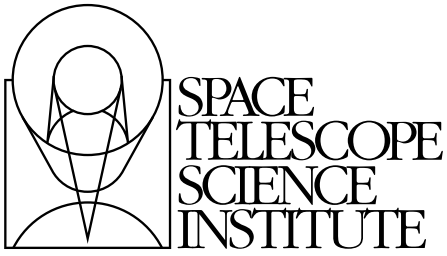
Version 5.0
November 2005

Synphot User's Guide



Space Telescope Science Institute
3700 San Martin Drive
Baltimore, Maryland 21218
help@stsci.edu

Operated by the Association of Universities for Research in Astronomy, Inc., for the National Aeronautics and Space Administration

User Support

For prompt answers to any question, please contact the STScI Help Desk.

- **E-mail:** help@stsci.edu
- **Phone:** (410) 338-1082
  (800) 544-8125 (U.S., toll free)

World Wide Web

Information and other resources are available on the STSDAS World Wide Web site:

- **URL:** http://www.stsci.edu/resources/software_hardware/stsdas

Revision History

Version	Date	Editors
5.0	November 2005	Vicki Laidler
4.0	December 1998	Howard Bushouse and Bernie Simon
3.0	March 1995	Howard Bushouse
2.0	September 1993	Howard Bushouse
1.0	November 1990	David Bazell

Contributors

Synphot is a large, complex software project based on Keith Horne's XCAL program which has benefited from the contributions of many people over the years, including David Bazell, Howard Bushouse, Vicki Laidler, Bernie Simon, and Anand Sivaramakrishnan. Special thanks to Susan Rose who assisted with the production of this version of the manual.

Citation

In publications, refer to this document as:

Laidler et al, 2005, "**Synphot** User's Guide", Version 5.0 (Baltimore: STScI).

Send comments or corrections to:
Space Telescope Science Institute
3700 San Martin Drive
Baltimore, Maryland 21218
E-mail:help@stsci.edu

Table of Contents

List of Figures vii

List of Tables ix

Preface xi

Chapter 1: About Synphot 1

1.1 Background 2
1.2 How Synphot Works 3
1.3 The Synphot Data Base 4
1.4 How Accurate are the Synthetic Photometry Results from Synphot? 5
1.5 Can Synphot be Used for Other Telescopes? 6

Chapter 2: Synphot and PyRAF 7

2.1 What is PyRAF? 7
2.2 General PyRAF advantages: 8
2.3 Synphot specific examples: 9
2.3.1 Replacing the "@filename" syntax 9
2.3.2 Replacing the "vzero" syntax 10
2.3.3 Generating a custom wavelength table 11

Chapter 3: Using Synphot 13

3.1 Synphot Tasks 13
3.2 Command and Argument Syntax 15
3.2.1 Syntax Examples 15
3.2.2 Observation Mode 16
3.2.3 Spectrum 18
3.2.4 Form 21
3.3 Wavelength Table 25
3.4 Variable Substitution (VZERO) 26

3.5 Reference Data 27
3.6 Table formats 28
3.7 Using EPAR to edit task parameters 28
3.8 About the Examples in This Manual 28

Chapter 4: Cookbook 31
4.1 Bandpasses 31
4.2 Spectra 34
4.3 Photometry 37

Chapter 5: Task Descriptions 41
5.1 Bandpar 42
5.1.1 Examples 44
5.2 Calcband 44
5.2.1 Examples 46
5.3 Calcphot 46
5.3.1 Examples 48
5.4 Calcspec 50
5.4.1 Examples 52
5.5 Countrate 54
5.5.1 Examples 57
5.6 Fitband 59
5.6.1 Example 62
5.7 Fitspec 63
5.7.1 Example 66
5.8 Fitgrid 68
5.8.1 Examples 70
5.9 Genwave 72
5.9.1 Examples 72
5.10 Grafcheck 73
5.10.1 Example 73
5.11 Graflist 74
5.11.1 Examples 74
5.12 Grafpath 75
5.12.1 Examples: 75

5.13 Imspec 76
5.13.1 If the input file is an image: 76
5.13.2 If the input file is a table: 77
5.13.3 Other parameters: 77
5.13.4 Examples 78
5.14 Mkthru 78
5.14.1 Examples 80
5.15 Modeinfo 80
5.15.1 EXAMPLES 81
5.16 Obsmode 81
5.16.1 Examples 82
5.17 Plband 82
5.17.1 Examples 84
5.18 Plspec 85
5.18.1 Examples 89
5.19 Plratio 94
5.19.1 Examples 97
5.20 Pltrans 98
5.20.1 Examples 100
5.21 Refdata 101
5.22 Showfiles 102
5.22.1 Example 103
5.23 Thermback 103
5.23.1 EXAMPLES 105

Chapter 6: Inner Workings 107
6.1 Syncalc 107
6.2 Graph and Component Tables 109

Chapter 7: Calibration Using Synthetic Photometry 115
7.1 Principles of calibration 115
7.2 Accuracy and errors in synthetic photometry 118
7.2.1 Groundbased photometric systems 119
7.2.2 Vegamags 119
7.2.3 HST bandpasses 120

Appendix A: On-Line Catalogs and Spectral Atlases 121
A.1 HST Calibration Spectra 122
A.2 Kurucz 1993 Atlas of Model Atmospheres 125
A.3 THE HST/CDBS VERSION OF THE 1993 KURUCZ ATLAS 126
A.4 USE OF KURUCZ ATLAS WITH **SYNPHOT** 129
A.5 Bruzual Spectrum Synthesis Atlas 131
A.6 Gunn-Stryker Spectrophotometry Atlas 132
A.7 Bruzual-Persson-Gunn-Stryker Spectrophotometry Atlas 132
A.8 Jacoby-Hunter-Christian Spectophotometry Atlas 135
A.9 Bruzual-Charlot Atlas 138
A.10 Kinney-Calzetti Atlas 139
A.11 Buser-Kurucz Atlas of Model Atmospheres 139
A.12 AGN Atlas 142
A.13 Galactic Atlas 142

Appendix B: Bibliography 143

Index 147

List of Figures

Figure 3.1: Standard photometric systems illustrated. 24
Figure 4.1: Plband Plot of Johnson V and WFPC2 Bandpasses 32
Figure 4.2: List of NICMOS Filters in File “nic_filters.lis” 33
Figure 4.3: Bandpar Results for a List of NICMOS Filters 34
Figure 4.4: Plotting Spectra with plspec 35
Figure 4.5: countrate Parameters for STIS CCD Observation 36
Figure 4.6: Predicted STIS CCD Spectrum Produced by countrate 37
Figure 4.7: Calcphot Results for a WFPC2 F439W Observation 38
Figure 4.8: Calcphot Results for a NICMOS F160W Observation 39
Figure 4.9: Commands for Plotting Redshifted Spectra and UB Bands 39
Figure 4.10: Plots of Redshifted Spectra and UB Bandpasses 40
Figure 4.11: Using calcphot to Compute U-B Colors of Redshifted Spectra 40
Figure 5.1: Bandpar Parameters 43
Figure 5.2: Calcband Parameters 44
Figure 5.3: Calcphot Parameters 46
Figure 5.4: Sample Calcphot Run 49
Figure 5.5: Second Sample Calcphot Run 49
Figure 5.6: Calcspec Parameters 51
Figure 5.7: FOS Count Rate Spectrum of G191-B2B 53
Figure 5.8: WFPC2 Blackbody Spectra 54
Figure 5.9: Countrate Task Parameters 55
Figure 5.10: Example Countrate Parameter Settings 57

Figure 5.11: Plot of Countrate Example Spectrum 58
Figure 5.12: Second Countrate Example Parameter Settings 59
Figure 5.13: Fitband Parameters 60
Figure 5.14: Results from Example Fitband Run 63
Figure 5.15: Fitspec Parameters 64
Figure 5.16: Example Fitspec Parameter Settings 67
Figure 5.17: Results from Fitspec 68
Figure 5.18: Fitgrid Parameters 69
Figure 5.19: Fitgrid Results 71
Figure 5.20: Genwave Parameters 72
Figure 5.21: Imspec Parameters 76
Figure 5.22: Mkthru Parameters 79
Figure 5.23: Plband Parameters 83
Figure 5.24: Results of First Sample Plband Run 84
Figure 5.25: Results of Second Sample Plband Run 85
Figure 5.26: Plspec Parameters 86
Figure 5.27: Plspec Results 90
Figure 5.28: Results from Second Sample Plspec Run 91
Figure 5.29: Plspec Comparison of STIS Sensitivities 92
Figure 5.30: Plotting Calcphot Results 93
Figure 5.31: Distribution of Detected Counts 94
Figure 5.32: Plratio Parameters 95
Figure 5.33: Sample Plratio Output 97
Figure 5.34: Pltrans Parameters 98
Figure 5.35: Results from Sample Pltrans Run 100
Figure 6.1: Structure of Instrument Graph Table 111
Figure 6.2: Structure of Component Lookup Table 112

List of Tables

Table 3.1: Primary Passband and Spectral Computation and Plotting Tasks 14
Table 3.2: Passband and Spectral Fitting Tasks 14
Table 3.3: Utility Tasks 14
Table 3.4: Passband Functions 18
Table 3.5: Spectrum Functions 20
Table 3.6: Reddening Laws Used by ebmvx 21
Table 3.7: Forms 22
Table 3.8: Wavelength Units 25
Table 3.9: vzero Examples 27
Table 3.10: Parameters in refdata Pset 27
Table 5.1: Bandpar Photometric Parameters 42
Table 5.2: Calcband Output Keywords 45
Table 5.3: Functions Supported by Calcphot 47
Table 5.4: Calcphot Output Table Columns 48
Table 5.5: Error Type Codes for Plspec 88
Table 6.1: Syncalc Functions 109
Table 6.2: Thermal Emissivity Table. 113
Table 6.3: The emissivity table also contains some crucial information in header keywords: 114
Table A.1: CDBS Flux Standards with Columns in Order of Preference 124
Table A.2: HST Calibration Spectra 125
Table A.3: Grid of temperatures for the models 126
Table A.4: Wavelength coverage for the models 126
Table A.5: Sample FITS Header from Kurucz 93 Model 128
Table A.6: Suggested models for specific stellar types 130

Table A.7: Bruzual Synthetic Spectral Atlas 131
Table A.8: Bruzual-Persson-Gunn-Stryker Spectral Atlas 133
Table A.9: Jacoby-Hunter-Christian Spectral Atlas 136
Table A.10: Kurucz K1200 Library 140
Table A.11: BK Atlas 141

Preface

This Guide describes the Space Telescope Science Data Analysis System (STSDAS) synthetic photometry (**synphot**) software package. STSDAS is developed and maintained by the STSDAS group at STScI and is an external package layered on the Image Reduction and Analysis Facility (IRAF), which is developed and maintained by NOAO. **synphot** is therefore portable to any host architecture supported by IRAF. **synphot**, like all other IRAF tasks, also runs under PyRAF, the new Python-based command language for IRAF.

In order to run **synphot**, you will need to have IRAF, STSDAS, TABLES (another IRAF package maintained by the STSDAS group at STScI), and the data files associated with **synphot** installed on your machine. You may also wish to have PyRAF installed. You can obtain these packages from:

Package	Download from
IRAF	http://iraf.noao.edu/iraf-homepage.html or call (520)318-8160
STSDAS	http://www.stsci.edu/resources/software_hardware/stsdas
TABLES	http://www.stsci.edu/resources/software_hardware/tables
PyRAF	http://www.stsci.edu/resources/software_hardware/pyraf
synphot data	http://www.stsci.edu/resources/software_hardware/stsdas/synphot

A variety of additional documentation is available for **synphot**, IRAF, and PyRAF. For more information, please consult the following documents:

For help with	Consult	Available from:
IRAF cl or STSDAS	The STSDAS User's Guide	www.stsci.edu/resources/software_hardware/stsdas
PyRAF	PyRAF tutorial, programmer's guide, and FAQ	stsdas.stsci.edu/pyraf/doc/index.html
Installing **synphot** data files	Data installation guide	ftp.stsci.edu/pub/software/stsdas/refdata/synphot/stdata_install.ps
HST keywords for **synphot**	**Synphot** Data User's Guide	www.stsci.edu/resources/software_hardware/stsdas/synphot/SynphotDataGuide.html
synphot problems	**synphot** FAQ	http://www.stsci.edu/resources/software_hardware/stsdas/synphot/FAQ

If you have questions or problems using IRAF, PyRAF, STSDAS, or **synphot**, you may also contact the STScI Help Desk by sending E-mail to help@stsci.edu.

CHAPTER 1:

About Synphot

In this chapter. . .

1.1 Background / 2
1.2 How Synphot Works / 3
1.3 The Synphot Data Base / 4
1.4 How Accurate are the Synthetic Photometry Results from Synphot? / 5
1.5 Can Synphot be Used for Other Telescopes? / 6

The Space Telescope Science Data Analysis System (STSDAS) **synphot** package simulates photometric data and spectra as they are observed with the Hubble Space Telescope (HST). **Synphot** tasks will:

- Plot HST sensitivity curves and calibration target spectra.
- Predict count rates for observations in any available HST instrument mode.
- Compute the photometric calibration for any HST instrument mode.
- Examine photometric transformation relationships among the various HST observing modes as well as conventional photometric systems, such as Johnson *UBV*.

Synphot can help HST General Observers (GOs) prepare Phase I and Phase II *observing proposals*. **Synphot**'s cross-instrument simulation ability is useful for planning and optimizing HST observing programs. **Synphot** provides an easy way to examine the transmission curve of the HST Optical Telescope Assembly (OTA), sensitivity curves for all modes of observing with the science instruments, and flux distributions of HST calibration targets.

Passbands for standard photometric systems are available, and users can incorporate their own filters, spectra, and data. A powerful spectrum calculator can create complicated composite spectra from various parameterized spectrum models, grids of model atmosphere spectra, and atlases of stellar spectrophotometry.

The rest of this chapter contains introductory information regarding the **synphot** package, and Chapter 2 provides a **synphot** user's introduction to PyRAF, the new Python-based command language for IRAF. Chapter 3 discusses the basic features that are common to most **synphot** tasks, and Chapter 4 demonstrates some simple examples of how they are used by the tasks. Detailed explanations of each task and examples of their use are given in Chapter 5. Chapter 6 contains a more in-depth look at internal operations, and Chapter 7 provides a mathematical description of some basic synthetic photometry concepts and its role in calibrating the HST instruments. Many of the examples contained in Chapter 4 and Chapter 5 make use of spectral data from the on-line catalogs of spectral atlases maintained at STScI. These atlases are also available to off-site **synphot** users. Appendix A:On-Line Catalogs and Spectral Atlases contains detailed information on the origin and contents of these atlases, as well as installation information for off-site users.

1.1 Background

The **synphot** package is modeled on Keith Horne's XCAL software—a suite of Fortran subroutines designed to be used as a dynamic throughput generator using files stored in the HST Calibration Data Base System (CDBS). Because the HST has a vast number of interrelated observing modes, it is impractical and inefficient to derive and maintain an independent calibration for every possible instrument configuration. Rather than restrict calibrations to a smaller number of *core* modes, the alternative is to provide a software tool that can generate the throughput function for *any* HST observing mode; this is how XCAL and **synphot** are implemented. Throughput functions and calibration data files for specific observing modes are dynamically generated as needed. This approach reduces the number of calibration data files that must be created and maintained to a manageable level, and ultimately saves considerable observing time since information from calibration observations in one mode can be easily transferred to other closely related modes.

Additional information and discussion of the synthetic photometry approach to HST calibration can be found in Koornneef et al. (1986) and Horne (1988).

1.2 How Synphot Works

The basic concepts, data structures, and software needed for dynamic throughput generation are discussed in detail by Horne, Burrows, and Koornneef (1986). Briefly, the throughput calibration of the HST observatory is represented in a framework consisting of:

- Component throughput functions for every optical component (e.g., mirror, filter, polarizer, disperser, and detector).
- A configuration graph describing the allowed combinations of the components.

A particular observing mode is specified by a list of keywords, which might be familiar names of filters, detectors, and gratings. The keywords are used to trace a path through the observatory configuration graph, thereby translating the keyword list into a list of pointers to data files that contain the individual component throughput functions. The grand throughput function of the requested observing mode is formed by multiplying together the individual component throughputs at each wavelength. (See Chapter 6 for more details on the functioning of the observing mode expression evaluator and the internal structure and functioning of the configuration graph and component throughput tables.)

To retrieve a particular HST passband, you furnish the passband generator with a couple of keywords, for example "WFPC2,4,F336W". The passband generator uses the keywords to trace a path through the graph, multiplies together the component passbands it encounters along the way, and returns the passband evaluated on a particular wavelength grid.

Passbands can then be convolved with spectral data to simulate HST observations of particular targets. Spectra may come from existing files containing lists of fluxes as a function of wavelength, or may be dynamically generated (individually or in combination) as simple blackbody, power-law, or Hydrogen continuum emission spectra of chosen temperatures and slopes.

Most **synphot** task data I/O is done via FITS files with binary table extensions. The HST instrument graph, component lookup, and component throughput tables are all in this format. All output files created by the **synphot** tasks are either in FITS format or STSDAS table format, depending on the extension specified for the output filename. Input data files, such as passband throughput and spectral data files, may be in either FITS files, STSDAS table format, or plain ASCII text tables.

Warning: STSDAS table format is a binary file format, and so care must be taken when working on multiple platforms with different endian architectures for floating point representation -- for example, Solaris and Linux. For this reason, we recommend that the use of STSDAS tables be deprecated, and that FITS binary tables be used instead. Existing STSDAS tables can easily be converted to FITS using the TABLES command
`tcopy mytable.tab mytable.fits`

1.3 The Synphot Data Base

As you may have already realized from the preceding discussion, the **synphot** package is entirely data driven. That is, no information pertaining to the physical description of instruments or their throughput characteristics is contained within the software, but is instead contained within an external "database." These data must be available in order to run any **synphot** tasks. The data set contains the HST instrument graph, component lookup, and component throughput tables, which are maintained and stored within the *HST Calibration Data Base System* (CDBS) at STScI. New versions of these tables are created whenever new or updated calibration information becomes available for the HST instruments.

Users at STScI have automatic access to the **synphot** data set on all science computing clusters. New versions of any of the tables become available within about 24 hours after they are installed in the CDBS.

Because the data set is not currently distributed along with the STSDAS software, off-site users must retrieve and install it separately before they will be able to use **synphot**. This can be done in one of two ways:

- ***Tar Files:*** Every time a new version of the STSDAS software is released, a "snapshot" of the current **synphot** data set is copied into a few tar files, which can then be retrieved, unpacked, and installed on your system. The STSDAS Site Manager's Installation Guide contains instructions for doing this. This method offers the convenience of having to transfer only a few files and automatically creates the necessary directory tree for the data. On the other hand, this "snapshot" is made only once or twice a year and therefore may not contain the latest data for the HST instruments.
- ***Individual CDBS Files***: An alternative method is to transfer the individual tables from the `/data/cdbs1/` directory at `ftp.stsci.edu` (see below), which is updated on a daily basis.

The best method is perhaps a combination of the two: first-time installers may wish to use the "snapshot" tar files to initially create and populate the directory structure, and then periodically check the CDBS area at STScI for updates to individual tables.

The latest versions of individual tables can be obtained via anonymous ftp in the `/data/cdbs2/` directory and its associated subdirectories on node `ftp.stsci.edu`, or through the web site

`http://www.stsci.edu/hst/observatory/cdbs/cdbs_throughput.html.`

The instrument graph and component lookup tables are contained in the mtab subdirectory and are named *_tmg.fits and *_tmc.fits. The component throughput tables are logically grouped into subdirectories of `/data/cdbs2/comp/` corresponding to each of the HST instruments (`acs,fgs`, `foc`, `fos`, `hrs`, `hsp`, `nicmos`, `nonhst`, `ota`, `stis`, `wfpc`, and `wfpc2`). Component throughput table names contain a three digit suffix indicating their version number. You can determine which tables are new by comparing either their names or creation dates with the corresponding set of tables installed at your site.

1.4 How Accurate are the Synthetic Photometry Results from Synphot?

Because the **synphot** package is entirely data driven, the accuracy of its results depends entirely on the accuracy of the bandpass sensitivity curves and zero points in the **synphot** database. The accuracy of these values is dependent on the instrument and photometric system under consideration.

As a general rule of thumb, synthetic photometry involving photometric systems that have been defined from the ground, or photometry that is given in vegamags, should only be considered accurate to about 5 percent. To achieve better accuracy, you should perform your own calibrations using known standard stars. (For a discussion of vegamags and other magnitude units, see section 3.2.4.)

Synthetic photometry with the stable HST instrumentation, flying above the atmosphere, when used in HST instrument natural systems, without reference to vegamags, can achieve accuracy much better than 5%; for example ACS broad band filters <=1% (De Marchi et al 2004).

For more detail, see Chapter 7:Calibration Using Synthetic Photometry, and the HST Data Handbook.

1.5 Can Synphot be Used for Other Telescopes?

Because the tasks in the **synphot** package are data driven, instrument observing modes can be changed and new instruments added without changing the software. To use **synphot** with non-HST instruments or components you would need to modify (or rebuild) only the instrument graph and component lookup tables.

Synphot requires:

- One instrument graph table.
- One component lookup table.
- One thermal component lookup table (only needed for thermal background calculations)
- One throughput table for each telescope and instrument component that appears in the graph and component lookup tables.

The names of the instrument graph and component lookup tables to be used by the **synphot** tasks are set by the parameters `grtbl` and `cmptbl`, in the `refdata` parameter set. The names of the individual component throughput tables are contained in the component lookup table and are located automatically when you run a task. See Chapter 6 for details on the structure of these tables. To build your own instrument graph and component lookup tables it is perhaps easiest to either start with a copy of the existing HST tables and modify or add to them, or at least use the HST tables as a model for your own tables. To make use of your own custom graph and component lookup tables in **synphot**, just change the values of the `grtbl` and `cmptbl` parameters (and the telescope area parameter, if appropriate) in the `refdata` parameter set.

CHAPTER 2:

Synphot and PyRAF

In this chapter. . .

2.1 What is PyRAF? / 7
2.2 General PyRAF advantages: / 8
2.3 Synphot specific examples: / 9

In this chapter we will illustrate some uses of **synphot** in the PyRAF environment, and demonstrate some techniques that can make scripting and data processing more efficient.

2.1 What is PyRAF?

PyRAF (Greenfield & White, 2000) is a new command language for IRAF that is based on Python. It has a number of advantages over the IRAF CL. Most importantly, with few exceptions, it allows use of exactly the same syntax that the IRAF CL accepts; but it is a Python module, so all python capabilities are available on the command line. Some of the advantages that it provides are:

- true command line recall (with arrow key editing)
- command and filename completion
- GUI-based graphics windows, previous plot recall, multiple graphics windows
- GUI epar editor with display of help in separate window

- IDL-like capabilities, when used together with numarray and PyFITS modules
- true error handling for scripts (shows which line the script fails at when errors occur)
- can script IRAF tasks in Python language
- exception handling capability

Since PyRAF is so highly compatible with the IRAF CL, virtually all of the examples shown in this handbook will work the same for PyRAF. Minor differences include the user prompt and the graphics windows appearance.

More information on PyRAF can be found at:

http://www.stsci.edu/resources/software_hardware/pyraf

For an introduction to PyRAF for IRAF users, and a general comparison of similarities and differences, see the Pyraf Tutorial and PyRAF FAQ available on the documentation pages from the above URL.

2.2 General PyRAF advantages:

- Simpler epar interface for XWINDOWS

  Under PyRAF, epar pops up a GUI window for editing task parameters. This can be particularly useful for **synphot** tasks that have many task parameters.

- true command line recall (with arrow key editing)

  Command line recall and editing are particularly useful when working with **synphot** interactively and experimenting with a variety of task parameters.

- Tab filename completion

  PyRAF provides tab filename completion. This can be particularly useful for selecting a reference spectrum from the crcalspec$ directory that contains dozens of available calibration spectra.

- Simpler specification of parameterized keywords and psets

  In the CL, observation modes containing a parameterized keyword (such as cont# for WFPC2) can be entered directly in epar, but must be enclosed in quotes if they are entered on the command line. PyRAF will accept parameterized keywords without surrounding them in quotes. Editing psets (parameter sets) is easy in PyRAF:

psets are presented as buttons in the epar widget. Selecting the button pops up a new epar window. In the cl, special key combinations must be used to begin and end the editing of psets.

2.3 Synphot specific examples:

These examples use Python syntax. In Python, indentation matters: instead of `begin` and `end` constructs, Python uses indentation to manage loops. A loop begins with a for statement ending in a colon:

```
for item in mylist:
```

or

```
for i in range(20):
```

In an interactive PyRAF session, the Python interpreter will respond to these lines by changing the prompt to three dots (`...`) indicating that it is waiting for indented lines to be executed inside the loop. To complete the loop, simply type a carriage return to produce a blank line.

The Python line continuation character is a backslash ("\"). We've used it in these examples for typesetting reasons, but you generally should not need to use it in interactive sessions.

2.3.1 Replacing the "@filename" syntax

Many **synphot** tasks will accept a file containing a list of filenames, one per line, to be processed. Using the scripting capabilities of Python, PyRAF can essentially duplicate this functionality without the need to generate an intermediate file.

Example:

Suppose you want to run calcspec to predict the observed spectrum for a variety of potential target spectra (which you have collected in the form of FITS files in the directory /data/planning/targets) as observed through the ACS wide field camera with the f606w filter.

After entering PyRAF, you need to import the **synphot** package in order to use the Python scripting capabilities

--> from pyraf.iraf import stsdas,hst_calib,synphot

To generate a list of the target filenames, you'll need the python package glob.

--> import glob

--> mytargets=glob.glob('/data/planning/targets/*.fits')

The variable mytargets is now a list of filenames, which you can look at in a print statement, use in a for loop, and operate on to produce the desired task parameters.

```
--> for targ in mytargets:
...      iraf.calcspec(spectrum='band(acs,wfc1,f606w)*'+targ,\
 output='output_'+targ,form='flam')
```

One significant difference between using this syntax, and the **synphot** @filename syntax, is the handling of the wavelength table on which the calculations are performed. (See Section 3.3, "Wavelength Table," on page 25 for a more detailed discussion of the wavelength table.) In a python loop, calcspec will be invoked separately for each input spectrum, and will therefore determine the optimal wavelength table to be used independently for each input spectrum. Using the @filename syntax invokes calcspec only once, and calcspec determines the optimal wavelength table based on the first input spectrum only, and uses it for all spectra in the list.

This default behavior is desirable in some cases and undesirable in others. If you want to ensure that all objects are run with the same wavelength table, you can generate it ahead of time and use it as another parameter in the call.

```
--> genwave mywavecat.fits 2000 10000 1.5
A/pixel = 1.5 Mean km/s/pixel = 90.5
Range = 2000. to 10000. Number of Pixels = 5334
--> for targ in mytargets:
...      iraf.calcspec(spectrum='band(acs,wfc1,f658n)*'+targ,\
output='output_'+targ,form='flam',wavetab='mywavecat.fits')
```

Note that, when running PyRAF interactively, it's perfectly possible to mix the use of ordinary IRAF style calls with the use of Python-scripted style calls. This allows you to use the terse interactive syntax most of the time, and use the more verbose scripted style only when necessary. (In a Python script, you cannot mix the styles and must use the Python style throughout.)

2.3.2 Replacing the "vzero" syntax

Some **synphot** tasks allow you to run in a loop using the vzero variable to specify values to substitute for $0 in another input parameter. If you use this ability infrequently enough that you find its syntax hard to remember, you might prefer to build a loop in python instead.

Example:

Let's look at the last example for calcspec, which illustrates the use of vzero, and use a python command instead. The goal is to generate a series of blackbody functions, covering the range of temperatures from 5000 to 20000 in steps of 1000K, as observed through the WFPC2 F439w filter with a flux of 1200 counts.

Here's the pure-**synphot** command:

sy> calcpsec "rn(bb($0),band(wfpc2,f439w,a2d7),1200,counts) \
bbody.fits form=flam vzero="5e3-20e3x1e3"

Here's the python looping command:

--> for t in range(5000,21000,1000):

rncmd="rn(bb(%d),band(wfpc2,f439w,a2d7),1200,counts)"%t

iraf.calcspec(spectrum=rncmd,out='bbody.fits',form='flam')

Note that the range runs from 5000 to 21000: this is because python ranges are defined including the low end but excluding the high end.

Because this invokes calcspec separately for each case, using the python loop produces a single FITS file with 16 extensions, each containing a table with columns for wavelength and flux. The vzero method produces a FITS file with a single table, with one column for wavelength and 16 columns for flux.

2.3.3 Generating a custom wavelength table

The **synphot** task **genwave** can be used to generate wavelength tables with uniform spacing; but what if non-uniform spacing is desired? In this case, we can use PyFITS to create a new table for use with **synphot**.

Suppose we want a wavelength set that ranges from 2000 to 8000, with single Angstrom spacing over most of the range, but 0.1 Angstrom spacing around the OIII forbidden lines at 4959 and 5007.

First, we import numarray and create the three pieces separately:

```
--> import numarray
--> lowwave=numarray.arange(2000.,4950.)
--> hiwave=numarray.arange(5010.,8000.)
--> midrange=numarray.arange(4950.,5010.,0.1)
```

then concatenate them

```
--> wave=numarray.concatenate( (lowwave,midrange,hiwave) )
```

Now we need to create a FITS file from this array. First we import PyFITS and define a column:

```
--> import pyfits
--> col1=pyfits.Column(name='wavelength',\
unit='angstroms',format='E',array=wave)
```

Then construct a table from the column; a list of HDUs including the table, and write the new file out to disk.

```
--> tabhdu=pyfits.new_table([col1])
--> tabhdu.writeto('mywaveset.fits')
```

The FITS file mywaveset.fits can now be used as a wavelength table in a **synphot** task.

CHAPTER 3:

Using Synphot

In this chapter. . .

3.1 Synphot Tasks / 13
3.2 Command and Argument Syntax / 15
3.3 Wavelength Table / 25
3.4 Variable Substitution (VZERO) / 26
3.5 Reference Data / 27
3.6 Table formats / 28
3.7 Using EPAR to edit task parameters / 28
3.8 About the Examples in This Manual / 28

This chapter describes the basic structure of the **synphot** package, and how the tasks are used. Most of the chapter is devoted to a discussion of common parameters and command syntax. The use of each individual task is described in detail in Chapter 5.

3.1 Synphot Tasks

There are three categories of tasks in the **synphot** package.

- Tasks that create, manipulate, and plot passbands and spectra (Table 3.1). This is the main group of tasks.
- Tasks that fit model passbands and spectra to spectrophotometric and photometric data (Table 3.2). These tasks are intended to be used for passband reconstruction.
- General utility tasks that convert data and check or maintain the instrument graph and component tables (Table 3.3).

Table 3.1: Primary Passband and Spectral Computation and Plotting Tasks.

Task	Function
calcband	Calculate a model passband
plband	Plot passband data
calcspec	Calculate a model spectrum
plspec	Plot spectral and photometric data
countrate	Calculate the response of HST instruments to model spectra and passbands
calcphot	Calculate synthetic photometry for model spectra and passbands
bandpar	Calculate photometric parameters of a passband
plratio	Plot the ratio of observed and synthetic spectral data
pltrans	Plot photometric transformation (color-color, color-mag) diagrams
thermback	Calculate a model thermal background

Table 3.2: Passband and Spectral Fitting Tasks

Task	Function
fitband	Fit a model passband to known throughput data
fitspec	Fit a model spectrum to known spectral data
fitgrid	Fit a spectrum by interpolating within a grid of model or observed spectra, such as a spectral atlas

Table 3.3: Utility Tasks

Task	Function
imspec	Convert IRAF/STSDAS images to and from STSDAS or FITS tables
genwave	Generate a wavelength set
grafcheck	Check an instrument graph table for bad rows
graflist	List components downstream from a given component in an instrument graph table
grafpath	Print ordered list of throughput files used for a specified obsmode
mkthru	Converts an ASCII, FITS, or STSDAS table into a throughput table for installation into CDBS
modeinfo	Lists available keywords for an instrument or filter system & explains their use
obsmode	List the valid observation mode keywords for an instrument
showfiles	Print filenames used in a **synphot** expression

3.2 Command and Argument Syntax

There are five parameters that are common to many of the primary **synphot** tasks:

- *Observation mode* (`obsmode`) - Passband to be calculated (page 16).
- *Spectrum* (`spectrum`) - Spectrum to be calculated (page 18).
- *Form* (`form`) - Units of the output data; for example, counts or magnitudes (page 21).
- *Wavelength table* (`wavetab`) - Wavelength grid on which passband and spectrum will be calculated (page 24).
- *Reference data parameters* (`refdata`) - Pointer to reference data for instrument graph and component lookup tables (page 27).

Each of these parameters is discussed in detail in the following sections, but before getting into details, the next section provides some examples of general command syntax to show how some of these parameters are typically used.

3.2.1 Syntax Examples

In the command:

```
sy> plband acs,hrc,f555w
```

the string "acs,hrc,f555w" is the observation mode, or `obsmode`, and specifies that a passband be calculated by taking the product of the individual throughputs of the Advanced Camera for Surveys (ACS) high resolution camera (hrc), the f555w filter, the detector sensitivity, and the HST Optical Telescope Assembly (OTA). The OTA throughput is included automatically whenever an HST instrument mode is specified. (Whenever an `obsmode` involving the Faint Object Camera (FOC), Faint Object Spectrograph (FOS), or High Resolution Spectrograph (HRS) is specified, the throughput for COSTAR, the Corrective Optics Space Telescope Axial Replacement, is also included automatically.)

In this example, where the settings of the `wavetab` and `refdata` task parameters were not specified, the HST default values will be used. The command:

```
sy> calcband acs,hrc,f555w acs_f555w.fits
```

will calculate that same passband and write the resulting data to the table `acs_f555w.fits`, which will contain columns of wavelength and throughput values.

The next example simply plots the spectrum of the star HZ 44:

```
sy> plspec "" crcalspec$hz44_stis_001.fits flam
```

Here the `obsmode` is set to a null string (`""`) so that the spectrum is plotted without being multiplied by a passband. The spectral data for HZ 44 come from a library of HST calibration spectra in the directory referenced by the IRAF environment variable `crcalspec$`, so the `spectrum` parameter is simply the name (including the directory path) of the table. The `form` parameter is set to "flam", which specifies that the spectrum is to be plotted in units of f_λ (ergs/s/cm^2/Å).

The last example will calculate the expected countrate for observing the star HZ 44 using the Wide Field Planetary Camera 2 (WFPC2) CCD number 1 and F439W filter:

```
sy> calcphot wfpc2,1,f439w,a2d7 crcalspec$hz44_stis_001.fits counts
```

Here the `obsmode` is the WFPC2 chip 1 with the F439W filter and the A-to-D gain setting of 7. The `form` parameter is "counts". This will multiply the combined HST+WFPC2+F439W throughput with the spectrum of HZ 44 and compute the resulting count rate (6287 DNs per second!).

3.2.2 Observation Mode

The observation mode (`obsmode`) parameter defines the passband, i.e., the wavelength-dependent sensitivity curve of the photometer or spectrophotometer. The `obsmode` parameter can also be used to specify a color index, or a series of passbands or color indexes, as described further below. The `obsmode` is usually given as a string of keyword arguments to the band function, for example "band(wfpc2,4,f555w)". The list of keywords specifies a light path through the telescope and an instrument. The **modeinfo** task can be used to obtain a current list of all the valid keyword names for the HST instruments. The keywords may appear in any order and are *not* case sensitive.

As a convenience to users, if the `obsmode` expression consists of a single call to the band function, only the arguments to the function need be given. For example, the `obsmode` "band(wfpc2,4,f555w)" can also be given as just "wfpc2,4,f555w".

Passbands can also be read directly from files. Passband files may be in the form of either ASCII text, FITS binary tables, or STSDAS binary tables and must contain, at minimum, columns of wavelength and throughput values. The **synphot** expression calculator determines whether a file contains a passband or a spectrum by looking for the throughput column. If the file contains a throughput column it is read as a passband; if not, it is read as a spectrum.

If the file is an STSDAS or FITS table, **synphot** will look for columns named "WAVELENGTH" and "THROUGHPUT". Since ASCII text files do not contain column names, **synphot** assumes that the wavelengths are in

the first column and you must supply the column number of the throughput data in brackets at the end of the file name. For example, if the throughput data are in column 2 of the text file `my_filter.dat`, then the `obsmode` is specified as "my_filter.dat[2]". See Chapter 6:Inner Workings for more details on the allowed formats of input tables and how the **synphot** expression evaluator handles them.

A series of passbands may be processed in a single task execution by setting the `obsmode` parameter to "@*filename*", where *filename* is the name of an ASCII text file containing a list of desired passbands, one per line. The passbands designated on each line may be composed of either calls to the band function or specific file names containing throughput data.

Color indices are defined by setting the `obsmode` parameter to *MODE1–MODE2*, where *MODE1* and *MODE2* are specifications for the two passbands. For example, "band(wfpc2,f439w)-band(wfpc2,f555w)" gives you the WFPC2 color index that approximates *B-V*. The observation mode can also be a part of a larger expression used to calculate either a passband or spectrum. The syntax of these larger expressions is explained below.

The HST OTA transmissivity is included by default in the calculation of all HST-related observation modes. It can be included explicitly by adding the keyword `ota` or excluded by adding the keyword `noota`; for example "fos,blue,1.0,g130h,noota".

All current HST instruments (except the FGS) have built-in corrective optics to compensate for the spherical aberration of the primary. For the first-generation instruments FOC, FOS, and GHRS, the HST instrument graph allows the inclusion of the effects of COSTAR on the wavelength-dependent sensitivity. This includes the product of the reflectivity curves for each pair of COSTAR mirrors for each of these instruments, as well as the effects on instrument throughput and sensitivity due to the improved point-spread function that is achieved with COSTAR. Like the OTA, the COSTAR effects on passbands and count rates are included by default for these instruments when using versions of the HST graph table dated `950224` (24 February 1995) and later. In earlier versions of the graph table, `nocostar` is the default. To explicitly include COSTAR, use the keyword `costar` anywhere within your `obsmode` string, e.g., "fos,red,4.3,g270h,costar". To exclude it, use the `nocostar` keyword. It is not necessary to explicitly exclude COSTAR for current generation instruments; it will be excluded by default.

In addition to the HST instruments, filters, and gratings, the **synphot** graph table also contains entries for various standard passbands that are not specific to HST. The ANS (Astronomical Netherlands Satellite), WF/PC-1 Baum, Cousins *RI,* Johnson *UBVRIJK*, Landolt *UBVRI*, Stromgren *uvby*, Walraven *VBLUW*, Sloan Digital Sky Survey *ugirz,* and the Bessell, Kitt Peak, and Steward Observatory *JHK* bands are included. The **modeinfo**

task will return a complete list of available non-HST passbands. Note that the Landolt (1983) *UBVRI* system is made up of the Johnson *UBV* and the Cousins *RI* passbands. Non-HST filters are specified using the name of the filter system, followed by the desired band name, e.g., "stromgren,u" or "cousins,i". If the name of the filter system is omitted for any of the common *UBVRIJHK* filters, the defaults are Johnson *UBV*, Cousins *RI*, and Bessell *JHK*.

The **synphot** tasks will also accept function expressions in the `obsmode` parameter. The available passband function expressions are shown in Table 3.4.

Table 3.4: Passband Functions

OBSMODE	Passband
`filename`	Name of an ASCII text file or STSDAS table containing columns of wavelength and throughput values.
`@filename`	Name of an ASCII text file containing one string of passband commands per line.
`band(obsmode)`	Constructs a passband associated with the specified obsmode
`thru(filename)`	Used when the filename might be interpreted as a number or contain arithmetic operators or be interpreted as a spectrum. Otherwise just use the filename.
`box(mu,width)`	Rectangular window centered on wavelength *mu* with width *width*; both arguments in Angstroms.
`em(mu,fwhm,totflux,units)`	Emission line centered on wavelength *mu* with a Gaussian profile, full width half maximum *fwhm*, and total flux *totflux* specified in *unit*s
`gauss(mu,fwhm)`	Gaussian with mean wavelength *mu* and full width half maximum *fwhm*; both arguments in Angstroms.
`lgauss(mu,fwhm)`	Gaussian in log wavelength space.
`poly(mu,fwhm,a1,a2,...)`	A passband that is a function of Legendre polynomials. The value of this function is (1+*P*) if *P* is positive and exp(*P*) if *P* is negative. *P* is the sum of up to ten Legendre polynomials with the specified coefficients. The independent variable in the polynomial is *(wave - mu) / fwhm*, where *wave* is the wavelength.
`tilt(band,a1,a2,...)`	The same as poly (above), but values of *mu* and *fwhm* are calculated from the passband *band* and the passband is implicitly multiplied by the tilt function.

More than one function expression can be included in an `obsmode`. For example, the following command will calculate a Gaussian, multiply it by a third-order polynomial, and write the results to a table called `out.fits`.

```
sy> calcband "tilt(gauss(5000,1000),0,1,3)" out.fits
```

3.2.3 Spectrum

The `spectrum` parameter defines the spectrum to be created or operated upon for tasks such as **calcphot**, **countrate**, **calcspec**, **plspec**, and **plratio**. Spectra can be read in from disk files or calculated from analytic models. A powerful spectrum calculator allows you to construct

complicated composite spectra, apply or remove extinction, renormalize to a desired magnitude or flux, add and subtract spectra, and multiply spectra by passbands or constants.

Filename specifications

When reading a spectrum from a file, you have a choice of using ASCII text files, FITS tables, or STSDAS tables. When using FITS or STSDAS tables, **synphot** looks for the column names "WAVELENGTH" and "FLUX". If column units are not specified, units of Angstroms for wavelengths and units of "flam" (see "Form" on page 21) for fluxes are assumed. When using a text table, the wavelength and flux values must be in the first and second columns, respectively. Since text tables do not have column units definitions, wavelengths must be in units of Angstroms and fluxes in units of "flam".

To process a series of spectra in sequence, set the `spectrum` parameter to "@*filename*", where *filename* is the name of an ASCII file containing the list of desired spectra, each one specified on a separate line of the file.

When using the @*filename* syntax, the task will generate the wavelength table to be used based on the **first** spectrum only, and will use this wavelength set for all calculations.

This will frequently provide different results than running the task for each spectrum in the sequence independently. If this is undesirable, one may either specify the wavelength table to be used for all calculations, or use the Python looping syntax (see"Replacing the "@filename" syntax" on page 9) to use the optimal set for each spectrum.

Functional forms

Table 3.5 lists the functional forms that can be used to generate synthetic spectra and to manipulate spectra read from a file.

Blank spaces in a spectrum expression are not significant, except for the division operator, which must be surrounded by blanks so that it is not mistaken for part of a directory path name.

Note that the `pl` function produces a power law spectrum in the units specified by the third argument. The exponent of the power law may change when converted to `photlam` space, which is the native form for all **synphot** calculations.

The `ebmv` operator applies or removes extinction effects using the interstellar extinction curve from Seaton (1979), which is based on an adopted value of $R = A_V / E(B–V) = 3.20$. For wavelengths between 1000 and 3704 Å, the analytic fits given in Seaton's Table 2 are used to compute

Table 3.5: Spectrum Functions

SPECTRUM	Function
`filename`	Name of an ASCII text file, FITS table, or STSDAS table containing columns of wavelength and flux values.
`@filename`	Name of an ASCII file containing one string of spectrum commands per line.
`spec(filename)`	Used when the filename might be interpreted as a number.
`unit(val,form)`	Creates a constant spectrum of value *val* in units of *form* (`fnu`, `flam`, `counts`, etc.)
`bb(temperature)`	Blackbody spectrum with specified temperature, in Kelvin. The flux of the spectrum is normalized to a star of solar radius at a distance of 1 kpc.
`pl(refval,expon,form)`	Power-law spectrum of the form $f \propto (\lambda / refval)^{expon}$, where *refval* is in Angstroms. The spectrum is normalized to a flux of 1 in *form* units at *refval*.
`hi(temp,colden)`	Continuum emission spectrum of an LTE slab of hydrogen at temperature *temp* (Kelvin) and column density *colden* (baryons/cm^{-2}). Values of *colden* less than 80 are assumed to be $\log_{10}$(*colden*). The spectrum is normalized the same as the blackbody.
`cat(catalog,key1,...)`	Read a spectrum selected from *catalog*, according to the search criteria *key1*, ..., *keyn*.
`icat(catalog,key1,...)`	Interpolate a spectrum from *catalog*, selected by the search criteria *key1*, ..., *keyn*.
`grid(filename,seq)`	Interpolate between two spectra in a list of spectra. The list is named by *filename* and the interpolation position within the list is specified by *seq*. For example, "3.5" means interpolate midway between the third and fourth spectra.
`rn(spec,obsmode,value,form)`	Renormalize the spectrum *spec* to *value* with units *form* over the *obsmode* passband. The evaluator computes the integral of the spectrum over the specified passband and rescales the spectrum by the appropriate factor, forcing this integral to have the requested renormalized value.
`z(spectrum,z)`	Redshift a *spectrum* by the amount *z*.
`ebmv(val)`	Applies an extinction of *val E(B – V)* to a spectrum using the galactic ISM reddening law of Seaton (1979).
`embvx(val,law)`	Applies an extinction to the spectrum using one of a number of available reddening laws.
`+ -`	Add, subtract spectra with automatic unit conversion, if needed.
`* /`	Multiply, divide by a passband (`obsmode`), constant, or `ebmv`, `ebmvx` functions.

the extinction, while for wavelengths between 3704 and 10000 Å, the extinction is computed by linear interpolation (in 1/λ vs. magnitudes) within the tabulated values given in Seaton's Table 3. For wavelengths greater than 10000 Å, the extinction is computed via a linear extrapolation of the tabulated optical values.

The extended reddening function, `ebmvx`, supports a number of different reddening laws. The second argument selects the type of reddening law used to compute the extinction. The task supports three galactic reddening laws (`gal1` to `gal3`) and one law each for the Small Magellanic Cloud (`smc`), Large Magellanic Cloud (`lmc`), and

extra-galactic objects (`xgal`). The reddening law used in the `ebmv` function is the same as `gal1`. The laws are derived from the papers cited in Table 3.6.

Table 3.6: Reddening Laws Used by ebmvx

Reddening Law	Citation
`gal1`	Seaton (1979) *MNRAS*, vol. 187, p. 75
`gal2`	Savage & Mathis (1979) *ARA&A*, vol. 17, pp. 73-111
`gal3`	Cardelli, Clayton & Mathis (1989) *ApJ* vol. 345, pp.245-256
`smc`	Prevot, et al. (1984) *A&A* vol. 132, pp. 389-392
`lmc`	Howarth (1983) *MNRAS*, vol. 203, p. 301
`xgal`	Calzetti, Kinney & Storchi- Bergmann (1994) *ApJ*, vol. 429, p. 582

Note that dividing a spectrum by `ebmv(val)` is equivalent to multiplying the spectrum by `ebmv(-val)` and has the effect of de-reddening the spectrum.

Here are two examples that illustrate several of the spectrum expression features. The first command will calculate a blackbody spectrum of 5000 K and renormalize it to 100 counts in the WFPC2 F555W passband. The output is written to the table `rnbb.fits`:

```
sy> calcspec \
>>> "rn(bb(5000),band(wfpc2,f555w),100,counts)" rnbb.fits
```

The next command creates a power-law spectrum with a slope of ν^{-2}, normalized to 10^{-14} ergs/s/cm^2/Å (f_λ units) in the Johnson *V* passband, and applies Galactic extinction of $E(B–V) = 0.1$. The output spectrum is written to the table `pl.fits` in units of f_ν (`fnu`):

```
sy> calcspec \
>>> "rn(pl(5500,-2,fnu),band(v),1.e-14,flam)*ebmv(0.1)" \
>>> pl.fits form=fnu
```

3.2.4 Form

The `form` parameter specifies the units of the output spectrum or plot. The **synphot** package supports the forms listed in Table 3.7.

Input and output spectra may be in any of these forms. The output form can differ from that of the input data; conversion is automatic. If you specify an inappropriate form, **synphot** will display a list of valid forms to choose from.

Table 3.7: Forms

FORM	Output Units
`fnu`	ergs / s / cm^2 / Hz
`flam`	ergs / s / cm^2 / Å
`photnu`	photons / s / cm^2 / Hz
`photlam`	photons / s / cm^2 / Å
`counts`	detected counts / s
`abmag`	–2.5 log (`FNU`) – 48.60
`stmag`	–2.5 log (`FLAM`) – 21.10
`obmag`	–2.5 log (`COUNTS`)
`vegamag`	–2.5 log (f / f(`VEGA`))
`jy`	10^{-23} ergs / s / cm^2 / Hz
`mjy`	10^{-26} ergs / s / cm^2 / Hz

All spectra are operated on internally in units of `photlam`.

The following comments should clarify the meanings of the `vegamag`, `abmag`, `stmag`, and `counts` options; the others are self-evident.

The `vegamag` option, which is defined by setting the magnitude of Vega to zero in all bands, offers a reasonable approximation to many of the conventional photometric systems that use the spectrum of Vega to define magnitude zero in one or more passbands. In broadband photometry, the relevant passband integral is calculated first for the source spectrum and then again for the spectrum of Vega, and the ratio of the two results is converted to a magnitude. `vegamag` would not be a scientifically meaningful option to use for spectrophotometry. The adopted Vega spectrum can be found in the ASCII file `stsdas$lib/synphot/vega.dat`, and is defined over a wavelength range of 900 Å to 300 μm.

Oke's AB_ν magnitudes (`abmag`) are based on a constant flux per unit frequency (see Oke 1974), and the analogous magnitude based on a constant flux density per unit wavelength is the `stmag` (Space Telescope magnitude) system. The `abmag` and `stmag` options are appropriate for either spectrophotometry or photometry. The zeropoint values of 48.60 and 21.10 are chosen for convenience so that Vega has AB_ν and ST_λ magnitudes close to 0 in the Johnson *V* passband (see Figure 3.1). Because the `abmag` and `stmag` systems are defined such that they result in constant magnitudes for spectra having constant flux per unit frequency

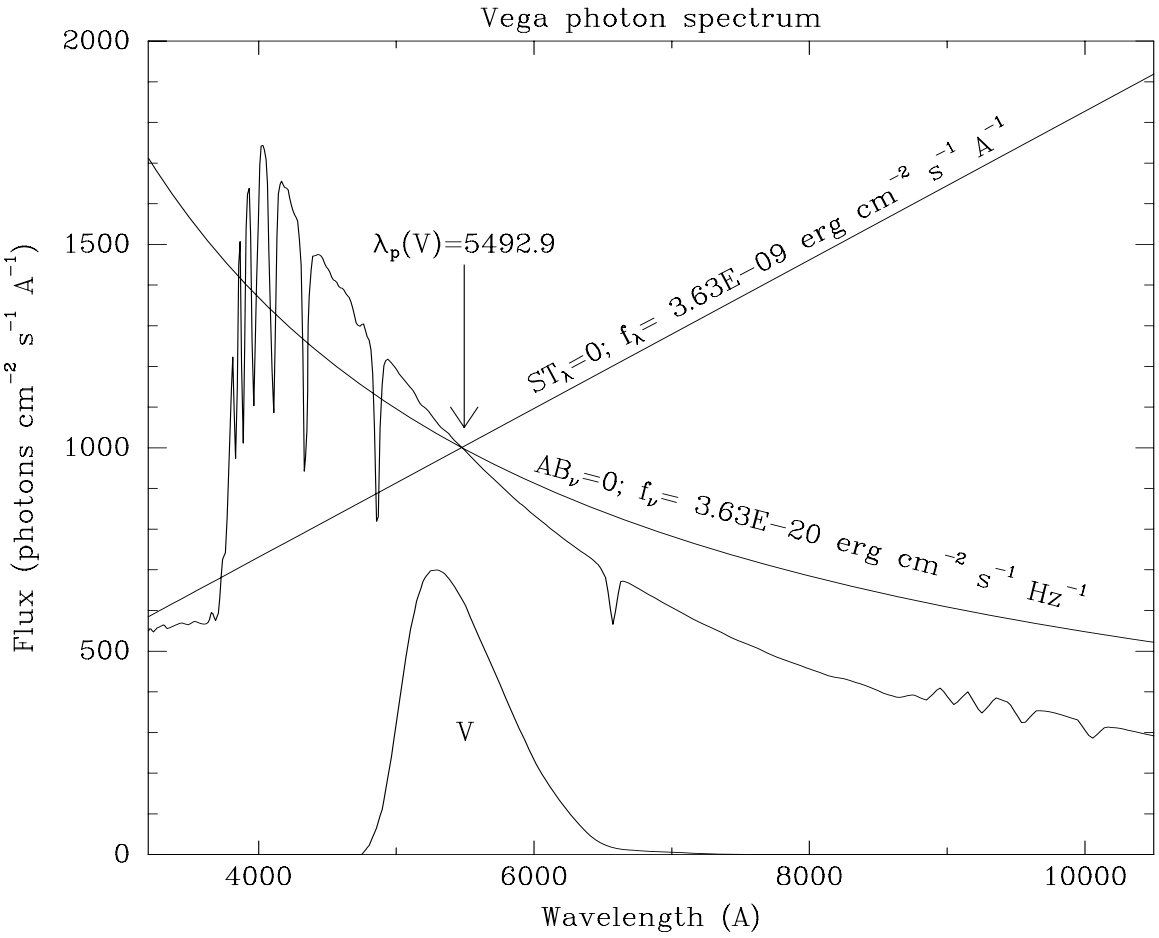
and wavelength, respectively, they will *not* provide magnitudes on a conventional system, such as *UBVRI*, without first deriving an appropriate transformation onto the desired standard system.

Because the adopted throughput functions for standard filter systems such as *UBVRI* are approximations to the complete instrumental passbands that were used to define these photometric systems, **synphot** results using the *UBVRI* passbands will only be accurate to a level of about 5% relative to the standard system, even when using `form=vegamag`. Results are particularly unreliable for the *U* filter, where the earth's atmosphere determines the blue edge of the passband in the standard ground-based system.

The `counts` option is used to predict detected count rates. When this `form` option is selected, tasks such as **countrate**, **calcspec** and **plspec** will calculate the predicted number of detected counts per second per wavelength interval within the passband, and **countrate** will also report the total number of detected counts integrated over the passband.

There are two important things to remember concerning spectra produced in units of `counts`. First, the number of counts per channel will depend directly on the width (in wavelength space) of the channels in the wavelength grid that is used (see below). All spectra are operated on internally in units of `photlam`. When output units of `counts` are requested, the `photlam` values are multiplied by the collecting area of the telescope and by the width (in Angstroms) of each channel in the wavelength grid. Therefore, in order to accurately predict the number of counts per channel for a spectroscopic instrument, it is necessary to use a wavelength grid that provides a good match to the dispersion properties of the particular instrument mode. The **genwave** task can be used to produce custom wavelength grids for this purpose, or one of the existing wavelength tables in the STSDAS directory `synphot$data` can be used (see below). The **countrate** task automatically uses these tables for STIS, FOS, HRS and NICMOS grism simulations, and for ACS grism/prisms.

Figure 3.1: Standard photometric systems illustrated.



Standard photometric systems generally use the spectrum of Vega to define magnitude zero. The spectrophotometric magnitudes AB_ν and ST_λ refer instead to spectra of constant f_ν and f_λ respectively. Magnitude zero in both systems is defined to be the mean flux density of Vega in the Johnson *V* passband. Thus all three of the spectra shown here produce the same count rate in the Johnson *V* passband. The pivot wavelength of Johnson *V* is defined to be the crossing point of the $AB_\nu = 0$ and $ST_\lambda = 0$ spectra.

Second, the form `counts` refers to actual detector counts for the FOC, FOS, HRS, and HSP instruments, while for the WF/PC-1, WFPC2, NICMOS, ACS, and STIS instruments, `counts` refers to *electrons*. In order to obtain counts in units of data numbers (DNs) for these instruments, include the keyword "dn" in the `obsmode` string for WF/PC-1 and NICMOS simulations, either of the keywords "a2d7" or "a2d15" for the WFPC2, and any of the keywords "a2d1", "a2d2", "a2d4", or "a2d8" for STIS (see the **modeinfo** task for complete lists of HST instrument keywords and their function). These keywords invoke the appropriate electron-to-DN conversion based on the A-to-D gain settings of the instruments. ACS WFC and HRC detectors are *always* in electrons.

3.3 Wavelength Table

Another parameter used by many **synphot** tasks is the wavelength table parameter (usually referenced as parameter `wavetab`). This parameter is used to specify the name of a file containing a list of wavelength values that determine the wavelength grid to be used in the calculations and plotting. The **genwave** task can be used to generate simple wavelength tables. If the chosen file is an STSDAS or FITS table, it should contain a column named "WAVELENGTH". The wavelength values may be in any of the units listed in Table 3.8 as long as the type of units is specified in the column units table keyword. If the units are not specified in a keyword, Angstroms is assumed. If it is an ASCII text file, the wavelength values must be in the first column and must be in units of Angstroms. In either case, the wavelength values must be in a monotonically increasing or decreasing order.

Since **synphot** works in single-precision arithmetic, care should be taken that the wavelength intervals specified are not so small as to produce an apparent duplicate entry. This violates the monotonicity constraint and will cause the task to fail.

There is a set of ASCII wavelength tables available for use with the HST

Table 3.8: Wavelength Units

Unit	in SI	Unit	in SI	Unit	in SI
angstroms	10^{-10} m	hertz	1 Hz	ev	1 eV
nanometers	10^{-9} m	kilohertz	10^{3} Hz	kev	10^{3} eV
microns	10^{-6} m	megahertz	10^{6} Hz	mev	10^{6} eV
millimeters	10^{-3} m	gigahertz	10^{9} Hz		
centimeters	10^{-2} m				
meters	1 m				

spectroscopic instruments in the STSDAS `synphot$data` directory area. This directory contains separate wavelength tables for each of the STIS, HRS and FOS grating modes and echelle orders, as well as the ACS and NICMOS grisms. The wavelength grid contained in each table covers the range and resolution that is appropriate for each instrument grating as used in a standard observational mode for that instrument.

You may set the `wavetab` parameter to "none" or leave it blank, in which case the task will use a default wavelength grid. The default grid consists of 10000 points covering the wavelength range where the calculated passband or spectrum is non-zero. The wavelengths are spaced logarithmically over this range, such that:

$$\log_{10}(\lambda_i) = \log_{10}(\lambda_{min}) + (i - 1) \times \Delta$$

where:

- λ_i is the i^{th} wavelength value
- $\Delta = (\log_{10}(\lambda max) - \log_{10}(\lambda min)) / (N - 1)$
- λmin = wavelength of first non-zero throughput or flux
- λmax = wavelength of last non-zero throughput or flux
- $N = 10000$

If more than one passband or spectrum is specified via the "*@filename*" syntax for the `obsmode` or `spectrum` parameters, the range of the default wavelength grid is computed from the **first** passband or spectrum in the list. Therefore if the wavelength ranges of the passbands or spectra differ significantly, it is best to specify a suitable wavelength table that covers the complete range of all items in the list.

3.4 Variable Substitution (VZERO)

Several **synphot** tasks support a method of variable substitution (often referred to as the "VZERO syntax") that can be used to run a series of simulations while varying a parameter over a stepped range.

This can now be accomplished more simply in PyRAF using the Python loop syntax (see Section 2.3.2, "Replacing the "vzero" syntax," on page 10).

If present, the `vzero` parameter is a list of values that are substituted for variable zero (`$0`) wherever it appears in the input expression. Each value in the list is substituted in turn. The values must be real numbers. Using `vzero` is equivalent to placing the input expression several times in a file, with each expression containing one of the values in the list. The list may contain single values or ranges. The endpoints of a range are separated by a dash. An optional step size may follow the range, preceded by the

letter “x”. If the step size is not given, it defaults to 1 or –1, depending on the order of the endpoints. Table 3.9 gives several examples of valid `vzero` lists.

Table 3.9: vzero Examples

vzero	Resulting list of values	Description
`".1,.2,.3,.4"`	0.1, 0.2, 0.3, 0.4	A list of values
`".1-.4x.1"`	0.1, 0.2, 0.3, 0.4	The same list expressed as a range
`"-1 - -4"`	–1, –2, –3, –4	A range with an implicit step size of –1
`"1-9,10-20x2"`	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 12, 14, 16, 18, 20	A list with more than one range

Variable zero ($0) is used by several **synphot** tasks. Variables 1 through 9 ($1 - $9) are also used by the fitting tasks.

3.5 Reference Data

Several common parameters are grouped in the `refdata` parameter set (pset). This pset contains the three parameters shown in Table 3.10

Table 3.10: Parameters in refdata Pset

Parameter	Default Value	Description
`area`	45238.93416	Telescope area in cm^2
`grtbl`	“mtab$*_tmg.fits”	Instrument graph table
`cmptbl`	“mtab$*_tmc.fits”	Instrument component table

The default value for `area` is the total collecting area of the HST and was computed from the 120 centimeter nominal radius of the HST entrance aperture (the obscuration factor 0.86 due to the secondary mirror is taken into account in the OTA throughput). This value should not be changed unless another telescope is being used. The telescope area is used to convert from counts per square centimeter to total counts. The graph table and component table names specify which tables are to be used by the **synphot** tasks. The table names may contain the wildcard character “`*`”, in which case the most recent table that matches the name template will be used. The default values for these parameters select the most current graph and component lookup tables installed in the CDBS.

The graph table describes all the possible light paths through the telescope and instruments. The component lookup table associates each

segment of the light paths listed in the graph table with the specific files containing the throughput data for each individual segment. You may use your own versions of the graph and component tables, if you wish. To do this, you would typically copy the default files to your own directory, edit them, and change the names in the `refdata` parameter set.

3.6 Table formats

Because **synphot** uses the TABLES package for all its table i/o, all **synphot** tasks presently support the use of ASCII tables, FITS tables, or STSDAS binary tables (`.tab` files).

However, we are planning to deprecate the `.tab` format.

While it will continue to be supported for at least two more releases or two years (whichever is longer), support will be ended at some point after that.

In this manual, we have retained the use of `.tab` files in some of our examples for historical reasons, but FITS files are strongly recommended as the table format for all current and future **synphot** use.

3.7 Using EPAR to edit task parameters

The EPAR interface makes it easy to enter task parameters, especially for the tasks that have long and complicated parameter lists. However, any values you enter using EPAR will be saved as new default values for the task. This can produce surprising results when you run the task again, especially for the case of "hidden" parameters for which you are not prompted in interactive mode.

To restore the default task parameters for a task, type `unlearn` *`taskname`*.

3.8 About the Examples in This Manual

Because **synphot** is a data-driven system, the results obtained by executing a task will change if the underlying data files (throughput files, graph table, etc) have been changed. Therefore, the actual results you

might obtain by executing the examples may differ from the results printed in the manual.

The examples included in this manual are intended only to illustrate the layout and general appearance of the task output.

You can check the date on the most recent reference files, and update your local copy of them, from the **synphot** data download webpage, ftp://ftp.stsci.edu/pub/software/stsdas/refdata/synphot/ .

CHAPTER 4:

Cookbook

In this chapter. . .

4.1 Bandpasses / 31
4.2 Spectra / 34
4.3 Photometry / 37

This chapter gives step by step examples of how some of the **Synphot** package tasks are commonly used to answer questions related to HST observing and instrument efficiencies and performance. Detailed descriptions of the tasks used in these examples can be found in Chapter 5.

4.1 Bandpasses

One of the simplest uses of the **synphot** tasks is to examine the characteristics of instrumental bandpasses. The three main tasks that work specifically with bandpasses are **bandpar**, **calcband**, and **plband**. The **bandpar** task computes and displays numerical parameters of a bandpass, while **calcband** and **plband** calculate and plot, respectively, functions of throughput vs. wavelength for a bandpass. All three tasks are useful for examining individual bandpasses, as well as making comparisons of several bandpasses.

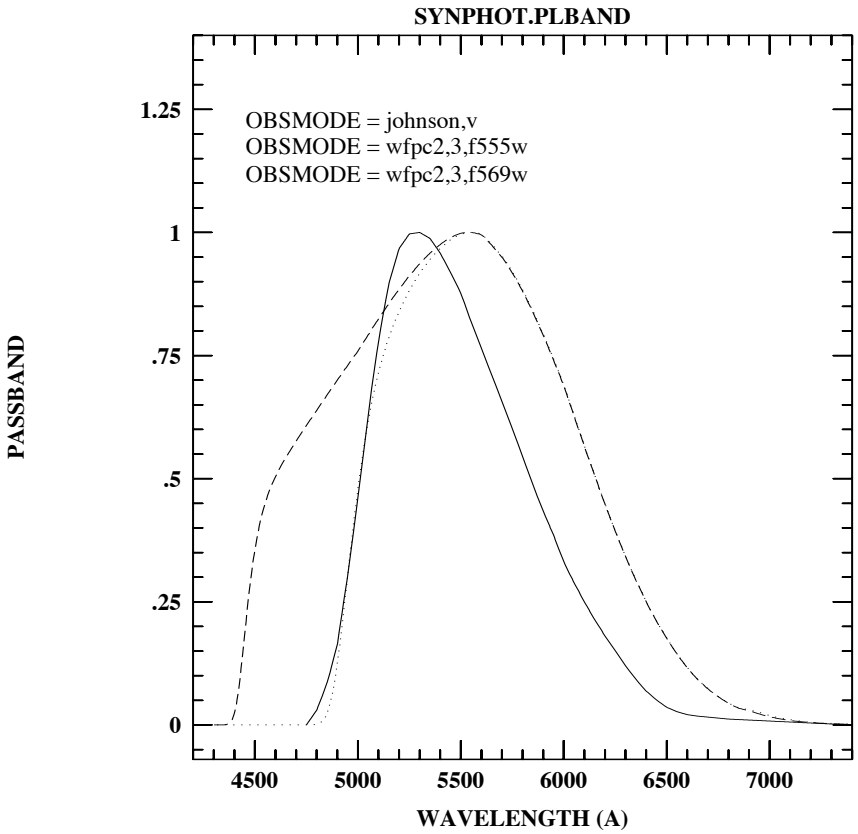
The only required user input to both **bandpar** and **plband** is the specification of the desired bandpass(es), which is done using the `obsmode` task parameter (see Section 3.2.2). The **calcband** task requires the `obsmode`, as well as the name of an output table in which the results will be stored.

For example, let's say you're interested in seeing which of the WFPC2 filters F555W and F569W is the closest match to the Johnson *V* bandpass. You can make a plot showing the relative throughput vs. wavelength for these three bandpasses using **plband** as follows:

```
sy> plband johnson,v
sy> plband wfpc2,3,f555w norm+ app+ ltype=dashed
sy> plband wfpc2,3,f569w norm+ app+ ltype=dotted
```

The resulting plot is shown in Figure 4.1. In this example we are plotting the response of the WFPC2 filters when used with CCD chip 3 of the camera. We have used the **plband** parameter `normalize` to set the peak transmission of all 3 bandpasses to a value of 1, the parameter `append` to overplot the WFPC2 bandpasses onto the first plot of the Johnson *V* bandpass, and the `ltype` parameter to change the line type for the WFPC2 bandpass plots.

Figure 4.1: Plband Plot of Johnson V and WFPC2 Bandpasses



As you can see, the F569W bandpass is a closer match to Johnson *V*, especially on the blue edge. However, since the F569W bandpass is narrower than the F555W, it will have a lower overall throughput than the F555W. But how much lower? You can use the **bandpar** task to find out.

```
--> bandpar wfpc2,3,f555w photlist=qtlam
     # OBSMODE           QTLAM
 wfpc2,3,f555w        0.030121


--> bandpar wfpc2,3,f569w photlist=qtlam
     # OBSMODE           QTLAM
 wfpc2,3,f569w        0.023435
```

Comparing the values of QTLAM, the dimensionless efficiency, for the two bandpasses we see that the F569W has about 78% of the throughput of the F555W.

Now let's do a similar experiment in which we wish to find out the relative efficiencies of several of the NICMOS filter bandpasses. First, for convenience, we create a text file called `nic_filters.lis`, which contains the list of bandpasses (`obsmodes`) we're interested in (see Figure 4.2).

Figure 4.2: List of NICMOS Filters in File "nic_filters.lis"

```
nicmos,1,f110w
nicmos,1,f110m
nicmos,1,f160w
nicmos,1,f165m
nicmos,2,f110w
nicmos,2,f160w
nicmos,2,f165m
nicmos,2,f222m
nicmos,3,f110w
nicmos,3,f160w
nicmos,3,f222m
```

We then run **bandpar**, using the *@filename* notation to specify the name of the file containing our list of obsmodes, and ask for only the QTLAM photometric parameter to be calculated and printed.

```
sy> bandpar @nic_filters.lis phot=qtlam
```

Figure 4.3 shows the resulting output from **bandpar**.

Figure 4.3: Bandpar Results for a List of NICMOS Filters

```
      # OBSMODE           QTLAM
 nicmos,1,f110w       0.057824
 nicmos,1,f110m       0.019782
 nicmos,1,f160w       0.048279
 nicmos,1,f165m       0.023963
 nicmos,2,f110w       0.066781
 nicmos,2,f160w       0.053258
 nicmos,2,f165m       0.026823
 nicmos,2,f222m        0.01811
 nicmos,3,f110w       0.060011
 nicmos,3,f160w        0.05143
 nicmos,3,f222m       0.017694
```

4.2 Spectra

The simplest thing to do with spectra is to just plot them using the **plspec** task. This is handy, for example, when you're interested in exploring the contents of the available STScI spectral libraries, which are located in the STSDAS `crgrid$` directory areas (see Appendix B). Let's say you're interested in making some HST observations of spiral galaxies and you want to see what sort of template galaxy spectra are available for making **synphot** predictions of observed count rates. A good place to look would be in either the `crgridbc95$` (Bruzual-Charlot atlas) or `crgridkc96$` (Kinney-Calzetti atlas) directories.

To plot one of the spectra, use the **plspec** task with no `obsmode` specified (so that the spectrum won't get multiplied by a bandpass), the `spectrum` parameter set to the name of the table containing the spectral data (see Section 3.2.3), and the output `form` set to the units of your choice (see Section 3.2.4). The following example plots the spectrum of a type Sb galaxy from the Kinney-Calzetti atlas, in units of `flam` (ergs/s/cm^2/Å).

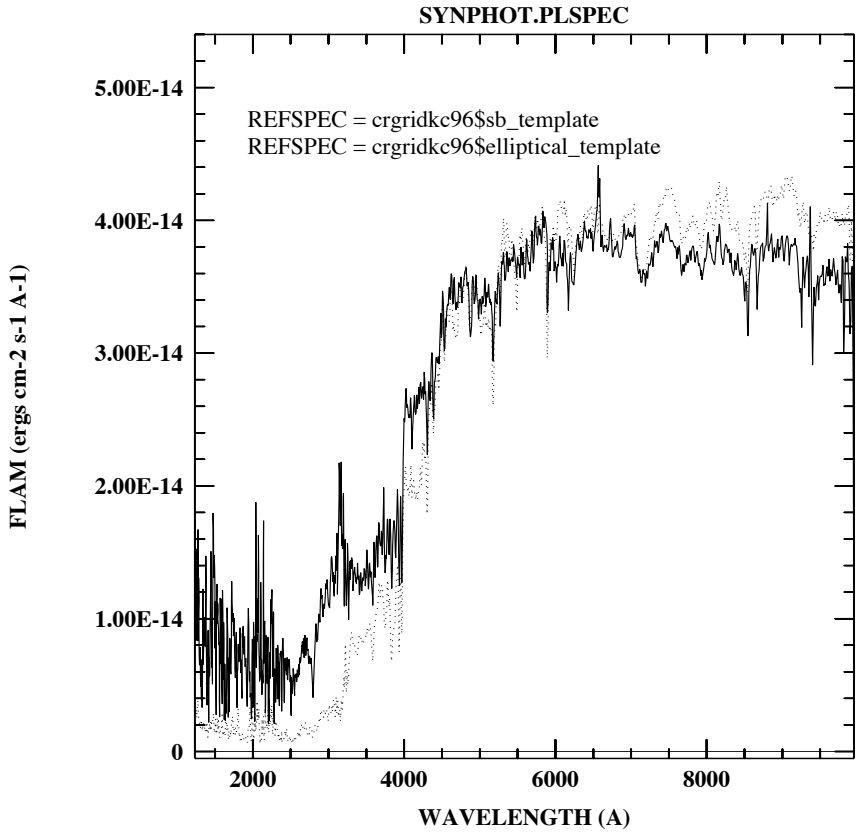
```
sy> plspec "" crgridkc96$sb_template.fits flam
```

How does this compare to the spectrum of a typical elliptical galaxy? You can find out by overplotting:

```
sy> plspec "" crgridkc96$elliptical_template.fits flam \
>>> app+ ltype=dotted
```

This uses the `append` parameter to overplot the spectra and the `ltype` parameter to change the line type of the second spectrum. The result is shown in Figure 4.4.

Figure 4.4: Plotting Spectra with plspec



Well, that's interesting, but what we'd really like to know is what will the spectrum of one of these targets look like when observed with an HST spectroscopic instrument? To create a predicted observed spectrum we could simply rerun **plspec** (or **calcspec)**, this time giving it an `obsmode` corresponding to one of the HST instrumental modes. However, because the detected count rate per channel in a spectroscopic mode depends directly on the width of each channel in wavelength space, we would have to be careful to first construct a wavelength set that matches the dispersion characteristics of the instrumental mode(s) we're interested in. It is much easier, however, to just use the **countrate** task, which automatically uses a library of existing wavelength sets appropriate for each HST spectroscopic mode.

The parameters for the **countrate** task are a little bit different than those of other **synphot** tasks, in that there is no `obsmode` parameter. Instead, the instrument mode is specified via a set of individual parameters for the instrument, detector, filter or grating, and aperture or slit. Let's say we want to simulate a STIS observation of the Sb-type galaxy we looked at earlier with **plspec**, using the STIS CCD detector, its low-resolution G430L grating, and the 50" x 0.5" entrance slit. We'll also assume that the galaxy

we want to observe has a *V* magnitude (within the slit area) of 18 and we'll try an exposure time of 500 seconds. Figure 4.5 shows how to set the **countrate** task parameters for this simulation.

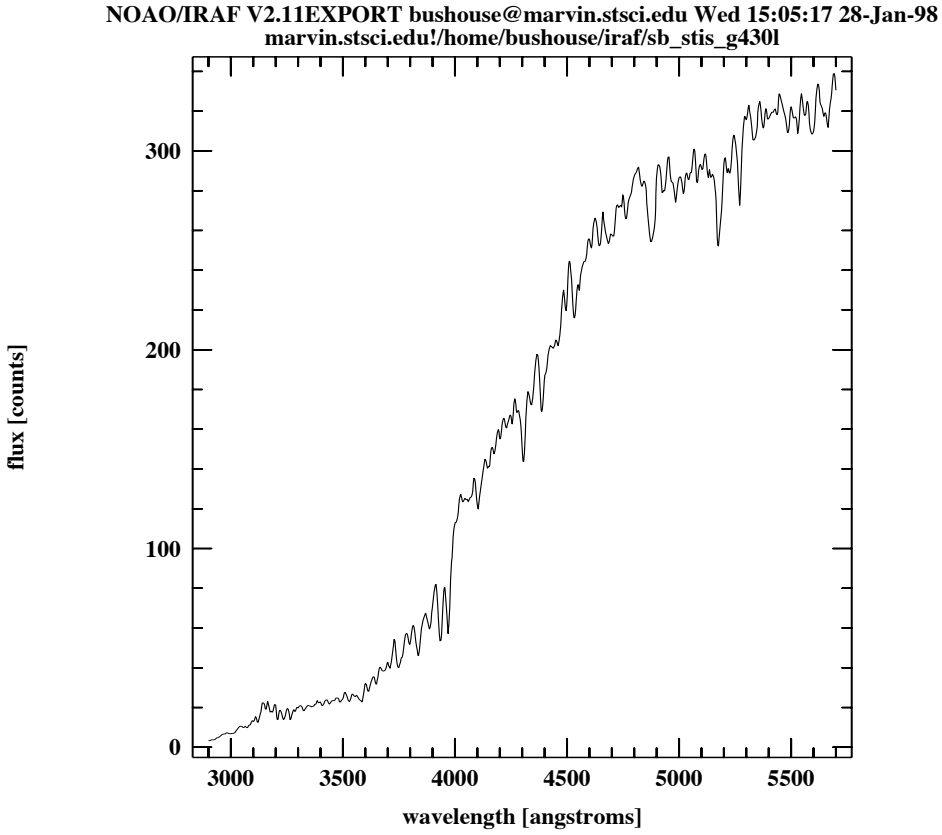
Figure 4.5: countrate Parameters for STIS CCD Observation

```
    spectrum = "crgridkc96$sb_template.fits"
   magnitude = "18 v"
  instrument = stis
    detector = ccd
   spec_elem = g430l
    aperture =  52x0.5
     cenwave = INDEF
     exptime = 500.
   reddening = 0.0
      redlaw = gal1
      output = sb_stis_g430l.fits
        form = counts
     magform = vegamag
     wavecat = synphot$data/wavecat.dat
     refwave = INDEF
     verbose = yes
    flux_tot = INDEF
    flux_ref = INDEF
     refdata = " "
```

You can then use **plspec**, or any STSDAS plotting task that can read tables, to plot the observed spectrum stored in the file "sb_stis_g430l.fits". Figure 4.6 shows the spectrum as plotted using the STSDAS **sgraph** task.

```
sy> sgraph "sb_stis_g430l wavelength flux"
```

Figure 4.6: Predicted STIS CCD Spectrum Produced by countrate



4.3 Photometry

The **calcphot** task is very useful for making quick estimates of the total number of counts that will be detected in an HST imaging observation of a particular source, normalized to an aperture of some (instrument-dependent) size. For example, let's again use that spectrum of an Sb galaxy to calculate the total counts that would be detected when observed with the WFPC2, in chip 3, using filter F439W (the WFPC2 analog to Johnson *B*).

```
sy> calcphot wfpc2,3,f439w \
>>> "rn(crgridkc96$sb_template.fits,band(v),18,vegamag)" \
>>> counts
```

Here we've used the `rn` (renormalize) function in the `spectrum` parameter to renormalize the Sb galaxy spectrum to a *V*-band magnitude of 18. We've also asked for the answer to be computed in units of counts, which for the WFPC2 is equivalent to electrons. It's also possible to get the answer in units of DNs (data numbers). To do this for the WFPC2, we

would need to include the additional keyword "a2d7" in the `obsmode` string (i.e., "wfpc2,3,f439w,a2d7"), which applies the appropriate electron-to-DN conversion ratio for the WFPC2 gain setting of 7. The **calcphot** results are shown in Figure 4.7.

Figure 4.7: Calcphot Results for a WFPC2 F439W Observation

```
Mode = band(wfpc2,3,f439w)
Pivot       Equiv Gaussian
Wavelength  FWHM
4311.664    476.1779    band(wfpc2,3,f439w)
Spectrum:   rn(crgridkc96$sb_template.fits,band(v),18,vegamag)
VZERO       (COUNTS s^-1 hstarea^-1)
 0.         42.3929
```

So this galaxy produces a total countrate of 42.4 e^-/sec, which means we'll need an exposure time of at least 235 seconds in order to approach an accuracy of 1% in measuring its total light.

Another question an HST observer might ask is "How faint can I go in an exposure time of *t* seconds if I want *x* percent photometry?" To answer this, you could either run **calcphot** over and over again the way we just did in the last example, picking different renormalization magnitudes for your source until you get the countrate that corresponds to your desired S/N, or you could run the equivalent simulation "backwards".

For example, if you're interested in observing red stars with the NICMOS, you can pick a suitable stellar spectrum from the `crgridbpgs$` area (Bruzual-Persson-Gunn-Stryker atlas), and then run **calcphot** as follows.

```
sy> calcphot h \
>>>"rn(crgridbpgs$bpgs_147.fits,band(nicmos,2,f160w),
10,counts)"\
>>> vegamag
```

Here we've picked star number 147 from the BPGS atlas, which is a K5 giant. We want to observe with NICMOS camera 2 and the F160W filter and want to see how faint we can go in 100 seconds and get 3% photometry. To obtain this measurement accuracy we need a total of 1000 electrons detected from the source, so for an exposure time of 100 seconds we need a countrate of 10 e^-/sec. We've again used the `rn` function to renormalize the spectrum, but this time we've renormalized it to our desired countrate of 10 in the "nicmos,2,f160w" band. Once it's been renormalized, **calcphot** then computes the *H* magnitude of the spectrum (in units of vegamag, since the *JHK* filter system uses Vega as its zeropoint). The results from **calcphot** are shown in Figure 4.8, where we see that we can reach an *H* magnitude of 21.4.

Figure 4.8: Calcphot Results for a NICMOS F160W Observation

```
Mode = band(h)
Pivot       Equiv Gaussian
Wavelength  FWHM
16448.      2040.596    band(h)
Spectrum:   rn(crgridbpgs$bpgs_147.fits,band(nicmos,2,f160w),10,counts)
VZERO       VEGAMAG     Mode: band(h)
0.          21.3681
```

The **calcphot** task is also useful for making photometric measurements of existing spectra, either in HST instrumental bands or standard ground-based filter systems. For example, let's say you'd like to demonstrate for some students the importance of the 4000 Å break on the *U–B* colors of redshifted galaxy spectra. First, you might want to produce a plot showing the *U* and *B* bandpasses relative to some redshifted spectra. The **plspec** and **plband** commands shown in Figure 4.9 will produce the plot shown in Figure 4.10. Note that the bandpass plots are multiplied by a scale factor to place them in the same range of data values as the spectra.

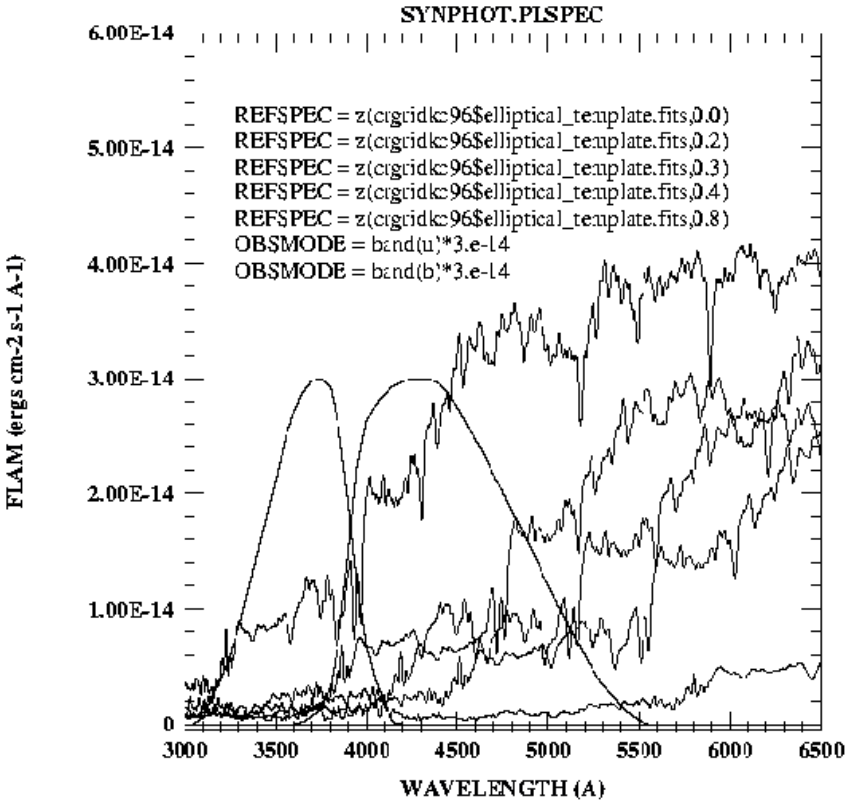
The commands below use the iraf cl syntax for specifying task parameters without invoking the task. This is sometimes useful for setting plotting ranges, as in this case.

In PyRAF, you must use the syntax `iraf.plspec.left = 3000` to specify task parameters in this way.

Figure 4.9: Commands for Plotting Redshifted Spectra and UB Bands

```
plspec.top = 6e-14
plspec.left = 3000
plspec.right= 6500
plspec "" "z(crgridkc96$elliptical_template.fits,0.0)" flam
plspec "" "z(crgridkc96$elliptical_template.fits,0.2)" flam app+
plspec "" "z(crgridkc96$elliptical_template.fits,0.3)" flam app+
plspec "" "z(crgridkc96$elliptical_template.fits,0.4)" flam app+
plspec "" "z(crgridkc96$elliptical_template.fits,0.8)" flam app+
plband "band(u)*3.e-14" app+
plband "band(b)*3.e-14" app+
```

Figure 4.10: Plots of Redshifted Spectra and UB Bandpasses



Now we run **calcphot**, having it compute the *U–B* colors of these spectra. We'll use the **synphot** expression variable `$0` for the redshift value in the `z` function, and then specify the range of redshifts that we want in the task parameter `vzero`. This tells **calcphot** to automatically run the calculation several times, each time substituting a different value for the redshift. The complete **calcphot** command and results are shown in Figure 4.11.

Figure 4.11: Using calcphot to Compute U-B Colors of Redshifted Spectra

```
--> calcphot "band(u)-band(b)" "z(crgridkc96$elliptical_template.fits,$0)"
vegamag vzero="0-0.8x0.2"
Mode = band(u) - band(b)
   Pivot       Equiv Gaussian
 Wavelength          FWHM
  3646.236         484.6001    band(u)
  4433.497         831.1038    band(b)
Spectrum:  z(crgridkc96$elliptical_template.fits,$0)
    VZERO      VEGAMAG(band(u)) - VEGAMAG(band(b))
       0.            0.645421
      0.2             0.90348
      0.4            1.022621
      0.6           -0.10024
      0.8           -0.55384
```

CHAPTER 5:

Task Descriptions

In this chapter. . .

5.1 Bandpar / 42
5.2 Calcband / 44
5.3 Calcphot / 46
5.4 Calcspec / 50
5.5 Countrate / 54
5.6 Fitband / 59
5.7 Fitspec / 63
5.8 Fitgrid / 68
5.9 Genwave / 72
5.10 Grafcheck / 73
5.11 Graflist / 74
5.12 Grafpath / 75
5.13 Imspec / 76
5.14 Mkthru / 78
5.15 Modeinfo / 80
5.16 Obsmode / 81
5.17 Plband / 82
5.18 Plspec / 85
5.19 Plratio / 94
5.20 Pltrans / 98
5.21 Refdata / 101
5.22 Showfiles / 102
5.23 Thermback / 103

This chapter describes in detail how each task in the **synphot** package operates and what parameters each task uses.

5.1 Bandpar

The **bandpar** task computes various photometric parameters, as shown in Table 5.1, for a selected passband. The computed photometric parameters can be saved to a FITS or STSDAS table.

Table 5.1: Bandpar Photometric Parameters

Parameter	Description	Formula
uresp	Unit response; flux (in flam) that produces 1 count/second in the passband	$(hc)/(area \int P_\lambda \lambda d\lambda)$
pivwv	Pivot wavelength of passband	$\sqrt{\int (P_\lambda \lambda d\lambda)/\int (P_\lambda d\lambda/\lambda)}$
bandw	RMS band width	$\bar{\lambda}\frac{\sqrt{\int P_\lambda \ln(\lambda/\bar{\lambda})^2 d\lambda/\lambda}}{\sqrt{\int P_\lambda d\lambda/\lambda}}$
fwhm	Full width half maximum of an equivalent gaussian	$\sqrt{8\ln 2} \times bandw$
wpeak	Wavelength at peak throughput	$\lambda where(P = max((P_\lambda))$
tpeak	Peak throughput of passband	$max(P_\lambda)$
avgwv	Passband average wavelength	$\frac{\int P_\lambda \lambda \, d\lambda}{\int P_\lambda \, d\lambda}$
qtlam	Dimensionless efficiency	$\int (P_\lambda d\lambda)/\lambda$
equvw	Equivalent width of passband	$\int P_\lambda d\lambda$
rectw	Rectangular width of passband	$\int (P_\lambda d\lambda)/max(P_\lambda)$
emflx	Equivalent monochromatic flux	$uresp \cdot rectw \cdot \left(\frac{tpeak}{tlambda}\right)$
tlambda	Throughput at reference wavelength	$P(refwave)$
refwave	Reference wavelength for tlambda	$\int (P_\lambda \lambda d\lambda)/\int (P_\lambda d\lambda)$

In the table above, P_λ is the (dimensionless) passband throughput as a function of wavelength, *area* is the telescope collecting area, *h* and *c* are the usual physical constants, and $\bar{\lambda}$ is the mean wavelength of the passband defined in Schneider, Gunn, and Hoessel (1983) as:

$$\bar{\lambda} = \exp\left[\frac{\int P_\lambda \ln(\lambda) d\lambda/\lambda}{\int P_\lambda d\lambda/\lambda}\right]$$

This rather unusual definition has the property that the correspondingly defined mean frequency is just $c/\bar{\lambda}$. Unless specified by the user, the refer-

ence wavelength (refwave) used in the computation of tlambda and emflx is set to the average wavelength of the passband (avgwav).

The **bandpar** task parameters are shown in Figure 5.1, below.

Figure 5.1: Bandpar Parameters

```
obsmode   =          Instrument observation mode(s)
(output   = "none")  Output table name
(photlist = "all")   Photometric parameters to calculate
(refwave  = INDEF)   Wavelength used in computing EMFLX
(wavetab  = "")      Wavelength table name
(refdata  = "")      Reference data
```

The obsmode parameter may be any valid string of HST instrument mode keywords, passband functions—such as gauss, box, or poly—or the name of a table (text or FITS or STSDAS binary) containing wavelength and throughput data. A series of obsmode strings may be processed by putting them in a text file, one per line, and setting obsmode=@*filename*. See Section 3.2.2 for more details about all of the available specifications for obsmode, and "Filename specifications" on page 19 for important side effects of using the @filename syntax.

If the output parameter is null or set to "none", no output table will be created. However, output will always be sent to STDOUT (terminal). The output table will contain columns for each of the photometric quantities specified in the photlist parameter (see below). The output table will also contain the table header keywords GRFTABLE, CMPTABLE, and APERAREA as a record of the settings used in the computations. If more than one obsmode is specified using an input list file, then the output table will have separate rows containing the parameters computed for each passband.

The photlist parameter contains a comma separated list of the names of the photometric parameters to be printed. The value "all" prints all of them. Placing a "~" in front of the list causes all of the parameters *except* those in the list to be printed. The two auxiliary parameters refwave and tlambda are printed by default if emflx is printed, and not printed by default if emflx is not printed.

The refwave parameter specifies the reference wavelength to be used in the computation of emflx. If this parameter is set to INDEF, the average wavelength (avgwv) will be calculated as shown in Table 5.1 and used as the reference wavelength. The units of a user-specified reference wavelength must be Angstroms.

Null values for the wavetab and refdata parameters will cause the task to use a default wavelength grid (see Section 3.3) and the default reference data for the HST observatory (see Section 3.5).

5.1.1 Examples

The following command computes all available photometric parameters for the passband of the ACS Solar Blind Channel (SBC) with longpass filter f150lp. The output is not saved in a table.

```
sy> bandpar acs,sbc,f150lp
```

The next example calculates only the pivot wavelength and the rms bandwidth for each of the Johnson *UBV* passbands, and saves the results in the table `ubv.fits`. The input file `ubv.lis` contains the lines:

```
band(u)
band(b)
band(v)
```

```
sy> bandpar @ubv.lis output=ubv.fits phot=pivwv,bandw
```

The last example calculates the photometric parameters for the WFPC2 detector 2 and F439W filter, setting the reference wavelength to 4300 Å. The reference wavelength is not printed.

```
sy> bandpar wfpc2,2,f439w refwave=4300 phot=~refwave
```

5.2 Calcband

The **calcband** task will calculate a passband by combining existing passbands, certain functional forms such as a Gaussian or a rectangular window, or throughput data read in from a file. These data can optionally be multiplied by a Legendre polynomial to modify the passband shape. The task takes any valid `obsmode` command string as input and produces a FITS or STSDAS table with two columns of data, called WAVELENGTH and THROUGHPUT, as its output.

The task parameters are shown in Figure 5.2, below.

Figure 5.2: Calcband Parameters

```
    obsmode =            Instrument observation mode passband
    output  =            Output table name
(wavetab =  )            Wavelength table name
(refdata = )             Reference data
```

The `obsmode` parameter may be any valid string of HST instrument modes, passband functions—such as `gauss`, `box`, or `poly`—or the name of a file (FITS file, STSDAS table or plain ASCII text) containing columns of throughput data. A series of `obsmode` strings may be processed by storing them in a text file, one per line, and setting `obsmode=@`*`filename`*. See Section 3.2.2 for more details about all of

the available specifications for `obsmode`, and "Filename specifications" on page 19 for important side effects of using the `@filename` syntax.

If more than one passband is specified via a list file, then a separate "THROUGHPUT*n*" column is created in the output table for each of the *n* passbands listed in the file. The output table also contains the following header keywords (listed in Table 5.2), many of which are the same photometric quantities computed by the **bandpar** task. The header can be read using **thedit** or **catfits**.

Table 5.2: Calcband Output Keywords

Keyword	Description	Formula
`grftable`	Name of the instrument graph table	
`cmptable`	Name of the component lookup table	
`expr`	Value of the `obsmode` parameter	
`aperarea`	Telescope collecting area, in cm^2, used to compute `uresp`	
`zeropt`	Photometric zeropoint of the STMAG system	
`uresp`	Unit response; flux (in flam) that produces 1 count/second in the passband	$(hc)/(aperarea \int P_\lambda \lambda d\lambda)$
`pivwv`	Pivot wavelength of passband	$\sqrt{\int(P_\lambda \lambda d\lambda)/\int(P_\lambda d\lambda/\lambda)}$
`bandw`	RMS band width	$\bar{\lambda}\frac{\sqrt{\int P_\lambda \ln(\lambda/\bar{\lambda})^2 d\lambda/\lambda}}{\sqrt{\int P_\lambda d\lambda/\lambda}}$
`tpeak`	Peak throughput of passband	$max(P_\lambda)$
`equvw`	Equivalent width of passband	$\int P_\lambda d\lambda$
`rectw`	Rectangular width of passband	$\int(P_\lambda d\lambda)/max(P_\lambda)$
`emflx`	Equivalent monochromatic flux	$uresp \cdot rectw \cdot \left(\frac{tpeak}{tlambda}\right)$

If more than one passband is specified via a list file, the last eight header keywords are repeated, once for each passband, with the names EXPR*n*, URESP*n*, PIVWV*n*, BANDW*n*, TPEAK*n*, EQUVW*n*, RECTW*n*, and EMFLX$_n$ for the n^{th} passband.

Null values for the `wavetab` and `refdata` parameters will cause the task to use a default wavelength grid (See "Wavelength Table" on page 25.) and the default reference data for the HST observatory (see Section 3.5). The default wavelength grid covers the wavelength range where the passband throughput is non-zero. Wavelengths are spaced logarithmically over this range. If there is more than one passband specified, the range of the default grid is computed based on the first passband. Therefore, if the

wavelength ranges of the passbands differ significantly, a suitable wavelength table that covers the range of all the passbands should be constructed using the **genwave** task, and supplied as input via the `wavetab` parameter.

5.2.1 Examples

The following command shows how to calculate a Gaussian centered at 4800 Å, multiply it by a first-order Legendre polynomial (which, by the way, produces a close match to the Johnson *V* passband), and write the results to a table called `gauss_tilt.fits`.

```
sy> calcband "tilt(gauss(4800,1300),10)" gauss_tilt.fits
```

The next example shows how to evaluate the total telescope plus instrumental throughput for an HST observation using the STIS FUVMAMA detector, and the e140h grating. The resulting throughput data are stored in the table `stis_thpt.fits`.

```
sy> calcband stis,e140h,fuvmama stis_thpt.fits
```

5.3 Calcphot

The **calcphot** task is useful for calculating the integrated counts or flux in a given passband for a particular spectrum. Passband information, such as pivot wavelength, FWHM, and rms bandwidth, can also be calculated. The output may be saved to a FITS or STSDAS table, if desired.

Task parameters include the observation mode and spectrum parameters that were discussed for the **calcband** and **calcspec** tasks, and discussed in detail in Section 3.2.2 and Section 3.2.3. The full list of task parameters is shown in Figure 5.3.

Figure 5.3: Calcphot Parameters

```
     obsmode =                Instrument observation mode
    spectrum =                Synthetic spectrum to calculate
        form = photlam        Form for output data
      (func = effstim)        Function of output data
      (vzero = )              List of values for variable zero
     (output = none)          Output table name
     (append = no)            Append to existing table?
    (wavetab = )              Wavelength table name
      (result = )             Result of synphot calculation for form
    (refdata = )              Reference data
```

The `form` parameter may be any of the accepted types (`fnu`, `flam`, `photnu`, `photlam`, `abmag`, `stmag`, `counts`, `obmag`, `vegamag`, `jy`, or

mjy). By default, the task calculates and reports the values of the pivot wavelength and FWHM in addition to the quantity chosen by form.

The func parameter determines what output function is computed. The default value (effstim) calculates the predicted count rate or flux. This is the function most users will use. The other functions compute various functions of the product of the passband throughput and the spectrum. The spectrum is first converted into the units specified by the form parameter before computing the function, so results will depend on whether the form has frequency or wavelength units. The sole exception is efflam, which is always equivalent to avglam computed in count units. The efflam form is included for compatibility with a previous version of **calcphot**.

Table 5.3: Functions Supported by Calcphot

Form	Description	Formula
avglam	Average wavelength	$\frac{\int f_\lambda P_\lambda \lambda \, d\lambda}{\int f_\lambda P_\lambda \, d\lambda}$
barlam	Mean log ("bar") wavelength	$\bar{\lambda} = \exp\left[\frac{\int f_\lambda P_\lambda \ln(\lambda) d\lambda/\lambda}{\int f_\lambda P_\lambda/\lambda}\right]$
efflam	Effective wavelength	$\frac{\int f_\lambda P_\lambda \lambda^2 \, d\lambda}{\int f_\lambda P_\lambda \lambda \, d\lambda}$
effstim	Effective stimulus [flam]	$\frac{hc \int f_\lambda P_\lambda \lambda \, d\lambda}{\int P_\lambda \lambda \, d\lambda}$
fwhmlam	FWHM bandwidth	$\sqrt{8\ln 2} \times rmslam$
rmslam	RMS bandwidth	$\frac{\sqrt{\int f_\lambda P_\lambda \ln(\lambda/\bar{\lambda})^2 d\lambda/\lambda}}{\sqrt{\int f_\lambda P_\lambda d\lambda/\lambda}}$

In the table above, P_λ is the (dimensionless) passband throughput, f_λ is the source flux distribution, and $\bar{\lambda}$ is the mean wavelength of the passband as defined in Schneider, Gunn, and Hoessel (1983).

If desired, the spectrum expression may contain variable zero ($0), so that the photometric calculations are repeated over the series of values specified by the vzero parameter. See the examples section below for details on how this can be used, and see Section 3.4, "Variable Substitution (VZERO)," on page 26 for more details.

The result parameter is an output parameter and contains the result of the requested photometric calculation. This can be the observed flux of the synthetic spectrum in the selected observation mode, or it can be a passband parameter as specified by the form parameter. This parameter contains the result of the last calculation performed, so if several spectra or

modes are given via a list file, or the calculation is repeated over a series of `vzero` values, then only the result of the last calculation is saved.

The results of calculations may be saved in a FITS or STSDAS table, if desired, via the `output` parameter. This table contains the columns of information described in Table 5.4.

Table 5.4: Calcphot Output Table Columns

Column	Contents
`COUNTRATE`	Result of photometric calculation in units of `form` or `FUNC`
`FORM`	Units of `COUNTRATE`
`OBSMODE`	Instrument observing mode
`TARGETID`	Synthetic spectrum specification

There is one table row for each calculation performed. This table can be used as input to the **plspec** task and other tasks using the `pfile` (photometry file) parameter. If `form` is set to `counts` or `obmag`, then the value of `COUNTRATE` as stored in this table is normalized to the telescope area. If calculating one of the passband parameters listed in Table 5.1, the name of the first column of the output table is the name of that parameter.

Null values for the `wavetab` and `refdata` parameters will cause the task to use a default wavelength grid (see Section 3.2.2) and the default reference data for the HST observatory (see Section 3.5). The default wavelength set covers the range where the `obsmode` and `spectrum` are non-zero. If there is more than one `obsmode` and `spectrum`, the range is computed based on the ***first pair***. If the wavelength ranges of the obsmodes and spectra differ significantly, a suitable wavelength table should be specified explicitly.

5.3.1 Examples

The first example (Figure 5.4) uses **calcphot** to calculate the integrated flux (in DNs per second) of a 5000 K blackbody in the passband defined by the WFC detector 2 and F555W filter. The blackbody spectrum is renormalized to have a *V* magnitude of 18.6. Specifying the generic instrument name “wfpc” in the `obsmode` string results in the use of the default WFC detector number 2. By default, the pivot wavelength and FWHM of the specified passband are also calculated.

Figure 5.4: Sample Calcphot Run

```
sy> calcphot wfpc,f555w,dn \
>>> "rn(bb(5000),band(v),18.6,vegamag)" counts

Mode = band(wfpc,f555w,dn)
   Pivot       Equiv Gaussian
 Wavelength          FWHM
  5467.652        1200.957    band(wfpc,f555w,dn)
Spectrum:  rn(bb(5000),band(v),18.6,vegamag)
    VZERO      (COUNTS s^-1 hstarea^-1)
       0.           55.41071
```

The next example (Figure 5.5) computes the flux of the same blackbody spectrum through the WFPC2 and its F439W and F555W filters (similar to Johnson *B* and *V*), and finds the color difference (in instrumental magnitudes) between the two. The calculation is repeated for values of *E*(*B*–*V*) of 0.0, 0.25, and 0.5 applied to the blackbody spectrum.

Figure 5.5: Second Sample Calcphot Run

```
sy> calcphot "band(wfpc2,4,f439w)-band(wfpc2,4,f555w)" \
>>> "bb(5000)*ebmv($0)" obmag vzero="0,0.25,0.5"

Mode = band(wfpc2,4,f439w) - band(wfpc2,4,f555w)
   Pivot       Equiv Gaussian
 Wavelength          FWHM
  4311.846        476.3737    band(wfpc2,4,f439w)
  5442.215        1229.98     band(wfpc2,4,f555w)
Spectrum:  bb(5000)*ebmv($0)
 VZERO OBMAG(band(wfpc2,4,f439w))-OBMAG(band(wfpc2,4,f555w))
     0.         2.535146
   0.25          2.77968
    0.5             3.03
```

As we might have expected, the change in the F439W-F555W color of the spectrum is about 0.25 mag for every 0.25 mag change in *E*(*B*–*V*)!

For a final example, we'll demonstrate the use of the ACS-specific parameterized keyword `aper#.` This keyword was implemented to permit the calculation of source counts as a function of photometric aperture. It simulates the performance of aperture photometry on the model spectrum. The keyword allows the user to specify the radius of the photometric aperture in arcseconds. This feature is particularly useful for near-IR filters where the aperture correction can be dependent on the spectral energy distribution of the object

First we calculate the integrated flux (in counts per second) of a K0V star in the passband defined by the Wide Field Channel and the f850lp filter. The star spectrum is renormalized to have a V magnitude of 24.

From the command line within **synphot** we type the usual syntax (common to all instruments):

```
sy>  calcphot acs,wfc1,f850lp "rn(icat \
(k93models,5250,0.0,5.0),band(v),24,vegamag)" \
counts func=effstim
```

In return we obtain the total number of counts for an infinitely large region. In fact, in our example we did not specify or ask for a special aperture.

```
Mode = band(acs,wfc1,f850lp)

   Pivot       Equiv Gaussian
Wavelength         FWHM
9054.795         1270.317    band(acs,wfc1,f850lp)

Spectrum:    rn(icat(k93models,5250,0.0,5.0),band(v),24,veg-
amag)
VZERO      (COUNTS s^-1 hstarea^-1)
0.           3.451657
```

The next example is to calculate the source counts that are in a user's specified aperture. We choose a 0.1 arcsec radius aperture.

```
sy> calcphot “acs,wfc1,f850lp,aper#0.1”\
"rn(icat(k93models,5250,0.0,5.0),band(v),24,vegamag)”|
counts func=effstim
```

In output we obtain:

```
Mode = band(acs,wfc1,f850lp,aper#0.1)
   Pivot       Equiv Gaussian
 Wavelength         FWHM
  8959.136        1158.454    band(acs,wfc1,f850lp,aper#0.1)
Spectrum:rn(icat(k93models,5250,0.0,5.0),band(v),24,vegamag)
VZERO      (COUNTS s^-1 hstarea^-1)
0.           1.709906
```

As expected, the amount of flux within a 0.1” radius aperture is significantly smaller. Because the definition of pivot wavelength depends on throughput (see Table 5.1), the pivot wavelength, and the FWHM of the bandpass centered on it, also change slightly.

5.4 Calcspec

The **calcspec** task will create a synthetic spectrum by combining existing models, such as a blackbody or power-law, as well as spectra read from a file. The spectral data can be manipulated in various ways, including multiplying by a passband, scaling by constants, adding or removing extinction, and normalizing to an arbitrary value in a specified passband. Unit conversion is performed automatically when adding or subtracting spectra that have different forms (units). Spectra cannot be multiplied

together, but they can be multiplied by arbitrary (unitless) constants and passbands.

The resulting spectral data are written to a FITS or STSDAS table containing columns of wavelength and flux. The wavelength data will be in units of Angstroms and the flux data will be in the unit system specified by the task parameter `form`. If the output form is chosen to be either `counts` or `obmag`, then the flux data written to the output table are scaled by the telescope area. The columns in the output table are labeled WAVELENGTH and FLUX.

The task parameters are shown in Figure 5.6 below.

Figure 5.6: Calcspec Parameters

```
spectrum =                Spectrum to calculate
output   =                Output table name
(form    = photlam)       Desired form of output spectrum
(vzero   = "")            List of values for variable zero
(wavetab = "")            Wavelength table name
(refdata = "")            Reference data
```

The `spectrum` parameter can be a string of any valid commands and arguments that specify the spectrum to be synthesized. Multiple spectrum commands may be placed in a text file, one per line, and specified as `spectrum=@`*`filename`*. See Section 3.2.3 for a detailed list and description of all the available `spectrum` commands, and “Filename specifications” on page 19 for important side effects of using the `@filename` syntax. If more than one spectrum expression is specified via a text file, then a separate “FLUXn” column is created in the output table for each spectrum.

The output table contains the header keywords GRFTABLE, CMPTABLE, and EXPR, which are the names of the instrument graph table, the component lookup table, and the `spectrum` parameter string. If more than one spectrum is specified via a file, each one will be listed as a separate header keyword, with the name EXPR*n*.

The `vzero` parameter contains a list of values that are substituted for variable zero (`$0`) wherever it appears in the spectrum expression. See Section 3.4, “Variable Substitution (VZERO),” on page 26 for a discussion of variable substitution.

Null values for the `wavetab` and `refdata` parameters will cause the task to use a default wavelength grid (See “Wavelength Table” on page 25.) and the default reference data for the HST observatory (see Section 3.5). The default wavelength grid covers the wavelength range where the spectrum is non-zero. Wavelengths are spaced logarithmically over this range. If there is more than one spectrum specified, the range of the default wavelength grid is computed based on the ***first spectrum***. Therefore, if the

wavelength ranges of the spectra differ significantly, a suitable wavelength table that covers the desired range of all the spectra should be created using the **genwave** task and supplied as input via the `wavetab` parameter.

5.4.1 Examples

The following example synthesizes a 5000 K blackbody spectrum and renormalizes it so that it produces an integrated flux of 100 counts (DN) per second in the WFPC2 F555W passband. The spectral data, in units of `flam`, are written to the table `rnbb.fits`.

```
sy> calcspec \
>>> "rn(bb(5000),band(wfpc2,f555w,a2d7),100,counts)" \
>>> rnbb.fits form=flam
```

The next example simulates an observation of the flux calibration standard star G191-B2B using the FOS blue-side detector with the 1.0" aperture and the G190H grating. The spectral data for G191-B2B are in the table `crcalspec$g191b2b_stis_001.fits`, in units of `flam`. First, in order to simulate the true count rate per diode as accurately as possible, we must either create a wavelength table, using the **genwave** task, that approximates the dispersion relation for the G190H grating, or use an existing wavelength table for this grating that is available in the STSDAS `synphot$data` directory called `fos_blue_g190h.dat`. To make our own table we use **genwave** as follows:

```
sy> genwave g190h.fits minw=1573.0 maxw=2330.0 dwave=1.47
```

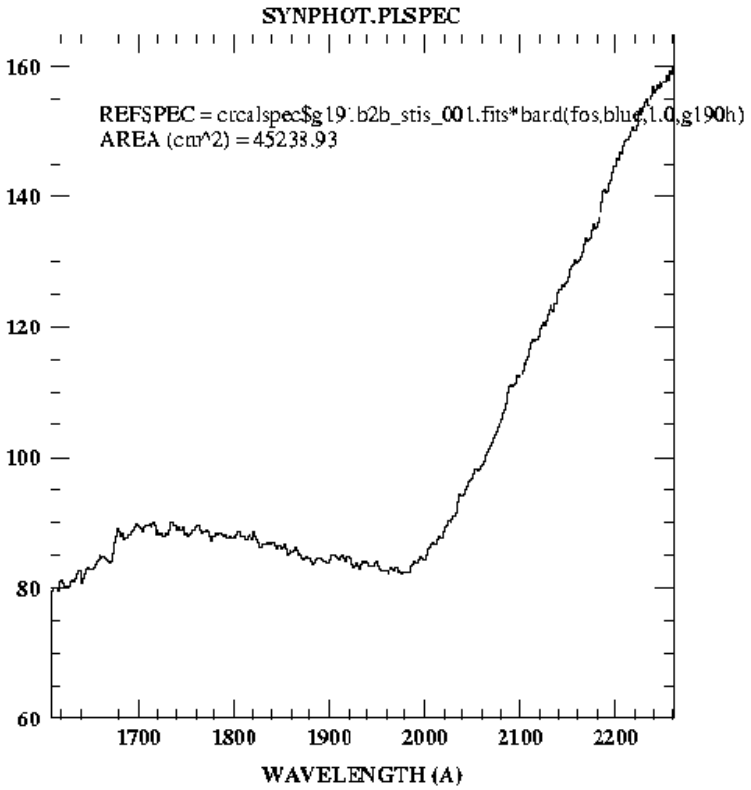
where the dispersion parameters have been obtained from the *FOS Instrument Handbook*.

Now we run **calcspec**, multiplying the spectrum of G191-B2B by the desired FOS observation mode. The simulated spectrum, in counts per second per diode, is written to the table `g191_fos.fits`.

```
sy> calcspec \
"crcalspec$g191b2b_stis_001.fits*band(fos,blue,1.0,g190h)" \
g191_fos.fits form=counts wavetab=g190h.fits
```

The resulting count rate spectrum is shown in Figure 5.7. You can create a similar plot by using the plspec command to display the resulting table (`plspec "" g191_fos.fits form=counts`). If we wanted to use the wavelength table from the `data` directory, we would specify "`wavetab=synphot$data/fos_blue_g190h.dat`" on the command line.

Figure 5.7: FOS Count Rate Spectrum of G191-B2B



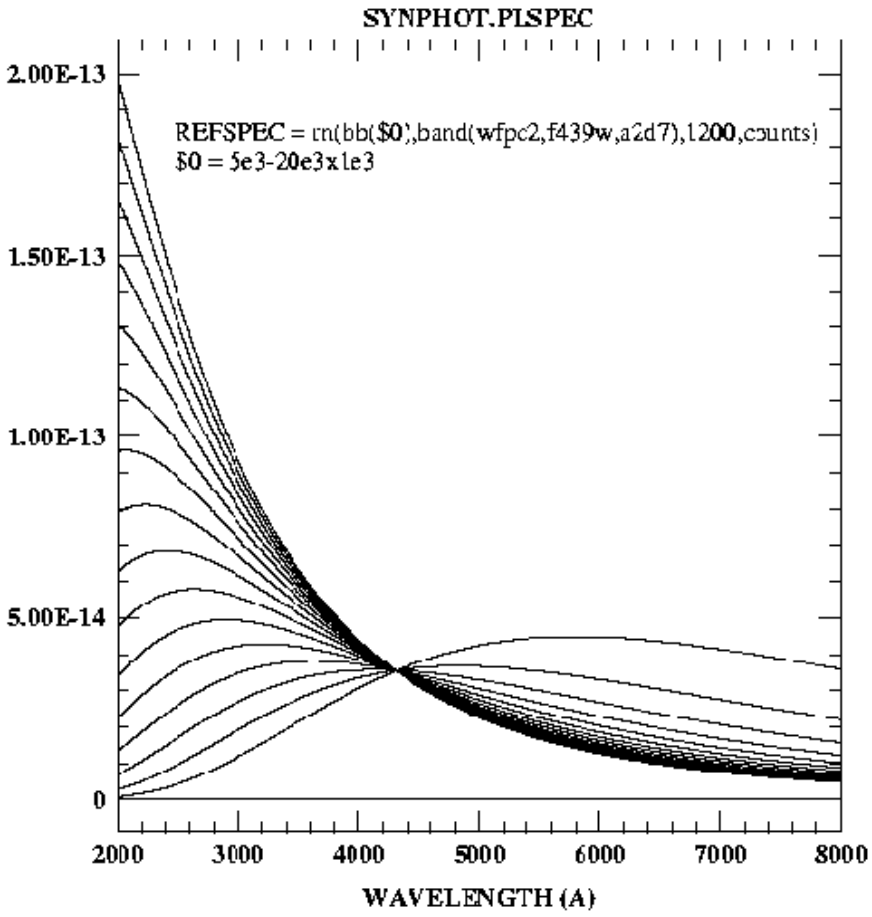
For a final example, let's suppose a star observed with the WFPC2 and filter F439W produces 1200 counts (DNs) per second. What would its spectrum be for various blackbody temperatures ranging from 5000 to 20000 K, in steps of 1000 K?

```
sy> calcspec \
>>> "rn(bb($0),band(wfpc2,f439w,a2d7),1200,counts)" \
>>> bbody.fits form=flam vzero="5e3-20e3x1e3"
```

The resulting spectra, in units of `flam`, are stored in the output table `bbody.fits`. Some of the spectra are shown in Figure 5.8.You can generate a similar plot by using the plspec command with equivalent parameters

```
(plspec "" \
 "rn(bb($0),band(wfpc2,f439w,a2d7),1200,counts)"\
 form=flam vzero="5e3-20e3x1e3" right=8000\
left=2000).
```

Figure 5.8: WFPC2 Blackbody Spectra



5.5 Countrate

The **countrate** task is similar to the **calcspec** and **calcphot** tasks in that, for a given input spectrum and HST observing mode, it will compute a countrate spectrum, as well as the total counts integrated over the passband defined by the observing mode. There are, however, two unique features to this task. First, the input parameters were originally structured to mimic what is contained in the exposure logsheets found in HST observing proposals under the RPS2 proposal entry system in use at the time. (Since then, the Astronomer's Proposal Tool (APT), which has a different user interface, has become the standard tool for Phase 2 proposals; so this feature is no longer especially useful.) Secondly, for the spectroscopic instruments, it will automatically search for and use a wavelength table from the STSDAS `synphot$data` directory that is appropriate for the selected instrumental dispersion mode. This remains extremely useful, and the **countrate** task therefore remains ideally suited to predicting exposure

times when writing HST proposals. **countrate** is the underlying engine for the Exposure Time Calculators (ETCs) that are provided by STScI for use in proposal preparation.

Figure 5.9 is a list of the task parameters.

Figure 5.9: Countrate Task Parameters

```
spectrum =                  Spectrum to calculate
magnitude =                 Magnitude and passband of spectrum
instrument =                Science instrument
(detector = " ")            Detector used
(spec_elem = " ")           Spectral elements used
(aperture = " ")            Aperture / field of view
(cenwave = INDEF)           Central wavelength (HRS and STIS only)
(exptime = 1.)              Exposure time in seconds
(reddening = 0.)            Interstellar reddening E(B-V)
(redlaw = "gal1")           Reddening law used (gal1|gal2|gal3|smc|lmc|xgal
(output = "none")           Output table name
(form = "counts")           Form for output data
(magform = "vegamag")       Form for magnitude
(wavecat = "synphot$data/wavecat.dat") Catalog of wavelength tables
(refwave = INDEF)           Reference wavelength
(verbose = yes)             Print results to STDOUT ?
(flux_tot = INDEF)          Estimated total flux (output)
(flux_ref = INDEF)          Estimated flux at reference wavelength (output)
(refdata = "")              Reference data
```

Unlike other **synphot** tasks in which the observing mode is specified via the single parameter `obsmode`, the **countrate** task uses separate `instrument`, `detector`, `spec_elem`, and `aperture` parameters to specify the observing mode. The `instrument` parameter specifies one of the HST instruments: `acs`,`fgs`, `foc`, `fos`, `hrs`, `hsp`, `nicmos`, `stis`, `wfpc`, or `wfpc2`. If you set the instrument parameter to `hrs` or `stis`, you should also specify a value for `cenwave`, the central wavelength of the spectrum (see below). The `detector` parameter specifies the name of the desired detector, if there is more than one available for the chosen instrument. The `spec_elem` parameter specifies the name of spectral elements, such as filters or gratings. Finally, the `aperture` parameter specifies the name of the instrument aperture, if there is more than one available for the chosen instrument. The `instrument` parameter must always be specified, but one or more of the `detector`, `spec_elem`, and `aperture` parameters may be left blank, depending on their applicability to a given instrument.

A standard `obsmode` string is constructed internally by the task by concatenating the `instrument`, `detector`, `spec_elem`, and `aperture` parameter strings. Therefore, if you want to include an extra mode keyword in the desired instrument mode, you can do so by including the keyword along with any others that are given for the four parameters mentioned above. For example, if you want to include the effects of contamination on MJD 50477 in a WFPC2 simulation using the f300w filter, you could specify "`spec_elem=f300w,cont#50477`".

The `cenwave` parameter specifies the central wavelength for the simulated observation, in Angstroms; the output spectrum will be centered at this wavelength. If set to INDEF, the output spectrum will contain the entire wavelength range covered by the chosen observation mode. This parameter only affects HRS and STIS instrument modes, because they are the only instruments for which the detector cannot cover the entire wavelength range of an observation. This parameter is ignored for all other instrument modes.

The target spectrum is specified as any valid **synphot** spectrum expression given in the `spectrum` parameter, to be renormalized to the magnitude given in the `magnitude` parameter using the form of the `magform` parameter. The `crcalspec$` and `crgrid$` directories contain tables of HST flux calibration star spectra and various spectral atlases (both models and observational data) that can be used as source spectra (see Appendix A:On-Line Catalogs and Spectral Atlases), or you can use a synthetic spectrum function to generate a spectrum at run-time. The `spectrum` parameter recognizes all of the valid functional forms that can be specified via to the `spectrum` parameter in other **synphot** tasks.

The `magnitude` parameter is used in conjunction with `spectrum` to renormalize the spectrum to a chosen magnitude. This is done by specifying the desired integrated broadband magnitude, in units of `magform`, for the spectrum in one of the standard *UBVRI* passbands. The syntax used to specify `magnitude` is a two-word string consisting of the desired magnitude value, followed by the name of the desired passband, e.g., "`magnitude=15.6 V`."

The `reddening` parameter may be used to add or remove the effects of interstellar reddening on the input spectrum and is specified in units of $E(B–V)$. It provides the same functionality as the `ebmvx` function in the spectrum interpreter of other **synphot** tasks. The `redlaw` parameter gives the reddening law to use in the `ebmvx` function. The default value, `gal1`, is Seaton's law and the same as by the `ebmv` function.

The `exptime` parameter is used to specify the desired exposure time, in seconds, to be applied in the calculation of the integrated counts and the countrate spectrum. Therefore, if `exptime=1`, the result will be a true count rate (counts per second), whereas if `exptime > 1`, the result will be the total counts accumulated in that time.

The output table produced by **countrate** is a FITS or STSDAS table containing columns of WAVELENGTH and FLUX, in units of Angstroms and `form`, respectively. This table will also contain the header keywords GRFTABLE, CMPTABLE, OBSMODE, SPECTRUM, and EXPTIME, corresponding to the values of the related task parameters. The `output` parameter may be set to either "none" or a null string (i.e., " "), in which case no output table will be created, but the integrated counts within the passband will still be printed to STDOUT (terminal).

In addition to computing the total counts integrated over the chosen instrument passband, it is possible to compute the number of counts at a particular reference wavelength through the use of the `refwave` parameter. If the chosen instrument is a spectrograph (FOS, HRS, or STIS), the task will compute the counts in the pixel containing the reference wavelength. If the chosen instrument is not a spectrograph, the task will compute the counts per Angstrom at the chosen reference wavelength.

If the `verbose` parameter is set to "yes", then the results of the total counts and counts at the reference wavelength calculations will be written to STDOUT (terminal). In addition, these two quantities are also written to the output parameters `flux_tot` and `flux_ref`.

5.5.1 Examples

This example computes the result of a 1800 second observation of Feige 110, using the HRS with the large science aperture (`lsa`) and the g160m grating. We'll also have it compute the number of counts in one pixel at 1504 Å. The spectral data for Feige 110 are read from the table `feige110_stis_001.fits` in the directory `crcalspec$`. The calculated spectrum, in units of counts per pixel, is written to the table `hrsobs.fits`. We use `epar countrate` to set the task parameters as follows:

Figure 5.10: Example Countrate Parameter Settings

```
spectrum = "crcalspec$feige110_stis_001.fits" Spectrum to calculate
magnitude = " "               Magnitude and passband of spectrum
instrument = "hrs"            Science instrument
(detector = " ")              Detector used
(spec_elem = "g160m")         Spectral elements used
(aperture = "lsa")            Aperture / field of view
(cenwave = 1495.)             Central wavelength (HRS and STIS only)
(exptime = 1800.)             Exposure time in seconds
(reddening = 0.)              Interstellar reddening E(B-V)
(redlaw = "gal1")             Reddening law used (gal1|gal2|gal3|smc|lmc|xgal
(output = "hrsobs.fits")           Output table name
(form = "counts")             Form for output data
(magform = "vegamag")         Form for magnitude
(wavecat = "synphot$data/wavecat.dat") Catalog of wavelength tables
(refwave = 1504.)             Reference wavelength
(verbose = yes)               Print results to STDOUT ?
(flux_tot = INDEF)            Estimated total flux (output)
(flux_ref = INDEF)            Estimated flux at reference wavelength (output)
(refdata = "")                Reference data
```

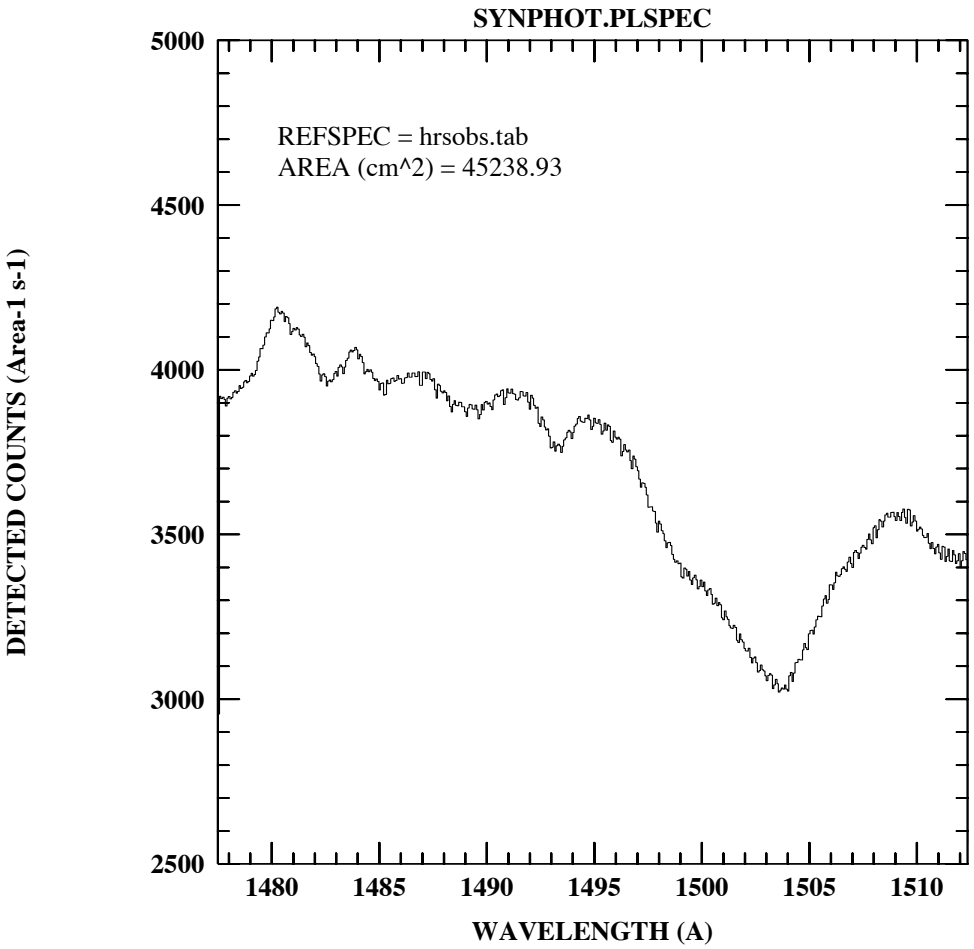
The output on your screen will be the following:

```
Mode = band(hrs,g160m,lsa)
Spectrum:  crcalspec$feige110_stis_001*1800.
   Pivot       Equiv Gaussian  Total Flux  Flux at 1504 A
 Wavelength         FWHM          COUNTS        COUNTS
  1579.611 636.7933       1902314.      3226.052
```

These results indicate an integrated flux of about 1.9 million counts over the whole spectrum, with approximately 3226 total counts in the pixel at 1504 Å.

Figure 5.11 shows the resulting spectrum as plotted using the STSDAS task **plspec** (e.g., `plspec none hrsobs counts wave=hrsobs top=5000 bottom=2500`).

Figure 5.11: Plot of Countrate Example Spectrum



For a second example, let's calculate the total number of counts expected in a 600 second integration of a star with a spectrum similar to that of HZ 2 having a *V* magnitude of 17.5, using the WFPC2 PC chip (chip 1) and F785LP filter. We'll also redden the spectrum by *E(B-V)*=0.05. Here

are the parameter settings and the resulting total counts, in units of DNs, recorded in the `flux_tot` parameter:

Figure 5.12: Second Countrate Example Parameter Settings

```
    spectrum = "crcalspec$hz2_005.fits" Spectrum to calculate
   magnitude = "17.5 V"        Magnitude and passband of spectrum
  instrument = "wfpc2"         Science instrument
   (detector = "1,a2d7" )      Detector used
  (spec_elem = "f785lp" )      Spectral elements used
   (aperture = " " )           Aperture / field of view
    (cenwave = INDEF)          Central wavelength (HRS and STIS only)
    (exptime = 600.)           Exposure time in seconds
  (reddening = 0.05)           Interstellar reddening E(B-V)
     (redlaw = "gal1")         Reddening law used (gal1|gal2|gal3|smc|lmc|xgal)
     (output = "none")         Output table name
       (form = "COUNTS")       Form for output data
    (magform = vegamag)        Form for magnitude
    (wavecat = synphot$data/wavecat.dat) Catalog of wavelength tables
    (refwave = INDEF)          Reference wavelength
    (verbose = yes)            Print results to STDOUT ?
   (flux_tot = INDEF)          Estimated total flux (output)
   (flux_ref = INDEF)          Estimated flux at reference wavelength (output)
    (refdata = )               Reference data
```

The output to the screen will be:

```
Mode = band(wfpc2,1,a2d7,f785lp)
Spectrum:        rn(crcalspec$hz2_005.fits*ebmvx(0.05,gal1),
band(v), 17.5,vegamag)*600.
    Pivot       Equiv Gaussian  Total Flux
 Wavelength          FWHM         COUNTS
  8458.758         937.9294       8353.222
```

5.6 Fitband

The **fitband** task fits a model passband to a known passband function stored in a throughput table. The user specifies a model expression containing up to nine free variables and initial values for the variables. The task then searches for values that minimize the residuals between the model and known passband. When the task finds the optimal solution, it writes the final values of the variables back to the parameter file.

Fitting is done by one of two available methods: the Levenberg-Marquardt method or the downhill simplex method, sometimes referred to as the “amoeba” method. The downhill simplex method is slower than the Levenberg-Marquardt method because it requires more iterations to converge to a solution. In compensation, however, it converges

to the solution over a larger range of initial values than the Levenberg-Marquardt method. In either case, the initial values for the free variables should be as accurate as possible, as neither method will converge to a solution from arbitrarily chosen initial values. If the initial values are outside the range of convergence, the task may either compute a false solution, or wander outside the range where the model expression is defined and terminate with an error. The Levenberg-Marquardt code is from the minpack library at Argonne National Laboratory. The downhill simplex method was adapted from *Numerical Recipes* by Press et al. (1992).

The task has a relatively long list of parameters (Figure 5.13).

Figure 5.13: Fitband Parameters

```
input =                  Observed bandpass
     obsmode =                  Model bandpass
     (output = none)            Table containing fitted bandpass
       (ftol = 1e-05)           Fractional tolerance termination condition
    (maxiter = 500)             Maximum number of iterations
    (nprint = 0)              Number of iterations between diagnostic prints
       (slow = no)              Use slow method (simplex) to compute fit?
      (equal = no)              Use equal weighting on the data points?
       (vone = INDEF)           Value of variable one
       (vtwo = INDEF)           Value of variable two
     (vthree = INDEF)           Value of variable three
      (vfour = INDEF)           Value of variable four
      (vfive = INDEF)           Value of variable five
       (vsix = INDEF)           Value of variable six
     (vseven = INDEF)           Value of variable seven
     (veight = INDEF)           Value of variable eight
      (vnine = INDEF)           Value of variable nine
    (refdata = )                Reference data
```

The `input` parameter specifies the name of a table containing a known throughput function. This table must contain, at minimum, columns of wavelength and throughput values, and may contain an optional error column. If the table is in FITS or STSDAS format, the column names must be WAVELENGTH, THROUGHPUT, and ERROR. If it is a text file, the wavelength, throughput, and error values must be in the first, second, and third columns, respectively. If present, the ERROR column data are used for weighting the data points during the fit (see below).

The `obsmode` parameter specifies the model passband expression to be fitted to the `input` throughput table data. The `obsmode` may contain any valid **synphot** passband functions, with up to nine variables substituted for the arguments of those functions. The variables in the expression are indicated by a dollar sign followed by a digit from 1 to 9 (e.g., "$3"). The initial values of the variables are given in the task parameters `vone`

through `vnine` (see below). All variables not used should be set to INDEF. Variables in the expression should be consecutive; for example if the model contains three variables, they should be `$1`, `$2`, and `$3`, not `$1`, `$3`, `$7`.

The `obsmode` expression may be placed in a file, if desired, in which case the name of the file is specified as `obsmode=@`*`filename`*. (See "Filename specifications" on page 19 for important side effects of using the `@filename` syntax.) Placing the `obsmode` expression in a file is necessary if the expression is too long to fit in the task parameter (63 characters max). If the expression is placed in a file, it may be split over more than one line wherever a blank is a legal character in the expression.

The `output` parameter specifies the name of the output table that will contain the final fitted passband function. If `output` is set to "none" or left blank, then no output table will be produced. The output table contains the model passband (`obsmode`) expression evaluated with the fitted values of the free variables. The header of the table also contains the names of the graph and component lookup tables and the model expression.

The `ftol` parameter sets the fractional tolerance convergence criterion. Iteration of the least squares fit ceases when the scaled distance between two successive estimates of the free variables is less than this value. Each component of the scaled distance is scaled by dividing the difference between the two estimates by half their sum. Note that the fit solution may not converge to an arbitrarily small value; instead it may cycle between several values. Setting `ftol` to too small a value may therefore result in failure to converge.

The `maxiter` parameter sets the maximum number of iterations to be performed. If convergence is not achieved in this number of iterations, then the task stops with a warning message to that effect.

The `nprint` parameter specifies the number of iterations to perform between successive printings of diagnostic information to STDERR (usually your terminal). If `nprint`=0, there will be no diagnostic information printed. The diagnostic prints contain the iteration number, the chi-squared value, and the model passband with the current values of the variables.

The `slow` parameter selects which fitting method to use. If `slow=no` (the default), the Levenberg-Marquardt method is used. If `slow=yes`, the downhill simplex method is used.

The `equal` parameter indicates whether or not to weight the data points when computing the fit. If `equal=no` and the input table contains a column of error values, then the data points will be weighted by the errors. Points with indefinite, negative, or zero-valued errors are not used in the fit. If `equal=yes` or no valid error values are specified in the input table, then the data points will receive equal weights.

The vone through vnine parameters are used to specify the initial values of the variables used in the model passband (obsmode) expression. Initial (non-INDEF) values must be set for variables in use before running the task. At the conclusion of the fitting process, these parameters will be updated with the final variable values. Any variables that are not used in the model should be set to INDEF. See Section 3.4, "Variable Substitution (VZERO)," on page 26 for a discussion of variable substitution.

5.6.1 Example

This example demonstrates how to determine values for the central wavelength, FWHM, and scale factor of a gaussian passband that best fits the shape of the F555W filter from the WFPC2. The equal parameter is set to "yes" because the error values in the throughput table for the F555W filter are all zero. We start with initial guesses of 5500 and 500 Å for the central wavelength and FWHM, respectively, and 1.0 for the scale factor. Diagnostic information is printed after each iteration and the final fitted passband data is saved in the table fit555w.fits.

```
sy> fitband crwfpc2comp$wfpc2_f555w_006_syn.fits
"gauss($1,$2)*$3" \
>>> out=fit555w.fits nprint=1 vone=5500 vtwo=500 vthree=1
equal+
```

While the task is running, the following information will appear:

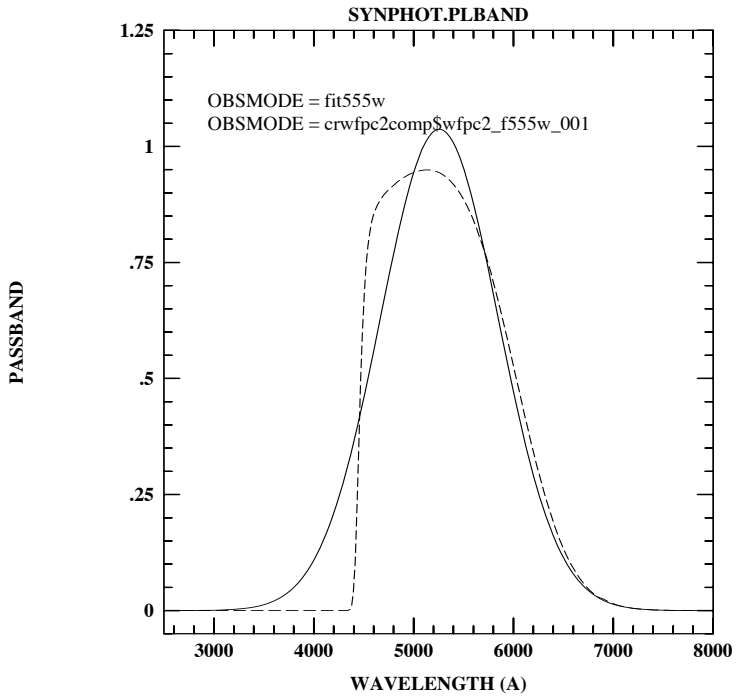
```
irep = 1 chisq = 0.069752 exp = gauss(5500.,500.)*1.01
irep     =     2     chisq     =     0.038892     exp     =
gauss(5447.017,1048.002)*0.6551079
irep     =     3     chisq     =     0.012754     exp     =
gauss(5204.142,1900.48)*0.8528795
irep     =     4     chisq     =     0.008139     exp     =
gauss(5317.027,1230.885)*1.017857
irep     =     5     chisq     =     0.005766     exp     =
gauss(5250.305,1477.974)*0.9963291
irep     =     6     chisq     =     0.005529     exp     =
gauss(5265.997,1366.231)*1.050334
irep     =     7     chisq     =     0.005493     exp     =
gauss(5256.394,1402.266)*1.039902
irep     =     8     chisq     =     0.005492     exp     =
gauss(5259.135,1388.974)*1.045587
irep     =     9     chisq     =     0.005491     exp     =
gauss(5258.057,1393.395)*1.043876
irep     =     14     chisq     =     0.005491     exp     =
gauss(5258.118,1393.203)*1.04391
irep     =     15     chisq     =     0.005478     exp     =
gauss(5258.145,1393.12)*1.033594

Final solution:

gauss(5258.118,1393.203)*1.033575
```

You can plot the fitted gaussian and overplot the F555W passband for comparison using the **plband** task:

```
sy> plband fit555w.fits left=2500 right=8000
sy>  plband  crwfpc2comp$wfpc2_f555w_006_syn.fits  append+
ltype=dashed top=1.25
```

The resulting plot is shown in Figure 5.14.

Figure 5.14: Results from Example Fitband Run



5.7 Fitspec

The **fitspec** task allows you to fit a spectrum model to spectral data stored in a table. You specify a model spectrum using any valid **synphot** spectrum expression, including up to nine free variables and their initial values. The task then searches for values for those variables that minimize the residuals between the model spectrum and the known spectrum stored in the table. When the task finds the optimal solution, it writes the fitted values of the variables back to the parameter file and prints the expression with the final variable values. Using the final parameter values you could restart the task to perform more iterations.

Fitting is done by one of two available methods: the Levenberg-Marquardt method or the downhill simplex method, sometimes

referred to as the "amoeba" method. The downhill simplex method is slower than the Levenberg-Marquardt method because it requires more iterations to converge to a solution. In compensation, however, it converges to the solution over a larger range of initial values than the Levenberg-Marquardt method. In either case, the initial values for the free variables should be as accurate as possible, as neither method will converge to a solution from arbitrarily chosen initial values. If the initial values are outside the range of convergence, the task may either compute a false solution, or wander outside the range where the model expression is defined and terminate with an error. The Levenberg-Marquardt code is from the minpack library at Argonne National Laboratory. The downhill simplex method was adapted from *Numerical Recipes* by Press et al. (1992).

The **fitspec** task parameter list (Figure 5.15) is nearly identical to that of the **fitband** task.

Figure 5.15: Fitspec Parameters

```
input =                  Observed spectrum
    spectrum =                  Model spectrum
     (output = none)            Table containing fitted spectrum
       (ftol = 1e-05)           Fractional tolerance termination condition
    (maxiter = 500)             Maximum number of iterations
    (nprint = 0)              Number of iterations between diagnostic prints
       (slow = no)              Use slow method (simplex) to compute fit?
      (equal = no)              Use equal weighting on the data points?
       (vone = INDEF)           Value of variable one
       (vtwo = INDEF)           Value of variable two
     (vthree = INDEF)           Value of variable three
      (vfour = INDEF)           Value of variable four
      (vfive = INDEF)           Value of variable five
       (vsix = INDEF)           Value of variable six
     (vseven = INDEF)           Value of variable seven
     (veight = INDEF)           Value of variable eight
      (vnine = INDEF)           Value of variable nine
    (refdata = )                Reference dat
```

The `input` parameter specifies the name of a table containing a known spectrum. This table must contain, at minimum, columns of wavelength and flux values, and may contain optional error and spectral resolution column. If the table is in FITS or STSDAS format, the column names must be WAVELENGTH, FLUX, STATERROR, and FWHM. If it is a text file, the wavelength, flux, error, and FWHM values must be in the first, second, third, and fourth columns, respectively. If present, the contents of the STATERROR column are used for weighting the data points during the fit (see below). The FWHM column data are not used by this task.

The `spectrum` parameter specifies the model spectrum expression to be fitted to the `input` spectrum table data. The `spectrum` may contain any valid **synphot** spectrum functions, with up to nine variables substituted for the arguments of those functions. The variables in the expression are indicated by a dollar sign followed by a digit from 1 to 9 (e.g. "$3"). The initial values of the variables are given in the task parameters `vone` through `vnine` (see below). All variables not used should be set to INDEF. The model expression should not skip variables; for example if the model contains three variables, they should be `$1`, `$2`, and `$3`, not `$1`, `$3`, and `$7`.

The `spectrum` expression may be placed in a file, if desired, in which case the name of the file is specified as `spectrum=@`*`filename`*. (See "Filename specifications" on page 19 for important details about the `@filename` syntax.) Placing the `spectrum` expression in a file is necessary if the expression is too long to fit in the task parameter (63 characters max). If the expression is placed in a file, it may be split over more than one line wherever a blank is a legal character in the expression.

The `output` parameter specifies the name of the output table that will contain the final fitted spectrum. If `output` is set to "none" or left blank, then no output table will be produced. The output table contains the model spectrum expression evaluated with the fitted values of the free variables. The flux units for the data in the output table are the same as that of the input spectrum. The header of the output table also contains the names of the graph and component lookup tables and the model expression.

The `ftol` parameter sets the fractional tolerance convergence criterion. Iteration of the least squares fit ceases when the scaled distance between two successive estimates of the free variables is less than this value. Each component of the scaled distance is scaled by dividing the difference between the two estimates by half their sum. Note that the fit solution may not converge to an arbitrarily small value; instead it may cycle between several values. Setting `ftol` to too small a value may therefore result in failure to converge.

The `maxiter` parameter sets the maximum number of iterations to be performed. If convergence is not achieved in this number of iterations, then the task stops with a warning message to that effect.

The `nprint` parameter specifies the number of iterations to perform between successive printings of diagnostic information to STDERR (usually your terminal). If `nprint=0`, there will be no diagnostic information printed. The diagnostic prints contain the iteration number, the chi-squared value, and the model spectrum with the current values of the variables.

The `slow` parameter selects which fitting method to use. If `slow=no` (the default), the Levenberg-Marquardt method is used. If `slow=yes`, the downhill simplex method is used.

The `equal` parameter indicates whether or not to weight the data points when computing the fit. If `equal=no` and the input table contains a column of error values, then the data points will be weighted by the errors. Points with indefinite, negative, or zero-valued errors are not used in the fit. If `equal=yes` or no valid error values are specified in the input table, then the data points will receive equal weights.

The `vone` through `vnine` parameters are used to specify the initial values of the variables used in the model spectrum (`spectrum`) expression. Initial (non-INDEF) values must be set for variables in use before running the task. At the conclusion of the fitting process, these parameters will be updated with the final variable values. Any variables that are not used in the model should be set to INDEF. See Section 3.4, "Variable Substitution (VZERO)," on page 26 for a discussion of variable substitution syntax.

5.7.1 Example

Let's say you have an observed spectrum of an A-type star that's reddened by some unknown amount. We'll use **fitspec** to compute the amount of extinction in the observed spectrum by comparing it to an unreddened spectrum of the prototypical A0 V star Vega that is in the HST standards `crcalspec$` directory. For the purposes of this demonstration we'll create a spectrum of our hypothetical reddened star using **calcspec** as follows:

```
sy> calcspec "crgridjac$jc_19.fits*ebmv(0.073)" \
Astar_ebv073.fits form=flam
```

This uses the observed spectrum of an A-type star from the Jacoby-Hunter-Christian spectral library (see section A.8) and applies reddening equivalent to $E(B–V)$=0.073. The reddened spectrum is written to the table `Astar_ebv073` in units of `flam`.

Now we run **fitspec** using the `Astar_ebv073.fits` table as our observed spectrum (`input`) and ask **fitspec** to compute a model based on the unreddened Vega spectrum, solving for the amount of extinction that best fits the observed spectrum. Since our observed star is much fainter than Vega, we'll also need to solve for a renormalization value. Using **calcphot** we determine that the observed star has a *V* magnitude of about 9; we'll use that as the initial value for the renormalization variable. We use **epar** to set the **fitspec** parameters as follows:

Figure 5.16: Example Fitspec Parameter Settings

```
    input = Astar_ebv073.fits Observed spectrum
 spectrum = rn(crcalspec$alpha_lyr_stis_002.fits,band(v),$1,vegamag)*ebmv($2)  Model spectrum
  (output = Astar_fit.fits) Table containing fitted spectrum
    (ftol = 1e-05)          Fractional tolerance termination condition
 (maxiter = 500)            Maximum number of iterations
  (nprint = 1)              Number of iterations between diagnostic prints
    (slow = no)             Use slow method (simplex) to compute fit?
   (equal = yes)            Use equal weighting on the data points?
    (vone = 8.99492783993)  Value of variable one
    (vtwo = 0.0850384839994)Value of variable two
  (vthree = INDEF)          Value of variable three
      ...
   (vnine = INDEF)          Value of variable nine
 (refdata = )               Reference data
```

Final solution: rn(crcalspec$alpha_lyr_stis_002.fits,band(v),8.994928, vegamag)*ebmv(0.08503857):

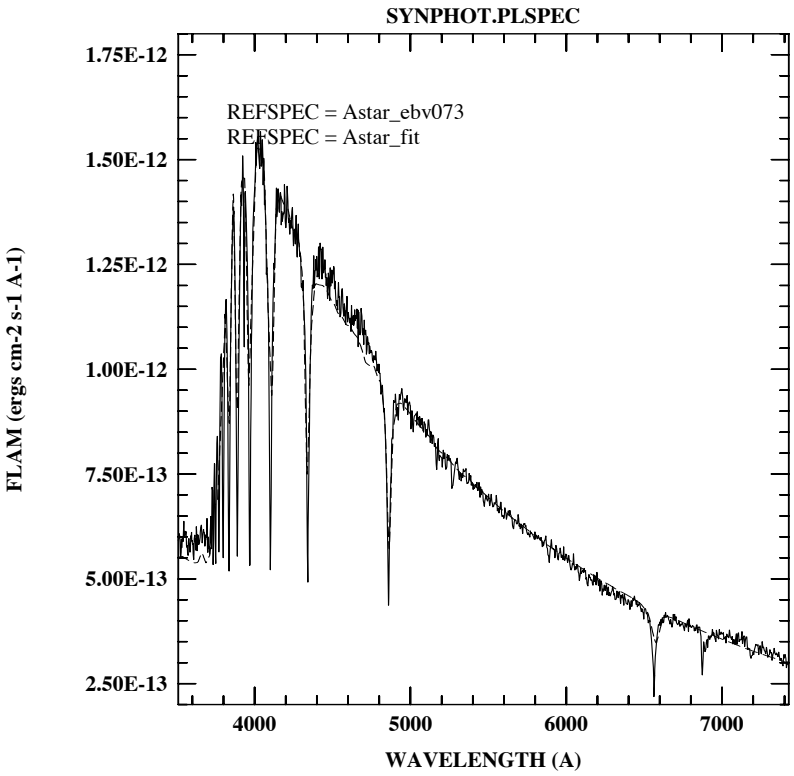
```
1 chisq=0.003964 exp=rn(alpha...,9.,vegamag)*ebmv(0.01)
2 chisq=2.307E-4 exp=rn(alpha...,9.0587,...)*ebmv(0.0624)
3 chisq=1.797E-4 exp=rn(alpha...,9.0333,...)*ebmv(0.0795)
4 chisq=1.799E-4 exp=rn(alpha...,9.0336,...)*ebmv(0.0799)
5 chisq=1.796E-4 exp=rn(alpha...,9.0336,...)*ebmv(0.0791)

Final solution:
rn(alpha_lyr_001,band(v),9.0336,vegamag)*ebmv(0.07914)
```

We can plot the results, comparing the observed spectrum with the fitted spectrum, using **plspec**:

```
sy> plspec "" Astar_ebv073.fits flam
sy> plspec "" Astar_fit.fits flam append+ ltype=dotted
```

This plot is shown in Figure 5.17.

Figure 5.17: Results from Fitspec



5.8 Fitgrid

The **fitgrid** task is useful for finding the best match to a given input spectrum from a list (or grid) of other spectra. The task scales every spectrum in the list to the same flux as the input spectrum, and keeps track of the best fits. The task determines a final fit by interpolating between the two best fits in the input list. This is useful for getting a quick scale factor to fit a spectrum to data, for example, for use as a starting point of a more refined spectrum fit to be done with the **fitspec** task. The resulting fit can optionally be saved to an output spectrum table.

The **fitgrid** task uses the same fitting methods as the **fitband** and **fitspec** tasks, namely the Levenberg-Marquardt and downhill simplex (“amoeba”) methods. For more details about these methods please see the description of the **fitband** or **fitspec** task. The parameter list for **fitgrid** (Figure 5.18) is similar to those of the **fitband** and **fitspec** tasks.

Figure 5.18: Fitgrid Parameters

```
input    =           Observed spectrum
spectrum =           List of spectra
(output  = "none")   Table containing the fitted spectrum
(vzero   = " ")      List of values for variable zero
(ftol    = 1.0E-5)   Fractional tolerance termination condition
(maxiter = 500)      Maximum number of iterations
(nprint  = 0)        Number of iterations between diagnostic prints
(slow    = no)       Use slow method (simplex) to compute fit?
(equal   = no)       Use equal weighting on the data points?
(refdata  = "")       Reference data
```

The `input` parameter specifies the name of a table containing a known spectrum. This table must contain, at minimum, columns of wavelength and flux values, and may contain optional error and spectral resolution column. If the table is in FITS or STSDAS format, the column names must be WAVELENGTH, FLUX, STATERROR, and FWHM. If it is a text file, the wavelength, flux, error, and FWHM values must be in the first, second, third, and fourth columns, respectively. If present, the contents of the STATERROR column are used for weighting the data points during the fit (see below). The FWHM column data are not used by this task.

The `spectrum` parameter specifies the list of spectra to be compared to the `input` spectrum data. The `spectrum` string may contain any valid **synphot** spectrum expressions, with the option of using the variable `vzero` (`$0`) as an argument within the string. During execution, the values of `vzero` will be substituted for `$0`, automatically creating a list of spectra. Alternatively, several individual spectrum expressions may be stored in a file, one per line, in which case the name of the file is specified as `spectrum=@`*`filename`*. (See "Filename specifications" on page 19 for important details about the `@filename` syntax).

The `output` parameter specifies the name of the output table that will contain the final fitted spectrum. If `output` is set to "none" or left blank, then no output table will be produced. The output table contains the best fit from the list of spectra. The flux units for the output data are the same as that of the input spectrum. The header of the output table also contains the names of the graph and component lookup tables and the model expression.

The `vzero` parameter contains a list of values that are substituted for variable zero (`$0`) wherever it appears in the spectrum expression. See Section 3.4, "Variable Substitution (VZERO)," on page 26 for a discussion of variable substitution.

The `ftol` parameter sets the fractional tolerance convergence criterion. Iteration of the least squares fit ceases when the scaled distance between two successive estimates of the free variables is less than this value. Each

component of the scaled distance is scaled by dividing the difference between the two estimates by half their sum. Note that the fit solution may not converge to an arbitrarily small value; instead it may cycle between several values. Setting `ftol` to too small a value may therefore result in failure to converge.

The `maxiter` parameter sets the maximum number of iterations to be performed. If convergence is not achieved in this number of iterations, then the task stops with a warning message to that effect.

The `nprint` parameter specifies the number of iterations to perform between successive printings of diagnostic information to STDERR (usually your terminal). If `nprint=0`, there will be no diagnostic information printed. The diagnostic prints contain the iteration number, the chi-squared value, and the model spectrum with the current values of the variables.

The `slow` parameter selects which fitting method to use. If `slow=no` (the default), the Levenberg-Marquardt method is used. If `slow=yes`, the downhill simplex method is used.

The `equal` parameter indicates whether or not to weight the data points when computing the fit. If `equal=no` and the input table contains a column of error values, then the data points will be weighted by the errors. Points with indefinite, negative, or zero-valued errors are not used in the fit. If `equal=yes` or no valid error values are specified in the input table, then the data points will receive equal weights.

5.8.1 Examples

Let's fit a series of spectra from the Bruzual-Persson-Gunn-Stryker spectral library (see section A.7) to the spectrum of P177D, which is again taken from the HST standards directory `crcalspec`. The list of library spectra are specified in the file `grid.lis` as follows:

```
crgridbpgs$bpgs_33.fits
crgridbpgs$bpgs_35.fits
crgridbpgs$bpgs_36.fits
crgridbpgs$bpgs_45.fits
```

The fit results are saved in the table `p177d_fit.fits`. **fitgrid** is run as follows:

```
sy>    fitgrid    crcalspec$p177d_stis_001.fits    @grid.lis
out=p177d_fit.fits
```

The final solution is:

```
0.7155979 * 4.151929E-6 * (crgridbpgs$bpgs_45.fits) +
(1. - 0.7155979) * 4.382411E-6 * (crgridbpgs$bpgs_36.fits)
```

For a second example we'll make use of variable zero (`vzero`) to automatically generate a list of blackbody spectra of varying temperatures to be fit to the spectrum of the same star:

```
sy> fitgrid crcalspec$p177d_stis_001.fits "bb($0)" \
out=bb_fit.fits vzero=10e3-30e3x1e3 equal=yes
```

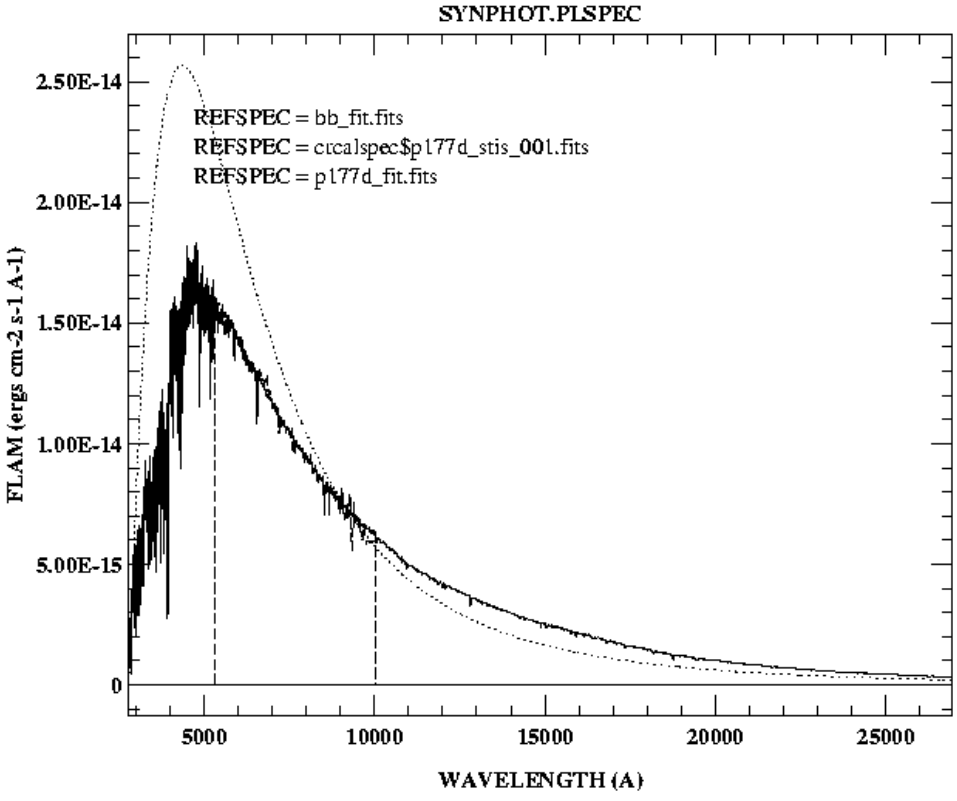
The `vzero` string gives us a list of blackbody spectra having temperatures from 10000 to 30000 K, in steps of 1000 K. The resulting spectrum is stored in the table `bb_fit.fits`. The final solution is:

```
5.574925 * 0.683679 * (bb(10000.)) +
(1. - 5.574925) * 0.5202777 * (bb(11000.))
```

We can then compare the models with the P177D spectrum using **plspec** as follows (see Figure 5.19):

```
sy> plspec "" bb_fit.fits flam ltype=dotted
sy> plspec "" crcalspec$p177d_stis_001.fits flam append+
sy> plspec "" p177d_fit.fits flam append+ ltype=dashed
```

Figure 5.19: Fitgrid Results



5.9 Genwave

The **genwave** task creates a wavelength set based on user-specified values for the minimum and maximum wavelengths and the sampling interval. The sampling interval may be expressed in terms of Angstroms per pixel or in terms of kilometers per second per pixel. The resulting wavelength set is stored in a FITS or STSDAS table. The task parameters are listed in Figure 5.20.

Figure 5.20: Genwave Parameters

```
      output =                  Wavelength set table name
    minwave =                Minimum of wavelength range (Angstroms)
    maxwave =                Maximum of wavelength range (Angstroms)
       dwave = INDEF           Wavelength interval (Angstroms/pixel)
(dvelocity = INDEF)            Velocity interval (km/s/pixel)
  (wavecol = WAVELENGTH)       Wavelength set table column name
```

The `output` wave set is a single column table with the column label specified by parameter `wavecol`. The default setting of `wavecol = WAVELENGTH` is the column name expected by other **synphot** tasks that use a user-specified wavelength set. The wavelength values written to the table are in units of Angstroms.

The wavelength interval between sample points, in Angstroms per pixel, is normally set by the value of the `dwave` parameter. If `dwave=INDEF`, then the value of the `dvelocity` parameter is used instead. In this case, the sample interval is expressed by the user in units of km per second per pixel, and the equivalent sampling in Angstroms per pixel is computed by the task. Note that the output wave set is always expressed in units of Angstroms, regardless of which sampling specification method was used.

There is a set of ASCII wavelength tables in the STSDAS `synphot$data` directory which can be used with any of the **synphot** tasks, so it is not always necessary to create custom wave sets. This directory contains tables for each grating and echelle order for the HRS and FOS instruments. The wavelength grids in these tables cover the range and are sampled at the (sometimes non-linear) resolution appropriate for each grating as used in a standard observing mode for each instrument.

5.9.1 Examples

The first example shows how to generate a wavelength set having a range of 3000 to 8000 Å, with a sampling interval of 2 Å per pixel. The wavelength set will be written to the table `wave2.fits` in a column labeled WAVELENGTH.

```
sy> genwave wave2.fits 3000 8000 2
```

Next, we'll generate a wavelength set having the same range as the one above, but have the sampling interval be equivalent to 100 km/s/pixel.

```
sy> genwave wave_kms.fits 3000 8000 INDEF dv=100
```

5.10 Grafcheck

The **grafcheck** task reads an instrument graph table and checks it for four types of errors:

- Undefined component name, keyword, input node number, or output node number.
- Output node number less than input node number.
- Identical keywords in two or more nodes with the same input node. (Leading blanks and case are ignored in determining uniqueness of keyword names; names that differ only in case are considered identical.)
- A row cannot be reached from the graph's starting node.

When errors are detected, a row will be printed for each type of error detected, along with the offending row. When a row is in error, the component name is displayed, followed by the keyword, the input node, and the output node numbers. Component names and keywords are converted to lower case in the output and are enclosed in quotation marks. No output is produced if no errors are detected. All output from the task is directed to STDOUT.

The only task parameter is `grftable`, which has the value "`mtab$*_tmg.fits`" by default, and is used to specify the name of the graph table to be checked.

5.10.1 Example

Check the graph table `hstgraph.fits` for errors:

```
sy> grafcheck hstgraph.fits
```

5.11 Graflist

The **graflist** task prints the names of all components in an instrument graph table downstream from a specified starting component. The specified starting component is referred to as the *root*. A component is considered to be *downstream* from the root if a path passes through both the root and the component and the root appears first in the path. A root is always considered to be downstream from itself, so at least one component name will always be printed when the task is executed.

The list of component names is printed in the order that they occur along the path. When component names are printed, each component name is indented to show the distance between it and the root.

There are two task parameters:

```
grftable="mtab$*_tmg.fits"     Instrument graph table
compname="mtab$*_tmc.fits"     Component name of starting
node
```

The `grftable` parameter specifies the name of the instrument graph table to be searched.

The `compname` parameter specifies the name of the root component. The component name is *not* case sensitive and leading and trailing blanks will not affect the match. If more than one component in the graph table matches `compname`, the component with the smallest value in the INNODE column will be used. The component name can be made unique by optionally specifying an INNODE number as part of `compname`. To do this, the INNODE number must follow the component name, separated by a white space. If no value is passed to `compname`, the entire graph will be listed.

5.11.1 Examples

List all components that are downstream from “hrs_echa”:

```
sy> graflist mtab$*_tmg.fits hrs_echa
```

List all components that are downstream from the component “clear” that has an INNODE value of 1000:

```
sy> graflist mtab$*_tmg.fits "clear 1000"
```

5.12 Grafpath

This task returns the ordered list of throughput filenames that comprise the model used by **synphot** when performing spectral calculations for a given observation mode. The other tasks compute the throughput for an observation mode by multiplying these individual files together. This task is useful for understanding the results returned by other tasks, and also for confirming that a graph table is returning the expected results for a specified obsmode.

The only task parameter (aside from the refdata PSET) is `obsmode`, which is specified in the usual way. All output from the task is directed to STDOUT.

5.12.1 Examples:

To examine the list of files used to model WFPC2 with the f122m filter, issue the command:

```
sy> grafpath wfpc2,f122m
```

This produces the following output:

```
crotacomp$hst_ota_007_syn.fits
crwfpc2comp$wfpc2_optics_006_syn.fits
crwfpc2comp$wfpc2_f122m_006_syn.fits
crwfpc2comp$wfpc2_dqewfc4_005_syn.fits
```

Here you can see the throughput files used for the HST OTA, the WFPC2 optics, the f122m filter, and the DQE of the detector. These filenames can then be examined using **tread** or by some other means.

The results for HST observation modes produce a list of files. Non-HST modes, such as those that simply specify a photometric bandpass, produce just one file that contains the bandpass definition as it is implemented in **synphot**. You can use **grafpath** to determine which file is used if the unadorned bandpass "r" is given:

```
sy> grafpath r
```

which produces as output

```
crnonhstcomp$cousins_r_003_syn.fits
```

5.13 lmspec

The **imspec** task can be used to convert one-dimensional images into STSDAS or FITS tables and to convert **synphot**-style tables into images. The type of conversion to perform (image to table, or table to image) is determined automatically from the type of the input file. More than one file may be converted at a time by using a list of input and output files. Wavelength information may also be copied along with flux values (see below). The task parameters are listed in Figure 5.21.

Figure 5.21: lmspec Parameters

```
input    =                List of input files
output   =                List of output files
(wave    = " ")           Optional wavelength images
(inform  = "counts")      Form of input spectrum
(outform = "counts")      Form of output spectrum
(olength = INDEF)         Length of output file
(badpix  = 0.)            Value to substitute for INDEF in output image
(refdata = "")            Reference data
```

The input and output file types may be either images or tables. The task will determine the type of the specified input file and create an output file of the opposite type.

If a wavelength file name is specified, the number of wavelength files must match the number of input and output files.

If both input and output files are FITS files, no header information is preserved. **imspec** constructs minimal headers for the output files.

5.13.1 If the input file is an image:

If the input file is an image, then the output file will be a FITS or STSDAS table containing columns of WAVELENGTH and FLUX. The wavelength values will be in units of Angstroms. The units of the flux values may be selected using the outform parameter (see below).

If no wavelength file is specified, the World Coordinate System (WCS) information in the input image header will be used to generate a wavelength set to be written to the output table.

If a wavelength image is supplied along with an input image, the WCS in the input image header is ignored and the wavelengths are copied from the supplied wavelength image, which are assumed to be in a one-to-one correspondence with the flux values in the input image.

If the input file is a multigroup (GEIS) image, only one data group can be copied at a time. If the group number is not specified in the input file name, the first group will be copied.

If the input file is a multi-extension FITS image, you must specify which HDU is to be copied in the filename, e.g. `manyimages.fits[2]` to copy the second extension.

If the output file is a FITS table, the table data will be placed in the first extension of the output FITS file. A primary HDU will be created with a minimal header.

5.13.2 If the input file is a table:

If the input file is a table, the output file will be a one-dimensional image. The number of output files must match the number of input files.

The input table must contain two columns labeled WAVELENGTH and FLUX (or FLUX1). All other columns in an input table will be ignored. The values in the wavelength column must be in monotonic order. If wavelength and flux units are specified in the table header, they must be units supported by the **synphot** package.

If the input file is a table and a wavelength image is specified, the flux column in the input table is copied to the output image, and the wavelength column in the input table is copied to the specified wavelength image.

If the output file is an image and no wavelength image is specified, WCS information will be computed from the input wavelength array and written to WCS header keywords in the output image.

If the input file is a multi-extension FITS table, and no extension is specified in the filename, its first extension will be copied into an image. Other extensions may be specified as part of the input filename, e.g. `manytables.fits[2]` will select the second extension.

If the output file is a FITS image, the image data will be placed in the primary HDU of the FITS file.

5.13.3 Other parameters:

The `inform` parameter is used to specify the flux units when the input file is an image. If the input file is a table, the flux units are read from the flux column units in the table header and this parameter is ignored (unless the column units are blank). The `outform` parameter is used to specify the desired output flux units. Units conversion, if necessary, is performed by the task.

The `olength` parameter may be used to specify the desired length of the output file. If set to INDEF (the default), the length of the output file

will be the same as the length of the input file. If the output file is an image, `olength` specifies the number of image pixels. If the output is a table, `olength` specified the number of table rows.

The value of the `badpix` parameter is used to replace INDEF flux values from an input table when they are copied to an output image.

As with all other **synphot** tasks, the parameter `refdata` specifies the name of the parameter set (pset) containing certain necessary reference data. The only parameter from this pset used by the **imspec** task is `area`, the telescope collecting area.

5.13.4 Examples

Copy the **synphot**-format table for the standard star GD71, contained in table `gd71_stis_001.fits` in the `crcalspec$` directory, to the image `gd71_image.fits`. Since no wavelength image name is specified, the wavelength data in the input table will be used to compute WCS header keyword values for the output image.

```
sy> imspec crcalspec$gd71_stis_001.fits gd71_image.fits out-
form=flam
```

Copy an FOS spectral image into a table with the same rootname as the input image, but an extension of `.fits`. Use the corresponding FOS wavelength image (image with a `.c0h` extension) to specify the wavelength array to be written to the output table. Because the input file is an image, we must tell the task what the input flux units (`inform`) are.

```
sy> imspec y15v0403t.c1h y15v0403t.fits wave=y15v0403t.c0h \
>>> inform=flam outform=flam
```

(Although only the header files (.c0h,.c1h) are explicitly mentioned in the command, the corresponding data files (.c0d, .c1d) must also be present in the directory.)

5.14 Mkthru

The **mkthru** task converts an ASCII file, FITS table, or STSDAS binary table into a throughput table for installation in CDBS. The table will have the required header keywords, column names, column units, and column formats.

If the necessary header keywords are not present in the input file, the task will query the user for the appropriate values. Because the task parameters that read this information use query mode, you will be queried for this information even if you set them in the parameter editor. (The values you set will be displayed as default values.)

Column names can be specified in ASCII files by placing them on the first line and setting the hidden parameter `title` to `yes.` The task uses default values for the column information if it is not present in the input file. The default column names are WAVELENGTH, THROUGHPUT, and ERROR. The default column units are ANGSTROMS, TRANSMISSION, and TRANSMISSION. The default print formats are%10.1f,%12.5g, and%12.5g. (If necessary, these values can be later modified by using the tchcol task on the output file.)

The task parameters are listed in Figure 5.22.

Figure 5.22: Mkthru Parameters

```
   input = filename template
instrume = instrument name
compname = Component name
useafter = Useafter date
pedigree = Reference file pedigree
 descrip = Description of file
 comment = Comment about file
  (title = no) First line of file has column names
(verbose = no) Verbose message switch
```

The `input` parameter is a list of input file names. File names may include wild cards. Output files have the same names as the input files, but their extension is changed to fits.

Except for title and verbose, all the remaining parameters for this task function as default values for the header keyword of the output file, as described above. The task will look for the matching header keyword in the input file first; only if it is not found will it use the value in the task parameter.

The `instrument` parameter specifies the name of the telescope instrument used in the observation.

The `compname` parameter specifies the name of the component associated with the throughput file.

The `useafter` parameter specifies the start of the date range for which the throughput file is valid. The date should be in the form "mmm dd yyyy" where "mmm" is a three letter month abbreviation.

Important: The `useafter` parameter functions differently for **synphot** files than for other CDBS reference files. The CDBS tool used to construct the component lookup table always selects the throughput file with the latest `useafter` date. Therefore, this parameter should always be set to the current date to ensure that the new throughput file will be picked up.

The `pedigree` parameter records the source of the information in the throughput file. It should have one of the following values: INFLIGHT, GROUND, MODEL, or DUMMY.

The `descrip` parameter contains a short description of the throughput file.

The `comment` parameter contains a comment on the throughput file.

If the `title` parameter is set to `yes`, the column names are taken from the first line of the input file. Otherwise, no column names are produced for the output file.

If the `verbose` parameter is set to yes, the task will display a message after each file is converted to FITS

5.14.1 Examples

Convert a set of ASCII files to FITS format, and let the task prompt for all necessary header keywords.

```
sy> mkthru *.dat
```

Convert an STSDAS binary table (which already contains the correct header keywords) to FITS format

```
sy> mkthru hst_dark.fits
```

5.15 Modeinfo

This task displays a list of the observation mode keywords available for a particular instrument or filter system, along with information about the use and application of the keywords. For example, `modeinfo stis` displays information about all the available observation mode keywords for the HST STIS instrument and explains their use and significance to the instrument. This task can also be used to display observation mode information for some of the non-HST filter systems that are supported in **Synphot**, such as the ground-based "UBV" system.

If the user does not supply an `instrument` parameter, or supplies one for which no information is available, the task will list the available instrument and filter system names.

The only task parameter is `instrument`, which is used to specify the name of the HST instrument to be checked.

5.15.1 EXAMPLES

To display observation mode information for NICMOS:

```
sy> modeinfo nicmos
```

To display observation mode information for the ground-based UBVRI filter system:

```
sy> modeinfo ubv
```

To display a list of instrument/filter system names for which information is available:

```
sy> modeinfo ""
```

5.16 Obsmode

The **obsmode** task displays a list of the observation mode keywords contained in the instrument graph table, usually for a single instrument. For example, "obsmode foc" displays the list of valid observation mode keywords for the FOC. The task is very helpful when you are having trouble remembering the names of all the available keywords for a given instrument.

The output is structured so that keywords representing a group of alternative elements at a given point within the instrument's optical path are placed on the same line. An observation mode (`obsmode`) string could contain no more than one of these keywords at a time. Long lists of keywords are wrapped, however, so that they are able to display on the terminal screen. It should be obvious from the keyword names when a long list has been wrapped.

Individual keywords from the graph table are used within the `obsmode` string of other **synphot** tasks to specify a unique light path through the telescope and instrument. The throughputs of the individual components in the light path are combined to determine a total throughput for a given observing mode. The keywords contained in the path string are dependent on the structure of the graph table. Default keywords are often allowed in the path string, but it is safest to explicitly include all desired components. In particular, in the current HST graph table, Johnson is the default filter system for *UBV*, and Cousins is the default for *RI*.

The **obsmode** task has the following two parameters:

```
path =                      Partial observation mode path
(refdata = "")              Reference data
```

The `path` parameter is used to specify a (partial) observation mode which selects the starting node for the output information. The keywords displayed by the task will be the descendants of the last node matched by

any of the keywords in the `path` string. Usually, the value of this parameter is a single keyword specifying the name of an instrument. For example, “acs” specifies that all the keywords that are descendants of the acs node in the graph table are to be displayed. If this parameter is left blank or set to “none”, all the keywords in the graph table will be displayed.

The `refdata` parameter specifies the name of the parameter set containing the necessary reference data for the task. In the case of the **obsmode** task, the only important reference data is the name of the instrument graph table (`grtbl`) to be searched.

5.16.1 Examples

Display the observation mode keywords for the WFPC2:

```
sy> obsmode wfpc2
```

Display the list of central wavelengths available for STIS with the e140h grating:

```
sy> obsmode stis,e140h
```

Display all the keywords in the graph table:

```
sy> obsmode ""
```

5.17 Plband

The **plband** task is similar to **calcband**, except that it produces a plot as its output rather than a table, and is used to plot a set of synthetic or instrumental passbands. The user has control over certain plot attributes, such as the displayed plot limits and line styles, and can append new plots to a previous one that was generated by **plband**.

The task parameters are listed in Figure 5.23.

Figure 5.23: Plband Parameters

```
      obsmode =                  Instrument mode or @list
        (left = INDEF)           x value for left side of plot
       (right = INDEF)           x value for right side of plot
      (bottom = INDEF)           y value for bottom of plot
         (top = INDEF)           y value for top of plot
   (normalize = no)              Normalization all curves to 1?
        (ylog = no)              Take log of y values?
      (append = no)              Append to existing plot?
     (ltype = solid)          Line type: clear,solid,dashed,dotted,dotdash
      (device = stdgraph)        Graphics device
    (wavetab =  )                Wavelength table name
    (refdata = )                 Reference data
```

The `obsmode` parameter specifies the passband(s) to be plotted. Individual passbands can be added, subtracted, multiplied or divided by one another using the +, -, *, and / operators, respectively. See Section 3.2.2 for more details concerning all of the supported mode functions. Lists of `obsmode` strings may be placed in a text file, one per line, and specified as `obsmode=@`*`filename`*. (See "Filename specifications" on page 19 for important side effects of using the `@filename` syntax).

The `left`, `right`, `bottom`, and `top` parameters specify the minimum and maximum wavelength and throughput limits for the plot. If set to INDEF, the task will set them based on the limits of wavelength set and passband values.

The `normalize` parameter can be used to normalize all passbands to a maximum value of 1. If `normalize=no`, each band is plotted at its intrinsic level. The `ylog` parameter can be used to create a plot with the y-axis in logarithmic units. If `append=yes`, the results will be overplotted on an existing plot.

The `ltype` parameter is used to specify the line type to be used for plotting the passband. The available line types are: `solid`, `dashed`, `dotted`, `dotdash`, and `clear`. If a list file is used to specify the bands to be plotted, then one line type is used for all the passbands in the list.

Null values for the `wavetab` and `refdata` parameters will cause the task to use a default wavelength grid (See "Wavelength Table" on page 25.) and the default reference data for the HST observatory (see Section 3.5). The default wavelength grid covers the wavelength range where the passband throughput is non-zero. Wavelengths are spaced logarithmically over this range. If there is more than one passband specified, the range of the default wave set is computed based on the first passband.

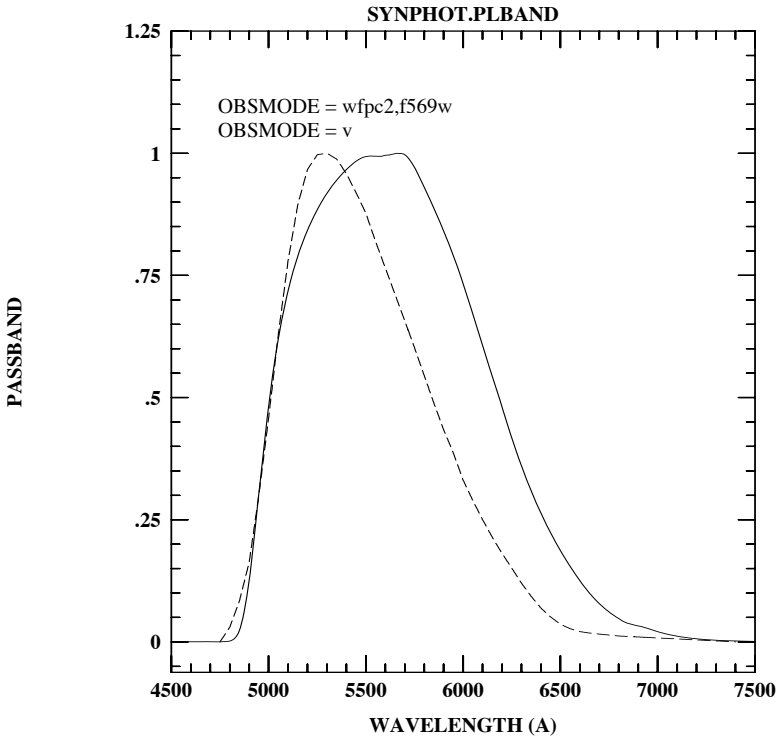
If the wavelength ranges of the passbands differ significantly, a suitable wavelength table that covers the range of all the passbands should be constructed using the **genwave** task, and supplied as input via the `wavetab` parameter.

5.17.1 Examples

Plot the WFPC2 F569W passband and compare it to the standard Johnson *V* passband (overplotted as a dashed line). Normalize both curves to a maximum of 1 and limit the wavelength range of the plot to 4000–7500 Å. Results are shown in Figure 5.24.

```
sy> plband wfpc2,f569w left=4000 right=7500 top=1.25 norm+
sy> plband v norm+ app+ ltype=dashed
```

Figure 5.24: Results of First Sample Plband Run



For the next example, plot the Johnson *UBVRI* passbands and overplot the Cousins *RI* passbands (using a dashed line) for comparison. The Johnson bands are listed in the file `johnson_ubvri.lis` as:

```
johnson,u
johnson,b
johnson,v
```

```
johnson,r
johnson,i
```

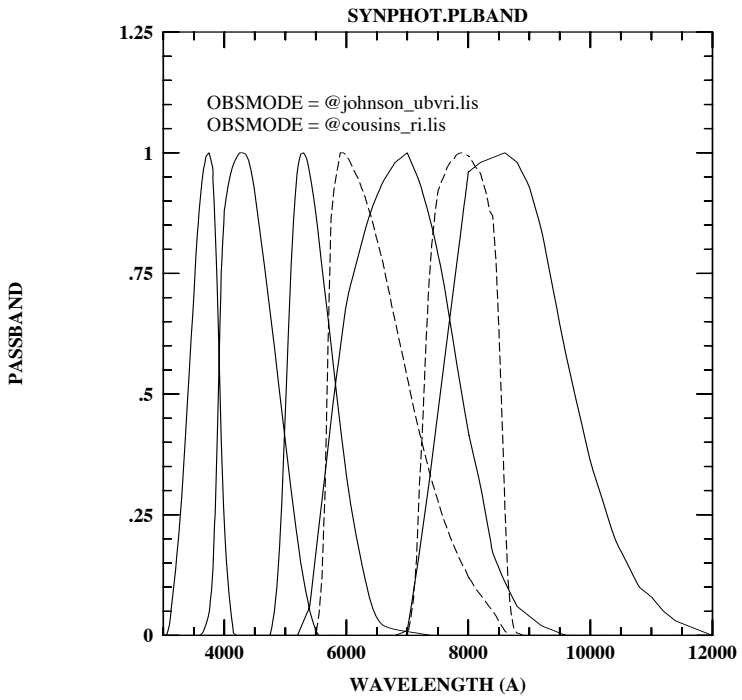
Similarly, the Cousins *RI* passband names are contained in the file `cousins_ri.lis` as:

```
cousins,r
cousins,i
```

Because the individual passbands have very different wavelength limits, it's necessary to generate a wave set (using **genwave**) that runs from 3000 to 12000 Å. This wave set is in table `johnson_wv.fits`. Results are shown in Figure 5.25.

```
sy> plband @johnson_ubvri.lis wav=johnson_wv.fits top=1.25
sy> plband @cousins_ri.lis wav=johnson_wv.fits app+
lt=dashed
```

Figure 5.25: Results of Second Sample Plband Run



5.18 Plspec

The **plspec** task is similar to the **calcspec** task, except that it plots its output instead of writing it to a table. It also has additional capabilities, such as being able to plot error bars if the input table contains error information and plotting photometric results from **calcphot**. In normal

operation **plspec** plots synthetic spectra generated from observation mode and spectrum expressions.

More than one spectrum can be plotted at a time by creating text files with one observation mode or passband on each line. The parameter `vzero` can also be used to substitute a set of values for variable zero (`$0`) in the spectra. All combinations of the observation modes, spectra, and values of `vzero` will be plotted.

The task parameters are shown in Figure 5.26.

Figure 5.26: Plspec Parameters

```
obsmode  =                  Observation mode or @list
spectrum =                  Synthetic spectrum or @list
form     =                  Form of output graph
(vzero   = " ")             List of values for variable zero values
(spfile  = "none")          Spectrophotometry data
(pfile   = "none")          Photometry data
(errtyp  = "n")             n[one] p[oint] c[ont] b[in] v[ert] h[oriz]
(left     =  INDEF)          x value for left side of graph
(right   = INDEF)           x value for right side
(bottom  = INDEF)           y value for bottom
(top     = INDEF)           y value for top
(append  = no)              Append to existing plot?
(ltype   = "solid")         Line type: clear, solid, dashed, dotted,dotdash
(device  = "stdgraph")      Graphics device
(wavetab = "")              Wavelength table name
(refdata  =  "")             Reference data
```

The `obsmode` parameter is used to specify the desired passband to be applied to the spectral data (see Section 3.2.2). Several `obsmode` strings may be processed by inserting them one per line in a text file and setting `obsmode=@`*`filename`*. (See “Filename specifications” on page 19 for important side effects of using the `@filename` syntax.) If this parameter is left blank or set to “none”, a default passband, equal to unity at all wavelengths, will be used.

The `spectrum` parameter is used to specify a synthetic spectrum to be generated (see Section 3.2.3). This parameter also accepts input from list files by setting `spectrum=@`*`filename`*. If this parameter is left blank or set to “none”, no synthetic spectra will be plotted.

The `form` parameter specifies the desired units for the plots and photometric calculations and can be any one of the standard **synphot** forms: `fnu`, `flam`, `photnu`, `photlam`, `counts`, `abmag`, `stmag`, `obmag`, `vegamag`, `jy`, or `mjy` (see Section 3.2.4). If `form` is set to either `counts` or `obmag`, all quantities are multiplied by the telescope area before plotting.

The `vzero` parameter contains a list of values that are substituted for variable zero (`$0`) wherever it appears in the spectrum expression. See Section 3.4, "Variable Substitution (VZERO)," on page 26 for a discussion of variable substitution.

The `spfile` parameter specifies the name of a table containing spectrophotometry data, i.e., an observed spectrum. A list of one or more files can be specified using the `spfile=@`*`filename`* syntax. Tables are expected to have three columns labeled WAVELENGTH, FLUX, and STATERROR. The STATERROR column can be all INDEF values. If the STATERROR column is not found, an array of INDEF error values will be generated by the task. The table may also contain a FWHM column, specifying the effective instrumental resolution for each data point. If a FWHM column is not found, an array of INDEF values will be generated. The STATERROR and FWHM data may be incorporated into the plot using the `errtyp` parameter (see below).

If an ASCII text file is used for `spfile`, the first through fourth columns must contain the wavelength, flux, staterror, and FWHM data. The third and fourth columns are optional. Because ASCII files cannot contain column units specifications, the wavelengths are assumed to be in Angstroms and the fluxes in units of `photlam`.

The `pfile` parameter is used to specify the name of a table containing photometric data in the same format as that produced by the **calcphot** task. A list of files can be passed as `pfile=@`*`filename`*. A table file must have the column names COUNTRATE (or DATUM or FLUX for compatibility with previous versions), FORM, OBSMODE, and TARGETID. An ASCII text file must have four columns in the following order: photometric data value, form, observation mode, and target ID string. The TARGETID column is not used by this task. The data are plotted as horizontal bars, with the midpoints marked by circles. The midpoint is located at the pivot wavelength of the passband. The length of the horizontal bar is equal to the FWHM of the equivalent gaussian of the passband.

The `errtyp` parameter may be used to specify how the statistical errors (if they exist) for the spectrophotometry (`spfile`) data should be plotted. The codes are described in Table 5.5.

Table 5.5: Error Type Codes for Plspec

Code	Plots Error As...
`n`	None (default)
`p`	Plot data as points
`b`	Plot data as bins (histogram)
`c`	Plot +/- vertical error as continuous lines
`v`	Plot vertical error bars
`h`	Plot horizontal error bars

Options "v" and "h" must be used in conjunction with one of the "p", "c", or "b" options. If only "p" is selected, then the flux data will be plotted as individual points. If "v" or "h" are included along with "p", e.g., `errtyp = pv`, or `ph`, or `pvh`, then vertical or horizontal error bars, or both, will be plotted with each flux data point. If the "c" option is selected, two continuous lines will be plotted at values corresponding to *flux + error* and *flux – error*. The "ch" option will plot horizontal error bars at the location of the flux values (between the continuous lines). The "b" option will plot the flux data in bins (histogram mode). The "bv" option will overplot vertical error bars on the histogram bins. In all cases, the vertical error corresponds to the statistical error values and the horizontal error corresponds to the FWHM values read from the `spfile`. Individual error values that are INDEF will not be plotted.

The `left`, `right`, `bottom`, and `top` parameters may be used to specify exact values for the wavelength and flux ranges to be plotted. If set to INDEF, the axes will be auto-scaled based on the combined ranges of the synthetic and observed spectral data and the photometric data. If `append=yes`, the results will be overplotted on an existing plot.

The `ltype` parameter specifies the line type to be used for plotting the data. The allowed values are `solid`, `dashed`, `dotted`, and `dotdash`.

The `device` parameter specifies the plotting device on which to produce the plot, e.g., `stdgraph` for on-screen display, or `stdplot` for hard copies.

The plot that is produced will contain the following:

- First, the spectral data specified by `spectrum` are plotted. These data are multiplied by the passband(s) specified by `obsmode`. If `form` is set to `counts` or `obmag`, the spectral data are plotted as a histogram. Data for all other `form` types are plotted as a continuous

line. If variable zero (`$0`) is used within the `spectrum` parameter, the spectral data are plotted multiple times, with each curve corresponding to a different value in the `vzero` list.

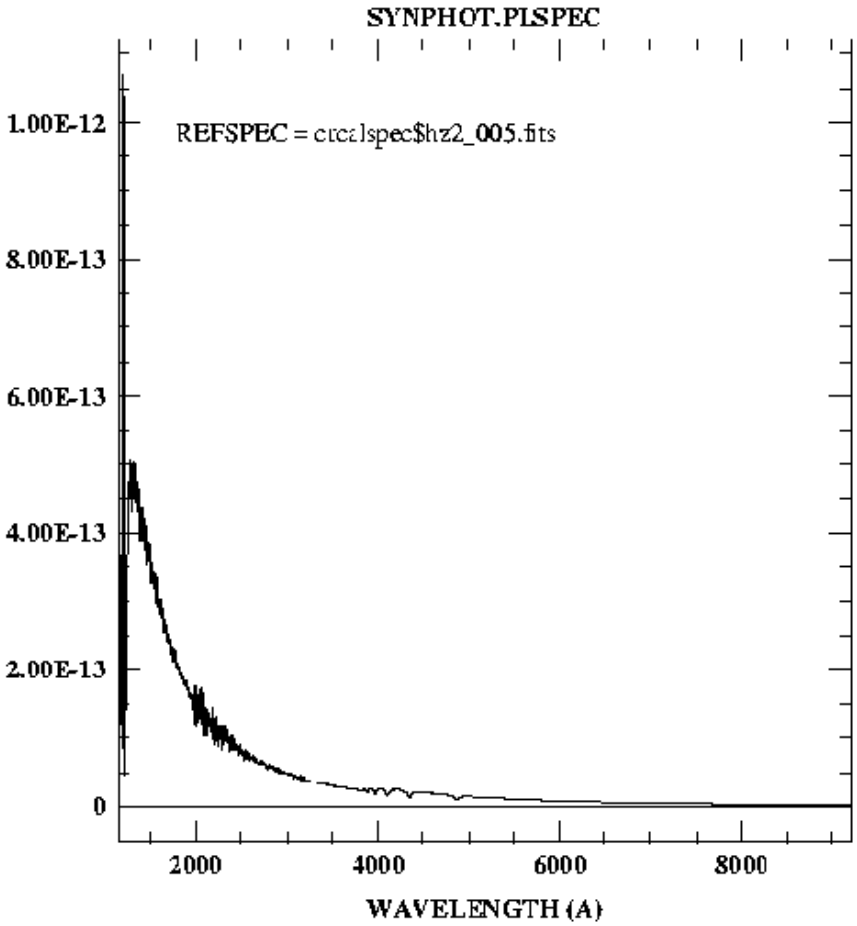
- Next, the spectrophotometry data (if any) specified via `spfile` will be plotted after any necessary form conversion. Error bars corresponding to the setting of `errtyp` will also be plotted.
- Finally, photometric data read from `pfile`, if any, are plotted after any necessary form conversion. The passband(s) corresponding to the `obsmode` of the `pfile` data will also be plotted at the bottom of the graph. The photometry data points will be plotted as horizontal bars at a vertical location corresponding to the photometric value of the data point. The bar will be centered horizontally at the pivot wavelength and will have a width equal to the FWHM of the `obsmode` corresponding to that data value. Note that photometric data that are the result of a two-mode calculation, such as a color index or flux ratio of two passbands, can *not* be plotted because they are the result of calculations involving two passbands.

5.18.1 Examples

The first example simply plots the spectrum, in units of `flam`, of the flux calibration standard star HZ 2from our directory of HST calibration spectra. Note that we leave the `obsmode` parameter null (“ ”) so that the spectrum is not multiplied by a passband. The resulting plot is shown in Figure 5.27.

```
sy> plspec "" crcalspec$hz2_005.fits flam
```

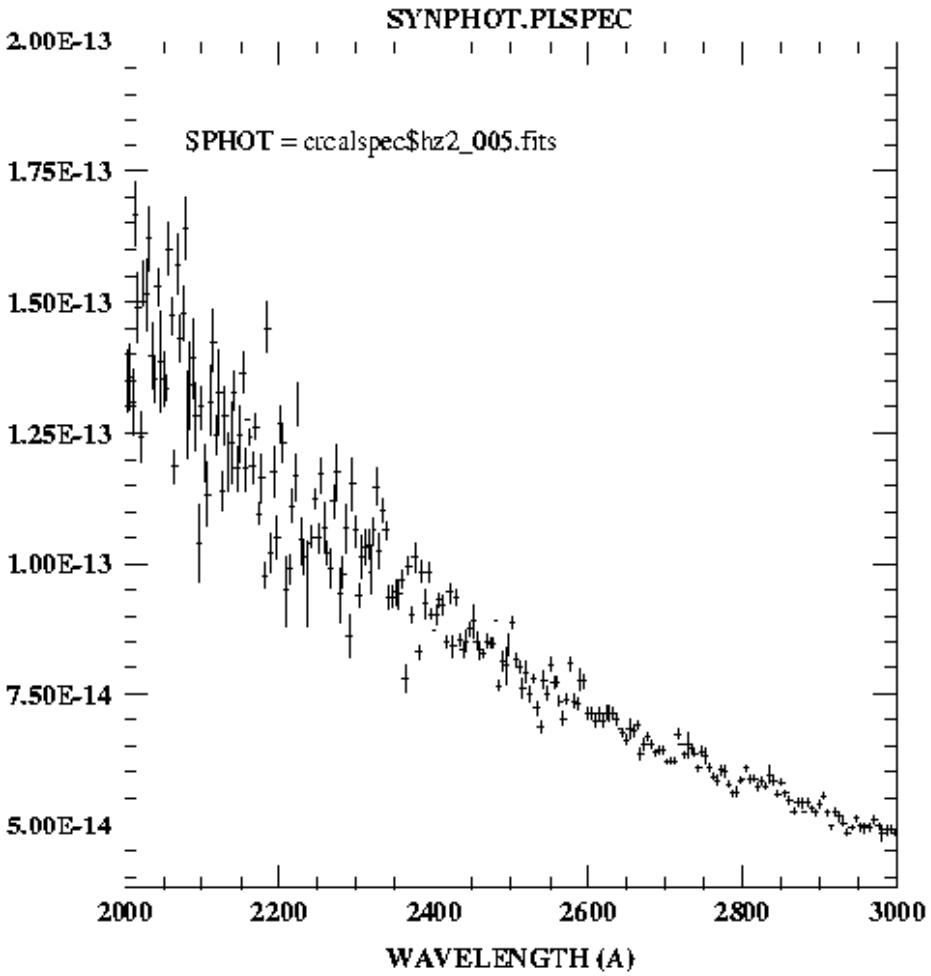
Figure 5.27: Plspec Results



The second example plots a portion of the spectrum for the same star, but this time we specify the spectrum table name using the `spfile` parameter (leaving the `spectrum` parameter blank) so that we can also plot the error data from the table. We use the `errtyp` parameter to plot the data as individual points with vertical and horizontal bars indicating the flux uncertainties and the effective wavelength resolution for each point. Figure 5.28 shows the results.

```
sy> plspec "" "" flam spfile=crcalspec$hz2_005.fits \
>>> err=pvh left=2000 right=3000 top=2e-13
```

Figure 5.28: Results from Second Sample Plspec Run

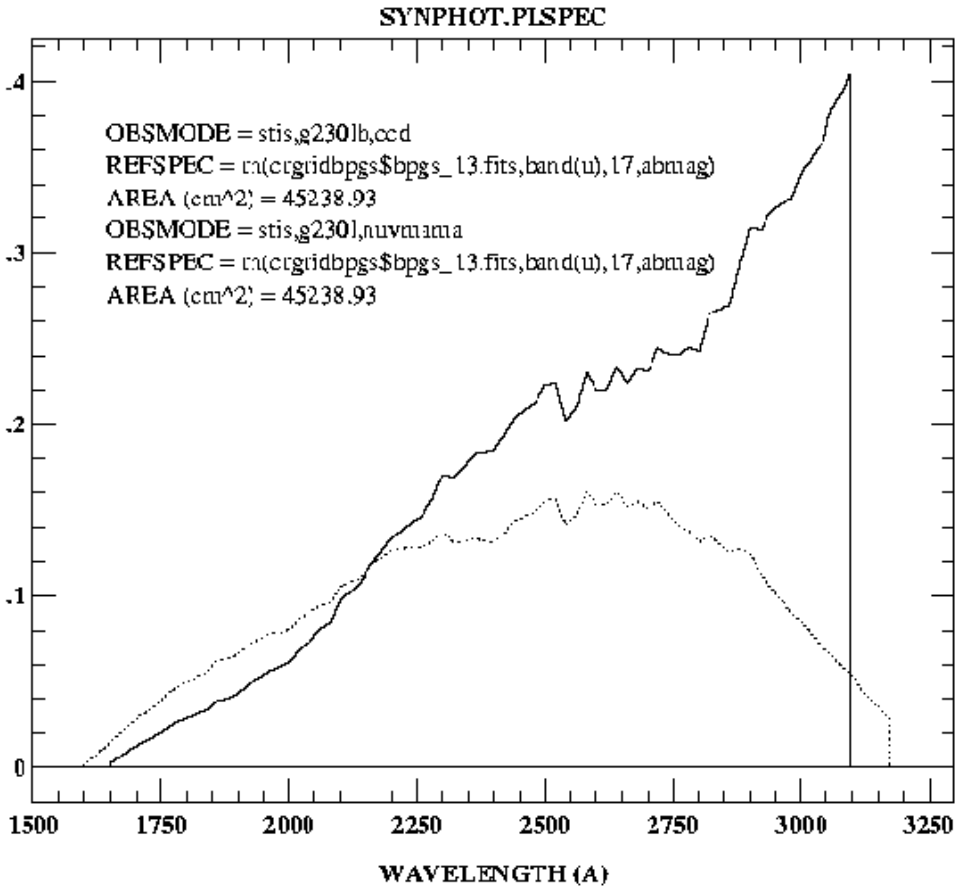


The next example performs a comparison of two STIS detection modes by overplotting the countrate spectra for a B9 star in the two modes. We'll compare the g230lb grating used with the CCD detector to the g230l grating used with the NUVMAMA.We'll use the spectrum of HD 189689, star number 13 in the Bruzual-Persson-Gunn-Stryker (BPGS) atlas, and renormalize it so that it has a U magnitude of 17. Here are the (somewhat lengthy) commands to perform this:

```
sy>        plspec        stis,g230lb,ccd        "rn(crgridb-
pgs$bpgs_13.fits,band(u),17,abmag)"     counts     left=1500
right=3300 ltype=solid

sy>       plspec       stis,g230l,nuvmama       "rn(crgridb-
pgs$bpgs_13.fits,band(u),17,abmag)"    counts    ltype=dotted
append=yes
```

The results are shown in Figure 5.29.

Figure 5.29: Plspec Comparison of STIS Sensitivities



The final example plots the results of some calculations made with **calcphot**. The integrated counts in the four WFPC2 filters F336W, F439W, F555W, and F814W were calculated for star number 23 in the BPGS atlas, renormalized to a *V* magnitude of 17.4. Because these filters cover disjoint wavelength ranges, **genwave** was used to create a wavelength table that covers the range 2800 to 11000 Å, in 10 Å steps. This table is called `wf_wave.fits`. The text file `wf_filters.lis` is also created, and contains the four `obsmodes`:

```
wfpc2,2,f336w,a2d7
wfpc2,2,f439w,a2d7
wfpc2,2,f555w,a2d7
wfpc2,2,f814w,a2d7
```

The **calcphot** command is as follows:

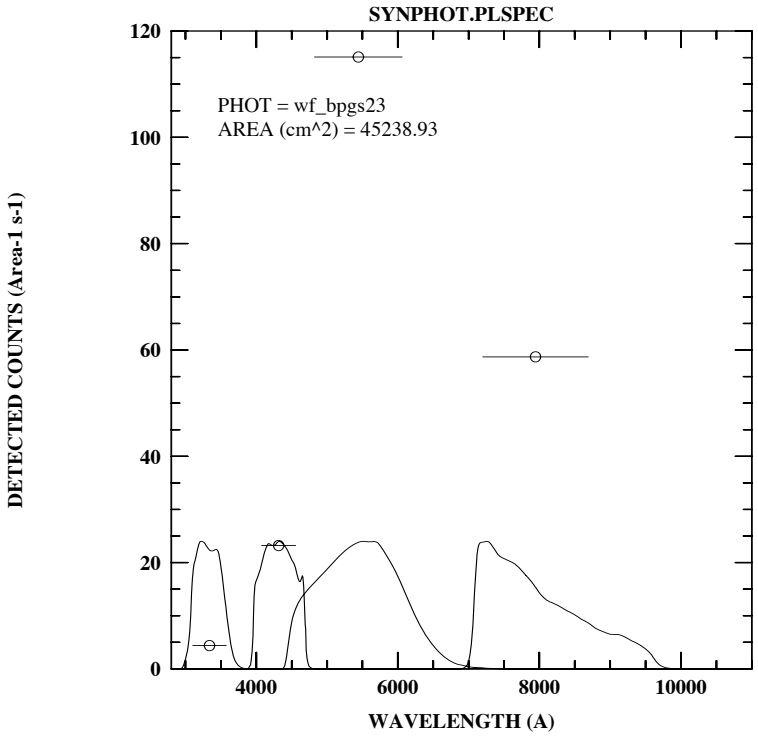
```
sy> calcphot @wf_filters.lis \
>>> "rn(crgridbpgs$bpgs_23.fits,band(v),17.4,vegamag)" \
>>> counts out=wf_bpgs23.fits wave=wf_wave.fits
```

Now **plspec** is executed to plot the results:

```
sy> plspec "" "" counts pfile=wf_bpgs23 wave=wf_wave.fits
```

The resulting plot, shown in Figure 5.30, shows that the integrated count rates range from about 5 DNs per second in the F336W filter, to a maximum of about 115 DNs per second in the F555W filter. Note that the passbands plotted at the bottom have been normalized to a common maximum value.

Figure 5.30: Plotting Calcphot Results

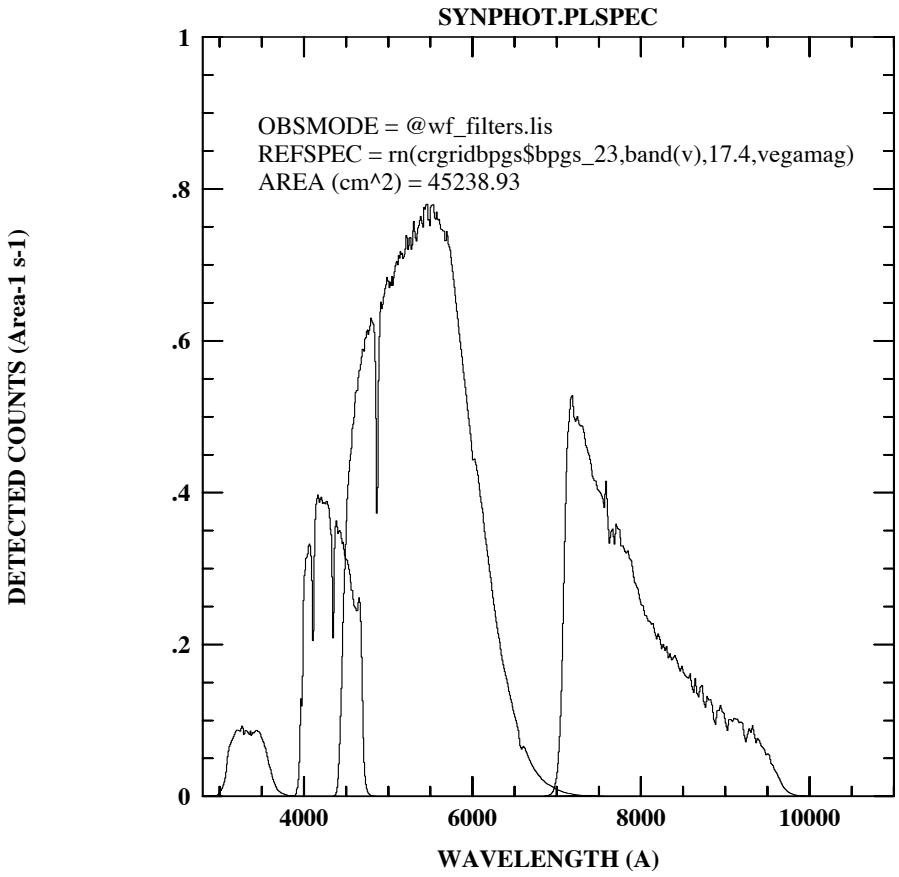


Attempting to combine these two operations (**calcphot** and **plspec**) into one execution of **plspec** would not produce the same type of results. For example, if we execute:

```
sy> plspec @wf_filters.lis \
>>> "rn(crgridbpgs$bpgs_23.fits,band(v),17.4,vegamag)"
counts \
>>> wave=wf_wave.fits top=1.1
```

we obtain plots of the *distribution* of detected counts over the range of each passband, as opposed to the total *integrated* counts within each passband, which is what we get from **calcphot** (Figure 5.31).

Figure 5.31: Distribution of Detected Counts



5.19 Plratio

The **plratio** task will calculate and plot the ratio of observed spectrophotometric data to a synthetic spectrum. It can also plot the ratios of previously calculated photometric data to the same photometric data calculated from the synthetic spectrum. This task produces plots similar to those of **plspec**, except that it always plots the *ratios* of observed data (the spectrophotometric and photometric table data) to synthetic data (the combination of obsmode and spectrum). The task also computes and plots the chi-squared error of the fit, and the bias and rms errors in magnitudes.

The task parameters are shown in Figure 5.32.

Figure 5.32: PIratio Parameters

```
 obsmode =                  Observation mode or @list
spectrum =                  Synthetic spectrum or @filename
    form =                  Form of output graph
  spfile =                  Spectrophotometry or @filename
  (pfile = none)            Photometry or @filename
  (vzero =  )               List of values for variable zero
   (left = INDEF)           x value for left side of graph
  (right = INDEF)           x value for right side
 (bottom = INDEF)           y value for bottom
    (top = INDEF)           y value for top
 (append = no)              Append to existing plot?
 (ltype = solid)         Line type: clear,solid,dashed,dotted,dotdash
 (device = stdgraph)        Graphics device
(wavetab = )                Wavelength table
(refdata = )                Reference data
```

Spectrophotometric ratios are plotted as histograms if `form` is set to `counts` or `obmag`, and as continuous lines otherwise. Photometric ratios are plotted as horizontal lines whose midpoints are marked by a circle. The x-location of the midpoint is equal to the pivot wavelength of the passband associated with the photometric value, and the length of the horizontal line is equal to the FWHM of the equivalent gaussian of the passband. The shapes of the passbands are overplotted at the base of the plot.

The type of plots produced by the task is governed by the settings of the `spfile` and `pfile` parameters. If spectrophotometric ratios are not desired, `spfile` should be set to "none" or left blank. If photometric ratios are not desired, `pfile` should be set to "none" or left blank. Spectrophotometric ratios are computed for every combination of synthetic spectra and spectrophotometric files. Synthetic spectra are matched to rows in the photometric (`pfile`) table by matching the `obsmode` and `spectrum` parameter strings to the strings in the OBSMODE and TARGETID columns in the photometric table (see below). The strings must be exactly the same to make a match.

The `obsmode` parameter is used to specify the desired passband to be applied to the spectral data (see Section 3.2.2). Several `obsmode` strings may be processed by inserting them one per line in a text file and setting `obsmode=@`*`filename`*. (See "Filename specifications" on page 19 for important side effects of using the `@filename` syntax). If this parameter is left blank or set to "none", a default passband, equal to unity at all wavelengths, will be used. Note that if photometric errors are being computed and `obsmode` is blank, the `form` parameter must be set to `counts` or `obmag`, otherwise the task cannot compute the effective stimulus of the synthetic spectrum.

The `spectrum` parameter is used to specify a synthetic spectrum to be generated (see Section 3.2.3). This parameter also accepts input from list files by setting `spectrum=@`*`filename`*. If this parameter is left blank or set to "none", no synthetic spectra will be plotted.

The `form` parameter specifies the desired units for the plots and photometric calculations and can be any one of the standard **synphot** forms: `fnu`, `flam`, `photnu`, `photlam`, `counts`, `abmag`, `stmag`, `obmag`, `vegamag`, `jy`, or `mjy` (see Section 3.2.4). If `form` is set to either `counts` or `obmag`, all computed quantities are multiplied by the HST collecting area before plotting.

The `vzero` parameter contains a list of values that are substituted for variable zero (`$0`) wherever it appears in the spectrum expression. See Section 3.4, "Variable Substitution (VZERO)," on page 26 for a discussion of variable substitution.

The `spfile` parameter specifies the name of a table containing spectrophotometry data, i.e., an observed spectrum. A list of one or more files can be specified using the `spfile=@`*`filename`* syntax. If the value of this parameter is set to "none" or left blank, the ratio of the spectrophotometric data to the synthetic spectra will not be plotted. In this case, the `pfile` parameter should have a value other than "none".

Spectrophotometric tables are expected to have three columns labeled WAVELENGTH, FLUX, and STATERROR. The STATERROR column can be all INDEF values. If the STATERROR column is not found, an array of INDEF error values will be generated by the task. The table may also contain a FWHM column, specifying the effective instrumental resolution for each data point. If a FWHM column is not found, an array of INDEF values will be generated. If an ASCII text file is used for `spfile`, the first through fourth columns must contain the wavelength, flux, staterror, and FWHM data. The third and fourth columns are optional. Because ASCII files cannot contain column units specifications, the wavelengths are assumed to be in Angstroms and the fluxes in units of `photlam`.

The `pfile` parameter is used to specify the name of a table containing photometric data in the same format as that produced by the **calcphot** task. A list of files can be passed as `pfile=@`*`filename`*. If the value of this parameter is "none" or left blank, the ratio of photometric data to the synthetic spectral data will not be plotted. A `pfile` table must have the column names COUNTRATE (or DATUM or FLUX for compatibility with previous versions), FORM, OBSMODE, and TARGETID. An ASCII text file must have four columns in the following order: photometric data value, form, observation mode, and spectrum (target ID) string. The task matches the photometric data to the data generated from the `obsmode` and `spectrum` parameters by the strings in the OBSMODE and TARGETID columns.

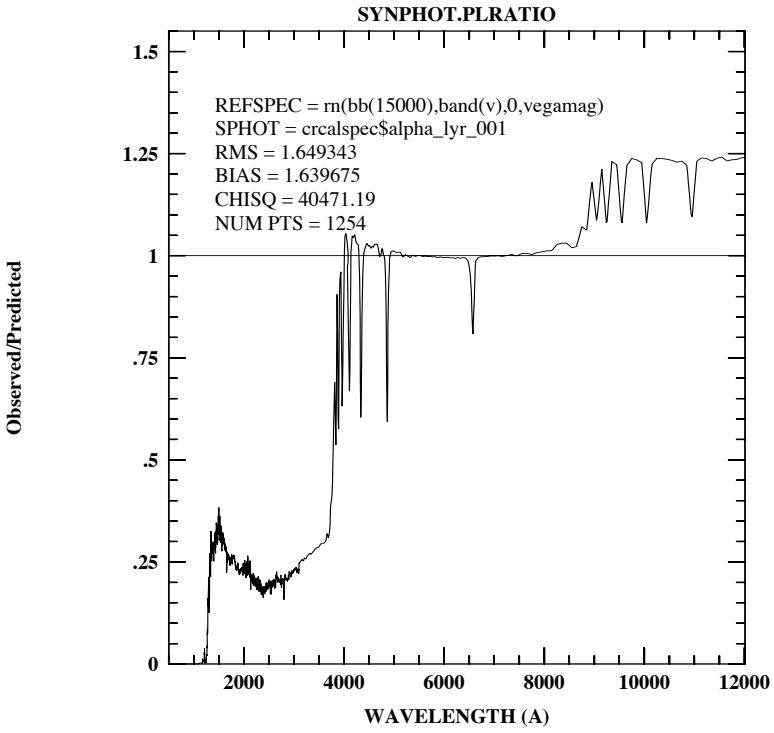
The `left`, `right`, `bottom`, `top`, `append`, `ltype`, `device`, `wavetab`, and `refdata` parameters are used in the same way as described previously for the **plband** and **plspec** tasks.

5.19.1 Examples

Plot the ratio of a 15000 K blackbody spectrum to that of Vega. Since the default normalization for synthesized blackbody spectra is to set it equal to the flux of a solar radius star at a distance of 1 kpc, we first renormalize it to have the same integrated *V* magnitude as Vega. We also use the wavelengths listed in the `alpha_lyr_stis_002` table as the wavelength grid for the calculations. The resulting ratio is plotted in units of `flam`.

```
sy> plratio "" "rn(bb(15000),band(v),0,vegamag)" flam \
>>> crcalspec$alpha_lyr_stis_002.fits wave=crcalspec
$alpha_lyr_stis_002.fits left=1500 right=12000
```

Figure 5.33: Sample Plratio Output



5.20 Pltrans

The **pltrans** task will create photometric transformation plots such as color-color and color-magnitude diagrams. The user specifies a set of spectra, and an observing mode and form for each axis. Usually the input "spectrum" will be a file containing a list of **synphot** spectrum expressions or filenames. Pre-calculated photometric data can be read in from a file specified by the `input` parameter. Results can be saved in a table specified by the `output` parameter.

The task parameters are shown in Figure 5.34.

Figure 5.34: Pltrans Parameters

```
      xmode =                  X axis mode
      ymode =                  Y axis mode
      xform =                  X axis form
      yform =                  Y axis form
     (vzero =  )               List of values for variable zero
     (input = none)            Input table
    (output = none)            Output table
      (left = INDEF)           x value for left side of graph
     (right = INDEF)           x value for right side
    (bottom = INDEF)           y value for bottom
       (top = INDEF)           y value for top
    (append = no)              Append to existing plot?
    (mktype = plus)            Marker type
    (device = stdgraph)        Graphics device
   (wavetab = )                Wavelength table name
   (refdata = )                Reference data
```

The `spectrum` parameter is used to specify a set of synthetic spectra to be generated (see Section 3.2.3). The commands can be placed in a file, one per line, with `spectrum=@`*`filename`*. (See "Filename specifications" on page 19 for important side effects of using the `@filename` syntax). Alternatively, variable zero (`$0`) may be used within a single spectrum expression so that a set of spectra is generated automatically using the list of values from the `vzero` parameter (see below). If this parameter is left blank or set to "none", the only points plotted will be those read from the table specified by the `input` parameter.

The `xmode` and `ymode` parameters contain `obsmode` strings that specify either a single passband or a color (difference of two passbands) to be calculated and plotted on the x- and y-axes of the plot. Single passbands are specified using the syntax "band(mode)", or simply "mode". Colors are specified using the syntax "band(mode1)-band(mode2)" (See "Observation Mode" on page 16.). If these parameters are set to "none" or left blank, a default observation mode that is unity at all wavelengths will be used.

The `xform` and `yform` parameters specify the `form` or units for the x- and y-axis data. They can be any one of the standard **synphot** forms: `fnu`, `flam`, `photnu`, `photlam`, `counts`, `abmag`, `stmag`, `obmag`, `vegamag`, `jy`, or `mjy` (see Section 3.2.4). If `form` is set to either `counts` or `obmag`, all computed quantities are multiplied by the HST collecting area before plotting.

The `vzero` parameter contains a list of values that are substituted for variable zero (`$0`) wherever it appears in the spectrum expression. See Section 3.4, "Variable Substitution (VZERO)," on page 26 for a discussion of variable substitution.

The `input` table contains calculated x-y pairs of data values for plotting. If the file is a FITS or STSDAS table, the x and y values are read from columns labeled FLUX1 and FLUX2, and the values of `xmode`, `ymode`, `xform`, and `yform` are read from table header keywords of the same name. The mode and form values read from the header keywords will supersede any values passed via the task parameter file. If the input file is an ASCII table, the x and y values are read from the first two columns of the table. If this parameter is set to "none" or left blank, no file will be read.

The `output` table, if created, will have the same format as that described above for the input table. Because it has the same format as the input table, output generated by this task can be replotted later without recalculating it. If `append` is set to "yes" and a table of the same name already exists, the data will be appended to the existing table. If `append` is set to "no" and a table of the same name exists, it will be overwritten. If the value of this parameter is "none" or left blank, no file will be written.

The `mktype` parameter specifies the marker type to be used for plotting the results. Lines, point markers, or text strings may be plotted. The recognized line types are `solid`, `dashed`, `dotted`, and `dotdash`. If a line style is chosen, a line will be plotted through the data points. The recognized point marker types are: `point`, `fill`, `box`, `plus`, `cross`, `diamond`, `hline`, `vline`, `hebar`, `vebar`, and `circle`. If a point style is chosen, individual markers will be plotted at each data point. Literal text strings may be plotted at each data point by setting `mktype=!`*string* (the exclamation point will *not* be printed with the text string).

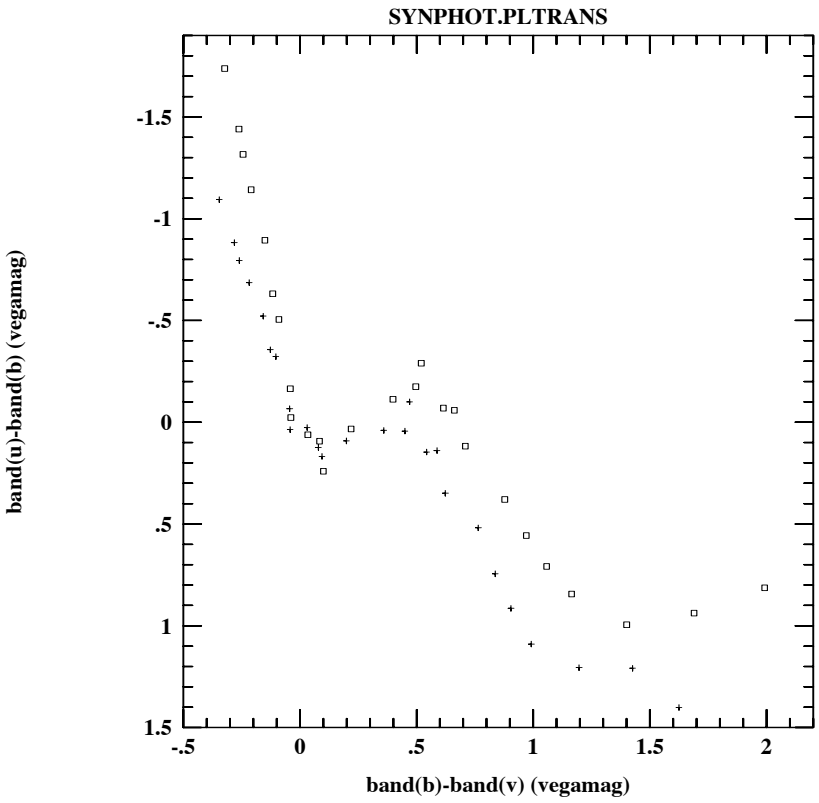
The `left`, `right`, `bottom`, `top`, `append`, `device`, `wavetab`, and `refdata` parameters function in their usual ways as described for other plotting tasks such as **plband** and **plspec**.

5.20.1 Examples

Use **pltrans** to create a color-color diagram of a selection of main sequence stars contained in the BPGS spectral atlas (see Table A.8). We'll calculate and overplot two sets of colors: first the Johnson *U-B* and *B-V* colors; then the WFPC2 equivalents, F336W-F439W and F439W-F555W. The names of the tables containing the spectral data for the stars are stored in the text file `bpgs_ms.lis`. The Johnson colors will be plotted as plus signs, while the WFPC2 colors will be plotted as boxes. Figure 5.35 shows the results.

```
sy> pltrans @bpgs_ms.lis "band(b)-band(v)"\
>>> "band(u)-band(b)" vegamag vegamag bottom=-1.9 top=1.5
>>> left=-0.5 right=2.2

sy> pltrans @bpgs_ms.lis \
>>> "band(wfpc2,f439w)-band(wfpc2,f555w)" \
>>> "band(wfpc2,f336w)-band(wfpc2,f439w)" \
>>> vegamag vegamag mktype=box append+
```

Figure 5.35: Results from Sample Pltrans Run



5.21 Refdata

Refdata is a parameter set (pset) and not an executable task. It defines a set of parameters that are common to most of the **synphot** tasks. It contains the following three parameters:

```
(area   = 45238.93416)          Telescope area in cm^2
(grtbl  = "mtab$*_tmg.fits")    Instrument graph table
(cmptbl.= "mtab$*_tmc.fits")    Instrument component table
```

The first, `area`, is the telescope collecting area, in square centimeters. The default value corresponds to the nominal 120 centimeter radius of the HST aperture. The obscuration factor of 0.86 due to the secondary is taken into account in the throughput data for the optical telescope assembly. The parameter `grtbl` specifies the name of the instrument graph table to be searched when interpreting `obsmode` strings. Similarly, the `cmptbl` parameter specifies the name of the component lookup table that is used to associate component keywords in the graph table with actual throughput data table names. For these last two parameters, the default string, containing the "*" wildcard character, will cause the latest available versions of the tables to be used.

Invoking the `refdata` pset by name, i.e., typing `refdata` at the IRAF system prompt, will run the **eparam** editor on the parameter set, allowing you to modify the parameters. Alternatively, the parameters may be modified on the command line by specifying the pset name and parameter name. For example, you can type `refdata.area=7853.98` to set the area to that of a 1-meter telescope. Parameters within `refdata` can also be modified by using **eparam** on any of the other **synphot** tasks that call for the `refdata` pset, e.g., by typing `epar calcphot`. In this case, `refdata` will appear as one of the **calcphot** task parameters.

In the cl, the `refdata` parameter set may then be edited by positioning the cursor on the line containing the `refdata` name and typing ":e" to edit that pset. After editing the `refdata` parameters, type ":q" to indicate that you are done editing and to return to the main task parameter menu.

In PyRAF, simply clicking on the `refdata` parameter set button will open up a new epar window that can be used to edit the values in the usual way.

5.22 Showfiles

The **showfiles** task produces a list of filenames used in evaluating a **synphot** expression. The purpose of this task is to allow the user to better understand the results that **synphot** produces by listing the files that go into computing this result.

There are several functions in **synphot** expressions which use files. The principal functions are the `band` and `cat` functions. The `band` function evaluates the combined throughput for an observation mode by multiplying the individual throughputs of the components in the optical path together. These component throughputs are stored in FITS tables. This task shows you the component tables that **synphot** uses for a specified observation mode. The `cat` and `icat` functions select a spectrum from a catalog of spectra. This task prints the name of the spectrum or spectra. This task will also print the names of files used by other functions, such as the `grid, spec,` and `thru` functions, as well as filenames embedded in the **synphot** expression.

The **showfiles** task has the following two parameters:

```
expr = ""                       Synphot expression
(refdata = ""                   Reference data
```

The expr parameter is an expression used by the **synphot** expression evaluator to compute a synthetic spectrum or passband. If the expression consists of a single call to the band function, only the arguments to the function need be given. For example, the expression "`band(wfpc, f555w)`" can also be given as "`wfpc,f555w`".

The `refdata` parameter specifies the name of the parameter set (pset) containing the names of the instrument graph (`grtbl`) and component lookup (`cmptbl`) tables to be used by the task. This task and all other tasks in the **synphot** package determine the components in the optical path for a specified observing mode from the graph table. The tasks then find the corresponding FITS tables that contain the throughput data for these components by searching the component lookup table. The default values for `grtbl` and `cmptbl` in the `refdata` pset are set so that the most recently installed versions of each of these tables will be used.

5.22.1 Example

Write a list of component table names for the observing mode "`acs,wfc1,f435w`". The command and its output are shown below.

```
sy> showfiles acs,wfc1,f435w

#Throughput table names:
crotacomp$hst_ota_007_syn.fits
cracscomp$acs_wfc_im123_004_syn.fits
cracscomp$acs_f435w_004_syn.fits
cracscomp$acs_wfc_ebe_win12f_005_syn.fits
cracscomp$acs_wfc_ccd1_017_syn.fits
```

This list includes tables containing throughput (or sensitivity) data for the HST Optical Telescope Assembly (OTA), the internal imaging mirrors of the ACS, the f435w filter, the CCD dewar windows, and the CCD detector.

5.23 Thermback

This task computes the predicted thermal background flux for a specified telescope observing mode. At the user's option, it can also create an output spectrum containing the flux as a function of wavelength. Default temperatures are contained in the thermal emissivity tables (similar to the throughput tables) for each component.

```
obsmode =nicmos,2,f222m  Instrument observation mode(s)
(output =none)           Output table name
(tcmptb =mtab$*_tmt.fits) Instrument thermal component table
(detcat=synphot$simulators/data/detectors.dat)      Detector
pixel scale table
(wavetab =  )              Wavelength table name
(refdata = )               Reference data
(verbose = yes)            Print results to STDOUT ?
(form = counts)            Form for magnitude
```

thermback extends the usual parameterization syntax associated with the `obsmode` to permit the user to optionally specify a temperature associated with any component in the optical chain. For example, if the usual obsmode is `nicmos,2,f222m`, one can override the default temperature of the primary mirror to 300K with `nicmos,2,f222m,primary#300`. Legal names for specifying the components are contained in the THMODE keyword of each component emissivity file.

The output table is specified by the parameter '`output`'. If this parameter is set to "none" (the default) or left blank, no output table will be created. Otherwise, the task will create a table with two columns, WAVELENGTH and FLUX. The WAVELENGTH column will contain

wavelengths in Angstroms. The FLUX column will contain the flux at the corresponding wavelengths in the specified units (counts by default). The table will also contain the header parameters GRFTABLE, OCMPTBL, TCMPTBL, and OBSMODE. These will contain the name of the graph table, the optical component lookup table, the thermal component lookup table, and the observation mode.

The thermal component lookup table is specified in the `tcmptb` parameter. This table is analogous to `refdata.cmptbl`: it translates the thermal component names in the graph table to the filenames containing the emissivity tables and other relevant data for the component. For this parameter, the default string, containing the "*" wildcard character, will cause the latest available version of the table to be used. At present, CDBS only contains thermal related data files for the NICMOS instrument. **thermback** will generate the error "Component names not found in lookup table" for unsupported instruments.

The `detcat` parameter specifies a file containing the pixel scale values for the HST detectors. This value is used in computing the thermal background per pixel.

The `wavetab` parameter contains the name of the wavelength table. All flux calculations are done on the wavelength set provided by this table. If the wavetab parameter is set to "none" or left blank, a default wavelength set is computed, according to the algorithm described in Section 3.3.

The `refdata` parameter specifies the name of the parameter set (pset) containing the names of the instrument graph (`grtbl`) and component lookup (`cmptbl`) tables to be used by the task. This task and all other tasks in the **synphot** package determine the components in the optical path for a specified observing mode from the graph table. This task then finds the corresponding STSDAS tables that contain the throughput and emissivity data for these components by searching the component and thermal lookup tables. The default values for `grtbl` and `cmptbl` in the `refdata` pset are set so that the most recently installed versions of each of these tables will be used.

If the `verbose` parameter is set to yes, the task will print incremental results to STDOUT (usually your terminal) as it walks through the optical path and performs the thermal background calculation at each step. This can be useful to explore the individual contributions of each component to the predicted thermal background.

The `form` parameter permits the user to specify the units for the thermal background. It is strongly recommended, however, that the default value of counts (per second per pixel) be used.

5.23.1 EXAMPLES

Compute the thermal background rate for the NICMOS 2 camera using the f222m filter. Store the result in table 'nic2def.fits':

```
sy> thermback.obsmode = "nicmos, 2, f222m"
sy> thermback.output = "nic2def.fits"
sy> thermback
```

The results are printed to STDOUT:

```
trate is    11.65 counts per sec per pixel
```

Repeat the calculation, but override the temperature of the "spider" to see what effect it would have if this component were warmer:

```
sy> thermback nicmos,2,f222m,spider#330
```

The results are printed to STDOUT, including a warning that one of the component temperatures has been overridden

```
Overriding spider# temperature to 330.
trate is    21.13 counts per sec per pixel
```

CHAPTER 6:

Inner Workings

In this chapter. . .

6.1 Syncalc / 107
6.2 Graph and Component Tables / 109

This chapter describes in detail the operation of the internal **synphot** expression evaluator, `syncalc`, as well as the structure and use of the instrument graph and component lookup tables.

6.1 Syncalc

`syncalc` is a function used internally by all **synphot** tasks. Every `obsmode` and `spectrum` parameter string used in the **synphot** tasks is parsed and evaluated by the spectrum expression evaluator `syncalc`. This expression evaluator is written to work like FORTRAN, so that the format of expressions will be easy to understand and use. `Syncalc` supports the four basic arithmetic operations and negation, as well as several functions. Expressions can be parenthesized in order to change the default order of evaluation. Spaces are not significant, except that the division operator must be surrounded by blanks so that it is not mistaken for part of a file name.

`Syncalc` evaluates expressions containing file names, constants, and variables. When `syncalc` sees a file name, it determines if the file is a passband or a spectrum, and then reads it interpolated onto the given wavelength grid. Constants are either numbers or strings. String constants are *not* surrounded by quotation marks. Numeric constants are interpreted

as real numbers and all mathematical operations between file names and constants are legal. Variables are represented as dollar signs followed by a positive integer; for example, `$3`. Variables may occur in an expression wherever a numeric constant is legal. The variable `$0` is used by several tasks for looping over an expression, substituting in turn the values from the list given in the `vzero` parameter. The variables `$1` through `$9` are used by the fitting tasks.

`Syncalc` prevents physically meaningless expressions from being computed by keeping track of the "degree" of the expression during computation. Constants, variables, and passbands have a degree of zero. Spectra have a degree of one. Each function also has an associated degree of either zero or one, depending on the result of the function. Multiplying two subexpressions yields a result whose degree is the sum of the degrees of the subexpressions. Dividing two subexpressions yields a result whose degree is the difference between the degrees of the subexpressions. Adding or subtracting two subexpressions yields a result whose degree is the same as the degrees of the subexpressions. Adding or subtracting two subexpressions that have different degrees is forbidden and causes an error exit. Negation gives a result whose degree is equal to that of the subexpression. The degree of the entire expression must be either zero or one. An expression with a degree of zero is considered to be a passband and an expression with a degree of one is a spectrum.

`Syncalc` determines whether a file contains a passband or a spectrum by looking for a column of throughput values. If the file contains a throughput column, `syncalc` reads it as a passband; if not, it reads it as a spectrum. If the file is in FITS or STSDAS table format, the throughput column is identified by one of two methods: 1) the user supplies the name of the throughput column by appending it within square brackets to the end of the file name, e.g., "my_detector[dqe]"; or 2) if no column name is supplied by the user, `syncalc` looks for a column labeled "THROUGHPUT". Since ASCII files do not contain column names, the column number of the throughput data must be appended in square brackets to the file name, e.g., "my_detector[2]". If a column number is not supplied with an ASCII file, then the data are assumed to be a spectrum.

The heart of `syncalc` is the set of functions that it supports. The following table lists these functions and their degree (see Section 3.2.2 and Section 3.2.3 for more details on the use and performance of these functions).

Table 6.1: Syncalc Functions

Function	Degree	Description
band	0	Telescope passband
bb	1	Blackbody spectrum
box	0	Rectangular passband
cat	1	Spectrum in catalog
ebmv	0	Galactic extinction curve
ebmvx	0	Extended extinction curve
em	0	Gaussian emission line
gauss	0	Gaussian passband
grid	1	Interpolated spectrum in grid
hi	1	HI continuum emission spectrum
icat	1	Interpolated spectrum in catalog
lgauss	0	Log gaussian passband
pl	1	Power-law spectrum
poly	0	Legendre polynomial
rn	1	Renormalized spectrum
spec	1	Spectrum file
thru	0	Throughput file
tilt	0	Legendre polynomial product
unit	1	Constant spectrum
z	1	Redshifted spectrum

6.2 Graph and Component Tables

Tasks in the **synphot** package go through the following steps when computing the combined throughput for an instrument mode:

1. The instrument mode is broken into individual keywords. Any parameters which may be in the instrument mode are extracted.

2. The instrument graph is searched starting at the row with the lowest `innode` value. All rows with this `innode` are examined. The row whose keyword matches one of the keywords in the instrument mode is selected. If no such row is found, the row with the keyword "default" is selected. If there is no such row, and there is only one

row with the current `innode`, it is selected. Otherwise the task exits with an error.

3. Once the row is selected, the component name is saved in an array, unless the component name is "clear". Clear components are discarded because their throughput will not modify the combined throughput.

4. The `outnode` of the selected row becomes the new `innode` and the process of searching the graph continues until the `outnode` is not found in the instrument graph.

5. Once all of the component names are found in the instrument graph table, the component lookup table is searched for the actual names of the throughput tables. The throughput values are read from the component throughput tables, resampled onto the specified wavelength set if necessary, and multiplied together to form the combined throughput.

The thermback task, unlike all other **synphot** tasks, traverses the graph table twice: once in the usual fashion, and once to obtain the optical train necessary for thermal calculations. These trains differ because of the existence of opaque but emissive components in the optical path, such as the spider that supports the secondary mirror. The graph table is used to obtain the train of thermally emissive components in the same way it is used for optical components, except that:

- obsmode keywords are compared to the thermal component name (THCOMPNAME) instead of the usual COMPNAME
- Once all of the component names are found in the instrument graph table, the thermal component lookup table is searched for the actual filenames of the emissivity tables.

The graph table

The instrument graph table has six columns:

- Component name (COMPNAME)
- Instrument keyword (KEYWORD)
- Input node (INNODE)
- Output node (OUTNODE)
- Thermal component name (THCOMPNAME)
- Comment (COMMENT)

The format of the instrument graph table is shown in Figure 6.1 below. The comment column is not used by **synphot**. It is there simply for documentation. The input and output node columns specify the light path

through the HST and are used in the process of searching the graph, as explained above. Node numbers should increase as one goes down the light path in the instrument. The instrument keyword column is used to match the keywords in the instrument mode string. Sometimes a component is known by several names. It can be represented in the graph table by several rows that are identical except for the instrument keyword column, which should contain the different names that the component is known by. The component name column contains the name of the instrument component. Each component should have a unique name. The name of the component is used to link the instrument graph table to the component lookup table.

Figure 6.1: Structure of Instrument Graph Table

```
Instrument Graph Table
COMPNAME      CH*20
KEYWORD       CH*12
INNODE        I
OUTNODE       I
THCOMPNAME    CH*20
COMMENT       CH*68
```

The optical and thermal component lookup tables

The component lookup table has four columns:

- Time (TIME)
- Component name (COMPNAME)
- Throughput file name and column (FILENAME)
- Comment (COMMENT)

The format of this table is shown in Figure 6.2 below. The insertion time and comment columns are not used by the **synphot** tasks. The time column contains the time the component file was created. It is included for documentation and to simplify the job of delivery of new **synphot** tables to the Calibration Data Base System (CDBS). The time is in CDBS date-time format (yyyymmdd:hhmmss). The comment field is used solely for documentation. The component name column is used to link the instrument graph with the component name table. The throughput file and column names could have been stored directly in the graph table. However, the component throughputs are usually updated more frequently than the structure of the instrument itself (at least for HST!), so to keep from having to update the instrument graph more than is necessary, the throughput file and column names are placed in a separate table and the component name is placed in the instrument graph table.

The component name column provides the link between the two tables. The file name column provides the name of the table, and optionally the

column, of the component throughput table. The filename is the name of the binary table containing the throughput. The file name should include the directory path name. The throughput column name is placed in brackets after the file name. If there is no column name in brackets, the default column name of THROUGHPUT is used. Allowing the column name to appear in the file name column was a later enhancement to the **synphot** package. This modification was made in order to allow more than one throughput to be stored in a single binary table.

The thermal component lookup table is identical in format to the component lookup table, except that it contains the THCOMPNAME and emissivity filename in place of the COMPNAME and throughput filename, and it does not support column names in the filename column.

Figure 6.2: Structure of Component Lookup Table

```
Component Lookup Table
TIME            CH*26
COMPNAME        CH*18
FILENAME        CH*50
COMMENT         CH*68
```

The throughput table

The throughput table contains the component throughput as a function of wavelength. It may also contain an optional column of estimated errors or uncertainties associated with the throughput values. If the default throughput column name of THROUGHPUT is used, the error column should be named ERROR. If another throughput column name is used, the error column has the same name as the throughput column plus the suffix "_ERR". For example, if the throughput column name is DN1, the error column would be named DN1_ERR. The wavelength information is kept in a column called WAVELENGTH. The throughput table is sorted on the wavelength column in either ascending or descending order. INDEF is a legal value for the throughput or error columns and indicates that the throughput is not known at that wavelength. The wavelength column, however, should *never* have a value of INDEF.

All columns in the throughput table should contain 32-bit floating-point numbers. The units must be specified for the wavelength column. The **synphot** tasks use Angstroms internally; however, the wavelength units may be Angstroms, nanometers, microns, millimeters, centimeters, meters, hertz, kilohertz, megahertz, gigahertz, eV, KeV, or MeV. Any unique abbreviation of the preceding units is also allowed. The throughput and error column units may be blank, however the following units are allowed for documentation: transmission, qe, or dn/photon. As with the wavelength units, any unique abbreviation is allowed.

A component throughput may also be parameterized, meaning that the throughput is a function of some other variable besides wavelength. An example of parameterized components are the WFPC2 ramp filters. The throughput of these filters varies as a function of position on the filter. The throughput table for a parameterized component contains several throughput and error columns, evaluated at different values of the parameter. The **synphot** package interpolates between these columns when computing the throughput for a given value of the parameter. The column name is a root name followed by one or more parameter values. Each parameter is preceded by a hash mark (#). The parameter values are those at which the column is evaluated. For example, suppose a ramp filter is parameterized on its peak wavelength, which varies between 3500 and 4000 Å. If the throughput is evaluated at intervals of 100 Å, the throughput column names would be WFPC2_RAMP#3500, WFPC2_RAMP#3600, ... WFPC2_RAMP#4000. The error column associated with each of these throughputs has the suffix "_ERR" added to the root name. The name of the error columns for the ramp filters in the previous example would be WFPC2_RAMP_ERR#3500 ... WFPC2_RAMP_ERR#4000.

Entries in the graph and component tables indicate that the component is parameterized by containing one or more hash marks. The number of hash marks indicates how many parameters the component takes. To continue with the previous example, the keyword column in the graph table might contain the value RAMP# and the filename column in the component lookup table might contain the name "wfpc2_ramp[ramp#]". Parameterized components can be placed in the same throughput table as other components.

The thermal emissivity table

The thermal emissivity table contains the component emissivity as a function of wavelength. It may also contain an optional column of estimated errors or uncertainties associated with the emissivity values. The column names should be WAVELENGTH, EMISSIVITY, and ERROR. Restrictions on wavelength values, units, and numerical formats are the same as for the component table.

Table 6.2: Thermal Emissivity Table.

```
BEAMFILL = 1.           / Fraction of beam filled by this component
DEFT     = 77.          / Default temperature in kelvins
THTYPE   ='THRU'        / Thermal type (opaque/thru/numeric/clear)
THCMPNAM = 'nic2_f110w' / Name of thermal component
THMODE   = 'f110w'      / Keyword in obsmode to specify temperature
```

Table 6.3: The emissivity table also contains some crucial information in header keywords:

```
BEAMFILL= 1.          / Fraction of beam filled by this component
   DEFT = 77.         / Default temperature in kelvins
 THTYPE ='THRU'       / Thermal type (opaque/thru/numeric/clear)
THCMPNAM='nic2_f110w' / Name of thermal component
THMODE  ='f110w'      / Keyword in obsmode to specify temperature
```

BEAMFILL specifies the fraction of the optical beam filled by this component. This value is usually one, but it depends on the precise optical layout of the instrument.

DEFT specifies the default temperature of the component, in Kelvins. This is the temperature that will be used in the thermal calculations unless it is overridden in the obsmode.

THTYPE specifies the type of thermal component that is described by this file. Opaque components are those which partially obstruct the beam: they emit, but do not pass, radiation. Thru components are those which have both throughput and emissivity: all the usual optical elements are of this sort. Numeric elements are those components which do not correspond to a physical device in the instrument, but are represented as such for convenience, such as the detector quantum efficiency. Clear elements are those which do not contribute to either throughput or emission: usually this value is associated only with rows in the graph table that organize the flow.

THCOMPNAME and THMODE are the associated pair of values used when traversing the graph table. THMODE contains the obsmode keyword which points to the associated THCOMPNAME.

The usual `keyword#` parameterization syntax is overridden for use with the thermal emissivity tables. Instead, this syntax is used to specify a component temperature (overriding the default temperature present in the DEFT header keyword).

CHAPTER 7:

Calibration Using Synthetic Photometry

In this chapter. . .

7.1 Principles of calibration / 115
7.2 Accuracy and errors in synthetic photometry / 118

7.1 Principles of calibration

Spectrophotometric data taken with modern array detectors are simpler to calibrate than broadband photometry because the spectrophotometric pixels have very narrow passbands across which the detector sensitivity, atmospheric extinction, and standard star continuum are effectively constant. The number of detected counts per exposure time Δt in a pixel at wavelength λ covering the wavelength interval $\Delta\lambda$ is given by:

$$\frac{C(\lambda)}{\Delta t} = A \cdot P(\lambda) \cdot f_{\lambda}(\lambda) \cdot \frac{\lambda}{hc} \cdot \Delta\lambda \quad \text{(counts/sec)}$$

where A is the nominal collecting area of the telescope, $f_{\lambda}(\lambda)$ is the flux density of the target star, $P(\lambda)$ is the dimensionless bandpass throughput function, and the division by $h\nu = hc / \lambda$ converts the energy flux to a photon flux as is appropriate for photon-counting detectors. To calibrate, we want to determine the unit response curve $U(\lambda)$, which gives as a function of wavelength the factor that converts observed count rate to source flux density. $U(\lambda)$ is determined from observed count rate spectra of spectrophotometric standard stars whose absolute flux spectra are known, and can

then be used to convert count rate spectra observed on other targets to absolute fluxes. Typically, this is done by dividing the known spectrum of a standard star by its observed count rate spectrum, so that $U(\lambda)$ is given by:

$$U_\lambda(\lambda) = \frac{f_\lambda(\lambda)}{C(\lambda)/\Delta t} = \frac{hc/\mathrm{A}}{P(\lambda)\cdot\lambda\cdot\Delta\lambda} \quad (\mathrm{ergs/cm^2/\AA})$$

Once $U(\lambda)$ is determined, the absolute flux distribution of any observed source can be computed by simply multiplying the source's count rate spectrum, $C(\lambda)/\Delta t$, by $U(\lambda)$:

$$f_\lambda(\lambda) = U(\lambda)\cdot C(\lambda)/\Delta t \quad (\mathrm{ergs/sec/cm^2/\AA})$$

Synthetic photometry permits a precise generalization of this procedure from the narrow passbands of spectrophotometric pixels to a passband $P(\lambda)$ of arbitrary width and shape. For a broadband photometric mode the analogous expression giving the detected count rate through passband P is:

$$\frac{C(P)}{\Delta t} = \frac{A}{hc}\int P(\lambda) f_\lambda(\lambda)\lambda\, d\lambda$$

The precise definition of mean flux density in a broad passband must then be:

$$f_\lambda(P) = \frac{\int P(\lambda) f_\lambda(\lambda)\lambda\, d\lambda}{\int P(\lambda)\lambda\, d\lambda}$$

so the count rate to flux density conversion factor for a broad passband is:

$$U_\lambda(P) = \frac{f_\lambda(P)}{C(P)/\Delta t} = \frac{hc/\mathrm{A}}{\int P(\lambda)\lambda\, d\lambda}$$

The calibration procedure for broadband photometry is now essentially parallel to that for spectrophotometry. The only difference is that in the spectrophotometric case the source's flux distribution can be considered constant across the narrow bandwidth of each spectroscopic pixel, whereas in the photometric case the integral over wavelength must be performed more carefully. So the normal method of calibrating photometric modes is to first calculate the mean flux densities of spectrophotometric standard stars in the broad bands, using the known flux spectra and passband shapes, and then derive the flux conversion factors *U(P)* by comparing the mean flux densities against the corresponding observed count rates.

Notice, however, that the last equation above shows us that, if we know the passband function *P(λ)*, we can derive *U(P)* directly, without the need

for standard star observations. This is the technique used by **synphot** (and incorporated into the WFPC2, ACS calibration pipelines) to calibrate HST observing modes. The value of the PHOTFLAM header keyword that appears in calibrated WFPC2, ACS, and STIS images is the flux conversion factor *U(P)*, computed directly from the passband function of the observation mode.

Once known, the flux conversion factor can be used to express the count rate observed on an unknown target to the precise flux density of that target, so that for an observed number of counts C in an exposure time of Δt, the corresponding mean flux density within the passband is simply:

$$f_\lambda(P) = U_\lambda(P) \cdot C/\Delta t$$

Corresponding magnitudes are given by:

$$m_\lambda(P) = -2.5 \log(U_\lambda(P) \cdot C/\Delta t) + K$$

where K is chosen for convenience to be –21.10 (for U_λ in units of ergs cm^{-2} $Å^{-1}$) so that m_λ (5500 Å) is approximately equal to the Johnson V magnitude.

A potential difficulty arises in the conversion between $f_\lambda(P)$ and $f_\nu(P)$, since it is not at first obvious from their definitions that their relationship is independent of the source spectrum. Fortunately, such conversions can be made precisely by the relation

$$f_\lambda(P) = f_\nu(P)\frac{c}{\lambda_P(P)^2}$$

where $\lambda_P(P)$ is the passband's pivot wavelength. This parameter is source-independent and defined as

$$\lambda_P(P) = \sqrt{\frac{\int P(\lambda)\lambda \, d\lambda}{\int P(\lambda) \, d\lambda/\lambda}}$$

The HST magnitude system, based on f_λ, can then be converted to an analogous f_ν (ergs cm^{-2} s^{-1} Hz^{-1}) system by calculating

$$m_\nu(P) = m_\lambda(P) - 5\log\lambda_P(P) + 18.70$$

where λ_P is the pivot wavelength of the passband in Angstroms; m_ν will then be approximately equal to AB_ν from the ground-based optical AB79 system of Oke and Gunn (1983). It is important to use the pivot wavelength when making this conversion, since a 1% wavelength error would cause a 5% error in the converted broadband flux.

For any non-zero passband width, the pivot wavelength differs somewhat from the more traditional average wavelength,

$$\lambda_0 = \frac{\int \lambda\ P(\lambda)\ d\lambda}{\int P(\lambda)\ d\lambda}$$

which is also source-independent. As a measure of the width of a passband we can use the rms width

$$\lambda_{\rm rms} = \sqrt{\frac{\int (\lambda - \lambda_0)^2 P(\lambda)\ d\lambda}{\int P(\lambda) d\lambda}}$$

All of these source-independent characteristics can be computed for a given passband using the **bandpar** task in the **synphot** package. The pivot wavelength, average wavelength, and rms width are known as "pivwv", "avglam", and "bandw", respectively.

It is also useful to define a source-*dependent* passband parameter that can be used to estimate shifts of the effective wavelength with source characteristics (e.g., temperature, reddening, redshift, etc.). We define the effective wavelength as the mean wavelength of the detected photons,

$$\lambda_{\rm eff} = \frac{\int \lambda\ f_\lambda(\lambda) P(\lambda) \lambda\ d\lambda}{\int f_\lambda(\lambda) P(\lambda) \lambda\ d\lambda}$$

Since this is a source-dependent parameter, it is calculated by **calcphot** (because **bandpar** and **calcband**, by definition, only support passband data and not spectral data), where it is referred to as "efflam".

7.2 Accuracy and errors in synthetic photometry

The accuracy of **synphot** results depends on the accuracy of the zero points and the shapes of the sensitivity curves for each bandpass in the database. In particular, zero point issues arise whenever one compares synthetic photometry to real photometry. Of course, the accuracy also depends on the spectrum or model used for the calculation.

7.2.1 Groundbased photometric systems

synphot supports a number of groundbased photometric systems, including Johnson, Cousins, Bessell, SDSS, and more. Each of these standard systems comes with its own set of corrections for airmass, for objects of extreme color, and so on. Accurate calibrations of groundbased data can only be achieved by measuring photometric standard stars along with the object(s) of interest, and performing your own measurement of the zero point offsets.

Ground-based U band photometry is particularly problematic, as the short-wavelength side of its bandpass cutoff is determined by atmospheric conditions.

For more detail on calibrating groundbased photometry, see for example Bessel et al, 1998 (UBVRIJHKL system) and Cohen et al, 2004 (2MASS JHK system).

"*As the standard systems have been established from natural system colors using linear and non-linear corrections of at least a few percent, we should not be reluctant to consider similar corrections to synthetic photometry to achieve good agreement with the standard system across the whole temperature range of the models.*"

Bessell et al, 1998

7.2.2 Vegamags

synphot offers the option of producing photometry in units of vegamags, defined such that Vega has magnitude zero in all passbands. To accomplish this transformation, the passband integral is calculated first for the source spectrum and then again for the spectrum of Vega built into the system. The ratio of the two results is converted to a magnitude.

The magnitude of Vega in "vegamags" is defined to be zero in all passbands, even if the real value is known to be different. (e.g., Vega = 0.026 mags in Johnson V

Naturally the accuracy of such results depends on the accuracy of the Vega spectrum used. As of October 2004, the Vega spectrum in the **synphot** system is a composite flux standard (Bohlin & Gilliland 2004) which consists of IUE data from 1255-1675A, STIS CCD fluxes from 1650-4200A (Bohlin & Gilliland 2004), and a specially tailored Kurucz model longward of 4200A and from 900-1255A (Kurucz 2003).

7.2.3 HST bandpasses

The bandpasses for HST instruments are more stably defined than those of groundbased systems, for several reasons. HST bandpasses are instrumental bandpasses, defined by a single specific combination of telescope/detector/filter hardware. There are of course no variations in atmospheric conditions such as air mass and humidity, which affect the sensitivity of groundbased instruments. Temperature conditions on orbit are also extremely stable, and sets of standard stars are routinely observed for calibration purposes. However, care must still be taken when attempting accurate calibrations.

Discussions of the photometric zero points and stability for each of the instruments can be found in the relevant sections of the HST Data Handbook. Detailed information can also be found for ACS in DeMarchi et al, 2004; for NICMOS and STIS in Bohlin et al, 2001; and for WFPC2 in Heyer et al, 2004.

APPENDIX A:

On-Line Catalogs and Spectral Atlases

In this appendix . . .

A.1 HST Calibration Spectra / 122
A.2 Kurucz 1993 Atlas of Model Atmospheres / 125
A.5 Bruzual Spectrum Synthesis Atlas / 131
A.6 Gunn-Stryker Spectrophotometry Atlas / 132
A.7 Bruzual-Persson-Gunn-Stryker Spectrophotometry Atlas / 132
A.8 Jacoby-Hunter-Christian Spectophotometry Atlas / 135
A.9 Bruzual-Charlot Atlas / 138
A.10 Kinney-Calzetti Atlas / 139
A.11 Buser-Kurucz Atlas of Model Atmospheres / 139
A.12 AGN Atlas / 142
A.13 Galactic Atlas / 142

There are several spectral atlases consisting of both observed and model data that are available in FITS table format for use with **synphot**. These are available at STScI on all science computing clusters in the `crrefer` directory areas. Off-site users can obtain these data via anonymous ftp on the node `ftp.stsci.edu` where they are located in several subdirectories under the /`cdbs` directory.

Current descriptions and access to all available calibration spectra and astronomical catalogs can also be found at the CDBS web site, which is updated more frequently than this manual:

`http://www.stsci.edu/hst/observatory/cdbs/astronomical_catalogs.html`.

The subdirectories /`cdbs/calspec/` and /`cdbs/grid/` contain the data tables for all of the atlases described here. For easy access from the **synphot** tasks, the on-line directories containing the various atlases are defined via environment variables within STSDAS. All subdirectories are defined relative to the `crrefer$` directory environment variable, which is

set in the IRAF file `hlib$extern.pkg`. Off-site users should define `crrefer` in this file to point to the local directory area that contains the atlas tables. The environment variables `crcalspec$`, `crgridbk$`, `crgridbpgs$`, `crgridbz77$`, `crgridgs$`, and `crgridjac$` are subsequently defined in the STSDAS file `stsdaslib$zzsetenv.def` to point to subdirectories beneath `crrefer$` which contain the spectral data tables for each of the individual atlases.

A.1 HST Calibration Spectra

The directory `crcalcspec$` contains the composite stellar spectra that are the fundamental flux standards for HST calibrations. All files are in machine independent binary FITS table format. Information about the pedigree of a given spectrum is in the header of the FITS table file, which can be read with the IRAF hedit task or examined usying pyFITS.

In some cases, the spectrum is truncated in the blue (or in the red) at wavelength longer (shorter) than the sensitivity limit of the instrument. As a result, **synphot** may underestimate the total counts. Users should check that the wavelength range of the spectrum/model they are using is compatible with the wavelength range of the calculation they require.

Table A.1 below summarizes the set of recommended standard star spectra. Columns 2 and 3 give the V and B magnitudes of the stars. More documentation on these stars, e.g. coordinates, finding charts and spectral types can be found in Turnshek et al. (1990), Bohlin, Colina & Finley (1995), and Colina & Bohlin (1997). Column 4 is the computer compatible file name with the plus and minus signs converted to underscores. Thus, the actual CALSPEC file name is the prefix in column 4, plus one of the suffixes in columns 5-8, plus ".fits". For example, a standard that has STIS data is "bd_28d4211_stis_001.fits".

These spectra can be used with **synphot** tasks by setting either the `spectrum` or `spfile` task parameters to the name of the desired file, e.g., `crcalspec$alpha_lyr_stis_002.fits`. The unit of flux in all files is erg s^{-1} cm^{-2} A^{-1}.

The pure hydrogen WD model LTE spectra of Bohlin (2000) have been replaced by Hubeny NLTE models calculated with his Tlusty code (Bohlin 2003). These fundamental standards GD71, GD153, G191B2B, and HZ43 are listed by Calibration Data Base System (CDBS) suffix in column 5 of the Table and provide the basis for the secondary flux standards. The observational spectra from columns 5-8 can be compared with the models;

and in the case of G191B2B, there is ultraviolet line blanketing at the few percent level. The model calculations extend to 30 microns and cover the long wavelength limits of 2.7 microns for NICMOS, 1.1 microns for STIS and ACS, and 27 microns for JWST. Vega was observed by STIS (Bohlin & Gilliland 2004); and the new composite flux standard alpha_lyr_stis_002.fits consists of IUE data from 1255-1675A, STIS CCD fluxes from 1650-4200A, and a specially tailored Kurucz model longward of 4200A (Kurucz 2003) and from 900-1255A.

Column 6 lists the 20 CDBS suffix names for the second choice standard stars with STIS data from Bohlin, Dickinson, & Calzetti (2001). Tabulated in column 7 are the next best standard star flux distributions, which are composed of FOS spectra in the UV and Oke spectra at the longer wavelengths. Also appearing in column 7 are the three solar analogs that are comprised of FOS observation but do not have "_FOS" in the CALSPEC file name.

The names for the last, but largest, set of standard stars appear in column 8 of the Table. The application of corrections to the original IUE and optical fluxes produces a consistent set of spectrophotometric standards from 1150 to 9200A (Bohlin 1996 and references therein). This set of standards is composed of IUE+Oke data only.

CALSPEC also contains the ultraviolet to near-infrared absolute flux distribution of the Sun (filename: sun_reference_stis_001.fits) to 2.7 microns, which is used to model the IR spectrum of the three solar analog stars. The solar reference spectrum combines absolute flux measurements from satellites and from the ground with a model spectrum for the near-infrared (Colina, Bohlin, & Castelli 1996).

The `SUPPLEMENTAL_CALSPEC` directory contains previous CALSPEC versions and spectrophotometry that may be useful for special purposes.

The order of preference for the choice of a standard flux distribution is from left to right in the Table A.1, i.e. from the best in column 5 to the last choice with the lowest quality in column 8.

Table A.2 contains the spectral type of some of the HST calibration sources.

Table A.1: CDBS Flux Standards with Columns in Order of Preference

Star name (1)	V (2)	B-V (3)	CDBS name (4)	Model (5)	STIS (6)	FOS+Oke (7)	IUE+Oke (8)
AGK+81D266	11.94	-0.34	agk_81d266		_STIS_001		_005
ALPHA-LYR	0.03	0.00	alpha_lyr		_STIS_002		_004
BD+25D4655	9.69	-0.31	bd_25d4655				_002
BD+28D4211	10.51	-0.34	bd_28d4211		_STIS_001	_FOS_003	_005
BD+33D2642	10.83	-0.17	bd_33d2642			_FOS_003	_004
BD+75D325	9.55	-0.33	bd_75d325		_STIS_001	_FOS_003	_005
FEIGE110	11.83	-0.30	feige110		_STIS_001		_005
FEIGE34	11.18	-0.34	feige34		_STIS_001		_005
FEIGE66	10.51	-0.29	feige66				_002
FEIGE67	11.82	-0.34	feige67				_002
G191B2B	11.77	-0.33	g191b2b	_MOD_004	_STIS_001	_FOS_003	_005
G93-48	1 2.74	-0.01	g93_48				_004
GD108	13.56	-0.22	gd108				_005
GD153	13.35	-0.29	gd153	_MOD_004	_STIS_001	_FOS_003	
GD50	14.06	-0.28	gd50				_004
GD71	13.03	-0.25	gd71	_MOD_005	_STIS_001	_FOS_003	
GRW+70D5824	12.77	-0.09	grw_70d5824		_STIS_001		_005
HD93521	6.99	-0.27	hd93521		_STIS_001		_005
HZ21	14.69	-0.33	hz21		_STIS_001		_005
HZ2	13.88	-0.09	hz2				_005
HZ43	12.91	-0.31	hz43	_MOD_004	_STIS_001	_FOS_003	
HZ43B	14.30	...	hz43b		_STIS_001		
HZ44	11.67	-0.29	hz44		_STIS_001	_FOS_003	_005
HZ4	14.51	+0.09	hz4		_STIS_001		_004
LB227	15.32	+0.06	lb227				_004
LDS749B	14.68	-0.04	lds749b				_005
LTT9491	14.10	+0.03	ltt9491				_002
NGC7293	13.53	-0.37	ngc7293				_005
P041C	12.00	0.62	p041c		_STIS_001	_001	
P177D	13.47	0.66	p177d		_STIS_001	_001	
P330E	13.00	0.64	p330e		_STIS_001	_001	
SUN	-26.75	0.63	sun_reference		_STIS_001		

(1) The unit of flux in all files is erg $s^{-}1$ $cm^{-}2$ $A^{-}1$.

Table A.2: HST Calibration Spectra

Star	Type
AGK +81 266	sdO
alpha Lyra	A0 V
BD +28 4211	Op
BD +33 2642	B2 IV
BD +75 0325	O5p
Feige 34	DO
Feige 110	D0p
G93-48	DA3
G191-B2B	DA0
GD50	DA2
GD108	sdB?
GRW +70 5824	DA3
HD 93521	O9 Vp
HZ 2	DA3
HZ 4	DA4
HZ 21	DO2
HZ 44	sdO
LB 227	DA4
LDS 749B	DB4
NGC 7293	

A.2 Kurucz 1993 Atlas of Model Atmospheres

The atlas contains about 7600 stellar atmosphere models for a wide range of metallicities, effective temperatures and gravities. These new LTE models have improved opacities and are computed with a finer wavelength and temperature resolution than the previous Buser-Kurucz atlas installed in the CDBS (crgridbk). The micro-turbulent velocity is 2 km s^{-1}.

The new atlas installed in the CDBS is from the Kurucz database at Goddard Space Flight Center. The original atlas (CD-ROM No. 13) was created on August 22, 1993 and can be obtained from Dr. R. Kurucz.

The atlas includes models of abundances (log_Z) relative to solar of +1.0, +0.5, +0.3, +0.2, +0.1, +0.0, -0.1, -0.2, -0.3, -0.5, -1.0, -1.5, -2.0, -2.5,-3.0, -3.5, -4.0, -4.5, and -5.0. The grid of models cover the gravity range from log_g= 0.0 to +5.0 in steps of +0.5. The range in effective

temperature from 3500 K to 50000 K is covered with an uneven grid (see Table A.3). The model spectra cover the ultraviolet (1000A) to infrared (10 microns) spectral range with non-uniform wavelength spacing (see Table A.4).

Table A.3: Grid of temperatures for the models

Temperature Range	Grid Step
K	K
3000 - 10000	250
10000 - 13000	500
13000 - 35000	1000
35000 - 50000	2500

Table A.4: Wavelength coverage for the models

Wavelength Range	Grid Step
microns	A
0.10 - 0.29	10
0.29 - 1.00	20
1.00 - 1.60	50
1.60 - 3.20	100
3.20 - 8.35	200
8.35 - 10.0	400

A.3 THE HST/CDBS VERSION OF THE 1993 KURUCZ ATLAS

The atlas is divided in 19 independent subdirectories, according to metallicity. Within each subdirectory the stellar atmosphere models are given in STDAS multicolumn table format. Each table consist of 12 different columns, the first one containing the wavelength grid and each of the rest containing the spectrum of a star with the same effective temperature but different gravity, ranging from log_g= 0.0 to +5.0. Columns filled with zeros indicate that the model spectrum for that particular metallicity, effective temperature and gravity combination is not covered by the atlas.

The names of the table files are given as kszz_ttttt.fits where "k", for Kurucz, is the first letter of the atlas; "szz" is the metallicity of the model (zz) with its sign (s); and "ttttt" is the model's effective temperature, using four or five digits depending on the value. For instance, models for an effective temperature of 5000 K with log_Z= -0.5 and log_Z= +3.5 are indicated by ttttt= 5000, s= m, zz= 05 and ttttt= 5000, s= p, zz= 35, i.e. km05_5000.fits and kp35_5000.fits.

Within each individual table file, each column is named "gyy" where "yy" corresponds to 10*log_g. For example, log_g= +0.5 and log_g= +4.0 models are located in columns named g05 and g40, respectively.

See Table A.5 for an example of a standard header of a table file. In this example the name of the file is kp00_8000.fits and contains all the models for a star of metallicity log_Z= 0.0 and effective temperature T_{eff}= 8000 K. Models cover a range of gravities from log_g= +1.0 (g10 in the header) to log_g= +5.0 (g50 in the header). Models for gravities log_g= +0.0 and +0.5 are not available for this particular metallicity and effective temperature combination, and therefore do not appear listed in the header. Their corresponding columns (g00 and g05) are filled with zeros. The models are in FLAM surface flux units, i.e. ergs cm^{-2} s^{-1} A^{-1}.

Table A.5: Sample FITS Header from Kurucz 93 Model

```
SIMPLE  =                    T / file does conform to FITS standard
BITPIX  =                   16 / number of bits per data pixel
NAXIS   =                    0 / number of data axes
EXTEND  =                    T / FITS dataset may contain extensions
COMMENT   FITS (Flexible Image Transport System) format defined in Astronomy and
COMMENT   Astrophysics Supplement Series v44/p363, v44/p371, v73/p359, v73/p365.
COMMENT   Contact the NASA Science Office of Standards and Technology for the
COMMENT   FITS Definition document #100 and other FITS information.
ORIGIN  = 'STScI-STSDAS/TABLES' / Tables version 1999-03-22
FILENAME= 'kp00_8000.fits'     / name of file
TEFF    =                 8000
LOG_Z   = 0.00000000000000E+00
HISTORY   g10
HISTORY   g15
HISTORY   g20
HISTORY   g25
HISTORY   g30
HISTORY   g35
HISTORY   g40
HISTORY   g45
HISTORY   g50
HISTORY   Kurucz model atmospheres (1993)
HISTORY   Fluxes tabulated in units of erg/s/cm^2/A
HISTORY   are surface fluxes. To transform to observed
HISTORY   fluxes multiply by (R/D)^2 where R is the
HISTORY   radius of the star and D the distance.
HISTORY   Each column in the table represents the
HISTORY   spectrum of a star for the same metallicity
HISTORY   and effective temperature but different gravity.
END
```

Physical fluxes of the spectra are given in FLAM surface flux units, i.e. ergs cm^{-2} s^{-1} A^{-1}. These flux units differ from those in the Kurucz CD-ROM by a factor of 3.336 x 10^{-19} x $lambda^2$ x $(4pi)^{-1}$, i.e. are converted from ergs cm^{-2} s^{-1} Hz^{-1} $steradian^{-1}$ to ergs cm^{-2} s^{-1} A^{-1}. To convert to observed flux at Earth, multiply by a factor of $(R/D)^2$ where R is the stellar radius, and D is the distance to Earth.

The names of the files located in each metallicity subdirectory are listed in the README file located in each subdirectory. The range in gravity covered by the models for the different temperatures is also indicated.

A.4 USE OF KURUCZ ATLAS WITH SYNPHOT

Synphot tasks now permit the use of spectra selected from one of many columns in a single STSDAS table file. One does this by specifying as the "spectrum" parameter the name of the disk file (as before), and appending the name of the column containing the flux in brackets. Thus, to select any model spectrum characterized by a given metallicity, effective temperature, and gravity, specify a "spectrum" of the form: crgridk93$m_directory/kszz_ttttt.fits[gyy], where m_directory is the name of the subdirectory for a given metallicity. For example, to select the spectrum of a star with a metallicity of +0.1, a temperature of 10,000 K, and log gravity of 3.0, the specification would be: crgridk93$kp01/kp01_10000.fits[g30].

Alternatively, one may select a model spectrum within a **synphot** expression using the function **icat** (which interpolates between the nearest matching models) or **cat** (which simply selects the closest match). The syntax for the Kurucz models is icat(k93models,T_{eff},metallicity,logG). See section 5.3.1 for an example that includes the use of this function.

The **icat** task can give incorrect results without warning if it is used at the edges of the defined grid of models. See above for a discussion of the available ranges for these values, and exercise caution when selecting a value near one of the extremes.

Please note that the model spectra in the atlas are in surface flux units. Thus, if the number of counts or the calculated absolute flux is needed, the model spectrum must be renormalized appropriately. One can do this in **synphot** with the "rn" function.

Since the entire atlas occupies close to 70MB of disk space, many applications could be satisfied by a copy of the solar metallicity spectra, only.

A list of solar metallicity stars of different spectral types and luminosity classes together with their closest Kurucz model spectrum is presented in Table A.6. The physical parameters, T_{eff} and log_g, characterizing each star are taken from Schmidt-Kaler's compilation of physical parameters of stars (Schmidt-Kaler 1982, Landolt-Bornstein VI/2b). The U-B and B-V colors of the closest model agree with the characteristic color of each star (see Schmidt-Kaler 1982) to better than 0.06 magnitude.

Table A.6: Suggested models for specific stellar types

Type	T_{eff}	log_g	Kurucz model
O3V	52500	+4.14	kp00_50000[g50]
O5V	44500	+4.04	kp00_45000[g50]
O6V	41000	+3.99	kp00_40000[g45]
O8V	35800	+3.94	kp00_35000[g40]
B0V	30000	+3.9	kp00_30000[g40]
B3V	18700	+3.94	kp00_19000[g40]
B5V	15400	+4.04	kp00_15000[g40]
B8V	11900	+4.04	kp00_12000[g40]
A0V	9520	+4.14	kp00_9500[g40]
A5V	8200	+4.29	kp00_8250[g45]
F0V	7200	+4.34	kp00_7250[g45]
F5V	6440	+4.34	kp00_6500[g45]
G0V	6030	+4.39	kp00_6000[g45]
G5V	5770	+4.49	kp00_5750[g45]
K0V	5250	+4.49	kp00_5250[g45]
K5V	4350	+4.54	kp00_4250[g45]
M0V	3850	+4.59	kp00_3750[g45]
M2V	3580	+4.64	kp00_3500[g45]
M5V	3240	+4.94	kp00_3500[g50]
B0III	29000	+3.34	kp00_29000[g35]
B5III	15000	+3.49	kp00_15000[g35]
G0III	5850	+2.94	kp00_5750[g30]
G5III	5150	+2.54	kp00_5250[g25]
K0III	4750	+2.14	kp00_4750[g20]
K5III	3950	+1.74	kp00_4000[g15]
M0III	3800	+1.34	kp00_3750[g15]
O5I	40300	+3.34	kp00_40000[g45]
O6I	39000	+3.24	kp00_40000[g45]
O8I	34200	+3.24	kp00_34000[g40]
BOI	26000	+2.84	kp00_26000[g30]
B5I	13600	+2.44	kp00_14000[g25]
AOI	9730	+2.14	kp00_9750[g20]
A5I	8510	+2.04	kp00_8500[g20]
F0I	7700	+1.74	kp00_7750[g20]
F5I	6900	+1.44	kp00_7000[g15]
G0I	5550	+1.34	kp00_5500[g15]
G5I	4850	+1.14	kp00_4750[g10]
K0I	4420	+0.94	kp00_4500[g10]
K5I	3850	+0.34	kp00_3750[g05]
M0I	3650	+0.14	kp00_3750[g00]
M2I	3450	-0.06	kp00_3500[g00]

A.5 Bruzual Spectrum Synthesis Atlas

Gustavo Bruzual has provided 77 stellar spectra frequently used in synthesis of galaxy spectra. They are available in directory `crgridbz77$` and are stored in 77 individual tables called `bz_`*nn*`.fits`, where *nn* runs from 1 to 77. A list of the file names and spectral types is shown in Table A.7.

Table A.7: Bruzual Synthetic Spectral Atlas

File	Type	File	Type	File	Type
bz_1	O5 V	bz_31	K9 V	bz_61	M4 III
bz_2	O7 V	bz_32	M0 V	bz_62	M5 III
bz_3	O8 V	bz_33	M1 V	bz_63	M6 III
bz_4	O9 V	bz_34	M2 V	bz_64	O7 I
bz_5	B0 V	bz_35	M6 V	bz_65	O9.5 I
bz_6	B1 V	bz_36	O8 III	bz_66	B0 I
bz_7	B2 V	bz_37	O9 III	bz_67	B0.5 I
bz_8	B3 V	bz_38	B0.5 III	bz_68	B5 I
bz_9	B5 V	bz_39	B1 III	bz_69	B8 I
bz_10	B7 V	bz_40	B2 III	bz_70	A2 I
bz_11	B8 V	bz_41	B5 III	bz_71	F0 I
bz_12	B9 V	bz_42	B7 III	bz_72	F2 I
bz_13	A0 V	bz_43	B8 III	bz_73	F8 I
bz_14	A1 V	bz_44	B9.5 III	bz_74	G0 I
bz_15	A2 V	bz_45	A0 III	bz_75	G2 I
bz_16	A3 V	bz_46	A2 III	bz_76	G8 I
bz_17	A5 V	bz_47	A3 III	bz_77	M1M2 I
bz_18	A7 V	bz_48	A5 III		
bz_19	F2 V	bz_49	A7 III		
bz_20	F5 V	bz_50	F0 III		
bz_21	F6 V	bz_51	G0 III		
bz_22	F7 V	bz_52	G8 III		
bz_23	F8 V	bz_53	G9 III		
bz_24	G0 V	bz_54	K0 III		
bz_25	G1 V	bz_55	K2 III		
bz_26	G2 V	bz_56	K5 III		
bz_27	G5 V	bz_57	K6 III		
bz_28	K0 V	bz_58	M0 III		
bz_29	K3 V	bz_59	M1 III		
bz_30	K5 V	bz_60	M3 III		

A.6 Gunn-Stryker Spectrophotometry Atlas

This is an optical spectrophotometric catalog of 175 stars covering a complete range of spectral types and luminosity classes from the observations of Gunn and Stryker (1983). The spectra cover the wavelength range 3130 to 10800 Å and are located in the directory `crgridgs$`. The list of stars contained in this atlas is identical to the BPGS atlas, and are shown in Table A.8. Note that when referring to Table A.8 for the names of files in the Gunn-Stryker atlas, the names are of the form `gs_`*`nnn`*`.fits`, instead of `bpgs_`*`nnn`*`.fits`.

A.7 Bruzual-Persson-Gunn-Stryker Spectrophotometry Atlas

This is an extension of the Gunn-Stryker optical atlas where the spectral data have been extended into both the UV and the infrared. They are available as 175 STSDAS tables in the directory `crgridbpgs$`. The IR data are from Strecker et al. (1979) and other unpublished sources. The IR and optical data were tied together by the *V–K* colors.

Note that the spectral data for all of the stars in this atlas have been arbitrarily renormalized to a *V* magnitude of zero. Therefore in order to use these data for calculations of absolute photometry they must be renormalized to their appropriate absolute levels.

The magnitudes and colors stored in header keywords in each of the tables in this atlas are *not* on the standard *UBVRI* system. They are "scanner" magnitudes and colors that were synthesized by the authors from the observed spectra (see Gunn and Stryker 1983).

The file names, star names, and spectral types for the BPGS atlas are listed in Table A.8..

Table A.8: Bruzual-Persson-Gunn-Stryker Spectral Atlas

File	Star	Type	File	Star	Type
bpgs_1	9 SGR	O5	bpgs_35	HD 35296	F8 V
bpgs_2	9 SGE	O8 F	bpgs_36	BD +26 3780	G0 V
bpgs_3	HR 8023	O6	bpgs_37	HD 148816	F9 V
bpgs_4	BD -01 0935	B1 V	bpgs_38	HD 155675	F8 V
bpgs_5	60 CYG	B1 V	bpgs_39	PRAESEPE 418	
bpgs_6	102 HER	B2 V	bpgs_40	HYAD 1	
bpgs_7	ETA HYA	B3 V	bpgs_41	HD 122693	F8 V
bpgs_8	IOTA HER	B3 V	bpgs_42	HD 154417	F8 V
bpgs_9	HR 7899	B4 V	bpgs_43	HYAD 2	
bpgs_10	38 OPH	A1 V	bpgs_44	HD 227547	G5 V
bpgs_11	HR 7174	B6 V	bpgs_45	HD 154760	G2 V
bpgs_12	9 VUL	B7 V	bpgs_46	HD 190605	G2 V
bpgs_13	HD 189689	B9 V	bpgs_47	HYAD 15	
bpgs_14	THETA VIR	A0 V	bpgs_48	HD 139777A	K0 V
bpgs_15	NU CAP	B9 V	bpgs_49	HD 136274	G8 V
bpgs_16	HR 6169	A2 V	bpgs_50	HYAD 26	
bpgs_17	HD 190849A	A1 V	bpgs_51	HD 150205	G5 V
bpgs_18	69 HER	A2 V	bpgs_52	HYAD 21	
bpgs_19	HD 190849B	A3 V	bpgs_53	BD +02 3001	G8 V
bpgs_20	58 AQL	A0 V	bpgs_54	HD 190571	G8 V
bpgs_21	78 HER	B9 V	bpgs_55	HYAD 183	
bpgs_22	HR 6570	A7 V	bpgs_56	HD 190470	K3 V
bpgs_23	HD 187754	A2 V	bpgs_57	HD 154712	K4 V
bpgs_24	THETA1 SER	A5 V	bpgs_58	HYAD 185	
bpgs_25	PRAESEPE 276		bpgs_59	BD +38 2457	K8 V
bpgs_26	PRAESEPE 114		bpgs_60	HYAD 173	
bpgs_27	PRAESEPE 154		bpgs_61	GL 40	M0 V
bpgs_28	HD 190192	A5 V	bpgs_62	HYAD 189	
bpgs_29	PRAESEPE 226		bpgs_63	HD 151288	K7 V
bpgs_30	PRAESEPE 37		bpgs_64	HD 157881	K7 V
bpgs_31	HD 191177	F4 V	bpgs_65	HD 132683	M0 V
bpgs_32	PRAESEPE 332		bpgs_66	GL 15A	M0 V
bpgs_33	BD +29 3891	F6 V	bpgs_67	GL 49	M2 V
bpgs_34	PRAESEPE 222		bpgs_68	GL 109	M4 V

Table A.8: Bruzual-Persson-Gunn-Stryker Spectral Atlas

File	Star	Type	File	Star	Type
bpgs_69	GL 15B	M6 V	bpgs_104	HD 56176	G7 IV
bpgs_70	GL 83.1	M8 V	bpgs_105	HD 227693	G5 IV
bpgs_71	GL 65	M5 V	bpgs_106	HD 199580	K2 IV
bpgs_72	HR 7567	B1 IV	bpgs_107	HD 152306	G8 III
bpgs_73	HR 7591	B2 III	bpgs_108	PRAESEPE 212	
bpgs_74	20 AQL	B3 IV	bpgs_109	THETA1 TAU	G8 III
bpgs_75	HR 7467	B3 III	bpgs_110	HD 170527	G5 IV
bpgs_76	IOTA LYR	B7 IV	bpgs_111	HD 136366	K0 III
bpgs_77	HR 7346	B7 III	bpgs_112	HD 191615	G8 IV
bpgs_78	59 HER	A3 III	bpgs_113	HD 124679	K0 III
bpgs_79	HR 6642	A0 IV	bpgs_114	HD 131111	K0 III
bpgs_80	11 SGE	B9 IV	bpgs_115	HD 113493	K0 III
bpgs_81	60 HER	A3 IV	bpgs_116	HD 4744	G8 IV
bpgs_82	HD 192285	A4 IV	bpgs_117	HD 7010	K0 IV
bpgs_83	ALPHA OPH	A5 III	bpgs_118	46 LMI	K0 III
bpgs_84	HD 165475B	A5 IV	bpgs_119	91 AQR	K0 III
bpgs_85	HD 165475	A5 IV	bpgs_120	M67 F141	
bpgs_86	XI SER	F0 IV	bpgs_121	HR 8924A	K3 III
bpgs_87	HD 5132	F0 IV	bpgs_122	HD 140301	K0 IV
bpgs_88	HD 508	A9 IV	bpgs_123	HD 95272	K0 III
bpgs_89	HD 210875	F0 IV	bpgs_124	HD 72184	K2 III
bpgs_90	RHO CAP	F2 IV	bpgs_125	HD 119425	K2 III
bpgs_91	HD 7331	F7 IV	bpgs_126	HD 106760	K1 III
bpgs_92	BD +63 0013	F5 IV	bpgs_127	PSI UMA	K1 III
bpgs_93	HD 13391	G2 IV	bpgs_128	PHI SER	K1 III
bpgs_94	HD 154962	G8 IV	bpgs_129	HD 136514	K3 III
bpgs_95	HD 192344	G4 IV	bpgs_130	MU AQL	K3 III
bpgs_96	HR 6516	G6 IV	bpgs_131	HR 5227	K2 IIIp
bpgs_97	HR 7670	G6 IV	bpgs_132	HD 154759	K3 III
bpgs_98	HD 128428	G3 IV	bpgs_133	20 CYG	K3 III
bpgs_99	31 AQL	G8 IV	bpgs_134	ALPH SER	K2 III
bpgs_100	BD -02 4018	G5 IV	bpgs_135	MU LEO	K2 III
bpgs_101	M67 F143?		bpgs_136	BD +01 3131	K0 III
bpgs_102	HD 11004	G5 IV	bpgs_137	M67 F170	
bpgs_103	HD 173399A	G5 IV	bpgs_138	18 LIB A	K2 IIIp

Table A.8: Bruzual-Persson-Gunn-Stryker Spectral Atlas

File	Star	Type	File	Star	Type
bpgs_139	BD +28 2165	K1 IV	bpgs_161	BD -01 3097	M2 III
bpgs_140	NGC 188 1-69		bpgs_162	TX DRA	
bpgs_141	BD +30 2344	K3 III	bpgs_163	Z CYG	M8 III
bpgs_142	HD 83618	K3 III	bpgs_164	BD +01 3133	M5 III
bpgs_143	HD 158885	K3 III	bpgs_165	BD -02 3886	M5 III
bpgs_144	HD 166780	K5 III	bpgs_166	W HER	M6 III
bpgs_145	HD 148513	K4 III	bpgs_167	TY DRA	M8
bpgs_146	M67 T626		bpgs_168	SW VIR	M7 III
bpgs_147	HD 127227	K5 III	bpgs_169	RZ HER	M6 III
bpgs_148	M67 IV-202		bpgs_170	R LEO	
bpgs_149	HD 50778	K4 III	bpgs_171	AW CYG	N
bpgs_150	HD 62721	K5 III	bpgs_172	WZ CAS	N
bpgs_151	HD 116870	M0 III	bpgs_173	69 CYG	B0 Ib
bpgs_152	HD 60522	M0 III	bpgs_174	HR 7699	B5 Ib
bpgs_153	BD -01 3113	K5 III	bpgs_175	HR 8020	B8 Ia
bpgs_154	BD +02 2884	K5 III			
bpgs_155	BD -02 3873	M0 III			
bpgs_156	HD 104216	M2 III			
bpgs_157	HD 142804	M1 III			
bpgs_158	HD 30959	M3 III			
bpgs_159	HD 151658	M2 III			
bpgs_160	BD -02 4025	M2 III			

A.8 Jacoby-Hunter-Christian Spectophotometry Atlas

This is an optical spectrophotometric atlas of 161 stars having spectral classes O through M and luminosity classes V, III, and I and are from the observations of Jacoby, Hunter, and Christian (1984). The spectra cover the wavelength range 3510 to 7427 Å at a resolution of approximately 4.5 Å. The spectra are available in STSDAS tables in the directory `crgridjac$`. The file names, star names, and spectral types for the JHC atlas are listed in Table A.9

Table A.9: Jacoby-Hunter-Christian Spectral Atlas

File	Star	Type	File	Star	Type
jc_1	HD 242908	O5 V	jc_35	SAO 57199	F6 V
jc_2	HD 215835	O5.5 V	jc_36	HD 24189	F6 V
jc_3	HD 12993	O6.5 V	jc_37	HD 5702	F7 V
jc_4	HD 35619	O7 V	jc_38	HD 107132	F7 V
jc_5	HD 44811	O7.5 V	jc_39	HD 107214	F7 V
jc_6	HD 242935	O8 V	jc_40	HD 6111	F8 V
jc_7	HD 236894	O8 V	jc_41	HD 31084	F9 V
jc_8	HD 17520	O9 V	jc_42	HD 107399	F9 V
jc_9	HD 12323	O9 V	jc_43	HD 28099	G0 V
jc_10	BD +62 0249	O9.5 V	jc_44	HD 17647	G1 V
jc_11	HD 158659	B0 V	jc_45	HD 66171	G2 V
jc_12	HD 237007	B0 V	jc_46	BD +58 1199	G3 V
jc_13	HD 35215	B1.5 V	jc_47	Tr A14	G4 V
jc_14	HD 37767	B3 V	jc_48	HD 22193	G6 V
jc_15	FEIGE 40	B4 V	jc_49	HD 27685	G7 V
jc_16	HD 240344	B4 V	jc_50	HD 33278	G9 V
jc_17	HD 30584	B6 V	jc_51	HD 29050	G9 V
jc_18	O 1015	B8 V	jc_52	HD 23524	K0 V
jc_19	HD 116608	A1 V	jc_53	HD 5351	K4 V
jc_20	FEIGE 41	A1 V	jc_54	SAO 76803	K5 V
jc_21	HD 124320	A2 V	jc_55	HD 260655	M0 V
jc_22	FEIGE 28	A2 V	jc_56	BD +63 0137	M1 V
jc_23	HD 190785	A2 V	jc_57	YALE 1755	M5 V
jc_24	HD 221741	A3 V	jc_58	HD 13505	F0 IV
jc_25	HD 9547	A5 V	jc_59	HD 83140	F3 IV
jc_26	HD 21619	A6 V	jc_60	HD 78277	G2 IV
jc_27	HD 23863	A7 V	jc_61	HD 70178	G5 IV
jc_28	HD 111525	A7 V	jc_62	HD 227018	O6.5 III
jc_29	HD 9972	A8 V	jc_63	SAO 11810	O7.5 III
jc_30	HD 23733	A9 V	jc_64	HD 191978	O8.5 III
jc_31	HD 10032	F0 V	jc_65	HD 16429	O9.5 III
jc_32	Hz 948	F3 V	jc_66	HD 13494	B1 III
jc_33	HD 23511	F4 V	jc_67	HD 12727	B2 III
jc_34	Hz 227	F5 V	jc_68	O 2311	B2 III

Table A.9: Jacoby-Hunter-Christian Spectral Atlas (Continued)

File	Star	Type	File	Star	Type
jc_69	SAO 11885	B2.5 III	jc_104	HD 110964	M4 III
jc_70	HD 166125	B3 III	jc_105	SAO 62808	M5 III
jc_71	HD 39136	B4 III	jc_106	HD 14357	B2 II
jc_72	HD 56183	B4 III	jc_107	BD -14 4956	B2 II
jc_73	HD 256413	B5 III	jc_108	BD -00 3227	F5 II
jc_74	BD +61 0339	B7 III	jc_109	HD 249384	G8 II
jc_75	HD 28696	B8 III	jc_110	HD 250368	G9 II
jc_76	HD 20023	B9 III	jc_111	HD 249826	K6 II
jc_77	HD 12027	A3 III	jc_112	BD +19 1947	M3 II
jc_78	HD 240296	A6 III	jc_113	BD +36 1963	M7 II
jc_79	HD 12161	A8 III	jc_114	BD -11 4586	O8 I
jc_80	HD 64191	F0 III	jc_115	HD 225160	O8 I
jc_81	HD 5211	F4 III	jc_116	HD 16808	B0.5 Ib
jc_82	BD +61 0367	F5 III	jc_117	HD 167451	B0.5 Ib
jc_83	HD 56030	F6 III	jc_118	BD -14 5030	B1.5 Ia
jc_84	SAO 20603	F7 III	jc_119	HD 209678	B2 Ib
jc_85	HD 9979	F8 III	jc_120	SAO 20899	B3 I
jc_86	HD 15866	G0 III	jc_121	HD 192832	B5 Ia
jc_87	BD +30 2347	G0 III	jc_122	BD +61 0220	B7 I
jc_88	HD 25894	G2 III	jc_123	HD 17145	B8 Ia
jc_89	HD 2506	G4 III	jc_124	LSI V P24	B9 Ib
jc_90	BD +28 1885	G5 III	jc_125	HD 209900	A0 Ib
jc_91	HD 112872	G6 III	jc_126	SAO 11344	A0 Ib
jc_92	HD 26514	G6 III	jc_127	SAO 12149	A1 I
jc_93	HD 29883	G6 III	jc_128	42 LSI	A2 I
jc_94	HD 249240	G7 III	jc_129	SAO 87716	A3 Ia
jc_95	HD 245389	G8 III	jc_130	SAO 12096	A4 I
jc_96	SAO 55155	G9 III	jc_131	HD 9167	A7 I
jc_97	SAO 55164	K0 III	jc_132	HD 842	A9 I
jc_98	HD 33506	K2 III	jc_133	SAO 37370	F0 Ib
jc_99	SAO 77849	K2 III	jc_134	BD +58 0204	F2 I
jc_100	HD 26946	K3 III	jc_135	HD 12842	F3 I
jc_101	HD 21110	K4 III	jc_136	SAO 21536	F4 I
jc_102	SAO 21753	K7 III	jc_137	HD 9973	F5 Iab
jc_103	SAO 63349	M3 III	jc_138	HD 8992	F6 Ib

Table A.9: Jacoby-Hunter-Christian Spectral Atlas (Continued)

File	Star	Type	File	Star	Type
jc_139	HD 17971	F7 I	jc_151	SAO 23888	M1 I
jc_140	HD 187428	F8 Ib	jc_152	HD 13136	M2 Ib
jc_141	HD 25361	G0 Ia	jc_153	HD 94028	F4 V
jc_142	SAO 21446	G1 I	jc_154	SAO 102986	F7 V
jc_143	BD +56 0084	G2 I	jc_155	SAO 81292	M4.5 Ve
jc_144	HD 191010	G3 Ib	jc_156	HD 16691	O5 If
jc_145	HD 187299	G5 I	jc_157	HD 108	O6 If
jc_146	HD 186293	K0 I	jc_158	BD +40 4220	O7 If
jc_147	SAO 37325	K1 Ib	jc_159	HD 13256	B1 Ia
jc_148	HD 1069	K2 I	jc_160	HD 50064	B1 Ia
jc_149	HD 1400	K5 I	jc_161	BD +51 0710	B5 Ib
jc_150	HD 14330	M1 Iab			

A.9 Bruzual-Charlot Atlas

This is a library of galaxy spectra computed by Bruzual and Charlot using their Isochrone Synthesis Spectral Evolutionary Code. The December 1995 version of the Bruzual-Charlot atlas consists of 21 instantaneous bursts characterized by different IMF slope functions, and mass limits. Each instantaneous burst contains 221 spectral energy distributions (SEDs) corresponding to 221 time steps from 0 to 20 Gyr.

The spectra represent bursts characterized by a Salpeter initial mass function (IMF) with different ranges in lower and upper mass limits, and at several ages (10E5, 25E5, 50E5, 76E5, 10E6, 25E6, 50E6, 10E7, 50E7, 10E8, 50E8, 10E9 years) after the burst. Spectra for instantaneous and composite bursts are available. Each spectrum has 1187 wavelength points covering the 100 Angstroms to 100 microns wavelength range, and the fluxes are given in units of solar luminosity per Angstrom. The nebular contribution to the spectral energy distribution, i.e., emission lines and nebular continuum, is not included in the spectra.

The spectra have names of the form `bc95_a_XXXX` through `bc95_g_XXXX` and are located in the `crgrid$bc95/templates/` directory.

A.10 Kinney-Calzetti Atlas

This atlas consists of an homogeneous set of 12 spectral templates of galaxies covering the ultraviolet, optical and near-infrared wavelength range up to about 1 micron. Templates include various morphological types and starburst galaxies. The ultraviolet range of the spectral templates has been obtained with the large aperture (10" by 20") and low resolution spectrographs of the IUE satellite. The optical spectra were obtained through a long slit with a 10" width, where a window of 20" long was extracted to mach the IUE aperture.

The spectral templates cover various galaxy morphological types from elliptical to late type spiral. Starburst templates for low (E(B-V) < 0.10) to high (0.61 < E(B-V) < 0.70) internal extinction are also available. Several of the starburst galaxies used in the construction of the starburst templates are classified as irregulars. Thus, although irregular galaxies are not explicitly covered, the starburst templates can be used to cover this morphological type.

The flux of the spectral templates has been normalized to a visual magnitude of 12.5 (STMAG system). Details about each individual template can also be found in the header of the STSDAS binary file. The spectra are located in the `crgrid$kc96/` directory.

A.11 Buser-Kurucz Atlas of Model Atmospheres

Roland Buser has provided an extensive collection of Kurucz model atmosphere spectra covering a wide range in metallicity, effective temperature, and gravity. They are organized into several atlases, A, B, C, D, M, and S, located in the directory `crgridbk$`.

This atlas contains models computed by Kurucz in 1979. For a more complete and updated library of stellar atmosphere models, use the Kurucz 1993 atlas described above.

The BK atlas is a library of stellar flux spectra calculated by Kurucz and Buser from theoretical model atmospheres. The spectra were computed for a wide range of physical parameters, covering essentially the whole HR diagram observed for galactic stars. For all the spectra, fluxes are given mostly with a resolution of 25 Å on a uniform grid of wavelengths from the UV to the infrared. The BK atlas is thus especially suited for synthetic photometry applications (cf. Buser 1986, and references below), which

also plays a major role in the calibration and the interpretation of HST observations (Koornneef et al. 1986). Therefore, the BK atlas has been implemented in the Calibration Data Base System (CDBS) at STScI.

The BK atlas consists of 1434 files, each of which represents a metal-line blanketed flux spectrum for a theoretical stellar model atmosphere. There files were prepared from two original data libraries, K1200 and BKLATE.

The K1200 library was computed by Kurucz (1979a,b) from his "ATLAS" model atmospheres for early-type (O to G) stars. The 1200 models were computed from three different versions of the "ATLAS" code, allowing for changes or improvements in the underlying physics, as shown in Table A.10:

Table A.10: Kurucz K1200 Library

Models	Code/Description	Reference
K1200 1- 284	A /ATLAS6 original	Kurucz 1979a
K1200 285- 609	APR/ATLAS6 purely radiative	Kurucz 1979b
K1200 610-1200	AIC/ATLAS6 improved convective	Kurucz 1979b

K1200 models 1–284, including their flux spectra, have been fully published by Kurucz (1979a), while models 285–1200 were described by Kurucz (1979b), and have been widely distributed but not published.

The BKLATE library was computed by Buser and Kurucz (1985, 1988, 1992) for late-type (F to K) stars from published as well as unpublished "GBEN" model atmospheres calculated by Gustafsson et al. (1975) and by Eriksson et al. (1979), respectively. Line-blanketed flux spectra were computed following the same procedure and employing the same opacity source input as in Kurucz (1979a). In particular, all the spectra were calculated using the extensive "KP" atomic line lists compiled by Kurucz and Peytremann (1975), and assuming a fixed turbulent velocity, vturb=2.00 km/s, and a fixed helium abundance, N(He)=0.10 (i.e., a number fraction of 10%). The BKLATE library data have not been published yet.

For convenient use, the BK atlas flux spectra have been subdivided into separate blocks (or sub-atlases) corresponding to the physical distinctions of their underlying model atmospheres, as described above. Table A.11 summarizes the structure of the atlas.

Table A.11: BK Atlas

Block	Source of Spectra	# of Spectra	Models + Opacity	T_{eff}	Log G	[M/H]
A	K1200	279	A +KP	5500–50000	0.00–5.00	0/–1/–2
B	K1200	323	APR +KP	8000–20000	1.00–4.50	1/.5/–.5/–1
C	K1200	590	AIC +KP	5500– 8500	0.00–4.50	1 to –9.99
D	BKLATE	233	GBEN +KP	3750– 6000	0.75–5.25	.5 to –3
S	K1200	3	A/AIC+KP	5770,9400	4.44,3.95	0
	BKLATE	1	GBEN +KP	5780	4.44	0
M	K1200	5	A/APR+KP	9400–10000	3.90–4.30	0/.5/1

Notice that within each of blocks A through D, models are available for ranges of usually uniformly spaced parameter values. Models are not available, however, for all possible combinations of parameter values. Block S contains models for two of the most prominent standard stars: three models of the Sun, and one model of Vega. Block M contains five miscellaneous models with special parameter combinations.

The file names in the `crgridbk$` directory are of the form `bk_`*mnnnn*`.fits`, where *m* is the block code (a, b, c, d, s, or m), and *nnnn* is a running sequence number within the block. Within each block, the individual models are ordered starting with the lowest metallicity, [M/H], temperature, T_{eff}, and surface gravity, log G, and with log G increasing fastest and [M/H] increasing most slowly. A complete listing of all the BK atlas model spectra can be found in the README file in the `crgridbk$` directory.

For all 1434 models in the BK atlas, flux spectra are given for the same set of 342 wavelengths as was used with the published Kurucz (1979a) models. Most of these wavelength points cover the range between 229 Å in the UV and about 2 microns in the IR, with spacings between 25 and 100 Å.

Fluxes in the BK atlas tables, tabulated in units of `fnu`, are *surface* fluxes. Therefore, calculations of absolute luminosities for the BK atlas models require additional assumptions about the radii of the stars represented by the models.

A number of synthetic photometry applications have already been made using the BK atlas, mainly for the hotter Kurucz models from the K1200 library. A few pertinent references for theoretical calibrations of photometric systems are Lub and Pel (1977) for the Walraven *VBLUW*

system, Buser and Kurucz (1978) for the Johnson *UBV* system, Relyea and Kurucz (1978) and Lester et al. (1986) for the Stromgren *uvby* system, and Buser and Kurucz (1988, 1992) for the Johnson-Cousins *UBVRI* system.

A.12 AGN Atlas

This atlas consists of a few spectral templates of AGNs ranging from LINER to Seyfert and bright QSO. The LINER and Seyfert 2 templates have been obtained with the large aperture (10" by 20") and low resolution spectrographs of the IUE satellite. The optical spectra were obtained through a long slit with a 10" width, where a window of 20" long was extracted to mach the IUE aperture (Calzetti 1995, private comm.). The flux of the LINER and Seyfert2 templates is normalized to a Johnson visual magnitude of 12.5 (STMAG).

The Seyfert1 template consists of an UV spectrum obtained with the IUE low resolution spectrographs and of a ground-based optical spectrum. The bright QSO template is a composite spectrum from the Large Bright Quasar Survey of Francie and collaborators (1991). The Seyfert1 and QSO spectral templates are normalized to a Johnson blue magnitude of 12.5 (STMAG).

The NGC 1068 template is a composite spectrum. The continuum contains the nebular, stellar, and power-law contributions. The observed fluxes and FWHM of the UV, optical and near-IR emission lines are also incorporated into the template (J.R. Walsh, private comm; read also the header of the FITS table for further details).

The spectra are located in the `crgrid$agn/` directory.

A.13 Galactic Atlas

This atlas consists of model spectra of the Orion Nebula and of the NGC 7009 planetary Nebula.

The continuum of the Orion Nebula's template contains the nebular contribution plus a combination of Kurucz model atmospheres to simulate the stellar contribution. The fluxes of the UV, optical and near-IR emission lines from different sources are also incorporated into the template (J.R. Walsh, private comm.).

The continuum of the planetery nebula has a nebular component and a hot stellar component simulated by an 80000K black body. The fluxes of the UV, optical and near-IR emission lines, from different sources, are also incorporated into the template (J.R. Walsh, private comm.).

The spectra are located in the `crgrid$galactic/` directory.

APPENDIX B:
Bibliography

Bessell, M.S., 1983, *PASP*, 95, 480.

Bessell and Brett (1988) *PASP*, 100, 134

Bessel, Castelli, and Plez, 1998, A&A 333:231, "Model atmospheresbroad-band colors, bolometric corrections and temperature calibrations for O - M stars"

Bohlin, R.C. 1996, AJ, 111, 1743

Bohlin, R.C. 2000, AJ, 120, 437

Bohlin, R. 2003, 2002 HST Calibration Workshop, ed. S. Arribas, A. Koekemoer, & B. Whitmore, (Baltimore:STScI), p. 115

Bohlin, R. C., & Gilliland, R. L. 2004, AJ, 127, 3508, "HST Absolute

Spectrophotometry of Vega from the Far-UV to the IR."

Bohlin, R.C., Colina, L., & Finley, D.S. 1995, AJ, 110, 1316

Bohlin, R. C., Dickinson, M. E., & Calzetti, D. 2001, AJ, 122, 2118

Buser, R., 1986, in *Highlights of Astronomy*, Vol. 7, J.P. Swings, ed., Reidel, Dordrecht, p. 799.

Buser, R. and Kurucz, R.L., 1978, *A&A*, 70, 555.

Buser, R. and Kurucz, R.L., 1985, in *Calibration of Fundamental Stellar Quantities*, IAU Symp. No. 111, D.S. Hayes, L.E. Pasinetti, A.G. Davis-Philip, eds., Reidel, Dordrecht, p. 513.

Buser, R. and Kurucz, R.L., 1988, in *The Impact of Very High S/N Spectroscopy on Stellar Physics*, IAU Symp. No. 132, G. Cayrel de Strobel, M. Spite, eds., Reidel, Dordrecht, p. 531.

Buser, R. and Kurucz, R.L., 1992, *A&A*, 264, 557.

Cohen, Wheaton, and Megeath, 2003, ApJ 126:1090, "Spectral Irradiance Calibration in the Infrared XIV: The Absolute Calibration of 2MASS"

Colina, L., & Bohlin, R. 1994, AJ, 108, 1931

Colina, L., & Bohlin, R. 1997, AJ, 113, 1138

Colina, L., Bohlin, R.C., & Castelli, F. 1996, AJ, 112, 307

DeMarchi et al, 2004, ISR ACS 2004-08, "Detector Quantum Efficiency and Photometric Zero Points of the ACS"

Eriksson, K., Bell, R.A., Gustafsson, B., Nordlund, A., 1979, *Trans. IAU*, Vol. 17A, Part 2, Reidel, Dordrecht, p. 200

Francis et al. 1991, *ApJ* 373, 465 .

Greenfield, P. & White, R. L. 2000, in ASP Conf. Ser., Vol. 216, Astronomical Data Analysis Software and Systems IX, eds. N. Manset, C. Veillet, D. Crabtree (San Francisco: ASP), 59

Gunn, J.E. and Stryker, L.L., 1983, *ApJS*, 52, 121.

Gustafsson, B., Bell, R.A., Eriksson, K., Nordlund, A., 1975, *A&A*, 42, 407.

Harris, H., Baum, W., Hunter, D. and Kreidl, T., 1991, *AJ*, 101, 677.

Heyer et al 2004, WFPC2 ISR 2004-01,"The Accuracy of WFPC2 Photometric Zeropoints"

Hodge, P.E., 2004, "PyRAF Programmer's Guide"

Horne, K., 1988, in *New Directions in Spectrophotometry*, A.G.D. Philip, D.S. Hayes, and S.J. Adelman, eds., L. Davis Press, Schenectady NY, p. 145.

Horne, K., Burrows, C. and Koornneef, J., 1986, "Dynamic Generation of Throughput Functions: A Unified Approach," STScI Memo.

Howarth, I.D., 1983, *MNRAS*, 203, 301.

Jacoby, G.H., Hunter, D.A. and Christian, C.A., 1984, *ApJS*, 56, 257.

Johnson, H.L., 1965, *ApJ*, 141, 923.

Koornneef, J., Bohlin, R., Buser, R., Horne, K. and Turnshek, D., 1986, in *Highlights of Astronomy*, Vol. 7, J.-P. Swinds, ed., Reidel, Dordrecht, p. 833.

Krist, J. & Hook, R. 2004 http://www.stsci.edu/software/tinytim (Ver. 6.3)

Kurucz, R.L., 1979a, *ApJS*, 40, 1.

Kurucz, R.L., 1979b, in *Problems of Calibration of Multicolor Photometric Systems*, Dudley Obs. Rep. No. 14, A.G. Davis Philip, ed., p. 363.

Kurucz, R.L. and Peytremann, E., 1975, Smithsonian Astrophys. Obs. Rep. No. 362.

Kurucz, R. 2003, http://kurucz.harvard.edu/

Landolt, A.U., 1983, *AJ*, 88, 439.

Lester, J.B., Gray, R.O. and Kurucz, R.L., 1986, *ApJS*, 61, 509.

Lub, J. and Pel, J.W., 1977, *A&A*, 54, 137.

Matsushima, S., 1969, *ApJ*, 158, 1137.

Oke, J.B., 1974, *ApJS*, 27, 21.

Oke, J.B., and Gunn, J.E., 1983, *ApJ*, 266, 713.

Oke, J.B. 1990, AJ, 99, 1621

Press, W.H., Teukolsky, S.A., Vetterling, W.T. and Flannery, B.P., 1992, *Numerical Recipes in FORTRAN: The Art of Scientific Computing*, Cambridge University Press, Cambridge.

Relyea, L.L. and Kurucz, R.L., 1978, *ApJS*, 37, 45.

Schneider, D.P., Gunn, J.E., and Hoessel, J.G., 1983, *ApJ*, 264, 337.

Seaton, M.J., 1979, *MNRAS*, 187, 73p.

Strecker, D.W., Erickson, E.F., and Whitteborn, F.C., 1979, *ApJS*, 41, 501.

Turnshek, D.A., Bohlin, R.C., Williamson, R., Lupie, O., Koornneef, J., & Morgan D. 1990, AJ, 99, 1243

White and Greenfield, 1999, "The PyRAF Tutorial"

Index

A

ABMAG
    form parameter 22
accuracy 5
AGN atlas 142

B

bandpar task 31
bandpass
    comparing 31
    examining 31
bb function 20
blackbody function 20
Bruzual spectrum synthesis atlas 131
Bruzual-Charlot atlas 138
Bruzual-Persson-Gunn-Stryker spectrophotometry atlas 132

C

calcband task 31, 34, 44
calcphot task 37, 46
calcspec task 50
cat function 20
combined throughput
    calculating 109
componenet table
    computation 109
component lookup table 6
component table
    default 27
component throughput table 6
countrate task 35, 37, 54
counts 22
    estimating total 37

D

data base
    synphot 4
directories
    location of catalogs and data 122

E

ebmv 20
em 18
embvx 20
emission line function 18
emissivity table 113
examples
    cookbook 31
    general syntax 15
    photometry 37
    spectra 34
expression evaluator
    syncalc 107

F

fitband task 59
fitgrid task 68
fitspec task 63
fitting
    tasks to perform 14
flam 22
fnu 22
form parameter 15, 21

G

galactic atlas 142

galactic extinction 20
gauss 18
genwave task 72
grafcheck task 73
graflist task 74
Grafpath 75
graph table
    components in 74
    computation 109
    default 27
grid function 20
Gunn-Stryker spectrophotometry atlas 132

H

help
    user support xii
hi function 20

I

icat function 20
imspec task 76
instrument graph table 6
    error check 73

J

Jacoby-Hunter-Christian spectrophotometry atlas 135
jy 22

K

keywords
    general discussion 3
Kinney-Calzetti atlas 139
Kurucz icat function 49
Kuruzc model atmospheres spectra
    using in synphot 139

L

lgauss function 18

M

mjy 22
Mkthru 78
Modeinfo 80

N

NICMOS
    filter bandpass efficiency 33

O

obmag 22
observation mode 15
observing mode
    specifying 3
obsmode parameter 15, 16

P

parameter
    form 15, 21
    refdata 15, 101
    spectrum 15, 18
    wavetable 15
passband
    creating 44
    obsmode parameter 15
    reconstruction, tasks to use 14
    retrieving 3
photlam 22
photnu 22
photometric transformation plot
    creating 98
photometry
    examples 37
pl function 20
plband task 31, 82
plot
    spectra 34
plratio task 94
plspec task 34, 85
pltrans task 98
poly function 18
power law function 20
proposal
    utility of synphot 1
PyRAF 7

R

ratio
    observed to synthetic spectra 94
reddening
    ebmvx 20
redshift
    spectra, plotting 39
redshift function 20
refdata parameter 15, 27
refdata pset 101
reference data
    default values 27
    refdata parameter 15
    refdata pset 27
renormalize 20
rn function 20

S

showfiles task 102
software
    obtaining xii
spectra
    examples 34
    photometric measurements 39
    plotting 34
    redshift 39
spectral atlases
    available in synphot 121
spectrum
    creating 50
spectrum parameter 15, 18
STIS
    CCD observation 35
stmag 22
syncalc
    expression evaluator 107
synphot
    capabilities 1
    tasks in package 41
syntax
    commands and parameters 15

T

tasks
    described 13, 41
    passband reconstruction 14
telescope
    HST area 27
    non-HST, using synphot with 6
thermal background 103
Thermback 103
throughput
    calculating combined 109
tilt function 18

U

unit function 20
units
    specifying 21
    specifying in form parameter 15
user support xii

V

variable substitution 26
VEGAMAG
    form parameter 22
VZERO 26

W

wavelength set
    creating 72
wavelength table
    wavetab parameter 25
    wavtable parameter 15
wavetab parameter 25
wavetable parameter 15
WFPC2
    calcphot results 37
    filter response 32

X

XCAL
    relation to synphot 2

Z

z function 20