STSDAS Version 1.3.3
March 1995

Synphot User's Guide

SPACE TELESCOPE SCIENCE INSTITUTE

Science Software Group
Science Support Division
3700 San Martin Drive
Baltimore, Maryland 21218

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

The STSDAS Group

Rick White	Lead, Science Software Group
Perry Greenfield	STSDAS Programming Supervisor
Dick Shaw	STSDAS Project Scientist
Ray Williamson	System Administration and Distribution
Howard Bushouse	NICMOS Calibration and Analysis, Synthetic Photometry
Jonathan Eisenhamer	FOS and GHRS Calibration and Analysis, Graphics, WCS Support
Phil Hodge	Table System, Fourier Analysis, STIS and FOC Calibration and Analysis
J.C. Hsu	HSP, FGS, and WF/PC Calibration and Analysis
Bernie Simon	Calibration Data Base, Table Editor, IRAF System Support
Nelson Zarate	FITS, IRAF System Support, Ports, Benchmarking
Mark Stevens	Technical Writing
Fred Romelfanger	X-Windows-Based User Interface
Jinger Mo	Software Testing, Image Restoration

First edition, November 1990
Written by David Bazell

Second edition, September 1993
Written and edited by Howard Bushouse

Third edition, March 1995
Written and edited by Howard Bushouse

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

Table of Contents

List of Figures

List of Tables

Preface

This Guide describes the Space Telescope Science Data Analysis System (STSDAS) synthetic photometry (**synphot**) software package. STSDAS is developed and maintained by the Science Software 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. If you don't already have IRAF, you may obtain it by sending e-mail to `iraf@noao.edu` or by calling (602) 323-4160. The STSDAS package may be obtained from STScI by sending e-mail to the HotSeat support staff at `hotseat@stsci.edu` or by calling (410) 516-5100. **Synphot** requires the external TABLES package which is also available from STScI.

This manual assumes that you already have some familiarity with the IRAF command language (`cl`). If you need help getting started with IRAF or STSDAS, read the *STSDAS User's Guide*, which is available from the STSDAS group at STScI. If you have questions or problems using IRAF, STSDAS, or **synphot**, contact the STSDAS HotSeat person by calling (410) 516-5100 or be sending e-mail to `hotseat@stsci.edu`.

CHAPTER 1: About Synphot...

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 Guest 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. Chapter 2 discusses the basic features that are common to most **synphot** tasks and 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 3. Chapter 4 contains a more in-depth look at internal operations, and Chapter 5 provides a mathematical description of some basic synthetic photometry concepts and its role in calibrating the HST instruments. Many of the examples contained in Chapters 2 and 3 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 B contains detailed information on the origin and contents of these atlases, as well as installation information for off-site users.

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).

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 4 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,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.

The majority of **synphot** task data I/O is done via STSDAS binary table files. The HST instrument graph, component lookup, and component throughput tables are all in this format. All output files created by the **synphot** tasks are also in STSDAS table format. Input data files, such as passband throughput and spectral data files, may be in either STSDAS table format or plain ASCII text tables.

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”. This 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. 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. An alternative method is to transfer the individual tables from the `/cdbs/comp/` 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 `/cdbs/comp/` directory and its associated subdirectories on node ftp.stsci.edu. The instrument graph and component lookup tables are contained in the top directory and are named according to the convention `hstgraph_yymmddx.tab` and `hstcomp_yymmddx.tab`, where “yymmdd” specifies the year, month, and day that the tables were created, and "x" is a one-letter suffix code. The component throughput tables are logically grouped into subdirectories of `/cdbs/comp/` corresponding to each of the HST instruments (fgs, foc, fos, hrs, hsp, nonhst, ota, 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.

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 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 4 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` pset.

CHAPTER 2: Using Synphot

In This Chapter...

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 3.

Synphot Tasks

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

- Tasks that create, manipulate, and plot passbands and spectra (Table 2.1). This is the main group of tasks.
- Tasks that fit model passbands and spectra to spectrophotometric and photometric data (Table 2.2). 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 2.3).

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

Table 2.1: Primary Passband and Spectral Computation and Plotting 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 2.2: Passband and Spectral Fitting Tasks

Task	*Function*
imspec	Convert IRAF/STSDAS images to and from STSDAS 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 keyword and component table names for a path through a graph table
grafplot	Plot components downstream from a given component in an instrument graph table
obsmode	List the valid observation mode keywords for an instrument

Table 2.3: Utility Tasks

Command and Argument Syntax

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

- *Observation mode* (`obsmode`) - Passband to be calculated (page 11).
- *Spectrum* (`spectrum`) - Spectrum to be calculated (page 14).
- *Form* (`form`) - Units of the output data; for example, counts or magnitudes (page 17).
- *Wavelength table* (`wavetab`) - Wavelength grid on which passband and spectrum will be calculated (page 20).
- *Reference data parameters* (`refdata`) - Pointer to reference data for instrument graph and component lookup tables (page 22).

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.

Syntax Examples

In the command

```
sy> plband hrs,lsa,g270m
```

the string “hrs,lsa,g270m” is the observation mode, or `obsmode`, and specifies that a passband be calculated by taking the product of the individual throughputs of the High Resolution Spectrograph (HRS) large science aperture (“lsa”), the g270m grating, the detector sensitivity, the HST Optical Telescope Assembly (OTA), and the Corrective Optics Space Telescope Axial Replacement (COSTAR). The OTA throughput is included automatically whenever an HST instrument mode is specified and the COSTAR throughput is also included automatically whenever an `obsmode` involving the Faint Object Camera (FOC), Faint Object Spectrograph (FOS), or HRS is specified. 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 hrs,lsa,g270m hrs_g270.tab
```

will calculate that same passband and write the resulting data to the table `hrs_g270.tab`, 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_002 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_002 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 (6290 DNs per second!).

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. Appendix A contains 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 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 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 4, "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 indexes 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”.

The HST intrument graph allows the inclusion of the effects of COSTAR on the wavelength-dependent sensitivity for observations with the FOC, FOS, and HRS. This includes the product of the reflectivity curves for each pair of COSTAR mirrors for a given instrument, 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 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 the COSTAR effects, use the keyword `costar` anywhere within your `obsmode` string, e.g. “fos,red,4.3,g270h,costar”. To exclude it, use the `nocostar` keyword.

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, WF/PC-1 Baum, Cousins *RI,* Johnson *UBVRI*, Landolt *UBVRI*, Stromgren *uvby*, and Walraven *VBLUW* bands are included. Appendix A contains 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. These 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 *UBVRI* filters, the defaults are Johnson *UBV* and Cousins *RI* (i.e., the Landolt system).

The higher-level **synphot** tasks, such as **calcband**, **calcspec**, **calcphot**, **plband**, and **plspec** will also accept function expressions in the `obsmode` parameter. The available passband function expressions are shown in Table 2.4.

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.
`box(mu,width)`	Rectangular window centered on wavelength *mu* with width *width*; both arguments in Angstroms.
`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.

Table 2.4: Passband Functions

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`.tab.

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

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.

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 reading a spectrum from a file, you have a choice of using ASCII text files or STSDAS tables. When using 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 "photlam" (see page 17) 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 "photlam".

Table 2.5 lists the functional forms that can be used to generate synthetic spectra and to manipulate spectra read from a file.

SPECTRUM	*Function*
`filename`	Name of an ASCII text file or STSDAS table containing columns of wavelength and flux values.
`@filename`	Name of an ASCII file containing one string of spectrum commands per line.
`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)`	Power-law spectrum of the form $f_{\nu} \propto (\lambda / refval)^{expon} \propto \nu^{-expon}$, where *refval* is in Angstroms. The spectrum is normalized to a flux of 1 Jansky 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}), accounting for bound-free and free-free emission. Values of *colden* less than 80 are assumed to be $\log_{10}$(*colden*). For example, “hi(1.e4,23)” is equivalent to “hi(1.e4,1.e23)”. The spectrum is normalized the same as the blackbody.
`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.
`ebmv(val)`	Applies an extinction of *val* $E(B-V)$ to a spectrum using the galactic ISM reddening law of Seaton (1979).
`embvl(val)`	Applies an extinction of *val* $E(B-V)$ to a spectrum using the LMC extinction curve of Howarth (1983).
`+ -`	Add, subtract spectra with automatic unit conversion, if needed.
`* /`	Multiply, divide by a passband (`obsmode`), constant, or `ebmv`, `ebmvl` functions.

Table 2.5: Spectrum Functions

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 of the form $f_\nu \propto \nu^{-expon}$, which is equivalent to $f_\lambda \propto \lambda^{(expon-2)}$ in `flam` 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 the extinction, while for wavelengths between 3704 and 10000 Å, the extinction is computed by linear interpolation (in $1/\lambda$ 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 `ebmvl` operator applies or removes extinction effects using the Large Magellanic Cloud (LMC) extinction curve of Howarth (1983) which uses an adopted *R* value of 3.10. For wavelengths between 912 and 3636 Å, we use Howarth's equation 1; for 3636 to 5464 Å we use Howarth's equation 2; for wavelengths greater than 5464 Å we use Howarth's equation 4.

Note that dividing a spectrum by `ebmv(val)` or `ebmvl(val)` is equivalent to multiplying the spectrum by `ebmv(-val)` or `ebmvl(-val)` and has the effect of dereddening 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.tab`:

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

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.tab` in units of f_ν (`fnu`):

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

Form

The `form` parameter specifies the units of the output spectrum or plot. The **synphot** package supports the following 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 / hstarea / dλ
`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

Table 2.6: Forms

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

The `vegamag` option offers a good approximation to many of the conventional photometric systems that use the spectrum of Vega to define magnitude zero in any passband. 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 229 to 14150 Å.

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 2.1). 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 groundbased system.

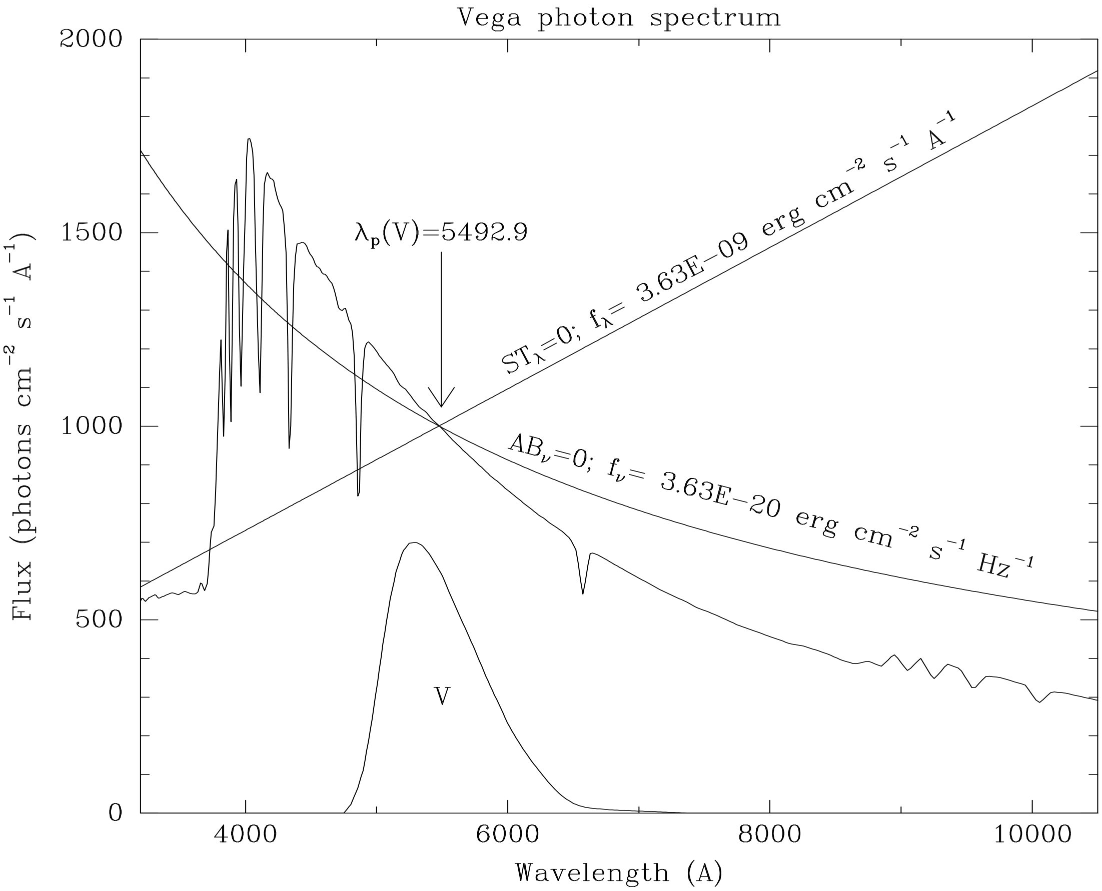

Figure 2.1: Standard photometric systems generally use the spectrum of Vega to define magnitude zero. The spectrophotometric magnitudes AB_ν and ST_λ refer instead to spectra of constant f_ν and f_λ respectively. Magnitude zero in both systems is defined to be the mean flux density of Vega in the Johnson *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.

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 FOS and HRS simulations.

Second, the form `counts` refers to actual detector counts for the FOC, FOS, HRS, and HSP instruments, while for the WF/PC-1 and WFPC2 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 simulations, and either of the keywords “a2d7” or “a2d15” for the WFPC2. These keywords invoke the appropriate electron-to-DN conversion based on the A-to-D gain settings of the instruments.

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 table, it should contain a column named “WAVELENGTH”. The wavelength values may be in any of the units listed in Table 2.7 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 both cases the wavelength values must be in a monotonically increasing or decreasing order.

FORM	*Units*	*FORM*	*Units*	*FORM*	*Units*
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				

Table 2.7: Wavelength Units.

There is a set of ASCII wavelength tables available for use with the HST spectroscopic instruments (HRS and FOS) in the STSDAS `synphot$data` directory area. This directory contains separate wavelength tables for each of the HRS and FOS grating modes and echelle orders. 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 1200 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
- $\Delta = (\log_{10}(\lambda_{max}) - \log_{10}(\lambda_{min})) / (N-1)$
- λ_{min} = wavelength of first non-zero throughput or flux
- λ_{max} = wavelength of last non-zero throughput or flux
- $N = 1200$

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.

Reference Data

Several common parameters are grouped in the `refdata` parameter set (pset). This pset contains the three parameters shown in Table 2.8

Parameter	*Default Value*	*Description*
`area`	45238.93416	Telescope area in cm^2
`grtbl`	"crcomp$hstgraph_*"	Instrument graph table
`cmptbl`	"crcomp$hstcomp_*"	Instrument component table

Table 2.8: Parameters in refdata Pset

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.

CHAPTER 3:
Task Descriptions

In This Chapter...

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

Bandpar

The **bandpar** task computes various photometric parameters, as shown in Table 3.1, for a selected passband. The computed photometric parameters can be saved to an STSDAS table.

Parameter	*Description*	*Formula*
`uresp`	Unit response; flux (in flam) that produces 1 count/second in the passband	$(hc)/\left(area\int P_\lambda \lambda d\lambda\right)$
`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	$uresp \cdot rectw \cdot \left(\frac{tpeak}{tlambda}\right)$
`tlambda`	Throughput at reference wavelength	$P(refwave)$
`refwave`	Reference wavelength for `tlambda`	$\int (P_\lambda \lambda d\lambda)/\int (P_\lambda d\lambda)$

Table 3.1: Bandpar Photometric Parameters

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 reference 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 3.1, below.

```
obsmode   =            Instrument observation mode
(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
```

Figure 3.1: Bandpar Parameters

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 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 "Observation Mode" on page 11 for more details about all of the available specifications for `obsmode`.

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 (`avgwav`) will be calculated as shown in Table 3.1 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 "Wavelength Table" on page 20)

and the default reference data for the HST observatory (see “Reference Data” on page 22).

Examples

The following command computes all available photometric parameters for the passband of the post-COSTAR FOC f/96 camera with filter F165W. The output is not saved in a table.

```
sy> bandpar foc,f/96,costar,f165w
```

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.tab`. The input file `ubv.lis` contains the lines:

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

```
sy> bandpar @ubv.lis output=ubv.tab 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
```

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 an STSDAS table with two columns of data, called WAVELENGTH and THROUGHPUT, as its ouput.

The task parameters are shown in Figure 3.2, below.

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

Figure 3.2: Calcband Parameters

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 (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 “Observation Mode” on page 11 for more details about all of the available specifications for `obsmode`.

If more than one passband is specified via a list file, then a separate “THROUGHPUTn” 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, many of which are the same photometric quantities computed by the **bandpar** task:

Keyword	*Description*	*Formula*
`grftable`	Name of the instrument graph table	
`cmptable`	Name of the component lookup table	
`aperarea`	Telescope collecting area, in cm^2, used to compute `uresp`	
`zeropt`	Photometric zeropoint of the STMAG system	
`expr`	Value of the `obsmode` parameter	
`uresp`	Unit response; flux (in flam) that produces 1 count/second in the passband	$(hc)/\left(aperarea\int P_\lambda \lambda d\lambda\right)$
`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	$uresp \cdot rectw \cdot \left(\frac{tpeak}{tlambda}\right)$

Table 3.2: Calcband Output Keywords

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 EXPRn, URESPn, PIVWVn, BANDWn, TPEAKn, EQUVWn, RECTWn, and EMFLXn 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 20) and the default reference data for the HST observatory (see "Reference Data" on page 22). 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.

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.tab`.

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

The next example shows how to evaluate the total telescope plus instrumental throughput for an HST observation using the FOS blue detector, the 4.3 arcsecond entrance aperture, and the G130H grating. The resulting throughput data are stored in the table `fos_thpt.tab`.

```
sy> calcband fos,blue,4.3,g130h,costar fos_thpt
```

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 an STSDAS table containing columns of wavelength, flux, and statistical error values. The wavelength data will be in units of Angstroms and the flux and error 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 statistical error values are set to INDEF. The columns in the output table are labeled WAVELENGTH, FLUX, and STATERROR.

The task parameters are shown in Figure 3.3 below.

```
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
```

Figure 3.3: Calcspec Parameters

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 “Spectrum” on page 14 for a detailed list and description of all the available `spectrum` commands. 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 EXPRn.

The `vzero` parameter is a list of values that are substituted for variable zero (`$0`) wherever it appears in the spectrum expression. Each value in the list is substituted in turn. The values must be real numbers. Using `vzero` is equivalent to placing the input spectrum 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. The following table gives several examples of valid `vzero` lists:

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

Table 3.3: vzero Examples

Null values for the `wavetab` and `refdata` parameters will cause the task to use a default wavelength grid (see “Wavelength Table” on page 20) and the default reference data for the HST observatory (see “Reference Data” on page 22). 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.

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.tab`.

```
sy> calcspec \
>>> "rn(bb(5000),band(wfpc2,f555w,a2d7),100,counts)" \
>>> rnbb.tab 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 `g191b2b.tab`, 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.tab 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.tab`.

```
sy> calcspec "g191b2b.tab*band(fos,blue,1.0,g190h)" \
>>> g191_fos form=counts wavetab=g190h.tab
```

The resulting count rate spectrum is shown in Figure 3.4. If we had wanted to use the wavelength table from the data directory, we would have specified “`wavetab=synphot$data/fos_blue_g190h.dat`” on the command line.

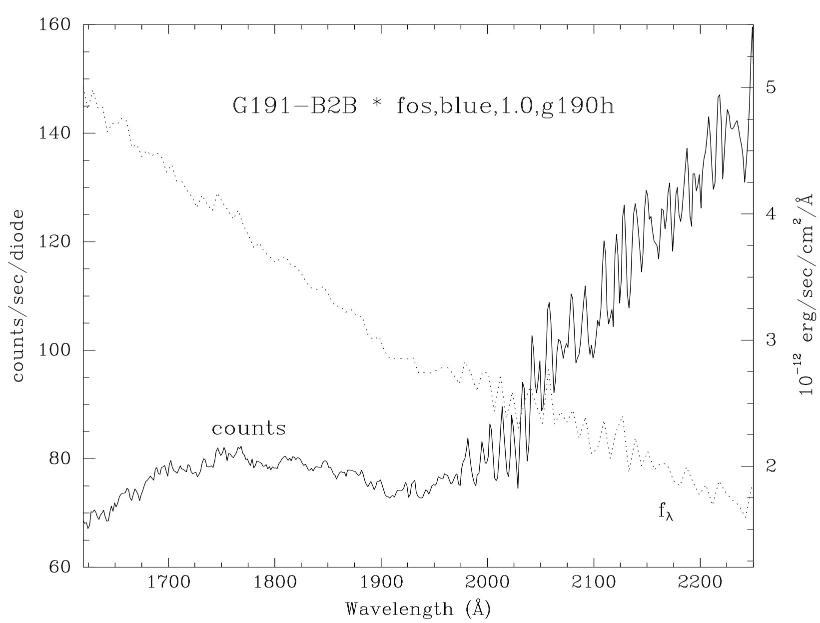

Figure 3.4: FOS Count Rate Spectrum of G191-B2B

For a final example, let's suppose a star observed with the WFPC2 and filter F439W produces 1200 counts (DNs) per second. What would it's 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.tab form=flam vzero="5e3-20e3x1e3"
```

The resulting spectra, in units of `flam`, are stored in the output table `bbody.tab`. Some of the spectra are shown in Figure 3.5.

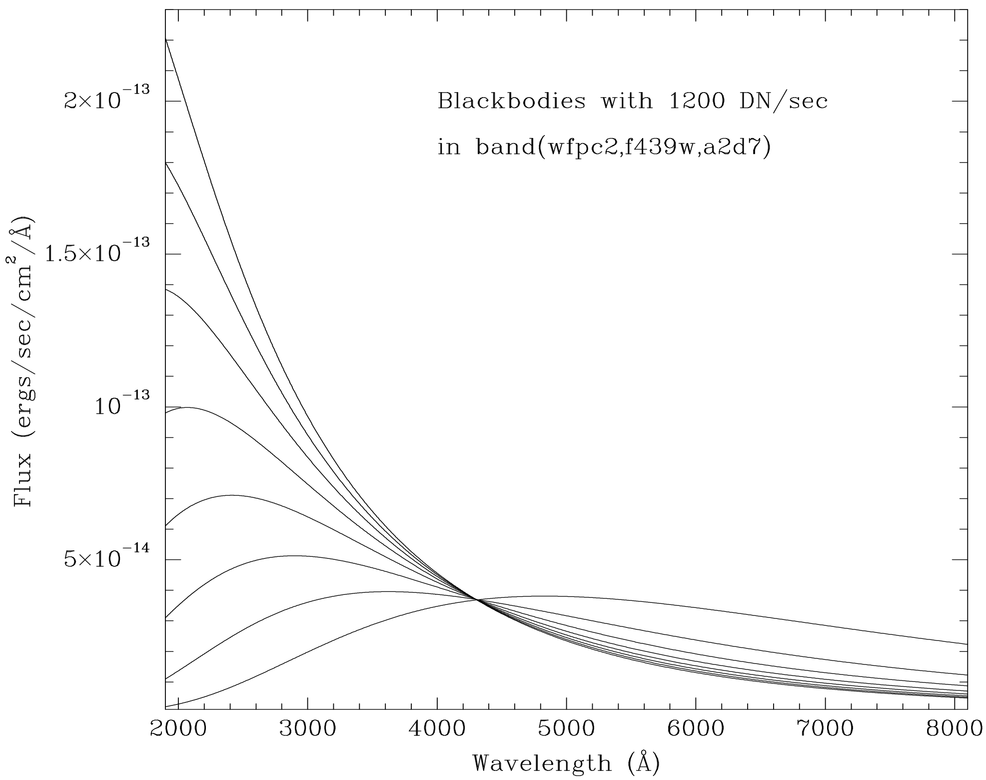

Figure 3.5: WFPC2 Blackbody Spectra

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 an 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 "Observation Mode" on page 11 and "Spectrum" on page 14. The full list of task parameters is shown in Figure 3.6.

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

Figure 3.6: Calcphot Parameters

The `form` parameter may be any of the accepted types (`fnu`, `flam`, `photnu`, `photlam`, `abmag`, `stmag`, `counts`, `obmag`, `vegamag`, `jy`, or `mjy`) in addition to several forms that are related to passband characteristics. The additional `form` types are listed in Table 3.4. See Chapter 5 for an explanation of these passband forms. By default, the task always calculates and reports the values of `pivlam` and `fwhmlam`, in addition to the quantity chosen by `form`.

Form	***Description***	***Formula***
`avglam`	Average wavelength	$\int (P_\lambda \lambda d\lambda) / \int (P_\lambda d\lambda)$
`barlam`	Mean log ("bar") wavelength	$\bar{\lambda} = \exp\left[\frac{\int P_\lambda \ln(\lambda)\, d\lambda/\lambda}{\int P_\lambda d\lambda/\lambda}\right]$
`efflam`	Effective wavelength	$\frac{\int \lambda f_\lambda(\lambda) P(\lambda) \lambda\, d\lambda}{\int f_\lambda(\lambda) P(\lambda) \lambda\, d\lambda}$
`fwhmlam`	Full width at half maximum of the equivalent Gaussian passband	$sqrt(8 \cdot \ln(2)) \cdot$ `rmslam`
`pivlam`	Pivot wavelength	$\sqrt{\int (P_\lambda \lambda d\lambda) / \int (P_\lambda d\lambda/\lambda)}$
`rmslam`	RMS bandwidth	$\bar{\lambda}\frac{\sqrt{\int P_\lambda \ln(\lambda/\bar{\lambda})^2 d\lambda/\lambda}}{\sqrt{\int P_\lambda d\lambda/\lambda}}$
`qtlam`	Dimensionless efficiency	$\int P(\lambda)\, d\lambda/\lambda$

Table 3.4: Additional Form Types for Calcphot

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). Note that all of the above parameters are souce-independent, except for `efflam`.

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.

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 an STSDAS table, if desired, via the `output` parameter. This table contains the columns of information described in Table 3.5.

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

Table 3.5: Calcphot Output Table Columns

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 3.1, 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 “Observation Mode” on page 11) and the default reference data for the HST observatory (see “Reference Data” on page 22). 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.

Examples

The first example (Figure 3.7) 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.

```
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        1200.943    band(wfpc,f555w,dn)
Spectrum:  rn(bb(5000),band(v),18.6,vegamag)
    VZERO      (COUNTS s^-1 hstarea^-1)
       0.           55.80261
```

Figure 3.7: Sample Calcphot Run

The next example (Figure 3.8) 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. The results are written to the table `bbcolor.tab`.

```
sy> calcphot "band(wfpc2,f439w)-band(wfpc2,f555w)" \
>>> "bb(5000)*ebmv($0)" obmag vzero="0,0.25,0.5"
Mode = band(wfpc2,f439w) - band(wfpc2,f555w)
   Pivot       Equiv Gaussian
 Wavelength         FWHM
  4313.585        477.4963    band(wfpc2,f439w)
  5449.512        1239.025    band(wfpc2,f555w)
Spectrum:  bb(5000)*ebmv($0)
  VZERO    OBMAG(band(wfpc2,f439w))-OBMAG(band(wfpc2,f555w))
     0.        2.52663
   0.25        2.77123
    0.5       3.022645
```

Figure 3.8: Second Sample Calcphot Run

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*)!

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 are structured to mimic what is contained in the exposure logsheets found in HST observing proposals. 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. It is therefore ideally suited to predicting exposure times when writing HST proposals.

Figure 3.9 is a list of the task parameters.

```
instrument =                Science instrument
(detector  = "")            Detector used
(spec_elem = " ")           Spectral elements used
(aperture  = " ")           Aperture or field of view
(cenwave   = INDEF)         Central wavelength (HRS only)
(userspec  = " ")           User supplied input spectrum
(synspec   = " ")           Synthetic spectrum
(synmag    = " ")           Magnitude of synthetic spectrum
(reddening = 0.)            Interstellar reddening E(B-V)
(exptime   = 1.)            Exposure time in seconds
(output    = "none")        Output table name
(form      = "counts")      Form for output data
(refwave   = INDEF)         Reference wavelength
(verbose   = yes)           Print results to STDOUT?
(flux_tot  = INDEF)         Estimated total counts
(flux_ref  = INDEF)         Estimated counts at refwave
(refdata   = "")            Reference data
```

Figure 3.9: Countrate Task Parameters

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: `fgs`, `foc`, `fos`, `hrs`, `hsp`, `wfpc`, or `wfpc2`. If you set the instrument parameter to `hrs`, 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, such as `costar`, 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 COSTAR in an HRS simulation using the small science aperture (`ssa`), you could specify "`aperture=ssa,costar`".

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 instrument modes, because it is the only instrument where the detector cannot cover the entire wavelength range of an observation. This parameter is ignored for all non-HRS instrument modes.

The target spectrum may be specified in one of two ways:

- As any valid **synphot** spectrum expression given in the `userspec` parameter
- As any valid **synphot** spectrum expression given in the `synspec` parameter, to be renormalized to the magnitude given in the `synmag` parameter

The synthetic spectrum (`synspec`) will only be used if the `userspec` parameter is blank. 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 B), or you can use a synthetic spectrum function to generate a spectrum at run-time. The `userspec` and `synspec` parameters recognize all of the valid functional forms that can be specified via the `spectrum` parameter in all other **synphot** tasks.

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

Note that the functionalities of the `userspec` and `synspec` parameters are a bit redundant, in that either of them may be used to specify any valid **synphot** spectrum expression. The only difference is that the `synspec/synmag` pair of parameters offers a somewhat simpler syntax (i.e., less typing) for specifying a desired renormalization to a standard magnitude. The same result can be accomplished by using the renormalization (`rn`) function within the spectrum expression given in the `userspec` parameter. For example, the two expressions:

```
userspec = rn(bb(8000),band(v),15.6,stmag)
```

and

```
synspec = bb(8000)
synmag = 15.6 v
```

will give identical results. The only advantage of using the `synspec` and `synmag` parameters is that you do less typing, assuming you want to renormalize the spectrum over one of the *UBVRI* passbands, in units of `stmag`. The advantage in using the `userspec` parameter is that it offers complete flexibility to renormalize (if desired) over any passband, in any units.

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 `ebmv` function in the spectrum interpreter of other **synphot** tasks. Either the user spectrum (`userspec`) or the synthetic spectrum (`synspec`) may be modified by this parameter.

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 an 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 or HRS), 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`.

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_003.tab` in the directory `crcalspec$`. The calculated spectrum, in units of counts per pixel, is written to the table `hrsobs.tab`. We use “epar countrate” to set the task parameters as follows:

```
instrument = hrs
(detector  = )
(spec_elem = g160m)
(aperture  = lsa)
(cenwave   = 1495.)
(userspec  = crcalspec$feige110_003)
(synspec   = )
(synmag    = )
(reddening = 0.)
(exptime   = 1800.
(output    = hrsobs)
(form      = counts)
(refwave   = 1504.)
(verbose   = yes)
(flux_tot  = INDEF)
(flux_ref  = INDEF)
(refdata   = )
```

Figure 3.10: Example Countrate Parameter Settings

The output on your screen will be the following:

```
Mode = band(hrs,g160m,lsa)
Spectrum:  crcalspec$feige110_003*1800.
   Pivot        Equiv Gaussian  Total Flux  Flux at 1504 A
 Wavelength         FWHM         COUNTS         COUNTS
  1494.484        23.83772      26276.89       44.24325
```

These results indicate an integrated flux of about 26000 counts over the whole spectrum, with approximately 44 total counts in the pixel at 1504 Å.

Figure 3.11 shows the resulting spectrum as plotted using the STSDAS task **stplot.sgraph** (e.g., `sgraph "hrsobs wavelength flux"`).

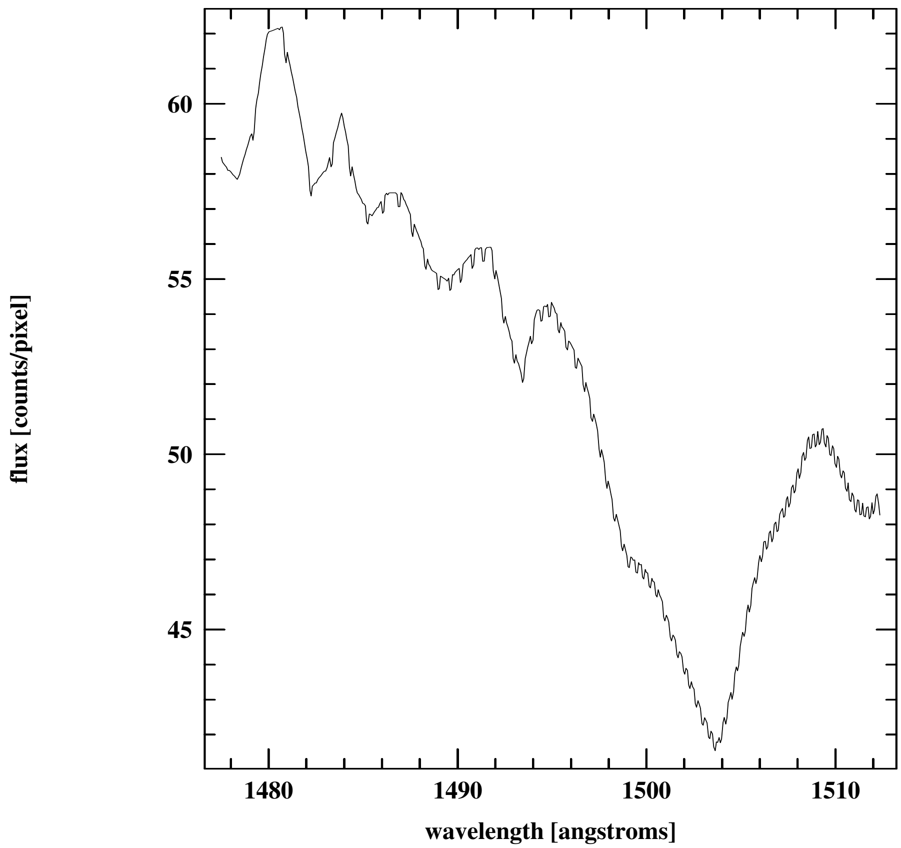

Figure 3.11: Plot of Countrate Example Spectrum

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:

```
instrument = wfpc2
(detector  = 1,a2d7)
(spec_elem = f785lp)
(aperture  = )
(cenwave   = INDEF)
(userspec  = )
(synspec   = crcalspec$hz2_003)
(synmag    = 17.5 v)
(reddening = 0.05)
(exptime   = 600.)
(output    = )
(form      = counts)
(refwave   = INDEF)
(verbose   = yes)
(flux_tot  = 8082.354)
(flux_ref  = INDEF)
(refdata   = )
```

Figure 3.12: Second Countrate Example Parameter Settings

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 refered 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 Recipies* by Press et al. (1992).

The task has a relatively long list of parameters (Figure 3.13).

```
input    =            Observed passband
obsmode  =            Model passband expression
(output  = "none")    Ouput passband table
(ftol    = 1.0E-5)    Convergence condition
(maxiter = 500)       Maximum number of iterations
(nprint  = 0)         Number of iterations between prints
(slow    = no)        Use slow method (simplex)?
(equal   = no)        Use equal weighting of 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
```

Figure 3.13: Fitband Parameters

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 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`*. 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 acheived 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 succesive 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.

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.tab`.

```
sy> fitband crwfpc2comp$wfpc2_f555w_001 "gauss($1,$2)*$3" \
>>> out=fit555w nprint=1 vone=5500 vtwo=500 vthree=1 equal+
```

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

```
irep=1 chisq=0.07018 exp=gauss(5500.,500.)*1.01
irep=2 chisq=0.03899 exp=gauss(5446.85,1049.77)*0.657212
irep=3 chisq=0.01281 exp=gauss(5203.74,1899.69)*0.856438
irep=4 chisq=0.00816 exp=gauss(5317.00,1231.66)*1.021223
irep=5 chisq=0.00579 exp=gauss(5250.33,1477.76)*0.999724
irep=6 chisq=0.00556 exp=gauss(5265.96,1366.36)*1.053696
irep=7 chisq=0.00552 exp=gauss(5256.39,1402.27)*1.04326
irep=8 chisq=0.00552 exp=gauss(5259.12,1389.02)*1.04894
irep=9 chisq=0.00552 exp=gauss(5258.05,1393.43)*1.04723
irep=14 chisq=0.00552 exp=gauss(5258.11,1393.24)*1.04726
irep=15 chisq=0.00551 exp=gauss(5258.13,1393.16)*1.03691

Final solution:
gauss(5258.11,1393.24)*1.03689
```

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

```
sy> plband fit555w left=2500 right=8000
sy> plband crwfpc2comp$wfpc2_f555w_001 append+ ltype=dashed
```

The resulting plot is shown in Figure 3.14.

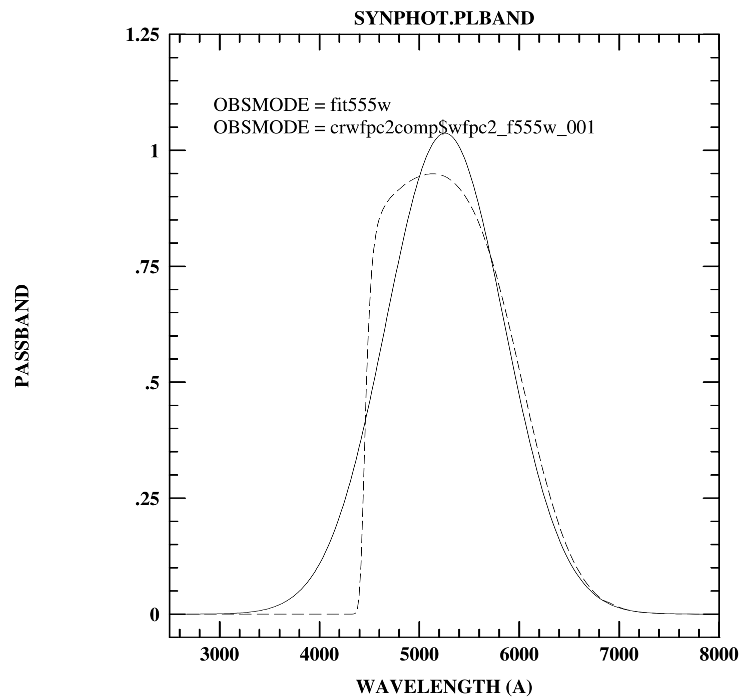

Figure 3.14: Results from Example Fitband Run

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 refered 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 3.15) is nearly identical to that of the **fitband** task.

```
input    =            Observed spectrum
spectrum =            Model spectrum expression
(output  = "none")    Output spectrum table
(ftol    = 1.0E-5)    Convergence condition
(maxiter = 500)       Maximum number of iterations
(nprint  = 0)         Number of iterations between prints
(slow    = no)        Use slow method (simplex)?
(equal   = no)        Use equal weighting of 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
```

Figure 3.15: Fitspec Parameters

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 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`*. 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 acheived 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 succesive 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.

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*embv(0.073)" Astar_ebv073 \
>>> form=flam
```

This uses the observed spectrum of an A-type star from the Jacoby-Hunter-Christian spectral library (see Appendix B), 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` 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 renormaliztion 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:

```
input    = Astar_ebv073
spectrum = rn(alpha_lyr_001,band(v),$1,vegamag)*ebmv($2)
(output  = Astar_fit)
(ftol    = 1.0E-5)
(maxiter = 500)
(nprint  = 1)
(slow    = no)
(equal   = yes)
(vone    = 9.)
(vtwo    = 0.)
(vthree  = INDEF)
 ...
(vnine   = INDEF)
(refdata = )
```

Figure 3.16: Example Fitspec Parameter Settings

Here is what the results should look like:

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

This plot is shown in Figure 3.17.

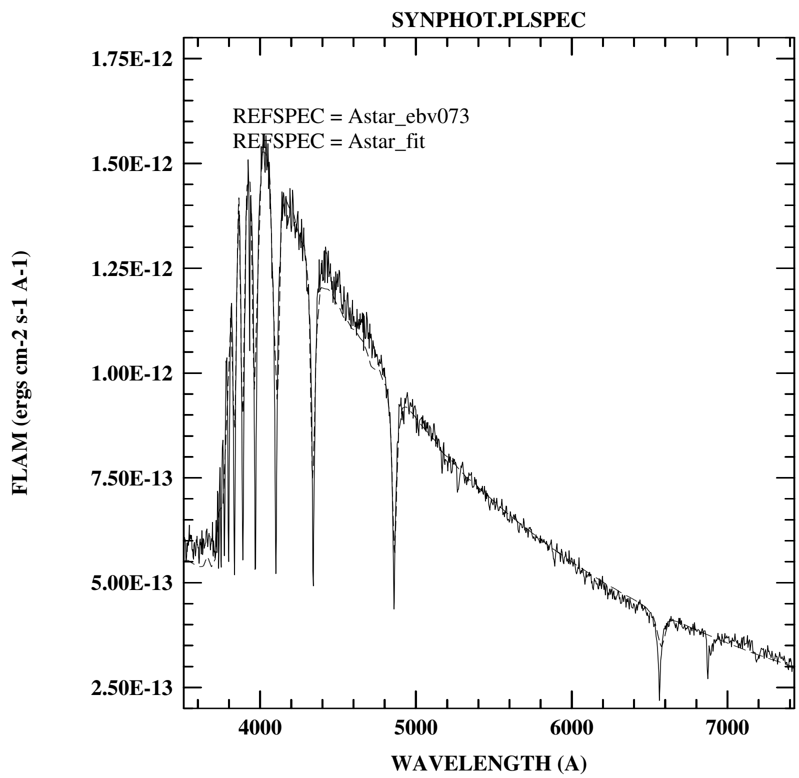

Figure 3.17: Results from Fitspec

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 3.18) is similar to those of the **fitband** and **fitspec** tasks.

```
input    =            Observed spectrum
spectrum =            List of spectra
(output  = "none")    Output spectrum table
(vzero   = " ")       List of values for variable zero
(ftol    = 1.0E-5)    Convergence condition
(maxiter = 500)       Maximum number of iterations
(nprint  = 0)         Number of iterations between prints
(slow    = no)        Use slow method (simplex)?
(equal   = no)        Use equal weighting of data points?
(refdata = "")        Reference data
```

Figure 3.18: Fitgrid Parameters

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 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`*.

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. Each value in the list is substituted in turn. The values must be real numbers.

Using `vzero` is equivalent to placing the spectrum 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. See Table 3.3 for examples of valid `vzero` strings.

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 acheived 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 succesive 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.

Examples

Let’s fit a series of spectra from the Bruzual-Persson-Gunn-Stryker spectral library (see Appendix B) to the spectrum of Eta Ursa Majoris, 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$bpgs_4
crgridbpgs$bpgs_9
crgridbpgs$bpgs_13
crgridbpgs$bpgs_20
```

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

```
sy> fitgrid crcalspec$eta_uma_002 @grid.lis out=etauma_fit
```

The final solution is:

```
       0.60996  * 0.13247 * (crgridbpgs$bpgs_9) +
 (1. - 0.60996) * 0.49853 * (crgridbpgs$bpgs_13)
```

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 Eta Ursa Majoris:

```
sy> fitgrid crcalspec$eta_uma_002 "bb($0)" out=bb_fit \
>>> 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`. The final solution is:

```
       0.733667  * 2665.36 * (bb(23000.)) +
 (1. - 0.733667) * 2354.4  * (bb(24000.))
```

We can then compare the model with the Eta Uma spectrum using **plspec** as follows (see Figure 3.19):

```
sy> plspec "" crcalspec$eta_uma_002 flam
sy> plspec "" bb_fit flam append+
```

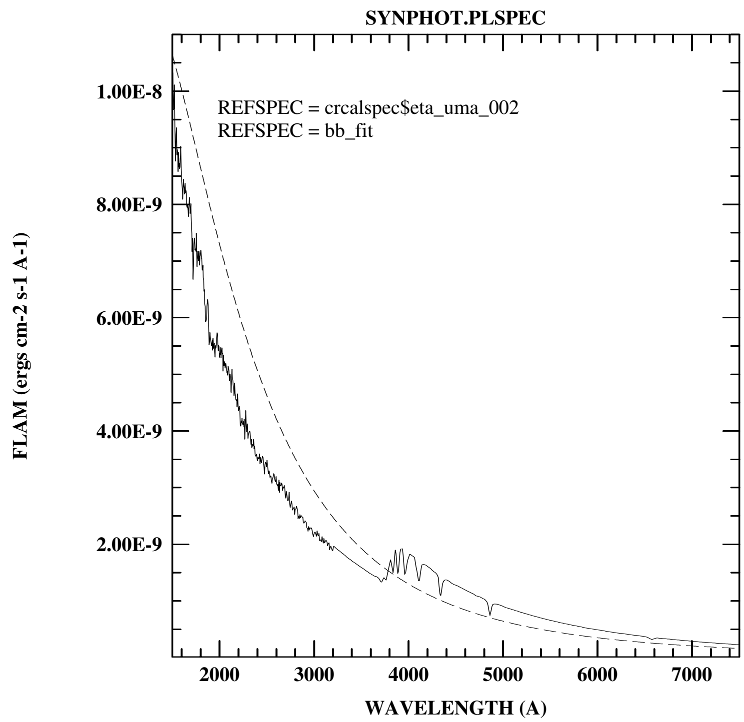

Figure 3.19: Fitgrid Results

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 an STSDAS table. The task parameters are listed in Figure 3.20.

```
output     =                  Wavelength set table name
minwave    =                  Minimum wavelength (Ang.)
maxwave    =                  Maximum wavelength (Ang.)
dwave      = INDEF            Wavelength interval (Ang.)
(dvelocity = INDEF)           Velocity interval (km/s)
(wavecol   = "WAVELENGTH")    Wavelength table column name
```

Figure 3.20: Genwave Parameters

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.

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.tab` in a column labeled WAVELENGTH.

```
sy> genwave wave2 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 3000 8000 INDEF dv=100
```

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 "`crcomp$hstgraph_*`" by default, and is used to specify the name of the graph table(s) to be checked. If more than one graph table matches the file name template, the contents of all the tables will be merged.

Examples

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

```
sy> grafcheck hstgraph
```

Check the combined graph composed of all graph tables listed in the file `graph.lis`:

```
sy> grafcheck @graph.lis
```

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="crcomp$hstgraph_*" Instrument graph table
compname="crcomp$hstcomp_*" Component name of starting node
```

The `grftable` parameter specifies the name of the instrument graph table to be searched. If more than one table matches the file name template, all tables will be merged.

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.

Examples

List all components that are downstream from “hrs_echa”:

```
sy> graflist crcomp$hstgraph_* hrs_echa
```

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

```
sy> graflist crcomp$hstgraph_* "clear 1000"
```

Grafpath

The **grafpath** task produces a list of throughput table names for the components that lie within the optical path of a specified observation mode. The other tasks in the **synphot** package compute the combined throughput for an observation mode by multiplying together the individual throughput values contained in these tables. This task will show you the specific tables that the other **synphot** tasks use to compute the combined throughput function for a given observation mode.

The two parameters for this task are:

```
obsmode  = ""                 Observing mode
(refdata = "")                Reference data
```

The `obsmode` parameter is a list of keywords, separated by blanks or commas, that uniquely specifies an optical path through the telescope. The keywords that make up the observing mode specify the instrument, detector, aperture, filters, and gratings used. These keywords may occur in any order, and are *not* case sensitive. Defaults are available for several standard components, such as the OTA and COSTAR, so that if some of these are omitted, the defaults will be used (see “Observation Mode” on page 11 for more details about the `obsmode` parameter).

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 searched by this 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 STSDAS 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.

Examples

Write a list of component table names for the observing mode “fos,costar, red,1.0,g270h” to the terminal screen as shown in Figure 3.21.

```
sy> grafpath fos,costar,red,1.0,g270h

crotacomp$hst_ota_004.tab
crfoscomp$fos_red_m1m2_002.tab
crfoscomp$fos_cir1p0_co_001.tab
crfoscomp$fos_rflgrza_001.tab
crfoscomp$fos_uvblk_002.tab
crfoscomp$fos_g270h_001.tab
crfoscomp$fos_g270h_r_co_001.tab
crfoscomp$fos_rflcola_002.tab
crfoscomp$fos_dqer_003.tab
```

Figure 3.21: Using Grafpath

This list includes tables containing throughput (or sensitivity) data for the HST Optical Telescope Assembly (OTA), the COSTAR FOS red-side m1 and m2 mirrors, the circular 1.0" entrance aperture, the deflection mirror, the UV blocking filter used with the g270h grating, the g270h grating, the red-side collimator, and the detective quantum efficiency (DQE) of the red Digicon.

Grafplot

This task adds plotting capabilities to the **graflist** task. It will create a block diagram showing all the instrument component names in a graph table that are downstream from a specified *root* component. As with **graflist**, a component is considered to be downstream from the root if a path passes through both it and the root component, and the root component appears first in the path. The root component is considered to be downstream from itself, so at least one component is always plotted.

Because any given graphics device has a limited resolution, there is an upper limit to the number of components that can be usefully plotted. This limit varies according to the size of the portion of the instrument graph being plotted. If the device limit is exceeded, the task will terminate with an error message and nothing will be plotted. **graflist** can be used to view that portion of the instrument graph and select a new root component further downstream.

The task parameters are shown in Figure 3.22.

```
grftable = "crcomp$hstgraph_*"   Instrument graph table
compname = ""                    Starting component name
(title   = "")                   Plot title
(device  = "stdgraph")           Graphics device
```

Figure 3.22: Grafplot Parameters

The `grftable` parameter specifies the name of the instrument graph table to be searched for the list of component names to be plotted. If more than one table matches the file name template, the contents of all the tables will be merged.

The `compname` parameter specifies the name of the desired root component at which to start plotting. The component name is *not* case sensitive and leading and trailing blanks do not affect the match. If more than one component matches `compname`, the component having the smallest value in the INNODE column will be used. In such a case, the match can be made unique by specifying an INNODE number as part of the `compname` string. To do this, the INNODE number must follow the component name, separated by a white space. If `compname` is null, the entire graph will be plotted.

The `title` and `device` parameters may be used to specify a plot title and the name of the desired graphics output device, respectively.

Example

Plot all components downstream from the FOC f320w filter (which is on the second of four f/96 filter wheels). Figure 3.23 shows the results.

```
sy> grafplot crcomp$hstgraph_* foc_96_f320w
```

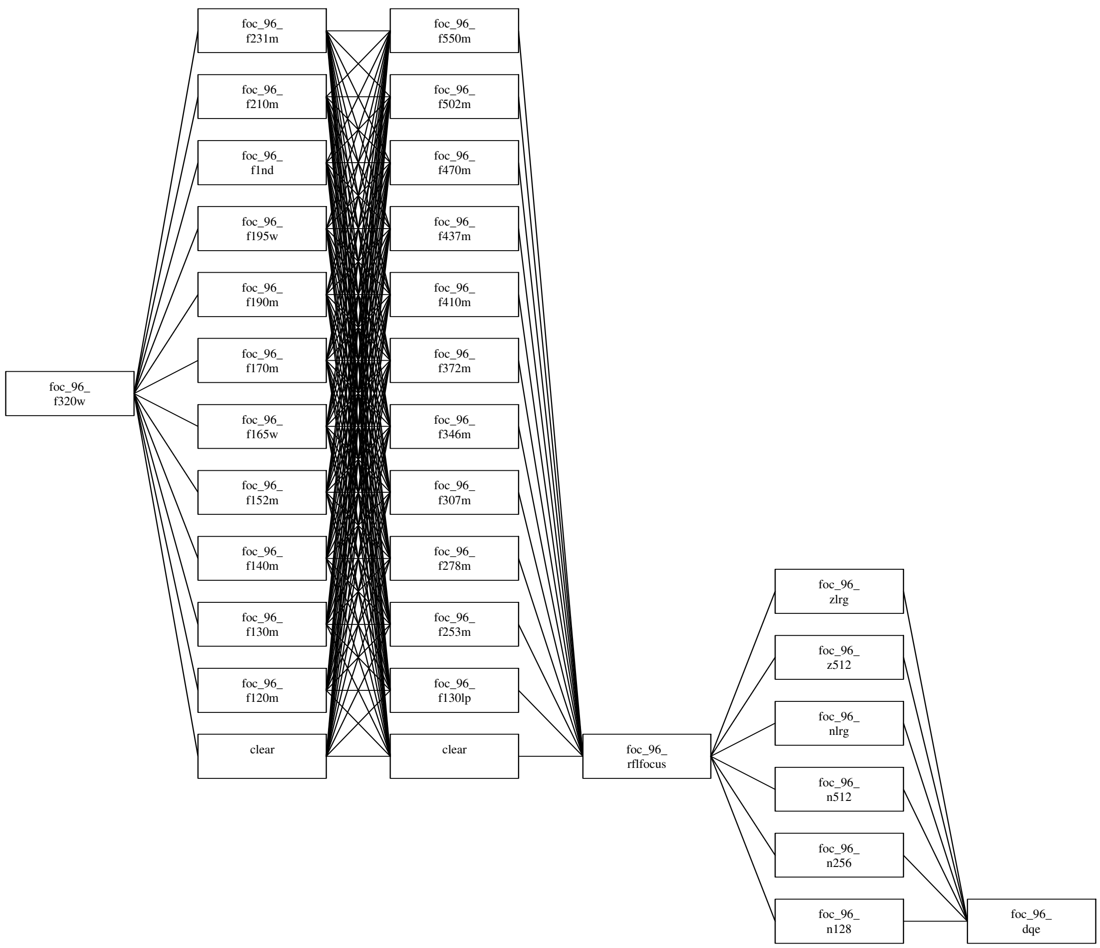

Figure 3.23: Output From Sample Grafplot Run

Imspec

The **imspec** task can be used to convert one-dimensional images into STSDAS 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 3.24.

```
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.)              Output value for INDEF pixels
(refdata = "")              Reference data
```

Figure 3.24: Imspec Parameters

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 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 table, it 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 an image, then the ouput file will be an 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 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.

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

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.

Examples

Copy the **synphot**-format table for the standard star Eta Uma, contained in table `eta_uma_002.tab` in the `crcalspec$` directory, to the image `eta_uma.hhh`. 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$eta_uma_002 eta_uma.hhh outform=flam
```

Copy an FOS spectral image into a table with the same rootname as the input image, but an extension of `.tab`. 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.tab wave=y15v0403t.c0h \
>>> inform=flam outform=flam
```

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*, Cousins is the default for *RI*, and `nocostar` is the default value of the COSTAR elements.

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, “foc” specifies that all the keywords that are descendants of the foc 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.

Examples

Display the observation mode keywords for the HSP:

```
sy> obsmode hsp
```

Display the list of echelle orders for the HRS echelle A:

```
sy> obsmode hrs,ssa,echa
```

Display all the keywords in the graph table:

```
sy> obsmode ""
```

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 3.25.

```
obsmode    =                    Observation mode or @list
(left      = INDEF)             Left plot limit
(right     = INDEF)             Right plot limit
(bottom    = INDEF)             Bottom plot limit
(top       = INDEF)             Top plot limit
(normalize = no)                Normalize all curves to 1?
(ylog      = no)                Plot log of y values?
(append    = no)                Append to existing plot?
(ltype     = "solid")           Line type
(device    = "stdgraph")        Graphics device
(wavetab   = "")                Wavelength table name
(refdata   = "")                Reference data
```

Figure 3.25: Plband Parameters

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 "Observation Mode" on page 11 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*.

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, and dotdash. 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 20) and the default reference data for the HST observatory (see "Reference Data" on page 22). 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. 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.

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 3.26.

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

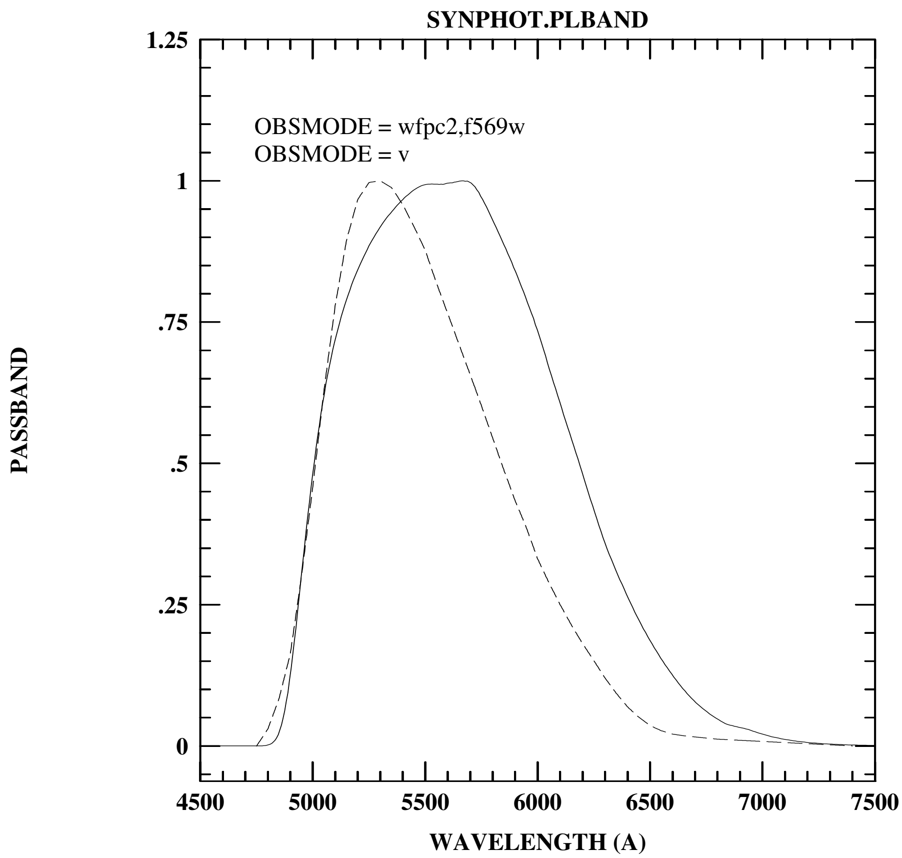

Figure 3.26: Results of First Sample Plband Run

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.tab`. Results are shown in Figure 3.27.

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

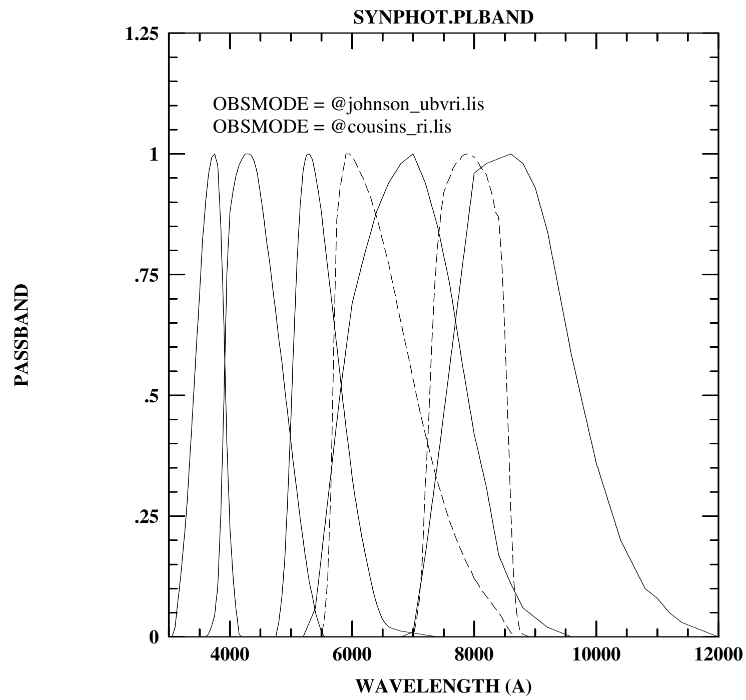

Figure 3.27: Results of Second Sample Plband Run

Plspec

The **plspec** task combines the capabilities of **calcband**, **calcphot**, and **calcspec** into one task that will produce graphical output. It can be used to plot synthetic and observed spectra, photometric data, and instrument passbands. This allows for comparisons of spectrophotometric and photometric data. **Plspec** automatically calculates the pivot wavelength and the FWHM of the specified observation mode so that you could use **plspec** to model both spectrophotometric and photometric observations.

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 3.28.

```
obsmode  =                     Observation mode or @list
spectrum =                     Synthetic spectrum or @list
form     =                     Form for output graph
(vzero   = " ")                Variable zero values
(spfile  = "none")             Spectrophotometry data
(pfile   = "none")             Photometry data
(errtyp  = "n")                Type of error bars to plot
(left    = INDEF)              Left plot limit
(right   = INDEF)              Right plot limit
(bottom  = INDEF)              Bottom plot limit
(top     = INDEF)              Top plot limit
(append  = no)                 Append to existing plot?
(ltype   = "solid")            Line type
(device  = "stdgraph")         Graphics device
(wavetab = "")                 Wavelength table name
(refdata = "")                 Reference data
```

Figure 3.28: Plspec Parameters

The `obsmode` parameter is used to specify the desired passband to be applied to the spectral data (see "Observation Mode" on page 11). Several `obsmode` strings may be processed by inserting them one per line in a text file and setting `obsmode=@`*`filename`*. 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 "Spectrum" on page 14). 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 "Form" on page 17). 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. Each value in the list is substituted in turn. The values must be real numbers. Using `vzero` is equivalent to placing the `spectrum` 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. See Table 3.3 for examples of valid `vzero` strings.

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 3.6.

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

Table 3.6: Error Type Codes for Plspec

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 autoscaled 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 hardcopies.

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.

Examples

The first example simply plots the spectrum, in units of `flam`, of the flux calibration standard star HD 60753 from 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 3.29.

```
sy> plspec "" crcalspec$hd60753_001 flam
```

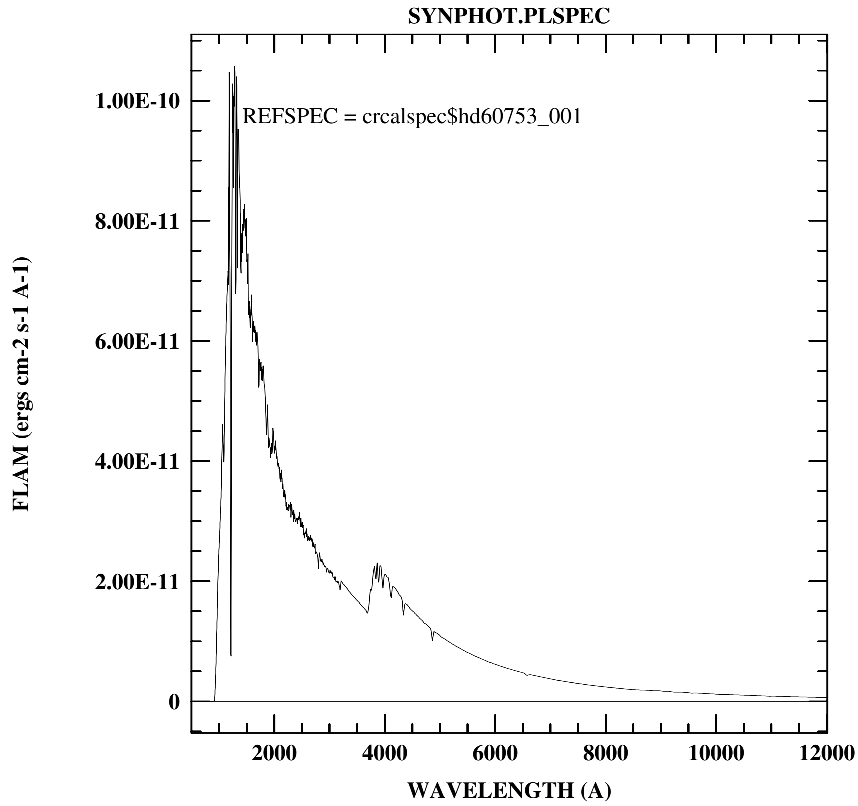

Figure 3.29: Plspec Results

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 3.30 shows the results.

```
sy> plspec "" "" flam spfile=crcalspec$hd60753_001 \
>>> err=pvh left=2000 right=3000
```

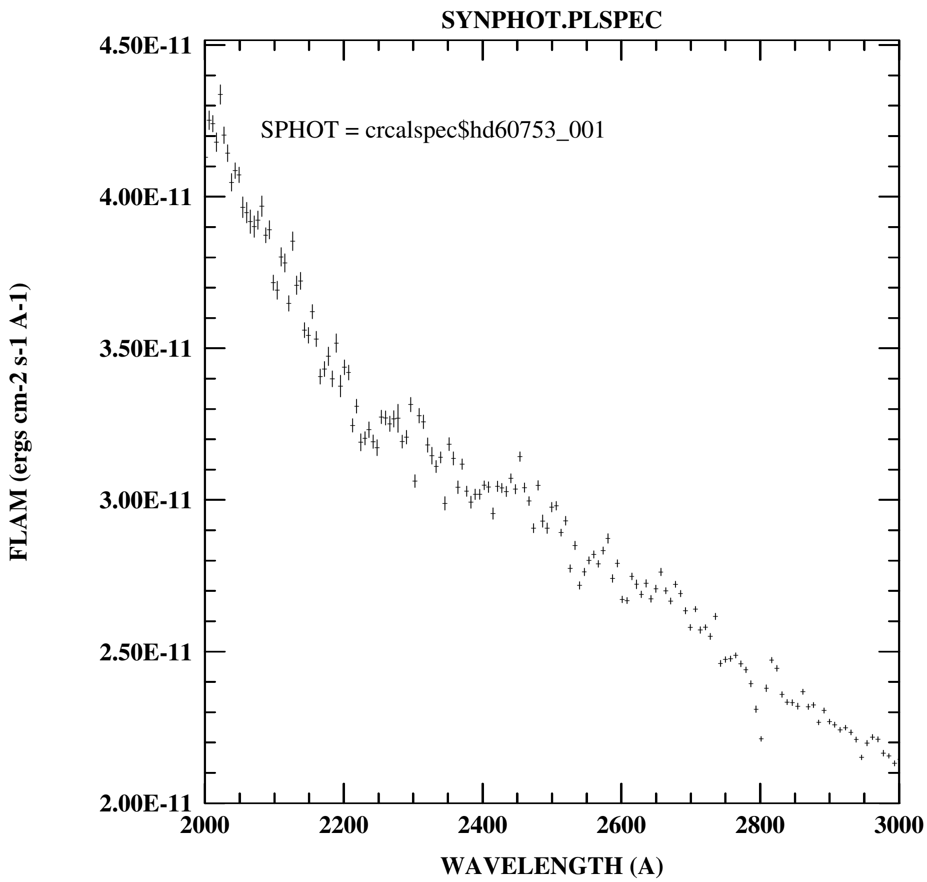

Figure 3.30: Results from Second Sample Plspec Run

The next example performs a comparison of the FOS red-side and blue-side sensitivities by overplotting the countrate spectra for a B9 star as observed with the FOS g270h grating and 1.0 arcsecond aperture (post-COSTAR) with the red and blue detectors. 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 flux of 2.5e-15 `flam` at 3000 Å. We'll also make use of the wavelength tables for the g270h grating that are in the `synphot$data` directory, so that we get a realistic count rate per channel. Here are the (rather lengthy) commands to perform this:

```
sy> plspec fos,red,g270h,1.0,costar \
>>> "rn(crgridbpgs$bpgs_13,box(3000,1),2.5e-15,flam)" \
>>> counts wave=synphot$data/fos_red_g270h.dat

sy> plspec fos,blue,g270h,1.0,costar \
>>> "rn(crgridbpgs$bpgs_13,box(3000,1),2.5e-15,flam)" \
>>> counts wave=synphot$data/fos_blue_g270h.dat \
>>> app+ ltype=dashed
```

The results are shown in Figure 3.31.

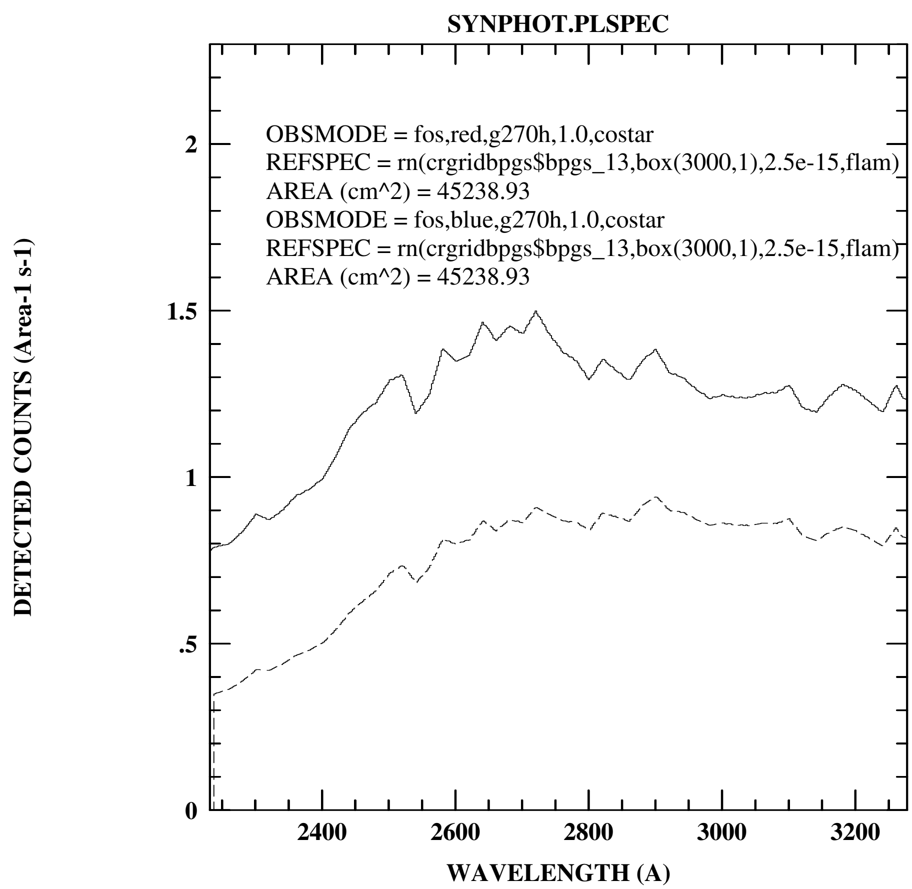

Figure 3.31: Plspec Comparison of FOS Red and Blue Side Sensitivities

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.tab`. 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,band(v),17.4,vegamag)" \
>>> counts out=wf_bpgs23 wave=wf_wave
```

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

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

The resulting plot, shown in Figure 3.32, 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.

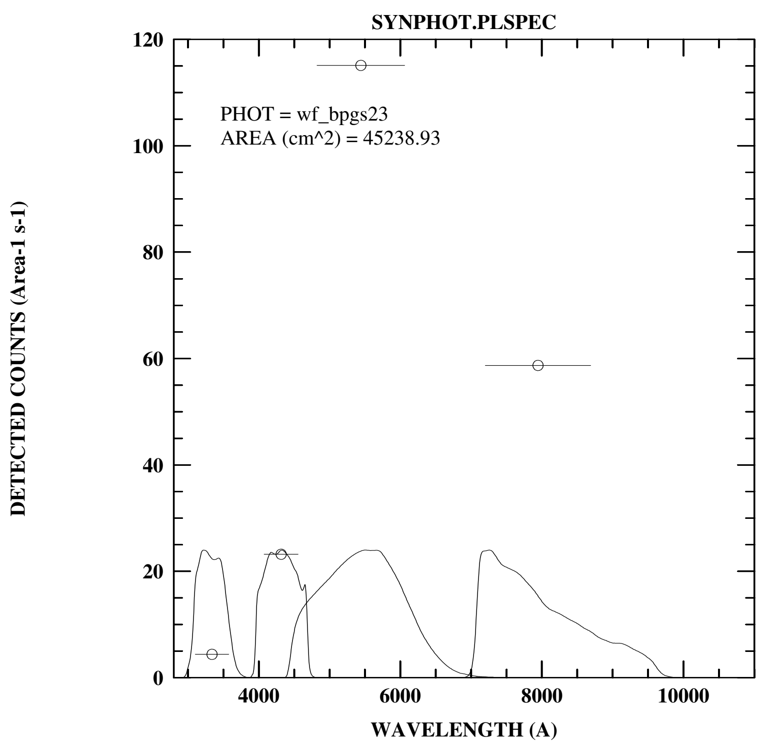

Figure 3.32: Plotting calcphot Results

Attempting to combine these two operations (**calcphot** + **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,band(v),17.4,vegamag)" counts \
>>> wave=wf_wave
```

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 3.33).

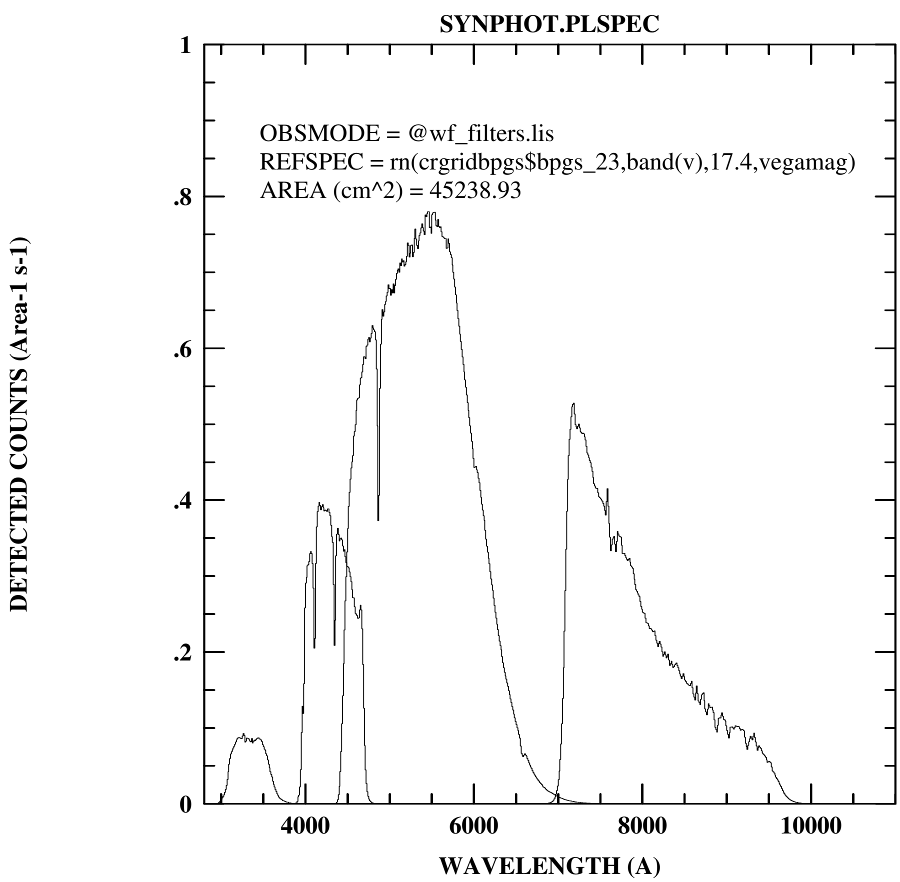

Figure 3.33: Distribution of Detected Counts

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 3.34.

```
obsmode  =                      Observation mode or @list
spectrum =                      Synthetic spectrum or @list
form     =                      Form for output plot
spfile   =                      Spectrophotometry data
(pfile   = "none")              Photometry data
(vzero   = " ")                 Variable zero values
(left    = INDEF)               Left plot limit
(right   = INDEF)               Right plot limit
(bottom  = INDEF)               Bottom plot limit
(top     = INDEF)               Top plot limit
(append  = no)                  Append to existing plot?
(ltype   = "solid")             Line type
(device  = "stdgraph")          Graphics device
(wavetab = "")                  Wavelength table
(refdata = "")                  Reference data
```

Figure 3.34: Plratio Parameters

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 "Observation Mode" on page 11). Several obsmode strings may be processed by inserting them one per line in a text file and setting obsmode=@*filename*. 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 "Spectrum" on page 14). 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 "Form" on page 17). 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. Each value in the list is substituted in turn. The values must be real numbers. Using vzero is equivalent to placing the spectrum 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. See Table 3.3 for examples of valid vzero strings.

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.

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_001` 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$alpha_lyr_001 wave=crcalspec$alpha_lyr_001
```

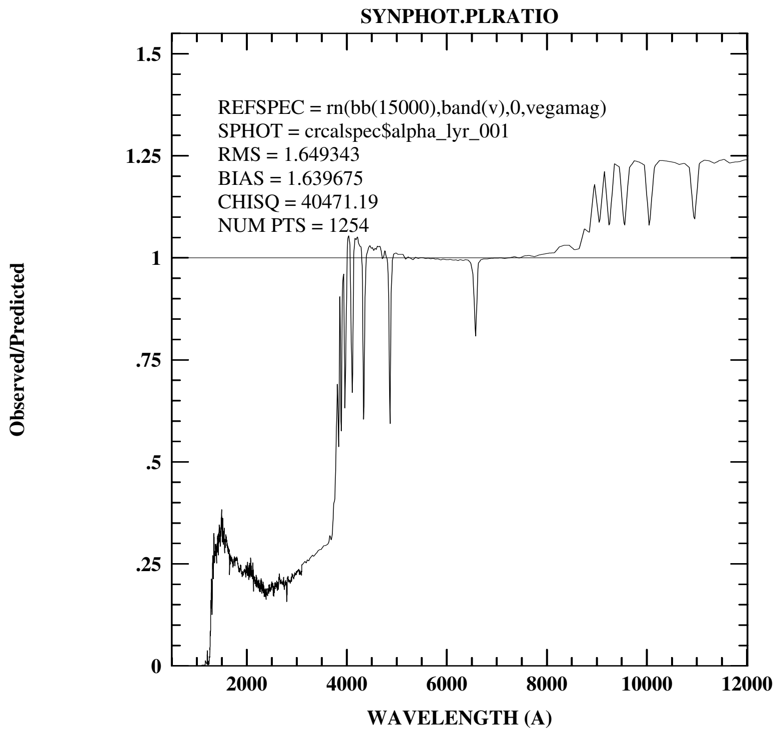

Figure 3.35: Sample Plratio Output

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. Alternatively, variable zero (`$0`) can be used within a single spectrum expression to automatically generate a list of spectra. Precalculated 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 3.36.

```
spectrum =                        Synthetic spectrum or @list
xmode    =                        X-axis mode
ymode    =                        Y-axis mode
xform    =                        X-axis form
yform    =                        Y-axis form
(vzero   = " ")                   Variable zero values
(input   = "none")                Input table
(output  = "none")                Output table
(left    = INDEF)                 Left plot limit
(right   = INDEF)                 Right plot limit
(bottom  = INDEF)                 Bottom plot limit
(top     = INDEF)                 Top plot limit
(append  = no)                    Append to existing plot?
(mktype  = "plus")                Marker type
(device  = "stdgraph")            Graphics device
(wavetab = "")                    Wavelength table name
(refdata = "")                    Reference data
```

Figure 3.36: Pltrans Parameters

The `spectrum` parameter is used to specify a set of synthetic spectra to be generated (see “Spectrum” on page 14). The commands can be placed in a file, one per line, with `spectrum=@`*`filename`*. 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 9). 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 “Form” on page 17). 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. Each value in the list is substituted in turn. The values must be real numbers.

Using `vzero` is equivalent to placing the `spectrum` 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. See Table 3.3 for examples of valid `vzero` strings.

The `input` table contains calculated x-y pairs of data values for plotting. If the file is an 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**.

Examples

Use **pltrans** to create a color-color diagram of a selection of main sequence stars contained in the BPGS spectral atlas (see Appendix B, page 116). 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 3.37 shows the results.

```
sy> pltrans @bpgs_ms.lis "band(b)-band(v)"
>>> "band(u)-band(b)" vegamag vegamag

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

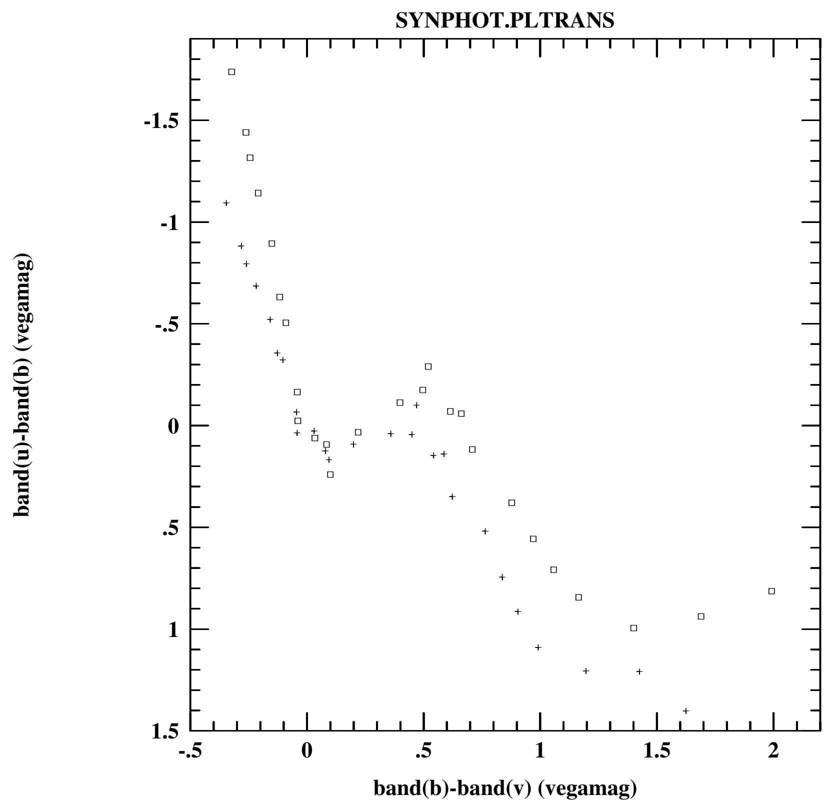

Figure 3.37: Results from Sample Pltrans Run

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  = "crcomp$hstgraph_*") Instrument graph table
(cmptbl = "crcomp$hstcomp_*")   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 lastest 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. 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.

CHAPTER 4: Inner Workings

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

Syncalc

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 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 "Observation Mode" on page 11 and "Spectrum" on page 14 for more details on the use and performance of these functions).

Function	*Degree*	*Description*
`band`	0	Telescope passband
`bb`	1	Blackbody spectrum
`box`	0	Rectangular passband
`ebmv`	0	Galactic extinction curve
`ebmvl`	0	LMC extinction curve
`gauss`	0	Gaussian passband
`grid`	1	Interpolated spectrum in grid
`hi`	1	HI continuum emission spectrum
`lgauss`	0	Log Gaussian passband
`pl`	1	Power-law spectrum
`poly`	0	Legendre polynomial
`rn`	1	Renormalized spectrum
`tilt`	0	Legendre polynomial product
`unit`	1	Constant spectrum

Table 4.1: Syncalc Functions

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 instrument graph table has five columns:

- Component name (COMPNAME)
- Instrument keyword (KEYWORD)
- Input node (INNODE)
- Output node (OUTNODE)
- Comment (COMMENT) columns.

The format of the instrument graph table is shown in Figure 4.1 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.

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

Figure 4.1: Structure of Instrument Graph Table

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 4.2 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 STSDAS 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 is a recent enhancement to the **synphot** package. This modification was made in order to allow more than one throughput to be stored in a single STSDAS table.

```
Component Lookup Table
TIME            CH*17
COMPNAME        CH*17
FILENAME        CH*40
COMMENT         CH*68
```

Figure 4.2: Structure of Component Lookup 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.

CHAPTER 5:

Calibration Using Synthetic Photometry

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 unit 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 it's observed count rate spectrum, so that $U(\lambda)$ is given by:

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

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 $U(\lambda)$:

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

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 WFPC and FOC calibration pipelines) to calibrate HST observing modes. The value of the PHOTFLAM header keyword that appears in calibrated WFPC and FOC 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

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

All of these source-independent characteristics can be computed for a given passband using the **bandpar, calcband,** or **calcphot** tasks in the **synphot** package. In the case of **bandpar** and **calcband**, the pivot wavelength, average wavelength, and rms width are known as “pivwv”, “avgwav”, and “bandw”, respectively, while in **calcphot** they are refered to as “pivlam”, “avglam”, and “rmslam”.

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,

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

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

APPENDIX A: HST Instrument Keyword Lists

In This Appendix...

This appendix contains current lists of all available component names for each of the HST science instruments, as well as some non-HST filter systems. These lists represent all of the valid component names that can be included as input to any of the **synphot** tasks via the `obsmode` parameter in order to define a passband. First, the list of allowed component names that represent the telescope, COSTAR, and science instruments are:

- *Telescope:* `ota noota`
- *COSTAR:* `costar nocostar`
- *Instruments:* `fgs foc fos hrs hsp pc wf wfc wfpc wfpc2`

As of September 1993, the default modes for the Telescope and COSTAR components are “ota” and “nocostar”, respectively. Soon after COSTAR is installed in the telescope, the default mode will be changed to “costar”. Note that for the WF/PC-1 instrument the names “wf”, “wfc”, and “wfpc” are all equivalent and correspond to the WFC mode of the WF/PC-1 instrument.

FGS

The FGS instrument keywords consist of a list of filters plus a dispersion axis. Some of the filter names are aliases for other filters. "astroclear" is an alias for "F605W", "clear" is an alias for "F583W", "red" is an alias for "F650W", and "yellow" is an alias for "F550W". One example of an FGS observation mode string would be "FGS,F583W,Y".

- *Filter:* `f550w (yellow) f583w (clear) f605w (astro-clear) f650w (red) nd5 pupil`
- *Axis:* `x y`

FOC

The FOC keywords consist of a list of detectors, filters, and miscellaneous keywords. The f/48 has two filter wheels and the f/96 detector has four filter wheels. Some of the filters have aliases. "fuvop" is an alias for PRISM1, "nuvop" is an alias for PRISM2, "fopcd" is an alias for PRISM3, "g450m" is an alias for F305LP, "g225m" is an alias for F220W, and "g150m" is an alias for F140W. The miscellaneous keywords include the COSTAR and the occulting fingers. An example of an FOC observation mode string would be "FOC,COSTAR,F/96,F410M"

- *Detector:* `f/48 f/96 f/288 spec`
- *f/48 Wheel 1:* `f140w (g130m) f150w (g150m) f175w f195w f220w (g225m) f305lp (g450m) prism3 (fopcd) (grat-prism)`
- *f/48 Wheel 2:* `f130lp f180lp f275w f342w f430w prism1 (fuvop) prism2 (nuvop)`
- *f/96 Wheel 1:* `f600m f630m f2nd f4nd f6nd f8nd pol0 pol0_par pol0_per pol0_unp pol60 pol60_par pol60_per pol60_unp pol120 pol120_par pol120_per pol120_unp prism1 (fuvop) prism2 (nuvop)`
- *f/96 Wheel 2:* `f140w f175w f220w f275w f320w f342w f370lp f430w f480lp f486n f501n`

- *f/96 Wheel 3:* `f120m f130m f140m f152m f165w f170m f190m f195w f210m f231m f1nd`
- *f/96 Wheel 4:* `f130lp f253m f278m f307m f346m f372m f410m f437m f470m f502m f550m`
- *Spectral Order:* `order1 order2 order3 order4`
- *Occulting Fingers:* `occ0p23 occ0p4 occ0p8`
- *Detector Formats:* `x48n256 x48n256d x48n512 x48nlrg x48zlrg x48zrec x96n128 x96n256 x96n512 x96z512 x96nlrg x96zlrg`

Note that the spectroscopic capabilities, and hence the related keywords `spec`, `order1`, `order2`, `order3`, and `order4`, are only available for the f/48 camera. Furthermore, the `occ0p23` keyword is only available with the f/48 camera, and the `occ0p4` and `occ0p8` keywords are only available with the f/96 camera.

The `x48*` and `x96*` keywords are used to account for the known dependency of DQE on the detector format (see the FOC Instrument Handbook for more details). These keywords invoke throughput tables that contain the (wavelength-independent) relative sensitivities for each format, where the 512x512 format (`x48n512` and `x96n512`) is set to 1.0. The following table lists the associations between formats and **synphot** keywords.

Camera	*Format*	*Keyword*
f/96	128 x 128	`x96n128`
	256 x 256	`x96n256`
	512 x 512	`x96n512`
	512z x 512	`x96z512`
	512z x 1024	`x96zlrg`
f/48	256 x 256	`x48n256`
	512 x 512	`x48n512`
	256z x 1024	`x48zrec`
	512 x 1024	`x48nlrg`
	512z x 1024	`x48zlrg`

Table A.1: FOC Detector Format Keywords

FOS

The FOS keywords consist of a list of detectors, apertures, gratings, and polarimeter waveplates and waveplate position angles. The waveplate keywords indicate whether waveplate A or B is being used and the angle of the waveplate. An example of an FOS observation mode string would be "FOS,BLUE,G160L".

- *Detectors:* `blue red`
- *Apertures:* `0.3 0.5 1.0 4.3 0.1-pair 0.25-pair 0.5-pair 1.0-pair upper lower 0.25x2.0 0.7x2.0-bar 2.0-bar blank failsafe`
- *Gratings:* `g130h g190h g270h g400h g570h g780h g160l g650l mirror prism order0`

- *Waveplates:* `pol0-a pol0-b pol22.5-a pol22.5-b pol45-a pol45-b pol67.5-a pol67.5-b pol90-a pol90-b pol112.5-a pol112.5-b pol135-a pol135-b pol157.5-a pol157.5-b pol180-a pol180-b pol202.5-a pol202.5-b pol235-a pol235-b pol257.5-a pol257.5-b pol270-a pol270-b pol292.5-a pol292.5-b pol315-a pol315-b pol337.5-a pol337.5-b`
- *COSTAR:* `costar nocostar`

The `upper` and `lower` aperture keywords are only recognized when used in conjunction with one of the paired apertures, e.g. "FOS,BLUE,G130H,UPPER,1.0-PAIR". The `order0` keyword is only available in conjunction with the `g160l` grating and the `blue` detector. The waveplate keyword syntax is *`POLpa-wp`*, where *pa* is the position angle in degrees, and *wp* is the A or B waveplate.

HRS

The HRS keywords consist of a list of detectors, apertures, gratings or mirrors, and Echelle mode orders. The Echelle mode orders are used with the keywords ECHA and ECHB. Orders 18 to 33 are valid with Echelle mode B, while orders 33 to 53 are valid with Echelle mode A. An example of a typical HRS obervation mode string would be "HRS,LSA,G160M".

- *Apertures:* `lsa ssa`
- *Gratings:* `echa echb g140l g140m g160m g200m g270m`
- *Mirrors:* `a1 a2 n1 n2`
- *Echelle Orders:* `18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 all`
- *COSTAR:* `costar nocostar`

HSP

The HSP keywords consist of a list of detectors, filters, apertures, and beams. The beams refer to the two beams that come out of the beam splitter. Not all apertures can be used with all detectors. Refer to the *HSP Instrument Handbook* for further information. The polarization detector also has angle and type keywords. An example of an HSP observation mode string would be “HSP,UV1,F220W,C”.

- *Detectors:* `pmt pol uv1 uv2 vis`
- *POL Filters:* `f160lp f216m f237m f277m f327m`
- *UV1 Filters:* `f122m f135w f140lp f145m f152m f184w f218m f220w f240w f248m f278n prism`
- *UV2 Filters:* `f122m f140lp f145m f152m f160lp f179m f184w f218m f248m f262m f278n f284m prism`
- *VIS Filters:* `f160lp f184w f240w f262m f355m f400lp f419n f450w f551w f620w prism`
- *Relay mirror:* `norelay relay`
- *Apertures:* `a b c d e f h j s t`
- *Beams:* `blue red`
- *Polarization Angles:* `0 45 90 135`
- *Polarization Types:* `ext ord par per`

WF/PC-1

The WF/PC-1 keywords consist of a list of filters, detectors, and miscellaneous keywords. The miscellaneous keywords include `baum`, which accounts for the Baum spot, `dn`, which converts units from electrons to data numbers (DNs), `cal`, which accounts for the flat-field correction, and `cont#`, which accounts for changes in sensitivity between decontamination events (see below).

The WF/PC-1 has twelve independently positionable filter wheels, each of which has five positions, including a `clear` position. Detectors 1 through 4 correspond to the Wide Field camera, while 5 through 8 are the

Planetary camera. A typical WF/PC-1 observation mode string would be "WFPC,F550W,4,DN".

- *Instrument:* `wfpc, wfc, wf` (all equivalent), `pc`
- *Detectors:* `1 2 3 4 5 6 7 8`
- *Miscellaneous:* `baum dn cal cont#`
- *Filter Wheel 1:* `f673n f8nd g450 g800`
- *Filter Wheel 2:* `f122m f336w f439w g200 g200m2`
- *Filter Wheel 3:* `pol0 pol60 pol120 f1083n`
- *Filter Wheel 4:* `f157w f194w f230w f284w`
- *Filter Wheel 5:* `f569w f658n f675w f791w`
- *Filter Wheel 6:* `f631n f656n f664n f702w`
- *Filter Wheel 7:* `f375n f437n f502n f588n`
- *Filter Wheel 8:* `f368m f413m f492m f622w`
- *Filter Wheel 9:* `f547m f555w f648m f718m`
- *Filter Wheel 10:* `f785lp f814w f875m f1042m`
- *Filter Wheel 11:* `f128lp f469n f487n f517n`
- *Filter Wheel 12:* `f606w f725lp f850lp f889n`

The detector specifications 1 through 4 are valid only when used in conjunction with the `wf`, `wfc`, or `wfpc` instrument keywords, and detectors 5 through 8 are only valid when used with the `pc` keyword. If a detector number is not specified, the default detector for WF is #2, and the default for PC is #6.

The `cont#` keyword references a "parameterized" component throughput table, meaning that the throughput is a function of some other variable besides wavelength (see page 92). In this case, the extra variable is the Modified Julian Date and is used to account for the gradual decline in throughput between decontamination events, as well as the sudden increase in throughput immediately after a decontamination. Data exists for 48384.0 < MJD < 49329.0 (approximately May 1991 to Dec 1993), in intervals of 20-30 days. To use this capability in simulations, include the string `cont#`*ddddd* in your `obsmode` specification, where *ddddd* is the desired MJD value (which can be given as either an integer or real value). For example, to get the throughput for the F336W filter on MJD 48753, use `obsmode="wfpc,f336w,cont#48753"`. The **synphot** expression evaluator interpolates between the dates for which data exists in the table to arrive at an estimate of the throughput on the requested date.

WFPC2

The WFPC2 keywords consist of a list of detectors, filters, and miscellaneous keywords. The miscellaneous keywords include `a2d7` and `a2d15`, which convert from electrons to data numbers (DNs), `cal`, which accounts for the flat-field response, `lrf#`, which is used for the linear ramp filters, and `cont#`, which accounts for changes in throughput between decontamination events (see below).

The WFPC2 has twelve filter wheels, each of which has five positions, including a `clear` position. Detector 1 is the PC chip, while 2 through 4 are the Wide Field chips. If a detector is not specified, the default is #4. A typical WFPC2 observation mode string would be "WFPC2,F450W,2".

- *Detectors:* `1 2 3 4`
- *Miscellaneous:* `a2d7 a2d15 cal lrf# cont#`
- *Filter Wheel 1:* `f122m f157w f160bw f953n`
- *Filter Wheel 2:* `f130lp f165lp f785lp f850lp`
- *Filter Wheel 3:* `f336w f410m f467m f547m`
- *Filter Wheel 4:* `f439w f569w f675w f791w`
- *Filter Wheel 5:* `f343n f375n f390n f437n`
- *Filter Wheel 6:* `f469n f487n f502n f588n`
- *Filter Wheel 7:* `f631n f656n f658n f673n`
- *Filter Wheel 8:* `f170w f185w f218w f255w`
- *Filter Wheel 9:* `f300w f380w f555w f622w`
- *Filter Wheel 10:* `f450w f606w f702w f814w`
- *Filter Wheel 11:* `f1042m fqunv fqunv33 fqch4n fqch4n33 fqch4n15 fqch4p15 polq polq_par polq_perp polqn33 polqn18 polqp15`
- *Filter Wheel 12:* `lrf#`

The `cont#` keyword references a "parameterized" component throughput table, meaning that the throughput is a function of some other variable besides wavelength (see page 92). In this case, the extra variable is the Modified Julian Date and is used to account for the gradual decline in throughput between decontamination events, as well as the sudden increase in throughput immediately after a decontamination. To use this capability in simulations, include the string `cont#`*ddddd* in your `obsmode` specification, where *ddddd* is the desired MJD value (which can be given as either an integer or real value). For example, to get the throughput for the F336W filter on MJD 49486, use `obsmode="wfpc2,f336w,cont#49486"`. The **synphot** expression evaluator interpolates between the dates for which data exists in the table to arrive at an estimate of the throughput on the requested date.

At this time, there are no calibration or throughput data available for the WFPC2 decontamination or ramp filter (see below) tables. Consequently, the current tables are set to unity. Calibration data used to populate these tables are expected to be available and analyzed by mid to late 1995.

Filter wheel 11 contains 3 specialized quadrant (quad) filters in which each quadrant corresponds to a facet of the pyramid, and therefore to a distinct camera relay. One quad filter, `fquvn`, contains four narrowband, redshifted [O II] filters. The second, `fqch4n`, contains four methane (CH4) band filters, and the third, `polq`, contains four polarizing elements. The instrument graph table is constructed so that distinct throughput values are automatically selected for a given quadrant of the `fquvn` and `fqch4n` filters based on the selected detector.

Note that all three of the quad filters were designed to map onto a four-faceted WFC configuration. However, in the final design of the instrument, with WF quadrant 1 replaced by the PC, it is necessary to rotate the quad filters by -33° in order to bring filter quadrant 1 into the WF 2 and 3 relays. In addition, partial rotations of -15° and +15° have been implemented for the methane (CH4) filter in order to bring two of its quadrants into the PC relay, and partial rotations of -18° and +15° have been implemented for the polarizer (pol) filter in order to allow observations with different polarization angles (see the WFPC2 Instrument Handbook for more details). Unique component keywords have been implemented in the instrument graph for the nominal and rotated positions of each of these filters. The nominal positions are designated as `fquvn`, `fqch4n`, and `polq`. The rotated positions are designated as `fquvn33`

(-33°), `fqch4n33` (-33°), `fqch4n15` (-15°), `fqch4p15` (+15°), `polqn33` (-33°), `polqn18` (-18°), and `polqp15` (+15°).

Filter wheel 12 contains four linearly variable narrowband ramp filters, which together cover a total wavelength range of 3700 to 9800 Å. The FWHM of the throughput at a given wavelength is typically about 1% of the central wavelength. The ramp filters have been implemented as a parameterized component in **synphot**. To use the ramp filters in simulations, use the keyword syntax `lrf#`*wwww*, where *wwww* is the central wavelength, in Angstroms, that you desire, e.g. `obsmode="wfpc2,3,lrf#4861"`.

Non-HST Filter Systems

In addition to the HST instrument components the graph table also supports various standard filter system passbands. The passband keyword string consists of the name of the system plus the passband name. An example of a passband observation mode string would be "COUSINS,I". If no system name is given for any of the *UBVRI* bands, the defaults are Johnson *UBV* and Cousins *RI* (i.e., the Landolt *UBVRI* system). For example, "v - r" will result in Johnson *V* minus Cousins *R*.

- *System:* `ans baum cousins johnson landolt stromgren walraven`
- *ANS Bands:* `1550 1550n 1800 2200 2500 3300`
- *Baum Bands:* `f336w f439w f547m f555w f569w f606w f622w f675w f702w f725lp f785lp f791w f814w f850lp f1042m`
- *Cousins Bands:* `R I`
- *Johnson Bands:* `U B V R I`
- *Landolt Bands:* `U B V R I` (= Johnson *UBV* + Cousins *RI*)
- *Stromgren Bands:* `u v b y`
- *Walraven Bands:* `V B L U W`

The Baum filter set is a set of 15 broadband and intermediate-band filters that are copies of some of the onboard WF/PC-1 filters and were used as part of a ground-based observing campaign to determine calibrations for the WF/PC-1 instrument. In order to match the response of the WF/PC-1 flight passbands as closely as possible, the throughputs for the Baum filters have been multiplied by the spectral response curve of the ground-based CCD (measured in the laboratory) and twice by the spectral reflectance of aluminum (see Harris et al., 1991, for details).

The throughput data for the Johnson *UBV* bands are the U3, B2, and V synthetic passband data of Buser and Kurucz (1978), while the Johnson *R* and *I* band throughputs are from Johnson (1965), Table A1. The Cousins *R* and *I* throughputs are taken from Bessell (1983), Table AII. The filter throughputs for the Stromgren system are taken from Matsushima (1969). The throughput data for the Walraven bands are from Lub and Pel (1977), Table 6.

APPENDIX B:
On-Line Catalogs and Spectral Atlases

In This Appendix...

There are several spectral atlases consisting of both observed and model data that are available in STSDAS 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. 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 conatains 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.

HST Calibration Target Spectra

The directory `crcalcspec$` contains spectra of HST flux calibration targets. These represent the combination of all the available optical and UV photometry and spectrophotometry archived in the HST CDBS. The spectral tables have names like *starname*_001.tab. These spectra can be used with **synphot** tasks by setting either the `spectrum` or `spfile` task parameters to the name of the desired file, e.g., `crcalspec$alpha_lyr_001.tab`. A list of the spectra in this directory is shown in Table B.3.

Kurucz 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$`.

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 and/or improvements in the underlying physics, as follows:

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

Table B.1: Kurucz K1200 Library

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. The following table summarizes the structure of the atlas.

Block	***Source of Spectra***	***# of Spectra***	***Models + Opacity***	***Teff***	***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

Table B.2: BK Atlas

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$` directory are of the form `bk_`*mnnnn*`.tab`, where *m* is the block code (a, b, c, d, s, or m), and *nnnn* is a running sequence number within the block. Within each block, the individual models are ordered starting with the lowest metallicity, [M/H], temperature, Teff, and surface gravity, log G, and with log G increasing fastest and [M/H] increasing most slowly. A complete listing of all the BK atlas model spectra can be found in the README file in the `crgridbk$` 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.

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 STSDAS tables called `bz_`*`nn`*`.tab`, where *nn* runs from 1 to 77. A list of the file names and spectral types is shown in Table B.4.

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 B.5. Note that when refering to Table B.x for the names of files in the Gunn-Stryker atlas, the names are of the form `gs_`*`nnn`*`.tab`, instead of `bpgs_`*`nnn`*`.tab`.

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 B.5.

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 B.6.

Table B.3: HST Calibration Spectra

File	*Star*	*Type*
`10_lac_001`	10 Lac	O9 V
`agk_81d266_003`	AGK +81 266	sdO
`alpha_lyr_001`	alpha Lyra	A0 V
`bd_28d4211_003`	BD +28 4211	Op
`bd_33d2642_002`	BD +33 2642	B2 IV
`bd_75d325_003`	BD +75 0325	O5p
`bpm16274_001`	BPM 16274	DA2
`eta_aur_001`	eta Aur	B3 V
`eta_uma_002`	eta UMa	B3 V
`feige34_003`	Feige 34	DO
`feige110_003`	Feige 110	D0p
`g93_48_002`	G93-48	DA3
`g191b2b_003`	G191-B2B	DA0
`gamma_uma_001`	gamma UMa	A0 V
`gd50_002`	GD50	DA2
`gd108_003`	GD108	sdB?
`grw_70d5824_003`	GRW +70 5824	DA3
`hd49798_002`	HD 49798	O6
`hd60753_002`	HD 60753	B3 IV
`hd93521_003`	HD 93521	O9 Vp
`hz2_003`	HZ 2	DA3
`hz4_002`	HZ 4	DA4
`hz21_003`	HZ 21	DO2
`hz44_003`	HZ 44	sdO
`lam_lep_001`	lambda Lep	B0.5 IV
`lb227_002`	LB 227	DA4
`lds749b_003`	LDS 749B	DB4
`mu_col_002`	mu Col	O9.5 IV
`ngc7293_003`	NGC 7293	
`zeta_cas_002`	zeta Cas	B2 IV

Table B.4: 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		

Table B.5: 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 B.5: 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 B.5: 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			

Table B.6: 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 B.6: Jacoby-Hunter-Christian Spectral Atlas

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 B.6: Jacoby-Hunter-Christian Spectral Atlas

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			

Bibliography

Bessell, M.S., 1983, *PASP*, 95, 480.

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.

Eriksson, K., Bell, R.A., Gustafsson, B., Nordlund, A., 1979, *Trans. IAU*, Vol. 17A, Part 2, Reidel, Dordrecht, p. 200.

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.

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.

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.

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.

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.

Index