also contain comments on any obvious features in the spectrum or image, as noted in the OPUS data assessment, or automatically extracted information about problems or oddities encountered during the observation or data processing. These comments may include correction to the keywords automatically placed in the obs logs files. Production of PDQ files was discontinued on May 9, 2002.

OCX Files

The suffix "ocx" denotes Observer Comment Files which were produced by STScI personnel to document the results of real-time commanding or monitoring of the observation, along with keywords and comments. Prior to April 17, 1992, OCX files were not always archived separately and, in some cases, were prepended to the trailer file.

After early February 1995, OCX files were produced only when an observation was used to locate the target for an Interactive Target Acquisition. At this time, mission and spacecraft information were moved to the PDQ reports and the Observation Logs (OMS jitter image and jitter table). Like PDQ files, production of OCX files was discontinued on May 9, 2002.

2.4 Associations

The ACS, NICMOS, STIS, and WFC3 calibration pipelines sometimes produce single calibrated images from *associations* of many exposures. For example, an ACS, NICMOS, or WFC3 observer might specify a dithering pattern in a Phase II proposal. Those instruments would then take several exposures at offset positions, and the pipeline would combine them into a single mosaic (suffix "mos" for NICMOS; suffix "drz" or "drc" for ACS and WFC3). In this case, the original set of exposures constitutes the association, and the mosaic is the *association product*.

Similarly, a STIS observer might specify a CR-SPLIT sequence in a Phase II proposal. STIS would gather several exposures at the same pointing, and the STIS pipeline would process this association of exposures into a single image, free of cosmic rays, that would be the association product (suffix "crj").

The COS calibration pipeline instead uses associations to process *all* COS science data. Please refer to Chapter 2 of the COS Data Handbook for more details.

When you search MAST for observations involving associations of exposures, your search will identify the final association product. The rootnames of association products always end in zero (see Table 2.1). If you request both raw and calibrated data from MAST, you will receive both the association product and the exposures that went into making it. The corresponding association table, stored in the file with suffix "asn" and the same rootname as the association product, lists the exposures or datasets belonging to the association. The list of files is stored as a binary table in the first extension (`EXTNAME = ASN`) of the association file (see Chapter 3 for more information about data formats). You can read this file using the **astropy.io.fits** Python package (see Section 4.4). The exposure IDs in the association table share the same "`ipppss`" sequence as the association rootname, followed by a base 36 number "`nn`" (`n` = 0-9, A-Z) that uniquely identifies each exposure, and a character `t` that denotes the data transmission mode for that exposure (see Table 2.1). For association products and sub-products, the last character will be a number between 0-8.

In practice, STIS stores the exposures belonging to associations differently than the other instruments. The exposures belonging to a STIS association all reside in the same file, while the exposures belonging to an ACS, COS, NICMOS, or WFC3 association reside in separate data files. See the relevant chapters concerning data structure in the instrument-specific data handbooks for more details.

3. HST File Formats

Chapter Contents

3.1 HST Data Files

The *HST* data calibration pipeline (HSTCAL; https://github.com/spacetelescope/hstcal) automatically processes and calibrates all the data received from *HST* and assembles them into a form suitable for most scientific analyses. Data from ACS, COS, NICMOS, STIS, WFC3, and WFPC2 are made available to observers as files in Multi-Extension FITS (MEF) format. For first generation instruments (FGS, FOC, FOS, GHRS, HSP, and WF/PC-1), data are stored in a format called "waivered" FITS. These waivered FITS files have only a single extension, therefore files containing data from multiple cameras must be stored as a three-dimensional data cube. For example, the four WFPC2 CCDs, each having a size of 800 x 800 pixels, would be stored in a waivered FITS file as a 800 x 800 x 4 pixel data cube, while in MEF format as four FITS extensions each containing a 800 x 800 pixel image.

⚠ WFPC2 data are stored in MAST as both MEF and waivered FITS files. Selecting the "minimum recommended products" will return the waivered FITS format files with suffixes c0f and c1f. Users may also retrieve the MEF format files by unchecking the "minimum recommended products" option and selecting explicitly the c0m and c1m suffixes.

Table 3.1: *HST* Instrument File Formats

Instrument	Status	Format from MAST
ACS	active	MEF
COS	active	MEF
FGS	active	waivered FITS
FOC	decommissioned SM3B	waivered FITS
FOS	decommissioned SM2	waivered FITS
GHRS	decommissioned SM2	waivered FITS
HSP	decommissioned SM1	waivered FITS
NICMOS	decommissioned SM4	MEF
STIS	active	MEF
WFC3	active	MEF
WFPC2	decommissioned SM4	waivered FITS & MEF
WF/PC-1	decommissioned SM1	waivered FITS

The remainder of this chapter describes FITS file formats in more detail.

✓ Throughout this document, references to FITS files will always mean *HST* Multi-Extension FITS format files. References to waivered FITS files will always be explicitly stated.

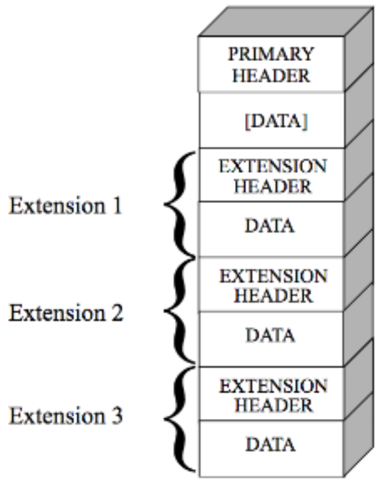
3.2 FITS File Format

3.2.1 Multi-Extension FITS Files

FITS[1] is a standard format for exchanging astronomical data, independent of the hardware platform and software environment. A file in FITS format consists of a series of Header Data Units (HDUs), each containing two components: an ASCII text header and the binary data. The header contains a series of keywords that describe the data in a particular HDU and the data component may immediately follow the header.

For *HST* FITS files, the first HDU, or primary header, contains no data. The primary header may be followed by one or more HDUs called extensions. Extensions may take the form of images, binary tables, or ASCII text tables. The data type for each extension is recorded in the `XTENSION` header keyword. Figure 3.1 schematically illustrates the structure of a FITS file and its extensions.

Figure 3.1: FITS format example.



Each FITS extension header contains the required keyword `XTENSION`, which specifies the extension type and has one of the following values: IMAGE, BINTABLE, and TABLE, corresponding to an image, binary table, and ASCII table, respectively.

Table 3.2: HST Header Keyword Descriptions

HDU Keyword	Description	Values
`XTENSION`	Data type for extension	• IMAGE • BINTABLE (binary table) • TABLE (ASCII table)
`EXTVER`	Imset number	Integer

`EXTNAME`	Extension names that describe the type of data component	SCI (science image) ERR (error image) DQ (data quality image) SAMP (number of sample) TIME[1] (exposure time) EVENTS[2] (photon event list) GTI[2] (good time interval) WHT (weight image) CTX (context image)
`PIXVALUE` [3]	Allowed for any extension except SCI, and used for an image with uniform value for all pixels	Real number

1. Only found in NICMOS and WFC3 data.
2. Only found in COS and STIS data.
3. When an image has the same value at all pixels (e.g., data quality value), the extension has no data component. Instead, the constant pixel value is stored in the header keyword `PIXVALUE`.

A set of FITS extension images which are logically related to one another is called an imset. For example, the error image and the data quality image are in the same imset as the science image itself. The keyword `EXTNAME` is used to specify the extension names of different images in the same imset. For example, the science data from the two ACS/WFC CCDs are stored in extensions 1 and 4 of a MEF file, but may also be referred to as ['sci', 1] and ['sci', 2], where 'SCI' is the extension name, and the integers 1 and 2 refer to the imset. Similarly, one may access the science data, error array, and data quality array of only the first imset located in extensions 1, 2, and 3, respectively, as ['sci', 1], ['err', 1], and ['dq', 1]. The data format chapters of each of the instrument-specific data handbooks provide specific information on extension names and imset numbers.

3.2.3 Waivered FITS Format

File formats for the first and second generation HST instruments (FGS, FOC, FOS, HSP, WF/PC-1, GHRS, and WFPC2) were developed before the standardization of MEF format. The waivered FITS format was developed in response to the need for a machine independent storage format for these data and was based on the idea of stacking multi-group data as a new dimension in a FITS image.

For example, a WFPC2 science data file containing data from four cameras (groups) has four 800 x 800 pixel images in its data file. The waivered FITS file containing these data has an image with dimensions 800 x 800 x 4 pixels in its primary HDU. Similarly, a FOS file may contain a two-dimensional spectrum stored as 2064 x 40 pixels in the primary HDU constructed from 40 groups of 2064 pixel one-dimensional spectra.

The first extension of a waivered FITS file will contain the header information as a binary table with each group represented as a row in the table. Each column in the table is a header keyword. For example, In the case of a WFPC2 image consisting of four groups, the first extension of the waivered FITS file is a table containing four rows.

[1]A description of FITS format and various supporting documents can be found at the website https://fits.gsfc.nasa.gov/fits_home.html.

4. HST Data Analysis

Chapter Contents

4.1 Analysis Options for HST Data

HST data can be manipulated with several different software packages. Most software provided and supported by STScI is written in Python. In this section, we introduce a few of the software language options for users to analyze their data. In subsequent sections, we will focus exclusively on Python tools.

As of version 9.0 of this document, we do not include any instructions for using IRAF/PyRAF to analyze *HST* data. **In October 2019, STScI stopped providing support for IRAF/PyRAF.** At the time of writing, IRAF is only available as part of PyRAF, which was written in Python 2.7. The end of support for these tools is consistent with the end of support for Python 2 by the Python Software Foundation on January 1, 2020.

Users still using IRAF/PyRAF to analyze *HST* data, unless having a strongly compelling reason to continue using those tools, are advised to instead use software written in Python 3. The Python 3 versions of analysis tools may contain bug fixes not present in the Python 2 versions, and continued use of Python 2 (i.e., PyRAF) for *HST* analysis may produce unexpected and undesired results.

Users needing assistance with converting from IRAF/PyRAF may find tutorials available at https://stak-notebooks.readthedocs.io/en/latest/ to be a helpful resource. These examples give specific IRAF task names and show how the same work may be accomplished in Python.

4.1.1 Python

Python is a freely available, general-purpose, dynamically-typed interactive language that provides modules for scientific programming, and is used for astronomical data reduction and analysis. The powerful and flexible nature of Python has led to its adoption in many fields, and thus there exists a large population of Python programmers continuously developing tools.

Python is the preferred programming language for analysis of *HST* data. The remaining text of this handbook will use only Python in discussion and examples.

In Python, the Astropy project (https://www.astropy.org/) contains many astronomy-specific tools provided with thorough documentation.

4.1.2 Interactive Data Language (IDL)

IDL is an array-based, interactive programming language that provides many numerical analysis and visualization tools. It is typically much easier to develop new analysis and visualization applications and utilities in IDL than in FORTRAN or C. IDL can be obtained from Harris Geospatial Solutions (https://www.harrisgeospatial.com/Software-Technology/IDL), and requires a paid license to use.

4.1.3 FORTRAN and C

For those who wish to write their own FORTRAN or C applications, we recommend using the FITSIO library for reading FITS files (https://heasarc.gsfc.nasa.gov/docs/software/fitsio/fitsio.html; note that the C library is called CFITSIO).

4.1.4 Software Support

Some software released by STScI is made available to the community, but unsupported (e.g., Tiny Tim). The *HST* Help Desk may be unable to help users with questions about unsupported software. Users unsure about if a particular tool is supported should contact the Help Desk for more information and possible recommendations for alternative tools.

4.2 Obtaining STScI Software

Software that supports *HST* is distributed through the conda package and environment management tool. Conda facilitates the installation and updating of software and dependencies in self-contained environments on a single computer. Conda *channels* are the paths that conda uses to obtain software packages.

stenv is a conda environment maintained by STScI to manage compatible versions for STScI-supported software. Please visit the stenv documentation for the latest information regarding installation and updates. In addition to *HST*-specific software, stenv also includes the Astropy project, which is a well-developed library of astronomy packages.

Users are advised to use Python 3 with STScI software for the most accurate results. Some packages may not be available or may produce unexpected results with earlier versions of Python.

Users may also wish to install some Python software using pip. For example, **astroquery** maintains a "bleeding edge" version of continuous releases through pip, while the conda version of **astroquery** is a tagged release version. Please refer to the installation instructions for specific Python modules for more information regarding any differences between conda and pip installations.

4.2.1 GitHub

All publicly available software from STScI is hosted on GitHub. GitHub is a "social coding" platform that facilitates version control and issue reporting for STScI software. Users who encounter problems with specific code packages are encouraged to review the open issues on the appropriate GitHub repository, and to open new issues to notify STScI about previously unknown bugs. Alternatively, users may also contact the *HST* Help desk for bug reporting and software assistance. Release notes for new versions of software and links to documentation may also be found in the GitHub repositories. The spacetelescope organization on GitHub contains all STScI-affiliated repositories and may be found at https://github.com/spacetelescope/.

Users unfamiliar with GitHub may find it intimidating at first. The *HST* Help Desk is prepared to offer assistance with software bugs and other related issues to users who prefer to not use GitHub.

4.3 Notable Python Packages for HST

Many of the software tools that are useful for analyzing *HST* data may be found in the Astropy library of Python modules. Additionally, some other Python modules are in Astropy-affiliated packages or are released directly by STScI. All of these packages are can be found in the stenv environment. For more information on stenv, see Section 4.2.

⚠ The following subsections list some Python tools that may be of interest to users. This is not an exhaustive list, but one that provides some of the most commonly used and recommended tools in topics of interest to *HST* users.

4.3.1 Data Retrieval

As discussed in Chapter 1, *HST* data may be obtained via the MAST portal. Data may also be obtained using the **astroquery** Python package for instances when one wishes to combine the data access with the analysis code. Furthermore, **astroquery** can be used to retrieve catalog information. The **astroquery** documentation (https://astroquery.readthedocs.io/en/latest/) contains information on supported archives and catalogs along with instructions for access and numerous examples.

4.3.2 Data Visualization

SAOImage DS9 (http://ds9.si.edu/site/Home.html) is a common tool for opening FITS data files and visualizing astronomical data.

As an alternative to DS9, some advanced Python users may be interested in using **ginga** (https://ginga.readthedocs.io/en/latest/). **Ginga** is a generic, customizable tool for visualizing **numpy** data arrays. **Ginga** functionality may be extended using plug-ins, and a set of plug-ins for functions common to *HST* data analysis are contained within **stginga** (https://stginga.readthedocs.io/en/stable/).

Users familiar with the IRAF task **imexam** will find useful the Python version of the tool with the same name. Documentation for Python **imexam** may be found at https://imexam.readthedocs.io/en/latest/.

⚠ The Python version of **imexam** contains all of the familiar functions from the IRAF task of the same name, however users may find that this implementation is more complex than its predecessor. Helpful examples are presented within the documentation linked above. Some basic functionality is demonstrated in Section 4.5, however users should always reference the code documentation for the most accurate and up-to-date syntax.

Glue is a tool for exploring your data. The previously discussed tools described how to visualize specifically imaging data, but **glue** is primarily made for exploring multi-dimensional datasets. More information about **glue** may be found on the website https://glueviz.org/.

4.3.3 AstroDrizzle

AstroDrizzle is used to combine dithered imaging data from ACS, NICMOS, STIS, WFC3, and WFPC2. Drizzling data corrects for geometric distortion effects and removes cosmic rays and other undesirable artifacts to create a clean mosaic. Note that STIS and NICMOS data are only sometimes combined via drizzling, and users should consult the data handbooks for each of the aforementioned instruments to verify appropriate data combination techniques. For more information on **AstroDrizzle**, which is part of the DrizzlePac package, please see http://drizzlepac.stsci.edu/.

Examples for drizzling data are provided as Jupyter notebooks in the STScI Notebooks GitHub repository (https://github.com/spacetelescope/notebooks). Jupyter notebooks are files that combine formatted text and blocks of Python code together making them useful tools for instruction. The linked GitHub repository contains both the Jupyter notebook files, which can be run on a computer with a properly configured environment, as well as pre-compiled versions rendered in HTML.

✓ All **AstroDrizzle** users are encouraged to review the Initialization notebook before proceeding to other example notebooks.

4.3.4 Synthetic Photometry

Synthetic photometry uses throughput information from the *HST* optical telescope assembly as well as the instrument optics, filters, and detectors to estimate count rates and fluxes associated with the observation of an astrophysical source. For proposing *HST* observations, users should use the Exposure Time Calculator (ETC; http://etc.stsci.edu/), which uses software called **PySynphot**. A new, generalized version of the synthetic photometry software called **synphot** (with an additional plug-in for STScI missions called **stsynphot**) is also available. **Synphot** is built upon Astropy tools and uses the **astropy.units** package along with Astropy model templates. For more information, please see the **synphot** documentation at https://synphot.readthedocs.io/en/latest/.

Tarballs containing all of the throughput tables, extinction curves, and spectral atlases required for synthetic photometry and simulated *HST* observations are available via https://archive.stsci.edu/hlsp/reference-atlases.

Alternatively, the throughput tables for *HST* are also contained within CRDS. See the "synphot" expandable section on the *HST* CRDS website (https://hst-crds.stsci.edu/) for access to these files. To download only the throughput tables via the command line, one can use the **crds.get_synphot** tool. Please see the help text for this tool for the most up-to-date information regarding its use.

4.3.5 Other Useful Resources

Here, we simply list other Python tools that may be helpful to astronomers. This list is by no means comprehensive.

- **astropy.stats** - Contains many basic and advanced statistical functions. See http://docs.astropy.org/en/stable/stats/.
- **astropy.units** - Allows for turning scalar values into a quantity object, which contains both the scalar value and an attached unit. Unit conversions are handled by Astropy. See https://docs.astropy.org/en/stable/units/.
- **matplotlib** - Flexible plotting tool for Python. There are other plotting packages, but this one is widely-used and well-documented. See https://matplotlib.org/.
- **numpy** - Numerical tools for Python. This package forms the basis of many scientific computing codes in Python. See https://numpy.org/.
- **pandas** - Another tool for displaying and working with tabular data that has advanced, SQL-like capabilities. See https://pandas.pydata.org/.
- **scipy** - SciPy is a container of many tools including, e.g., **numpy**, **pandas**, and **matplotlib**. See https://www.scipy.org/.

4.4 Working with FITS Data in Python

A comprehensive demonstration of all Python tools available for *HST* data analysis is outside the scope of this document. Users are encouraged to consult the documentation for the specific tools they are interested in using for examples and syntax, as well as any additional information such as limitations and known bugs.

⚠ Python arrays are zero-indexed, and the convention is to specify indices of two-dimensional data as [row, column] or [y, x]. This indexing order is different than other languages and tools familiar to astronomers such as IDL and IRAF. When specifying ranges of indices, the first element is inclusive, while the second is exclusive. For example, the indices specified by [:15] include all elements 0 through 14 with the zeroth index being implicit by omission. One can also use negative indices to count backwards from the end of an array. For example, we can select the first 2,000 rows and all columns of a 2048 row x 4096 column array using either [0:2001, :] or [:-48, :]. Notice that by not specifying any indices, we return all indices.

4.4.1 Opening and Updating FITS Files

As discussed in Section 3.2, FITS files are comprised of multiple HDUs, which are a combination of an ASCII header and a data array. In Python, FITS files can be read into a HDUList object using the **astropy.io.fits.open()** function (see http://docs.astropy.org/en/stable/io/fits/ for more information). These HDUList objects can be indexed like arrays to access the desired HDU, or alternatively indexed using the `EXTNAME` and `EXTVER` discussed in Section 3.2.

The header and data array of an HDU can be accessed using the ".data" and ".header" attributes. For example:

```
from astropy.io import fits

with fits.open('jdl803dxq_flt.fits') as hdu:

        science_data1 = hdu[1].data
        science_header1 = hdu[1].header

        error_data2 = hdu['err',2].data
        error_header2 = hdu['err',2].header
```

One can easily update the header and data array of an HDU as follows:

```
from astropy.io import fits
import numpy as np

with fits.open('jdl803dxq_flt.fits') as hdu:

        hdu[0].header['FILTER1'] = 'NEW_FILTER_NAME'
        hdu[1].data = np.zeros(2048, 4096)
```

In this example, we have updated the keyword `FILTER1` with the new value "NEW_FILTER_NAME" and we have replaced the first science extension with an 2048 row x 4096 column array of all zeros. Image data stored within FITS files is returned as a **numpy** ndarray object. Note that our changes will not be written out to the FITS file unless we call the **astropy.io.fits.open()** function with the optional argument `mode='update'`. Also notice that we have opened the FITS file inside of a "with block." Unindenting after this block will automatically close the file. We can repeat the above example, saving our changes to the FITS file, and explicitly close the file without using the "with block" syntax:

```
from astropy.io import fits
import numpy as np

hdu = fits.open('jdl803dxq_flt.fits', mode='update')

hdu[0].header['FILTER1'] = 'NEW_FILTER_NAME'
hdu[1].data = np.zeros(2048, 4096)

hdu.close()
```

Use of the "with block" is often considered more readable and safer for file input/output, and one may encounter it in many examples.

4.4.2 FITS Table Data

Binary FITS tables are stored just like image arrays inside the HDUList object, and we can also access these with the ".data" attribute. By simply returning the ".data" attribute of the HDUList object, we obtain a FITS_rec class. We can more easily access the binary table data using the **astropy.table.Table()** class (see http://docs.astropy.org/en/stable/table/ for more information). We can use this to access, for example, the table information contained within an *HST* association file.

```
from astropy.io import fits
from astropy.table import Table

with fits.open('icdm0a070_asn.fits') as hdu:
        asn_table = Table(hdu[1].data)

print(asn_table)
```

This will yield the resulting table:

```
   MEMNAME        MEMTYPE     MEMPRSNT
-------------- -------------- --------
ICDM0AEIQ      EXP-DTH            True
ICDM0AEKQ      EXP-DTH            True
ICDM0AEMQ      EXP-DTH            True
ICDM0A070      PROD-DTH           True
```

Astropy tables are indexed just like other array-like objects, but we can also use the column names to our advantage. For example, we can select the rootnames (the MEMNAME column) of the first three entries in the table (the individual exposures in the observation; the last row represents the combined product of the association) using `asn_table['MEMNAME'][:-1].data`. By including the ".data" attribute here, we convert the indexed result into a **numpy** array.

4.5 Displaying Imaging Data

This section will be of interest primarily to observers whose datasets contain two-dimensional images from the imaging detectors onboard *HST*. We will discuss two methods for displaying imaging data:

- Display of an image using **ds9**
- Creation of a figure using **matplotlib**
- Analysis of an image using **imexam**

4.5.1 Viewing Image Data in DS9

DS9 is a suitable option for displaying data if one simply wishes to view the imaging data contained within a FITS file, or do basic analysis work such as estimating the centroid position of a point source. The ds9 website (http://ds9.si.edu/site/Home.html) contains information on more advanced features including command line arguments. From a bash prompt, one can open ds9 by typing:

```
ds9 &
```

This will open ds9 in the background. If the file name is specified, e.g.,

```
ds9 icdm0a070_drc.fits &
```

then the image will be displayed. In the case of a MEF file, the first FITS extension (not the zeroth primary extension) will be displayed. To access other extensions, specify them at the command line:

```
ds9 icdm0a070_drc.fits[2] &
```

This will display the second extension in the MEF file, which in this case contains weight image of a drizzled observation set. Much of the functionality for adjusting scaling, limits, matching of multiple images using the World Coordinate System (WCS) or aligning in pixel-space can be done using menu options in the ds9 graphical user interface. Some of these features may also be accessed when opening data via the command line. See the ds9 documentation for more information.

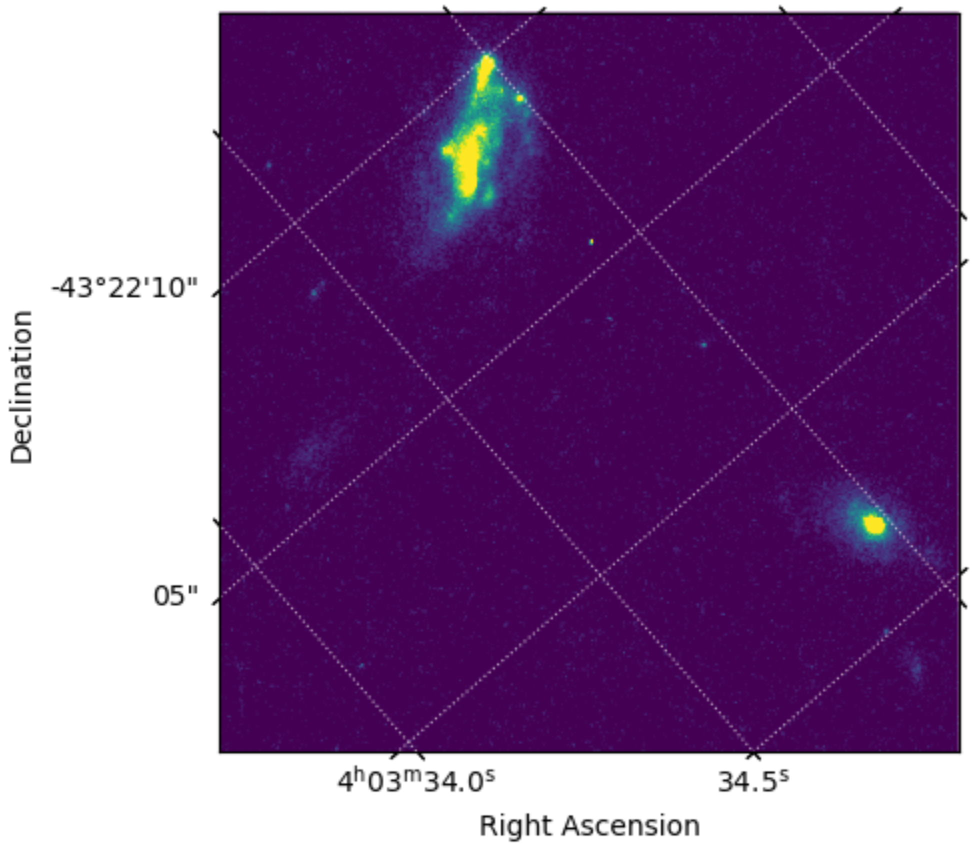
4.5.2 Plotting Imaging Data with Python

As discussed in Section 4.4, the data component of a FITS file HDU is read into Python as a **numpy** ndarray object, and thus it may be treated like any other data array for plotting purposes. We can use **matplotlib** to visualize the data and create figures.

In this example, we will create a plot of a subsection of a WFC3/UVIS drizzled image. We manually set the scaling to display the image, and include right ascension and declination axes. We then save the plot as a PNG image "galaxies_wcs.png".

```
from astropy.io import fits
from astropy.wcs import WCS
import matplotlib.pyplot as plt

# Open the FITS file and retrieve the data
# and WCS header keyword information from the
# first science extension. Save the WCS
# information into an Astropy WCS object.
with fits.open('icdm0a070_drc.fits') as hdu:
        data = hdu[1].data
        wcs = WCS(hdu[1].header)

# Select a subsection of the image to display.
# Here we have selected a 400 x 400 pixel section
# with x = [280:680] and y = [2290:2690].
cutout = data[2290:2690, 280:680]

# Create the plotting object with the WCS projection.
plt.subplot(projection=wcs)
plt.imshow(cutout, vmin=0, vmax=0.2)
plt.grid(color='white', ls=':', alpha=0.7)
plt.xlabel('Right Ascension')
plt.ylabel('Declination')

# Save the figure.
plt.savefig('galaxies_wcs.png')
```

Figure 4.1: Plotting image cutout with matplotlib



Notice in Figure 4.1 that the tick marks (and grid lines) are rotated. This is because the drizzled image we used for the example did not have North and East oriented up and to the left, respectively.

4.5.3 Python imexam

Users familiar with legacy code may recall a task called **imexam**. This function allowed users to interactively examine imaging data displayed in an image viewer such as ds9 and do quick analysis such as point spread function (PSF) fitting, simple photometry, and statistics. A Python version of **imexam** is now available as an Astropy-affiliated project, and documentation may be found at https://imexam.readthedocs.io/en/latest/.

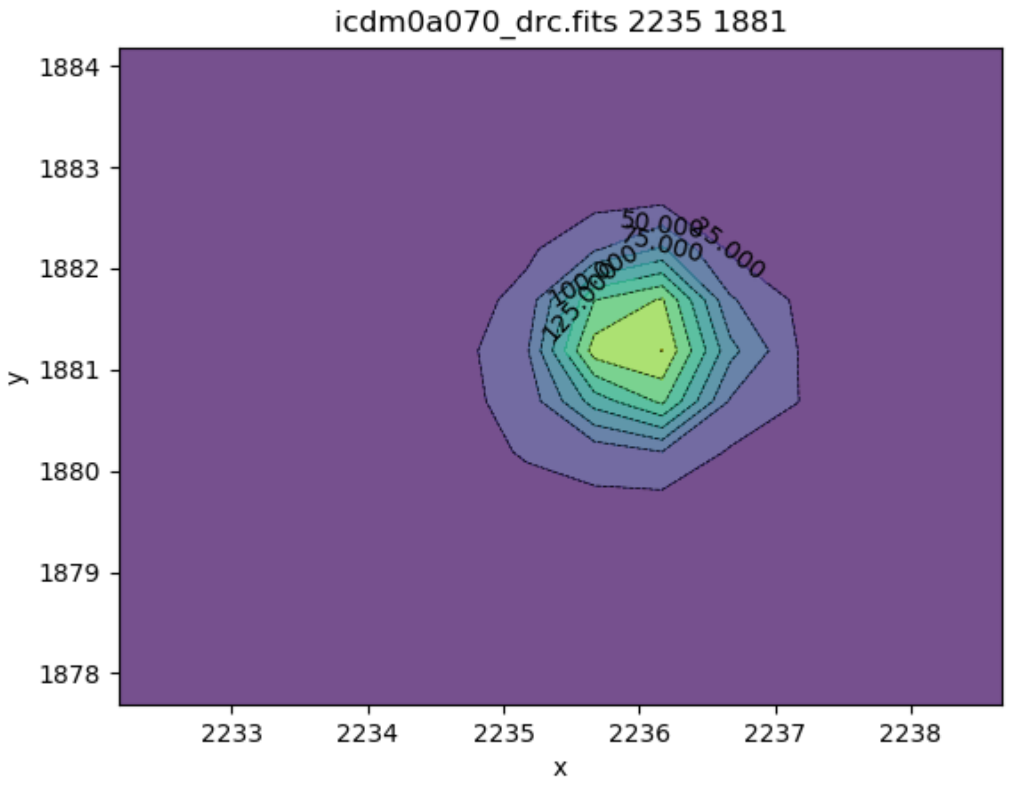
⚠ The Python version of **imexam** can be daunting at first, and therefore users new to Python should carefully consult the examples given within the documentation.

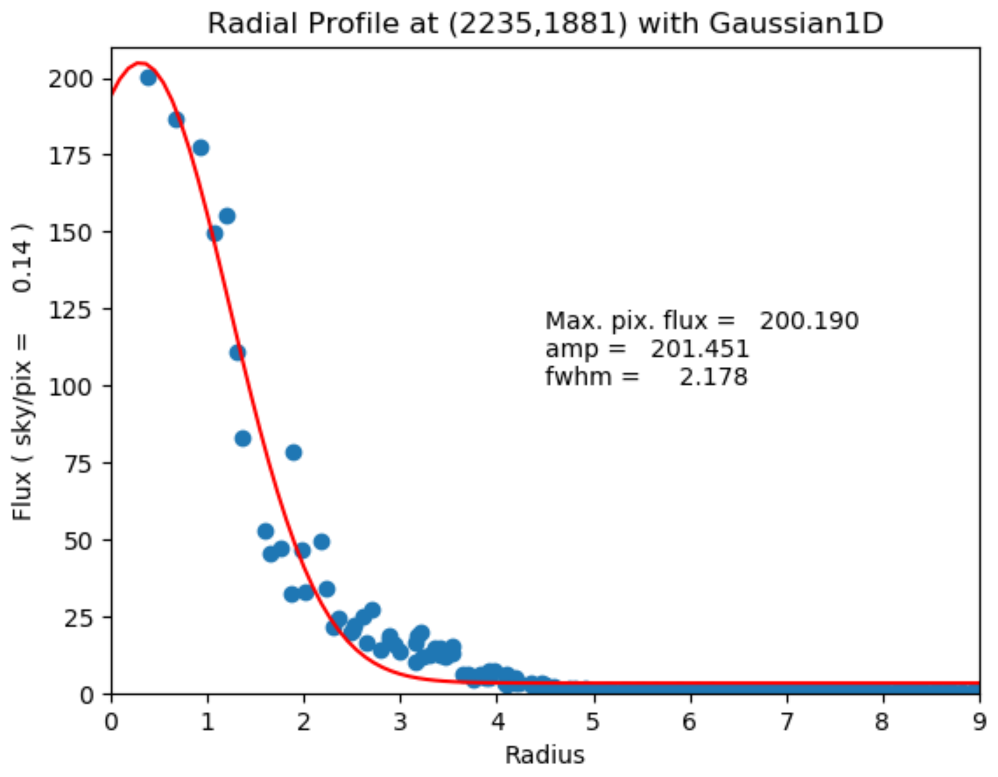
Using our previous WFC3/UVIS drizzled FITS file as an example, we can open ds9 with **imexam**, display the image, and do some simple analysis tasks using the following commands:

```
import imexam

# Launch ds9 and display the image.
# Also automatically scale the image.
viewer = imexam.connect()
viewer.load_fits('icdm0a070_drc.fits')
viewer.scale()

# Interactively examine the image.
viewer.imexam()
```

Typing this last command will change the cursor in the ds9 window to a circular, blinking cursor that should be positioned over pixels you wish to analyze. A list of available functions will be displayed in the bash shell, and users who had experience with the previous non-Python **imexam** will find these functions and their keyboard shortcuts familiar. In Figure 4.2 below, we show plots generated by **imexam** of the two-dimensional histogram and PSF fit of a star in the drizzled image, which were produced with the keyboard shortcuts "e" and "r", respectively.

Figure 4.2: imexam 2-D Histogram and PSF Fit





4.6 Analyzing HST Images

4.6.1 Basic Astrometry

The header of every calibrated *HST* two-dimensional image contains information to translate pixel positions in the image onto the World Coordinate System (WCS). Python can use this information to convert between pixel coordinates and right ascension and declination using the Astropy **wcs** package. For example, using WFC3/UVIS, we can convert from pixel position to WCS coordinates, and back again:

```
from astropy.io import fits
from astropy.wcs import WCS

with fits.open('icdm0a070_drc.fits') as hdu:
        wcs = WCS(fobj=hdu[1], header=hdu[1].header)

# Pixel position in (y, x)
pixel = (1000, 1200)

# Convert
ra, dec = wcs.all_pix2world(pixel[0], pixel[1], 1, ra_dec_order=True)
y, x = wcs.all_world2pix(ra, dec, 1, adaptive=False, ra_dec_order=True)

# Print results
print('Pixel ({}, {}) converts to RA: {}, Dec:{}'.format(y, x, ra, dec))
```

This yields the result:

```
Pixel (1000.0000000014072, 1200.0000000008781) converts to RA: 60.91377790287812, Dec:-43.370117840192
```

As you can see, our input pixel position of (y, x) = (1000, 1200) converted to (RA, Dec) in decimal degrees and back to (y, x). The optional argument `adaptive=False` in **wcs.all_world2pix** is advised for all *HST* data. By using the **wcs.all_*** functions, all WCS corrections contained within the headers will be used including SIP distortion keywords and lookup tables. For instruments that have geometric distortion corrections applied by the pipeline, these keywords will be present and the corrections will be applied during pixel-WCS coordinate transformations.

Differential astrometry is easy and relatively accurate for *HST* images. Absolute astrometry, on the other hand, is more difficult owing to uncertainties in the locations of the instrument apertures relative to the Optical Telescope Assembly (OTA) and inherent uncertainty in the guide star positions. Generally, observations obtained during the same visit using the same guide star acquisitions will typically have small shifts (< ~10 milli-arcseconds; see, e.g., TEL ISR 2005-02 and ACS ISR 2006-05).

✓ The DrizzlePac notebooks contain an example of aligning *HST* images to an absolute reference catalog, e.g., Gaia DR2. For more information and an example, please see the "Align to Catalogs" Jupyter noteboook (a compiled HTML version is also available).

4.6.2 Photometry

In this subsection, we discuss aspects of *HST* data that are important for photometry, but we do not provide a comprehensive example of photometric analysis. The Python package **photutils** provides tools for aperture photometry and background estimation. Other options for photometric measurements include, e.g., **dolphot** and **Source Extractor.**

⚠ No photometry software is officially supported by STScI. Users should find an option that works best for their scientific needs and consult the documentation for those tools as necessary.

Calibrated *HST* images obtained from MAST store signal in various units. Table 4.1 lists a selection of *HST* instrument image units. Refer to the appropriate instrument-specific data handbooks for more information.

Table 4.1: Selection of *HST* Instrument Image Units

Instrument	**Calibrated Data Units**	**Drizzled Data Units**
ACS/WFC	electrons	electrons / second
COS	DN / second	N/A
NICMOS	DN / second	DN / second
STIS	DN	N/A
WFC3/UVIS	electrons	electrons / second
WFC3/IR[1]	electrons / second	electrons / second
WFPC2	DN	DN / second

1. WFC3/IR SCAN mode data have calibrated units of electrons.

The pipeline calibration tasks do not alter the units in the images when performing the photometric correction step. Instead they calculate and write the sensitivity conversion factor (`PHOTFLAM`) and the ST magnitude zeropoint (`PHOTZPT`) into the header keywords in the calibrated data.

⚠ In some cases, instruments with multiple CCDs have different photometric calibrations for each individual CCD, e.g., in the case of WFCP2. Users should consult the appropriate instrument-specific data handbook for more information concerning if re-normalization of the sensitivity of the detectors is required.

`PHOTFLAM` is defined as the mean flux density F_λ in units of erg cm^{-2} s^{-1} Angstrom^{-1} that produces 1 count per second in the *HST* observing mode (`PHOTMODE`) used for the observation. Note that the word "counts" may refer to electrons or to data numbers (DN; also called analog digital unit or ADU) depending on the instrument used. Flux values measured from calibrated images (in units of counts s^{-1}) may be converted to physical fluxes in units of erg cm^{-2} s^{-1} Angstrom^{-1} by multiplying the count rate by the `PHOTFLAM` value. If a `PHOTNU` header keyword is present, then it may similarly be used to convert the count rate to a flux in units of Janskies.

If the F_λ of your spectrum is significantly sloped across the bandpass or contains prominent features, such as strong emission lines, you may wish to recalculate the inverse sensitivity `PHOTFLAM` using **pysynphot**. WF/PC-1 and WFPC2 users should note that the PHOTFLAM values calculated during pipeline processing did not include a correction for temporal variations in throughput owing to contamination buildup or charge transfer efficiency effects. Keywords were added to WFCP2 headers to allow users to correct for contamination (`ZP_CORR`) and charge transfer efficiency (`CTE1E2`, `CTE1E3`, and `CTE1E4`). Likewise, FOC observers should note that `PHOTFLAM` values determined by the pipeline before May 18, 1994 do not account for sensitivity differences in formats other than 512 x 512 pixels.

Similar to the AB magnitude system, there is an ST magnitude system that is based on a reference spectrum that is flat in *wavelength* (as opposed to flat in frequency for the AB magnitude system). The conversion from flux F_λ to magnitude is achieved in the familiar way:

$$m = -2.5\log_{10}(F_\lambda) + \mathrm{ZP},$$

where ZP for the ST magnitude system is -21.10. This value was chosen such that the star Vega has an ST magnitude of zero for the Johnson *V* filter. Further corrections are necessary for converting from the ST magnitude system to other systems like Johnson/Cousins, and depend on the spectral shape of the observed source. See specific photometry examples in the instrument-specific data handbooks.

4.6.3 Synthetic Photometry and Transmission Curves

The python packages **PySynphot** (https://pysynphot.readthedocs.io/en/latest/index.html) and **Synphot** (with the extension **STSynphot**; https://stsynphot.readthedocs.io/en/latest/index.html) can simulate *HST* observations of astronomical targets with known spectra. They make use of tables that contain throughput information for all of the *HST* optical components, such as mirror, filters, gratings, apertures, and detectors, and can generate passband shapes for any combination of these elements. They can also generate synthetic spectra of many different types of sources, including stellar, blackbody, power-law, and H II regions, and can convolve these spectra with the throughputs of the *HST* instruments. You can therefore use it to compare results in many different bands, to cross-calibrate one instrument with another, or to relate your observations to theoretical models.

There is functionally no difference in the results yielded by using **PySynphot** compared with **Synphot** + **STSynphot**. The latter two packages contain some quality of life improvements over **PySynphot** such as use of Astropy models and Astropy quantity objects to attach units to input and output values. At STScI, **PySynphot** is used in the exposure time calculators.

For *HST* instruments, the complete path through the optical elements from the pupil to the detector of interest is specified using the "obsmode." The obsmode contains information about the instrument, detector, filter elements, and in some cases additional specifications about the date of the observation (for time-dependent sensitivity) and central wavelength of the observation.

Appendix B in both the **PySynphot** (https://pysynphot.readthedocs.io/en/latest/appendixb.html) and **Synphot** (https://stsynphot.readthedocs.io/en/latest/stsynphot/appendixb.html#stsynphot-appendixb) documentation contains information on the obsmode specification. For example, an observation using the ACS/WFC detector (specifically we will choose the WFC1 CCD for this example) using the ramp filter FR782N tuned to wavelength 7730 Angstroms and obtained on date January 1, 2019 (MJD 58484) is specified as "acs,wfc1,fr782n#7730,mjd#58484".

4.7 Displaying and Analyzing HST Spectra

⚠ **This section is a work in progress.** Python code is being actively developed to support spectroscopy data analysis. This section will be updated as more information becomes available.

Users may wish to consult the previous version of the Introduction to the *HST* Data Handbooks sections 3.5 ("Displaying *HST* Spectra") and 3.6 ("Analyzing *HST* Spectra") for more information.

5. Observation Logs

Chapter Contents

5.1 Observation Log Files

Observation Log Files, also known as *jitter* files, record pointing, jitter, and other Pointing Control System (PCS) data taken during an *HST* observation. You can use them to assess the behavior of the *HST* spacecraft during your observation, and in particular, to evaluate the jitter of the spacecraft while it was taking data. Here we describe the contents and structure of the observation log files, how to retrieve them from MAST, and how to work with the data they contain.

These data files are produced by the Engineering Data Processing System (EDPS), an automated software system that interrogates the *HST* engineering telemetry and correlates the time-tagged engineering stream with *HST*'s Science Mission Schedule (SMS), the seven-day command and event list that drives all spacecraft activities. The EDPS replaced the Observatory Monitoring System (OMS) in February 2003. EDPS provides observers with information about guide star acquisition, pointing, and tracking that is not normally provided in the science headers.

The observation log files share the same rootname as the observation they are associated with, except for the final character, which for observation log files is always a "j" (see Chapter 2 for more information on the names of HST data files). The "jit" table accompanies the "jif" header. The "jit" table is the three-second average pointing data. The "jif" header is a two-dimensional histogram of jitter excursions during the observation, which includes the header plus some image related keywords.

⚠ A detailed description of observation log files can be found at the archived observation logs website.

⚠ Pointing and tracking information prior to October 1994 is not routinely available. Interested observers with data from this epoch should contact the *HST* Help Desk for assistance.

5.1.1 Jitter File Contents

The EDPS jitter files are limited to the engineering data that describe the performance of the Pointing Control System (PCS) including the Fine Guidance Sensors that are used to control the vehicle pointing. The jitter files report on PCS engineering data for the duration of the observation. The EDPS jitter files contain:

- *ipppssoo*`j_jif.fits`: The FITS header contains keywords providing information regarding the file structure, observation details, modeled background light, point control system, jitter summary, orbital geometry, and problem flags and warnings. The extension 0 header will contain parameters relating to the entire association or dataset, ending at the "ASSOCATION KEYWORDS" block. Each of the header values for extensions 1 and beyond represent a single exposure in the association, and will contain all the group 0 keywords as well as additional blocks of parameters relating to that particular exposure.
- *ipppssoo*`j_jit.fits`: This is the 3-second average jitter table. It contains the reconstructed pointing, guide star coordinates, derived jitter at the science instrument aperture, pertinent guiding-related flags, orbital data (e.g., latitude, longitude, limb angle, magnitude field values, etc.) and fine guidance sensor flags.

5.1.2 Retrieving Observation Log Files

Observation logs can be retrieved via the MAST Portal by deselecting the "Minimum Recommended Products" option in the download basket and then selecting the "jit" and "jif" options in the "Group" panel.

5.2 Using Observation Logs

Here are some examples of what can be learned from the observation log files. Some examples assume basic familiarity with FITS file structures and the Python programming language (see Chapters 3 and 4, respectively).

5.2.1 Guide Star Errors

Seldomly, the *HST* spacecraft pointing may be interrupted during an observation. Subsections 5.2.2 through 5.2.5 describe in detail the information contained within observation log files that can help diagnose the severity of pointing problems and their impact to observations. In general, problems with guide star lock and tracking can be checked using keywords in the "Problem Flags and Warnings" section of the "jif" file primary header. For example, an observation with no loss of guide star lock will have the following in the "jif" file primary header:

```
               / PROBLEM FLAGS and WARNINGS

T_GDACT =                    T / Actual guiding mode same for all exposures
T_ACTGSP=                    T / Actual Guide Star Separation same in all exps.
T_GSFAIL=                    F / Guide star acquisition failure in any exposure
T_SGSTAR=                    F / Failed to single star fine lock
T_TLMPRB=                    F / problem with the engineering telemetry in any e
T_NOTLM =                    F / no engineering telemetry available in all exps.
T_NTMGAP=                    0 / total number of telemetry gaps in association
T_TMGAP =             0.000000 / total duration of missing telemetry in asn. (s)
T_GSGAP =                    F / missing telemetry during GS acq. in any exp.
T_SLEWNG=                    F / Slewing occurred during observations
T_TDFDWN=                    F / Take Data Flag NOT on throughout observations
```

These keywords (except `T_NTMGAP` and `T_TMGAP`) are Boolean values describing basic tests of pointing quality, and are aggregated from the extensions following the primary extension that describe the individual exposures in the observation/association. Observation logs with deviations from the above may be associated with suspect data, and further examination of the observation logs is warranted.

5.2.2 Guiding Mode

Unless requested, all observations will be scheduled with "FINE LOCK" guiding, which may be one or two guide stars (dominant and roll). The spacecraft may roll slightly during an observation if only one guide star is acquired. The amount of roll depends upon the gyro drift at the time of the observation, the location during an orbit, and the lever arm from the guide star to the center of the aperture.

There are three commanded guiding modes: "FINE LOCK," "FINE LOCK/GYRO," and "GYRO." Observation file header keywords `GUIDECMD` (commanded guiding mode) and `GUIDEACT` (actual guiding mode) will usually agree. If there was a problem during the observation, they will not agree and the `GUIDEACT` value will be the guiding method actually used during the exposure. If the acquisition of the second guide star fails, the spacecraft guidance `GUIDEACT` may drop from "FINE LOCK" to "FINE LOCK/GYRO," or even to "GYRO," which may result in a target rolling out of an aperture. Check the observation log keywords to verify that there was no change in the requested guiding mode during the observation.

⚠ The `GUIDECMD` header keyword is located in the primary header of the "jif" files, while the `GUIDEACT` keyword appears in the subsequent extension headers that describe each exposure in the observation/association.

⚠ Until flight software version FSW 9.6 came online in September 1995, if the guide star acquisition failed, then the guiding dropped to "COARSE" track. After September 1995, if the guide star acquisition failed, then the tracking did not drop to "COARSE" track. Archival researchers may find older datasets that were obtained with "COARSE" track guiding.

The dominant and roll guide star keywords (`GSD*` and `GSR*`) in the observation log header can be checked to verify that two guide stars were used for guiding, or in the case of an acquisition failure to identify the suspect guide star. The dominant and roll guide star keywords identify the stars that were scheduled to be used, and in the event of an acquisition failure may not be the stars that were actually used. The following list of observation log keywords is an example of two-star guiding. These keywords are found in the primary header of the "jif" file in the "Pointing Keywords" section.

```
               / POINTING CONTROL DATA

GUIDECMD= 'FINE LOCK           ' / Commanded Guiding mode
GSD_ID  = 'N134000416'          / Dominant Guide Star ID
GSD_RA  =           270.236940 / Dominant Guide Star RA (deg)
GSD_DEC =            66.694810 / Dominant Guide Star DEC (deg)
GSD_MAG =            12.649000 / Dominant Guide Star Magnitude
GSD_FGS = '3'                   / Dominant GS Fine Guidance Sensor Number
GSR_ID  = 'N4E8000253'          / Roll Guide Star ID
GSR_RA  =           269.806080 / Roll Guide Star RA (deg)
GSR_DEC =            67.026590 / Roll Guide Star DEC (deg)
GSR_MAG =            13.326000 / Roll Guide Star Magnitude
GSR_FGS = '2'                   / Roll GS Fine Guidance Sensor Number
GS_PRIM = 'ROLL    '            / Guide star that is primary (dominant or roll)
PREDGSEP=             1340.940 / Predicted Guide Star Separation (arcsec)
MGSSPRMS=                   1. / maximum Guide Star Separation RMS in assoc.
TNLOSSES=                    0 / Number of loss of lock events
TLOCKLOS=                   0. / Total loss of lock time
TNRECENT=                    0 / Number of recentering events
TRECENTR=                   0. / Total recentering time
```

The guide star identifications, GSD_ID and GSR_ID, are different for the two Guide Star Catalogs: GSC2 IDs are 10-characters in length, like those of GSC1, but consist of both letters and numbers. GSC1 IDs consist entirely of numbers.

⚠ The GSC2 catalog is the default catalog since *HST* Cycle 15 (June 2006). The keyword `REFFRAME` in the primary science header indicates which catalog was in use for an observation. This keyword is included in all Cycle 15 and later observations, and is either "GSC1" for Guide Star Catalog 1 or "ICRS" for International Celestial Reference System, upon which GSC2 coordinates are based. The same information is added to the *HST* Archive catalog file `shp_refframe` of the `shp_data` database table since June 2006.

Some GHRS observations can span several orbits. If during a multi-orbit observation the guide star re-acquisition fails, the observation may be terminated with possible loss of observing time, or switch to other less desirable guiding modes. The `GSACQ` keyword in the "jif" header will state the time of the last successful guide star acquisition.

```
GSACQ   = '1997.037:03:45:03 ' /Actual time of GS Acquisition Completion
```

5.2.3 Guide Star Acquisition Failure

The guide star acquisition at the start of the observation set could fail if the FGS fails to lock onto the guide star. The target may not be in the aperture, or maybe only a piece of an extended target is in the aperture. The jitter values will be increased because "FINE LOCK" was not used. The following list of observation log header keywords in the extensions of the "jif" file corresponding to individual exposures indicate that the guide star acquisition failed.

```
V3_RMS = 19.3                                   / V3 Axis RMS (milli-arcsec)
V3_P2P = 135.7                                  / V3 Axis peak to peak (milli-arcsec)

              / PROBLEM FLAGS, WARNINGS, and STATUS MESSAGES
              /(present only if problem exists)

GSFAIL =  'DEGRADED'                  / Guide star acquisition failure!
```

The observation logs for all of the exposures in the observation set will have the "DEGRADED" guide star message. This is not a loss-of-lock situation, but an actual failure to acquire the guide star in the desired guiding mode. For the example above, the guiding mode was dropped from FINE LOCK to COARSE TRACK.

```
GUIDECMD= 'FINE LOCK'                  / Commanded Guiding mode
GUIDEACT= 'COARSE TRACK'          / Actual Guiding mode at end of GS acquisition
```

If an observation dataset spans multiple orbits, then the guide star will be re-acquired, but the guiding mode will not change from COARSE TRACK. In September 1995, the flight software was changed so that COARSE TRACK is no longer an option. The guiding mode drops from two guide star FINE LOCK to one guide star FINE LOCK, or to GYRO control.

5.2.4 Moving Targets and Spatial Scans

A type 51 slew is used to track moving targets (planets, satellites, asteroids, and comets). Observations are scheduled with FINE LOCK acquisition, i.e., with two or one guide stars. Usually, a guide star pair will stay within the FGS field of view during the entire observation set, but if two guide stars are not available, a single guide star may be used, assuming the drift is small or the proposer says that the roll angle is not important for that particular observing program. An option during scheduling is to drop from FGS control to GYRO control when the guide stars move out of the FGS. Also, guide star handoffs (which are not a simple dropping of the guide stars to GYRO control) will affect the guiding and may be noticeable when the "jitter ball" or "pointing ball" is plotted (see Figure 5.2).

⚠ When using less than three gyros ("reduced gyro mode"), all observations are scheduled with two guide stars. Proposers cannot request the use of only one guide star.

Guide star handoffs are not allowed in reduced gyro mode.

The jitter statistics are accumulated at the start of the observation window. Moving targets and spatial scan motion will be seen in the jitter data and image. Therefore, the observation log keywords `V2_RMS` and `V3_RMS` values (the root mean square of the jitter about the V2- and V3-axes[1]) can be quite large for moving targets. Also, a special anomaly keyword (`SLEWING`) will be appended to the observation log header indicating movement of the telescope occurred during the observation. This is expected when observing moving targets. The following list of observation log keywords is an example of expected values while tracking a moving target.

```
              / LINE OF SIGHT JITTER SUMMARY

V2_RMS   =  3.2       / V2 Axis RMS (milli-arcsec)
V2_P2P   =  17.3      / V2 Axis peak to peak (milli-arcsec)
V3_RMS   =  14.3      / V3 Axis RMS (milli-arcsec)
V3_P2P   =  53.6      / V3 Axis peak to peak (milli-arcsec)
RA_AVG   =  244.01757 / Average RA (deg)
DEC_AVG =  -20.63654 / Average DEC (deg)
ROLL_AVG=  280.52591 / Average Roll (deg)
SLEWING = 'T'         / Slewing occurred during this observation
```

5.2.5 High Jitter

The spacecraft may shake during an observation, even though the guiding mode is FINE LOCK. This movement may be due to a micro-meteoroid impact, jitter at a day-night transition, or for some other reason. The FGS is quite stable and will track a guide star even during substantial spacecraft motion. The target may move about in an aperture, but the FGS will continue to track guide stars and reposition the target into the aperture. For most observations, the movement about the aperture during a spacecraft excursion will be quite small, but sometimes, especially for observations with the spectrographs, the aperture may move enough that the measured flux for the target will be less than the previous observation. Check the "jif" header keywords `V2_RMS` and `V3_RMS` for the root mean square of the jitter about the V2- and V3-axes. The following list of header keywords, found in the "jif" file, is an example of typical guiding rms values.

```
              / LINE OF SIGHT JITTER SUMMARY

V2_RMS   = 2.6                  / V2 Axis RMS (milli-arcsec)
V2_P2P   = 23.8                 / V2 Axis peak to peak (milli-arcsec)
V3_RMS   = 2.5                  / V3 Axis RMS (milli-arcsec)
V3_P2P   = 32.3                 / V3 Axis peak to peak (milli-arcsec)
```

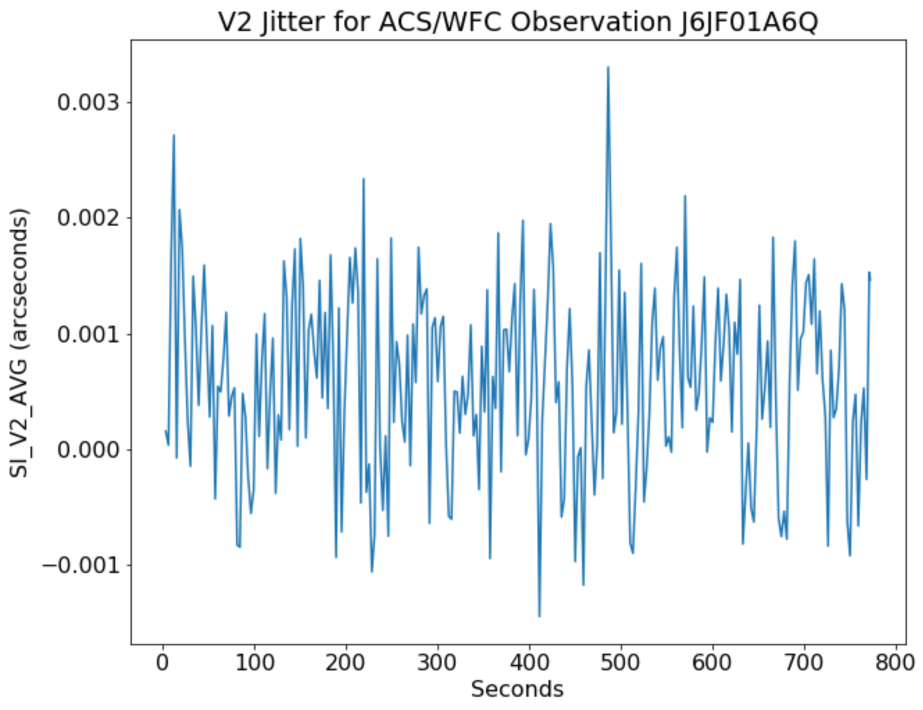
Re-centering events occur when the spacecraft software determines that shaking is too severe to maintain lock. The FGS will release guide star control and within a few seconds re-acquire the guide stars. It is assumed that the guide stars are still within the FGS field of view. During the re-centering time, "INDEF" will be written into the "jit" table. Re-centering events are tracked in the "jif" header file.

Care should be taken when interpreting loss-of-lock and re-centering events that occur at the beginning or at the end of the engineering telemetry window. The telemetry window is larger than the actual observation window. These events might not affect the observation since the observation will start after the guide stars are acquired (or re-acquired), and the observation may stop before the loss-of-lock or re-centering event that occurred at the end of a telemetry window.

Figure 6.1 shows a plot of the three second time-averaged jitter along the V2-axis of *HST* for an observation. The Python code used to produce this plot is shown below:

```
# Imports
from astropy.io import fits
from astropy.table import Table
import matplotlib.pyplot as plt

# Read the data from the jit file into an
# Astropy table
with fits.open('j6jf01a6j_jit.fits') as hdu:
    data = Table(hdu[1].data)

# Update the plot font size to 16
plt.rc('font', size=16)

# Plot the data
fig = plt.figure(figsize=(10, 8))
ax = fig.add_subplot(111)
ax.plot(data['Seconds'], data['SI_V2_AVG'])
ax.set_xlabel('Seconds')
ax.set_ylabel('SI_V2_AVG (arcseconds)')
ax.set_title('V2 Jitter for ACS/WFC Observation J6JF01A6Q')

# Save and close the figure
fig.savefig('v2_jitter.png')
plt.close(fig)
```

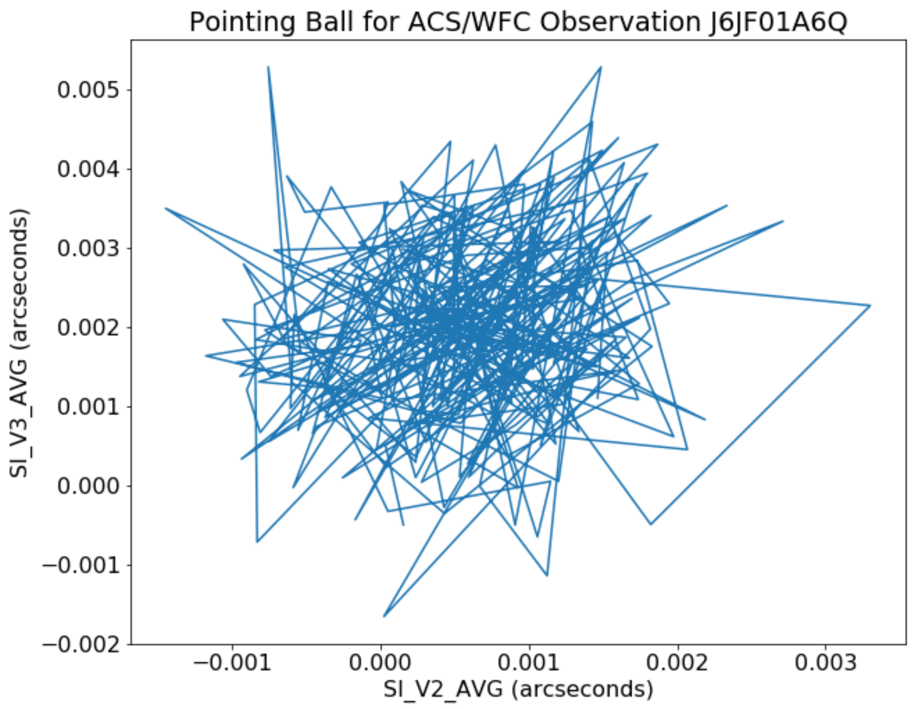
Figure 5.1: V2-axis Jitter vs. Time



A "pointing ball," which plots the V3-axis jitter against the V2-axis jitter, can also give insight into the stability of the pointing during the observation. Continuing from the same Python code used for Figure 6.1, we can create such a plot as shown below:

```
# Plot the data
fig = plt.figure(figsize=(10, 8))
ax = fig.add_subplot(111)
ax.plot(data['SI_V2_AVG'], data['SI_V3_AVG'])
ax.set_xlabel('SI_V2_AVG (arcseconds)')
ax.set_ylabel('SI_V3_AVG (arcseconds)')
ax.set_title('Pointing Ball for ACS/WFC Observation J6JF01A6Q')

# Save and close the figure
fig.savefig('pointing_ball.png')
plt.close(fig)
```

Figure 5.2: "Pointing Ball" or V3 jitter vs. V2 jitter in an observation



5.2.6 Spacecraft Information in Observation Logs

More information about the status of the spacecraft and other tracking diagnostics are contained within the observation logs. Here we discuss some of information available.

Spacecraft position information and measurements of the magnetic field are contained within the "jit" file data. Below we plot the orbital trace and the magnetic field amplitude in units of Gauss along the V2 axis. The result is shown in Figure 6.3.

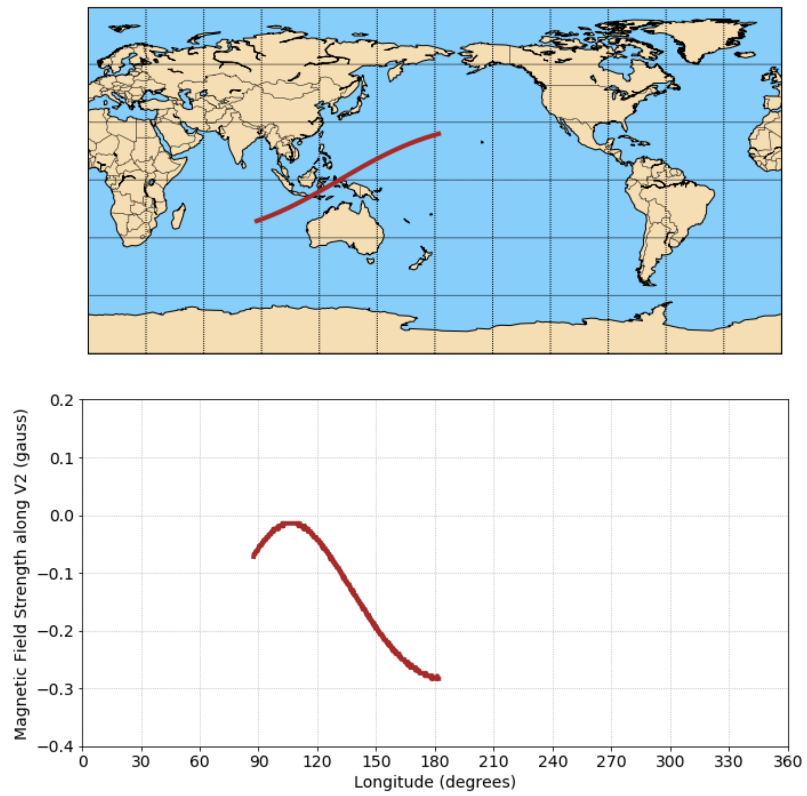
The example below uses a Python package not included in AstroConda to display a Mercator projection map. To install this package, simply type

```
conda install basemap
```

at a bash prompt, review the conda updates and installations, and then choose if you wish to proceed with the installation.

```
from astropy.io import fits
from mpl_toolkits.basemap import Basemap
import matplotlib.pyplot as plt

# Set the font size to 14
plt.rc('font', size=14)

# Read in the jitter file data
with fits.open('icau15esj_jit.fits') as hdu:
    data = hdu[1].data

# Set up the figure and subplots
fig = plt.figure(figsize=(10, 10))
ax1 = fig.add_subplot(211)
ax2 = fig.add_subplot(212)

# Create the map projection. This is a cylindrical equidistant projection.
# We have centered the map on longitude 180 degrees, which better suits
# this dataset. We have filled the landmasses with the matplotlib color
# 'wheat' and the water with 'lightskyblue.'
mapproj = Basemap(projection='cyl', lon_0=180, ax=ax1)
mapproj.drawcoastlines()
mapproj.drawcountries()
mapproj.drawparallels(np.arange(-90, 90, 30), alpha=0.1)
mapproj.drawmeridians(np.arange(-180, 180, 30), alpha=0.1)
mapproj.fillcontinents(color='wheat', lake_color='lightskyblue')
mapproj.drawmapboundary(fill_color='lightskyblue')

# Pull the variables we want from the jitter table.
latitude = data['Latitude']
longitude = data['Longitude']
mag_field_v2 = data['Mag_V2']

# Plot the data. Latitude and longitude are plotted on the map,
# while magnetic field strength along V2 is plotted as a function
# of longitude on a subplot below the map.
mapproj.plot(longitude, latitude, lw=4, color='brown')
ax2.plot(longitude, mag_field_v2, lw=4, color='brown')

# Configure the subplot axes and labels.
ax2.set_ylim(-0.4, 0.2)
ax2.set_xlim(0, 360)
ax2.set_xticks(np.arange(0, 361, 30))
ax2.grid(ls=':')
ax2.set_xlabel('Longitude (degrees)')
ax2.set_ylabel('Magnetic Field Strength along V2 (gauss)')

# Reduce whitespace around the plots and save the figure.
plt.tight_layout()
fig.savefig('orbit_magfield.png')
plt.close(fig)
```

Figure 5.3: HST Orbit and Magnetic Field along V2 Axis



During an exposure, a re-centering event may occur. During these events, the "Recenter" column in the "jit" table will contain a value of 1 (True). During re-centering events and losses of lock, the jitter and pointing data in the observation logs are set to INDEF or 1.6E+38, and these values should be excluded while examining observation logs.

On occasion, the telemetry format stream changes during an observation. During such switches, some engineering data may be lost.

During target acquisition peak-up, the telescope is moved to search for the target in a raster pattern. Once the target is found by the science instrument, the spacecraft automatically moves to center the target. During periods of peak-up, spatial scan, or in the case of a moving target observation, jitter statistics will be incorrect because they will incorporate the motion of the spacecraft.

[1] The V2 and V3 axes are part of the *HST* V (or ST) coordinate system, which is tied to the *HST* spacecraft. The V1 axes is the longitudinal axis of the spacecraft. The V2 and V3 axes thus form a plane on which the sky can be projected.