Lying Line Ratios: ISIS Plasma model simulations

Line ratios are commonly used as plasma diagnostics. Care must be used, however, in applying an isothermal ratio to a multithermal plasma — a ratio doesn't always mean what you think it means. ISIS makes it easy to simulate spectra for any given a plasma model, and to evaluate the model fluxes and ratios of lines. Here we will give the significant steps in a simulation and analysis scenario in which we

1. Define a multi-thermal plasma model;
2. Load HETGS ARFs and RMFs for first orders, set exposure, assign responses, make fake data;
3. Define fitting function for O VII HeLβ and O VIII HLα, fit each line, compute 1σ confidence intervals;
4. Evaluate the theoretical isothermal ratio of O VII HeLβ/HLα and derive a plasma temperature from it.

Define a Plasma Model

Defining a custom plasma model (not one provided by built-in intrinsic models or by xspec module functions) in ISIS is simple. You load the atomic database, define a temperature grid and emission measure weights, specify elements and abundances if you want a non-solar mix, put them into a plasma state variable, and create a fit function:[1]

```
plasma( aped ) ;   % load the database

logtgrid = [ 5.8 : 8.3 : 0.1 ] ;  % logarithmic temperature grid
tgrid = 10^logtgrid ;             % ... and in linear form

% make a log-gaussian 3-component emission measure distribution,
% using built-in integrated gaussian function:
%
%                   model    low grid   high grid   norm  center sigma
norms = eval_fun2( "gauss", logtgrid, logtgrid+0.1, [0.6, 6.3, 0.15 ] )
      + eval_fun2( "gauss", logtgrid, logtgrid+0.1, [1.3, 7.0, 0.30 ] )
      + eval_fun2( "gauss", logtgrid, logtgrid+0.1, [0.1, 7.7, 0.10 ] );
norms /= sum( norms ) ;   %  normalize the weights...
norms = 1.e-14 * 1.e54 / ( 30^2 * 1.1965e38 ) ; % and scale them via 1.e-14*VEM/(4*PI*d^2)

Plas = default_plasma_state ;     % get a copy of the plasma state variable...
Plas.norm = norms;                % and set it's fields to the model
Plas.temperature = tgrid ;
Plas.elem = [ Ne, Fe ] ;          % suppose abundance of Ne where high,
Plas.elem_abund = [ 1.5, 0.5 ] ;  %   and Fe low...

create_aped_fun( "plas", Plas ) ; % This defines your model fit function ...
fit_fun( "plas(1)" );             % ... and this makes it the current fit fun
```

[1]We will omit variable declarations in the S-Lang code examples; declarations are typically required in evaluated scripts, but are not in ISIS interactive mode. The % designates the start of a comment. The full script can be obtained from `http://space.mit.edu/%7Edph/isis_demo_line_ratios.sl`.

Create Fake Data

A simulated data set can be created by loading responses ("ARFs" and "RMFS") for the instrument of interest. You can set the exposure by overwriting the ARF's exposure. Then you assign the responses to an unused spectral histogram index, and then `fakeit()`, and you have your simulated counts. As a simple example, suppose you have no data or responses loaded, and you want to simulate MEG negative first order (see Figure 1):

```
load_arf( "MEG_-1.arf" ) ;          % HETGS MEG -1 effective area
load_rmf( "MEG_-1.rmf" ) ;          % HETGS MEG -1 redistribution (LSF)
set_arf_exposure( 1, 100000 ) ;     % Go deep!  100ks
assign_rsp( 1, 1, 1 ) ;             % ( arf_idx, rmf_idx, data_idx)
fakeit;
```

Fit line fluxes with a parametric model

To measure the line fluxes in our simulated model, we can now specify a parametric gaussian model which will be folded through the instrumental response. We will give the example for one line, the 18.627 Å O VII HeLβ line, fit with a Gaussian plus constant continuum. The width will be left at an infinitesimal value, since instrumental broadening dominates.

```
fit_fun( "gauss(1) + poly(1)" ) ;    %  specify a parametric line model

%         index or name      value  flag  min     max
set_par( "gauss(1).area",    1.e-4,  0,   0.0,    0.01  ) ; % set parameters
set_par( "gauss(1).center", 18.627,  0,   18.61,  18.64 ) ;
set_par( "gauss(1).sigma",   0.001,  1,   0.0001, 0.02  ) ;
set_par( "poly(1).a0", 3.e-4, 0, 0.0, 1.e-2 ) ;  % continuum, constant term
set_par( [5,6], 0, 1 ) ;   % linear, quadratic terms frozen to 0.0

ignore( all_data ) ;          % ignore everything
group_data( 3, 2 ) ;          % apply some grouping to histogram 3; by 2 bins.
xnotice( 3, 18.55, 18.9 ) ;   % notice 3 (as if 4 orders were loaded: H-1 H+1 M-1 M+1)
() = fit_counts( ) ;          % Do the fit

c0b = get_par( "gauss(1).area" ) ;          % get the flux value for use later.
( c1b, c2b ) = conf( "gauss(1).area", 0 ) ; % compute and save 1\sigma confidence
```

After doing this for both lines, you can compute the ratio, $r = c0b/c0a$, and also do the error propagation to get an uncertainty. (See Figure 1 for simulation and as fits.)

Plasma diagnostics, truth and fiction

The He/H ratio is temperature sensitive, so can be used as a temperature diagnostic. However, we know *a priori* that the simulated He/H line ratio we computed was from a simulation with a broad temperature distribution. Hence, we might doubt the meaning of *a temperature* derived from the line ratio. Let's take a look. We have the database and can easily examine line emissivities and isothermal ratios. We also have our assumed emission measure distribution and can look at the weighted emissivities. Here are a few key ISIS operations to support this:

```
% Determine the unique trans() levels for each line by selectively listing the database:
page_group( brightest( 3, where( el_ion(O, [7,8] ) and wl( 18.6, 18.97))));
#  index        ion   lambda    F (ph/cm^2/s)  A(s^-1)  upper lower   label
   2189 *     O VII    18.627    8.914e-05   9.333e+11    13      1 1s3p^^1P_{1} - 1s^2^^1S_{0}
 309111 *    O VIII    18.932    2.412e-06   0.000e+00  10037    17
   2373 *    O VIII    18.967    7.166e-04   2.525e+12     4      1 2p^^2P_{3/2} - 1s^^2S_{1/2}
```

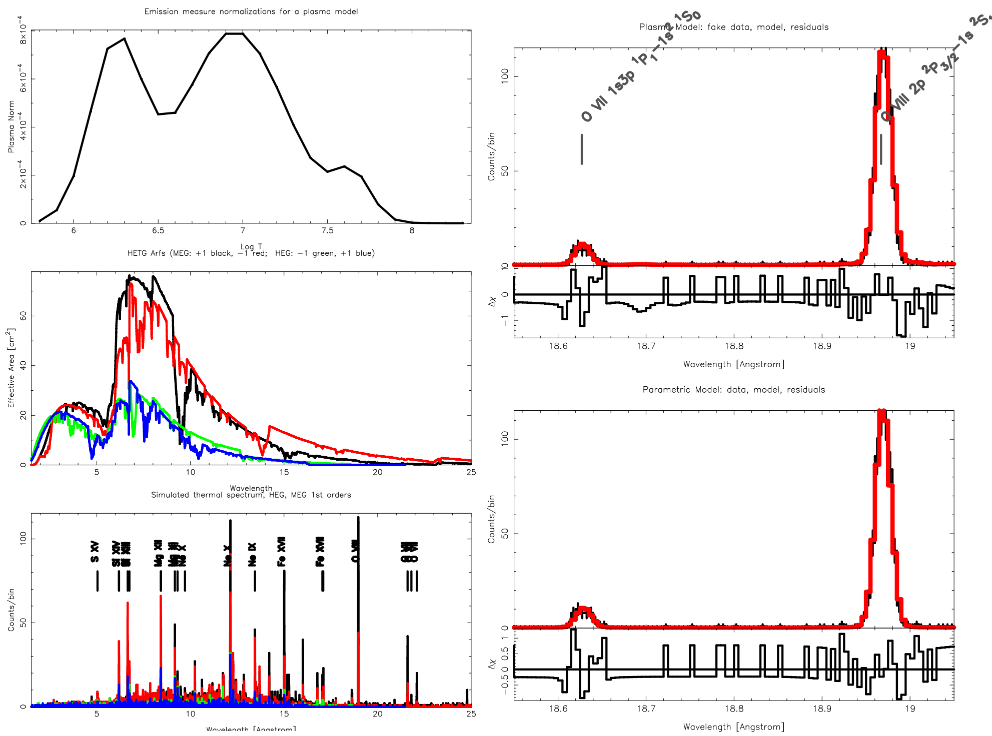

Figure 1: *Left: Model emission measure distribution, effective area curves, and simulated spectra for HETGS first orders. Right: The O* VII*-O* VIII *region as the plasma simulation (top) and as the parametric fit (bottom). Subtle differences can be seen in the residuals.*

```
l1 = where( trans( O, 7, 13, 1 ) ) ;    %  get the line index for O7 18.627A
l2 = where( trans( O, 8, [4,3], 1 ) ) ; %  get the line indices for O8 18.9A
rth = ratio_em( l1, l2, tgrid ) ;       % compute theoretical ratio vs T

xlimits; ylimits( 1.e-3, 1 ) ; xlog; ylog ; % set up to plot...
plot( tgrid, rth ) ;                        % and plot the theoretical ratio

fHeLb = line_info( l1[0] ).flux ;     % get plasma model fluxes  for the lines
fHLa  = line_info( l2[0] ).flux + line_info( l2[1] ).flux ;

eHeLb = line_em( l1, tgrid ) ;     % get the emissivities for each line.
eHLa =  line_em( l2, tgrid ) ;

yrange;                               % plot the normalized emissivities
plot( tgrid, eHLa / max(eHLa) );
oplot( tgrid, eHLb / max( eHLb ) ) ;
```

Figure 2 shows the theoretical ratio vs temperature, the normalized line emissivities, the emission measure distribution, and the emission measure weighted emissivities (whose integral is proportional to the flux). The vertical gray lines delimit the "temperature" derived from the value of the theoretical isothermal ratio and its uncertainty. Note that this "temperature" is actually nearer a local minimum in the emission measure than one of its peaks.

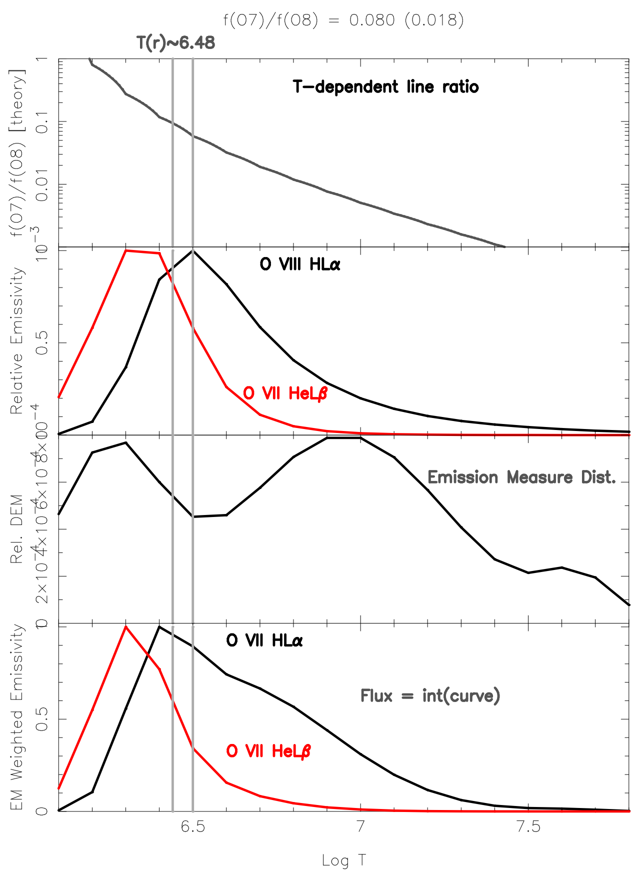

Figure 2: *(top to bottom) The theoretical ratio vs temperature; the normalized line emissivities; the emission measure distribution; and the emission measure weighted emissivities (whose integral is proportional to the flux). The vertical gray lines delimit the "temperature" derived from the value of the theoretical isothermal ratio and its uncertainty.*