Chapter 4

SIMULATIONS AND MODEL COMPARISONS

Many GO's will want to compare their spectra to models of the target spectrum. To make the two comparable, either the EUVE spectra must be flux calibrated, or else the instrument signature of the Spectrometer must be added to the model spectrum. Many EUVE spectra have relatively low signal to noise, even after the spectrum has been re-extracted, so that flux calibration is difficult. Comparisons may be done in these cases by converting the model spectrum to a set of three simulated EUVE channel spectra in counts/second or counts for a comparable exposure time.

4.1 Making Simulated EUVE Spectra

This section gives instructions and examples on creating simulated one-dimensional EUVE spectra from spectral models. Typically, the models consist of output files from a modeling program that simulates the flux from the target. However, the fluxes used in simulations should be estimates of flux at the telescope, including the effects of interstellar absorption along the line of sight.

4.1.1 ISM calculates transmission of the interstellar medium

To create simulated EUVE spectra from a theoretical model, one must determine the spectrum resulting from the model *at the telescope*. The task **euvspec.ism** is a tool for calculating the effect of absorption by the interstellar medium on spectra produced by modeling programs. The task currently supports a single ISM model, presented in the paper by Rumph, et. al[1]. The code also accounts for metals as described in the R. Morrison and D. MacCammon paper (*1983, ApJ, 270, 119*). The code uses the best available photoionization cross sections from the atomic data literature and combines them in an arbitrary mixture of HI, HeI, and HeII.

A typical parameter file for **ism** is shown in figure 4-1. The input parameters are the column density of neutral hydrogen in the direction of the target, in cm^{-2}, the ratios of HeI/HI, and HeII/HI, and a two-column list with wavelengths in Å in the first column and fluxes in the second, for which to compute the local flux. The fluxes may be expressed in any units. Each flux value is multiplied by a dimensionless transmission for that wavelength. The value of the parameter `model` should not be

[1] Interstellar Medium Continuum, Autoionization, and Line Absorption in the Extreme Ultraviolet, T. Rumph, S. Bowyer, and S. Vennes, *AJ*, **107**(6), 2108–2114, 1994

changed, as only the Rumph et. al. model is currently supported. Figure 4-2 shows a model of a hot DA white dwarf spectrum before and after processing by **ism**.

```
eu> lpar ism

    hcolumn = 1.0000000000000E18 neutral hydrogen column density (cm**-2)
 (he1hratio = 0.1)            HeI/HI ratio
 (he2hratio = 0.)             HeII/HI ratio
     (model = "rumph")        ism model
     (input = "hotDA")        Name of input file (or stdin)
    (output = "hotDA.ism")    Name of output file (or stdout)
  (document = yes)            output comment lines that document parameters
      (mode = "al")
```

Figure 4-1: Sample parameters for calculating the transmission of flux from a model white dwarf spectrum.

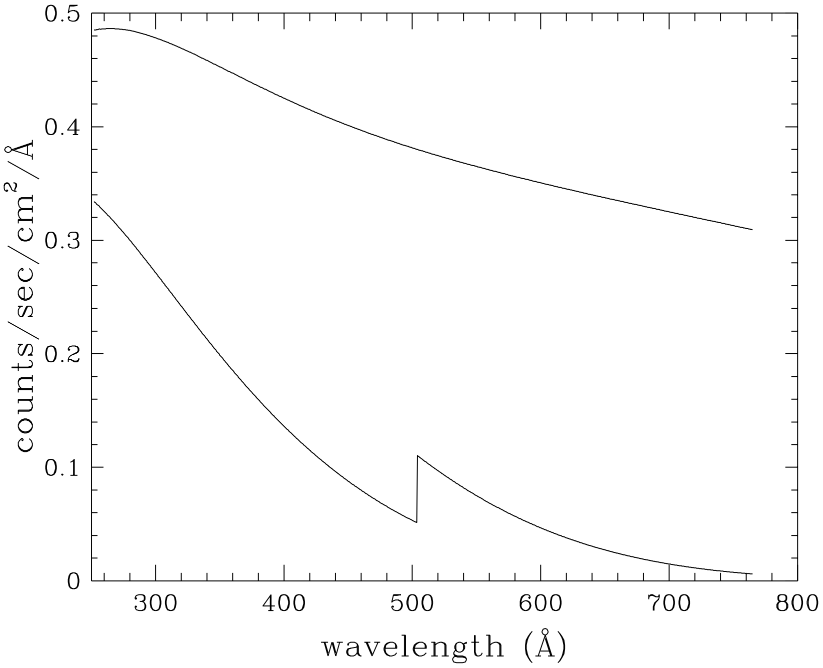

Figure 4-2: Effect of the task **ims** on a hot DA white dwarf model. The calculation assumes no HeII, and HeI/HI of .1. The offset between the spectra is entirely the result of estimated absorption.

Spectra created with **ism** are suitable for input to **specmod**.

4.1.2 SPECMOD makes simulated EUVE spectra

The task **euvspec.specmod** simulates the effect of the EUVE Spectrometer on model spectra, producing a spectrum of binned integrated counts from a model spectrum of flux *at the telescope*. Input model spectra must have columns of wavelengths and fluxes, and can be ASCII text files, IRAF images, or ST tables. Input may be either a list of emission lines, or a continuum whose values are sampled or integrated bins. The user defines the integration time, the binning of the output images, and can choose whether to convolve the spectra with a point spread function or the spectrometer collimator response, where appropriate.

The input model may be a file or image of: a) the centers and integrated fluxes of a number of narrow emission lines, or values of b) photons/cm^2/sec or c) photons/cm^2/sec/Å for any number of wavelengths. Fluxes input as ASCII files or ST tables need not be regularly spaced; **specmod** will perform a linear interpolation where necessary to create the specified output binning. The output from **specmod** is a one-dimensional spectrum for one or more Spectrometer channels, each with a linear WCS determined by the input values of the binning parameters in a detector `pset`. The output spectrum has units of counts integrated over the specified exposure time.

A sample parameter setup is shown in figure 4-3.

```
eu> lpar specmod


    (model = "W7")            Input model
  (wavecol = "")              Input model wavelength column
  (fluxcol = "")              Input model flux column
   (output = "wd7_spn")       Basename of output files
 (outtable = no)              Produce ST table as output?
  (detpars = "detsw.par,detmw.par,detlw.par") List of detector psets
  (refdata = "reference/detector") Reference dataset
 (linelist = no)              Is input model a linelist?
 (linefwhm = 1.0000000000000E-5) FWHM of lines in linelist (angstroms)
  (sampled = yes)             Is input model sampled?
 (convolve = yes)             Apply convolution?
      (psf = yes)             Use point spread function for convolution?
   (nsigma = 4.)              Number of sigma for convolution gaussian
 (addnoise = no)              Add poisson noise?
     (seed = 2)               Seed for random number generator
     (mode = "al")
```

Figure 4-3: **Specmod** parameters for making simulated EUVE spectra from an ASCII line list.

Notes on running **specmod**

- **specmod** reads the parameter `linelist` to determine whether the input is a line list, with wavelengths vs. integrated photons/cm^2/sec for emission lines of finite width, or a continuum in photons/cm^2/sec or photons/cm^2/sec/Å.

- If `linelist` is "yes," **specmod** uses the parameter `linefwhm` as the intrinsic line width. This parameter should not be changed from its default value ($<< FWHMofPSF$) unless the user knows that the lines are unusually broad. Very broad and/or blended lines should have fluxes specified at intervals of .05 Å or less.

- IRAF images can be used only for the two varieties of continuum input, and must have a WCS relating the pixel index to wavelength. Sampled continua are assumed to be in photons/cm^2/sec/Å and unsampled, or histogram images are assumed to be photons/cm^2/sec.

- The output may be stored in an ST table, as opposed to an image, by setting the boolean parameter `outtable`.

- If a continuum is input, the user must specify whether it is a sampled piecewise linear function (PLF), with pixel values in photons/cm^2/sec/Å or a binned histogram, in photons/cm^2/sec, using the boolean parameter `sampled`.

- Values for `wavecol` and `fluxcol` need not be supplied for ASCII files or ST tables that have wavelengths in the first column and fluxes in the second. For ASCII files with some other format, specify values of the column parameters using "c1," "c2," "c3," etc. For ST tables, use the column names.

- **specmod** produces an output spectrum in each channel for which it has a detector `pset` (short for 'parameter set') listed in the value of `detpars`. Each detector `pset` is a parameter file which is specific to one of the detectors, and must be created by the user before **specmod** can be run. The parameters that must be supplied by each `pset` are listed in the **specmod** help page. Information on creating detector `psets` is given in the help page, and below.

- **specmod** uses the instrument reference data: wavelength solution, effective areas, and PSF. The default value of `refdata`, *reference/detector.tab*, should be correct for the standard EUVE directory setup shown in Chapter 0 of the *EGO Center Data Products Guide*.

- The parameter `nsigma` determines the width of the kernel (gaussian PSF or triangular collimator transmission) used in the convolution, in units of the standard deviation. It should be set to at least 4, so that the convolution is not cut off in the wings, but higher values can make **specmod** run slowly.

Creating detector `psets`

Each `pset` supplies specialized information for processing the simulated output of one detector, including parameters like the exposure time, and the desired output binning. A typical detector `pset` is shown in figure 4-4 below.

Sample detector psets are included with every copy of the **euv** package in the library `euvlib`. The files are called *swdetpars.par, mwdetpars.par,* and *lwdetpars.par,* and are in the location indicated by the environmental variable `euvlib`. They have the default values for each detector.

```
eu> lpar detsw.par
```

```
   (detector = "sw")          Name of detector to process
    (exptime = 51964.9)       Exposure time (seconds)
    (binsize = 0.2)           Size of output bins (angstroms)
    (minwave = 70.)           Wavelength of first bin (angstroms)
    (maxwave = 190.)          Wavelength of last bin (angstroms)
      (nbins = INDEF)         Number of output bins
   (convstep = 0.1)           Wavelength step for convolution (angstroms)
   (backrate = 5.4000000000000E-4) Background rate (counts/sec/angstrom)
     (orders = "1")           List of spectrometer orders to include
  (outsuffix = "sw")          Suffix appended to output filename
       (mode = "al")
```

Figure 4-4: Example of a short wavelength detector `pset` for use with **specmod**. The filename —in this case *detsw.par*— must have the suffix *.par*.

Since each observation may require different parameter values, and **specmod** may be run several times for each, it is convenient to create separate `psets` for each EUVE observation. This can be done from the IRAF parameter file editor. Edit the short wavelength pset file to create your own SW pset by typing:

```
eu> epar euvlib$swdetpars.par
```

Edit the file to insert the desired values. Make sure you enter the data in the required units. When finished, write the parameters to a unique filename by typing

```
:w mysw.par
```

(insert an appropriate filename with the .par suffix). Then you can fetch and edit the MW file without leaving the editor:

```
:e euvlib$mwdetpars.par
```

and the same for the LW Spectrometer.

Notes on creating `psets`

- The value of `exptime` should match the value of the header card `EXPTIME` in the spectrum to which you wish to compare the simulation. Use **imhead** on your extracted EUVE spectra to determine the values.
- Three of the four binning parameters, `binsize`, `minwave`, `maxwave`, and `nbins`, must have values to completely determine the output range and binsize.
- The value of `orders` determines the number of spectral orders that are calculated, allowing the creation of simulated spectra comparable to the output

from **suborder**. The default files include all possible orders for each channel, producing spectra equivalent to the 1-d spectra supplied on the data tape. Of course which higher spectral orders can be calculated depends on the range of wavelength/flux data supplied.

- The default value of `convstep` should generally not be changed from the values in the default psets. (.1 for SW, .2 for MW, .4 for LW).

Since **specmod** has no "command line" parameters, it can be run repeatedly on the same model without much typing[2]. This facilitates making many simulations from the same model; one need change only the input or the detector/`pset` each time. Several sets of `psets` can be created and referenced with the "@filename" notation, along with a list of output image names. Most of the task parameters, and those from the relevant `pset`, are saved as header cards in the output image.

Example 4.1 : Creating spectra with specmod

The following command lines create simulated spectra with and without Poisson noise from a hot DA white dwarf model in a two-column ASCII file:

```
eu> specmod model=hotDA.ism output=DAmod3 detpars=detsw.par,detmw.par,detlw.par
linel- sampled+ addnoi-

eu> specmod output=DAmod3noi addnoi+
```

The model spectrum is the same one produced by the example in section 4.1.1. The two simulated EUVE spectra for the LW detector are shown in figure 4-5. Header cards preserve a record of most of **specmod**'s parameters, as can be seen from the LW image header shown below:

```
hotdanoi.lw[481][real]:
    No bad pixels, no histogram, min=unknown, max=unknown
    Line storage mode, physdim [512], length of user area 1297 s.u.
    Created Tue 19:44:45 30-May-95, Last modified Tue 19:44:45 30-May-95
    Pixel file "HDR$hotdanoi.lw.pix" [ok]
    WCSDIM  =                    1
    CTYPE1  = 'LINEAR  '
    CRVAL1  =                280.
    CRPIX1  =                  1.
    CDELT1  =                  1.
    CD1_1   =                  1.
    LTM1_1  =                  1.
    WAT0_001= 'system=world
    MODEL   = 'hotdamod.ism'
    WAVECOL = '        '
    FLUXCOL = '        '
```

[2] However, **specmod** will not overwrite files, as a rule, so unwanted output from previous runs must be deleted or moved to a new image name with **imrename**.

```
OUTPUT  = 'hotdapoi.lw'
OUTTABLE=                    F
REFDATA = 'reference$detector'
LINELIST=                    F
LINEFWHM=  1.00000000000000E-5
SAMPLED =                    T
CONVOLVE=                    T
PSF     =                    T
NSIGMA  =                   4.
ADDNOISE=                    T
SEED    =                    2
DETECTOR= 'lw      '
EXPTIME =              50658.3
BINSIZE =                   1.
MINWAVE =                 280.
MAXWAVE =                 760.
NBINS   =                  481
CONVSTEP=                  0.5
BACKRATE=  1.70000000000000E-4
ORDERS  = '1       '
BANDID1 = 'spectrum - generated from model'
```

Other simulation tasks

Users may also wish to try the tasks in the package **noao.artdata** which can be used to construct artificial spectra or add noise to spectra. Keep in mind that the actual extracted spectra have noise properties which depend on the number of lines in the spectrum aperture, the size of the background sample region, and the method of background subtraction.

4.2 Statistical Comparisons Between Spectra

4.2.1 COMPSPEC makes statistical comparisons

The task **euvtools.compspec**[3] compares two spectra in either counts or flux units, by performing pixel-by-pixel computation of the χ^2 statistic and the associated probability that the two spectra sample the same parent distribution. The task can compare models to data, models to models, or compare observation spectra to each other, using a logical hierarchy which chooses an appropriate data variance for each type of calculation. Multiple models can also be compared to a single spectrum and

[3] Since the **stsdas** package also has a task called **compspec**. You may need to reload the **euvtools** package or type the package and task names, e.g. **euvtools.compspec** when summoning **compspec**, its parameter file, or help file.

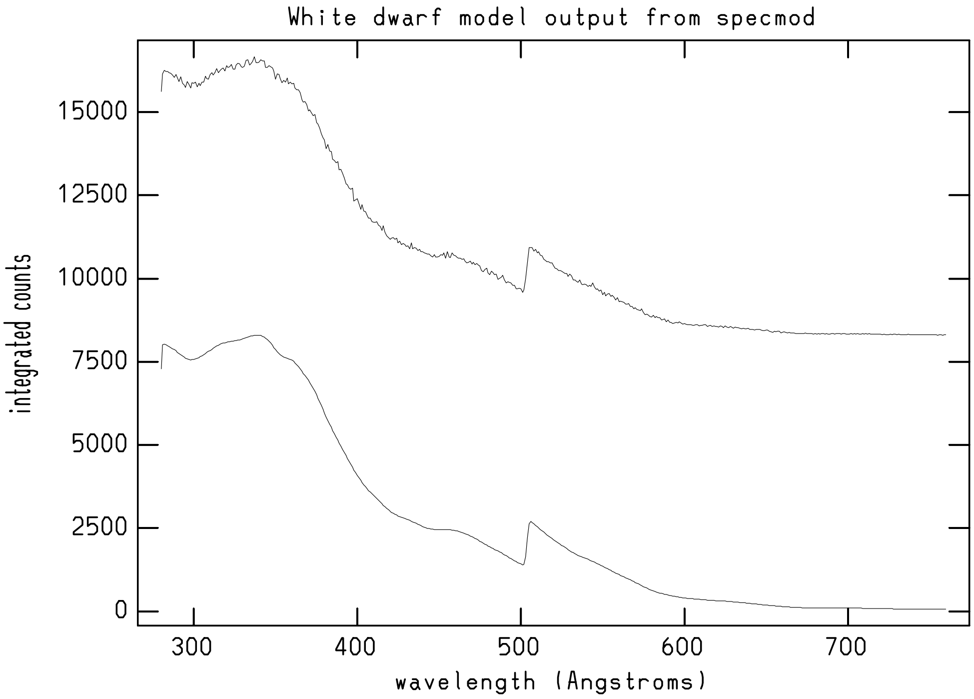

Figure 4-5: Output spectra from **specmod**. Input model spectrum is shown in figure 4-2. The upper spectrum has had Poisson noise added, and has been offset by 8230 on the intensity scale.

vice versa, if one of the input arguments is a list or file of image names. A sample parameter file for comparing a single observation spectrum, *euvspsm_lw*, to a list of model spectra is shown in figure 4-6.

```
eu> lpar compspec


    spectra1 = "euspsm_lw"      one set of spectral 1d images
    spectra2 = "@models_lw"     comparison set of spectral 1d images
  (poiss1errs = yes)            are errors poisson for spectra1
  (poiss2errs = yes)            are errors poisson for spectra2
  (resolution = INDEF)          Resolution
          (x1 = 290.)           Minimum wavelength
          (x2 = 740.)           Maximum wavelength
 (constraints = 0)              No. of constraints
       (title = yes)            Print output title
        (mode = "al")
```

Figure 4-6: Sample **compspec** parameters for comparing an EUVE spectrum to model spectra.

Notes on running **compspec**

- □ Images to be compared must be one-dimensional spectra or have 1-d spectra identified with a `BANDID#` card in the header.
- □ A number of model spectra can be compared to a single observation by using a comma-separated list or the *cl* pointer notation *@filename* for the value of `spectra1` or `spectra2`.
- □ If two lists of spectra are to be compared, they must have the same number of entries.
- □ The variance used in each χ^2 computation is determined by the presence of a `BANDID# = ...SIGMA...` header card indicating an error band with σ values, the values of `poiss1errs` and `poiss2errs`, and the types of the images being compared: observation data or model, Please see the figure 4-7 below, and the **compspec** help page.
- □ The values of `x1` and `x2` specify the upper and lower limits of the wavelength range to be compared, in Å.
- □ For best results, the value of `resolution` should be "INDEF", which forces a minimal rebinning to bring the input spectra to the same resolution. A value matching the observation spectrum may be specified if a model comparison spectrum has extremely small bins, but the χ^2 test results are not as meaningful if the data are substantially rebinned.

How compspec determines the test data variance

As noted above, **compspec** will compute χ^2 for different combinations of observation and model spectra. The χ^2 computation assumes one of the inputs, the "test data," is being tested against some other set of "comparison data". It is assumed that the test data will usually be a model of the target spectrum, with an EUVE observation spectrum as the comparison data. When **compspec** receives a pair of spectra, it first determines what to use as the variance of each input, then which input of the pair is the model fit being tested. Only the variance of the fit being tested is used in the calculation of χ^2.

compspec first checks the header of each input image for a card of the form `BANDID# = sigma`. If the card is not found, Poisson errors are assumed. Figure 4-7 shows what is used for σ^2 of each type of input spectrum. Whether the spectrum is considered a model or an observation depends on the value of the parameter `poisserrsX`. The symbol `#` stands for the band number of the error band, if present, and X is 1 or 2. Table 4-1 shows which spectrum's variance is used in the computation of χ^2, based on the spectrum type, model or observation.

If model spectra are being input as, say, `spectra2`, the value of `poiss2errs` should be "no". to force **compspec** to use each model variance in calculating χ^2. If either or both inputs are multiple spectra, the determination of which input is the "test data" is done each time **compspec** compares a new pair of spectra, although `poiss1errs` and `poiss2errs` have the same values for each pair.

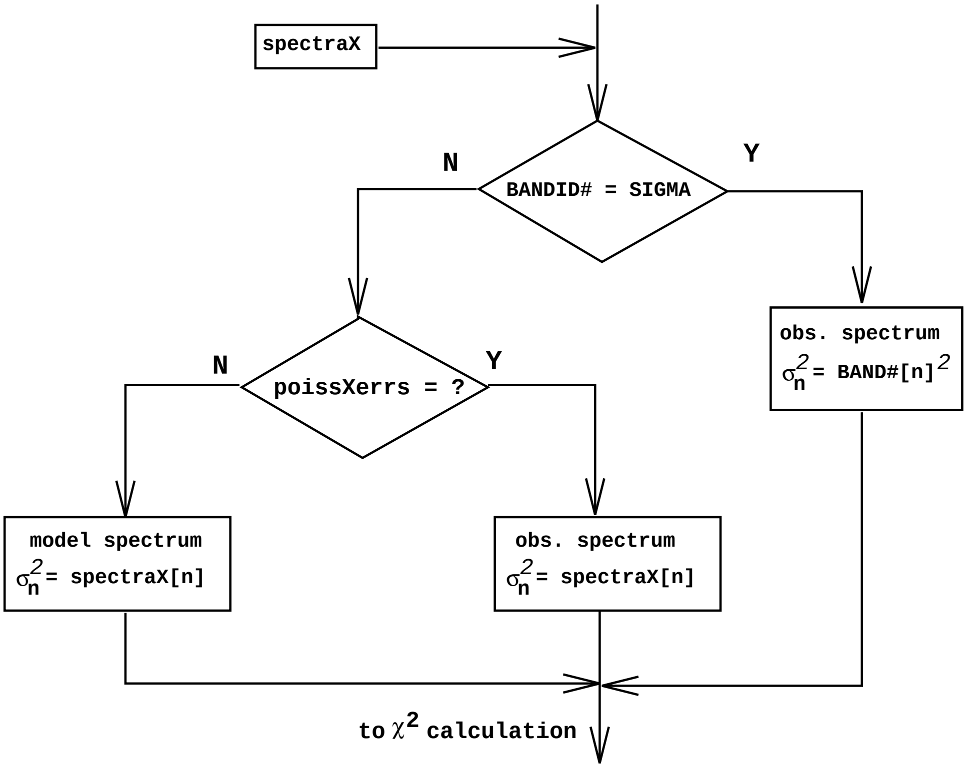

Figure 4-7: Logic to determine variance (σ^2) of spectrum input to **compspec**. If no error band is identified in the header, Poisson errors are assumed.

Table 4-1: Which input variance is used in χ^2?

`spectra1`	`spectra2`	σ^2 in χ^2
model spectrum	obs. spectrum	σ_1^2
obs. spectrum	model spectrum	σ_2^2
model spectrum	model spectrum	σ_1^2
obs. spectrum	obs. spectrum	$\left[\sigma_1^2 + \sigma_2^2\right]$

Example 4.2 : Compare specmod outputs with compspec

A nearly trivial use of **compspec** is to see how much difference the addition of Poisson noise makes in a spectrum produced by **specmod**. If we compare the simulated hot DA white dwarf spectra with and without noise to each other, assuming that the one without noise is a "model" while the other is not:

```
eu>  compspec hotda.lw hotdanoi.lw poiss1- poiss2+
```

The task assumes that `spectra1` is fit to the data in `spectra2`. The output shows that **compspec** concurs with the fit:

```
spectrum1 spectrum2 statistic probability degfree
=================================================
hotda.lw hotdanoi.lw 460.2904968 0.7440797 481
```

It gives a probability of nearly 75% that the two are sampling the same parent distribution. This is hardly surprising, given the brightness of the spectrum; the Poisson noise added by **specmod** to the model was relatively small.