```
;+
; NAME:
;   synthspec
;
; PURPOSE:
;   Construct synthetic spectrum from eigen-templates.
;
; CALLING SEQUENCE:
;   synflux = synthspec(zans, [ loglam=, hdr=, eigendir= ])
;
; INPUTS:
;   zans       - Structure(s) with redshift-fit information
(from SPREDUCE1D).
;
; OPTIONAL KEYWORDS:
;   loglam     - Log-10 wavelengths at which to synthesize
the spectrum;
;                this can either be a single vector if all
spectra have
;                the same wavelength mapping, or an array of
[NPIX,NOBJ].
;   hdr        - If specified, then use this header to
construct LOGLAM.
;                Either LOGLAM or HDR must be specified.
;   eigendir   - Directory for EIGENFILE; default to
$IDLSPEC2D/templates.
;
; OUTPUTS:
;   synflux    - Synthetic spectra
;
; COMMENTS:
;   The sub-pixel shifts are applied with the COMBINE1FIBER
procedure.
;
; EXAMPLES:
;
```

```
; BUGS:
;
; DATA FILES:
;   $IDLSPEC2D_DIR/etc/TEMPLATEFILES
;
; PROCEDURES CALLED:
;   combine1fiber
;   concat_dir()
;   poly_array()
;   readfits()
;   sxpar()
;
; REVISION HISTORY:
;   20-Aug-2000  Written by D. Schlegel, Princeton
;------------------------------------------------------------
------------------
function synthspec, zans, loglam=objloglam, hdr=hdr,
eigendir=eigendir

   ;----------
   ; If ZANS is an array, then call this routine recursively

   nobj = n_elements(zans)
   if (nobj GT 1) then begin
      for iobj=0, nobj-1 do begin
         ndim = size(objloglam, /n_dimen)
         if (ndim EQ 1) then thisloglam = objloglam $
          else if (ndim EQ 2) then thisloglam =
objloglam[*,iobj]
         newflux1 = synthspec(zans[iobj], loglam=thisloglam,
hdr=hdr, $
          eigendir=eigendir)
         if (iobj EQ 0) then newflux =
fltarr(n_elements(newflux1), nobj)
         newflux[*,iobj] = newflux1
      endfor
```

```
      return, newflux
   endif

   ;----------
   ; Get name of template file

   if (n_elements(eigendir) EQ 0) then $
    eigendir = concat_dir(getenv('IDLSPEC2D_DIR'),
'templates')
   tfile = strtrim(zans.tfile,2)

   ;----------
   ; Determine the wavelength mapping for the object spectra,
   ; which are the same for all of them.

   if (keyword_set(objloglam)) then begin
      naxis1 = n_elements(objloglam)
      objdloglam = objloglam[1] - objloglam[0]
   endif else if (keyword_set(hdr)) then begin
      naxis1 = sxpar(hdr, 'NAXIS1')
      objloglam0 = sxpar(hdr, 'COEFF0')
      objdloglam = sxpar(hdr, 'COEFF1')
      objloglam = objloglam0 + dindgen(naxis1) * objdloglam
   endif else begin
      print, 'Either LOGLAM or HDR must be specified'
      return, -1
   endelse

   ;----------
   ; If no template file, then return all zeros.

   if (tfile EQ '') then $
    return, fltarr(naxis1)

   ;----------
   ; Read the template file, and optionally trim to only
```

```
those columns
   ; specified by COLUMNS.
   ; Assume that the wavelength binning is the same as for
the objects
   ; in log-wavelength.

   starflux = readfits(djs_filepath(tfile,
root_dir=eigendir), shdr, /silent)
   if n_elements(starflux) LE 1 then begin
     polyflux = poly_array(naxis1,zans.npoly) #
zans.theta[0:zans.npoly-1]
     bspline_set = mrdfits(djs_filepath(tfile,
root_dir=eigendir), 1, shdr, /silent)
     rloglam = objloglam - alog10(1+zans.z)

     newflux = bspline_valu(rloglam, bspline_set,
x2=objloglam) * polyflux
     return, newflux
   endif


   starloglam0 = sxpar(shdr, 'COEFF0')
   stardloglam = sxpar(shdr, 'COEFF1')

   ndim = size(starflux, /n_dimen)
   dims = size(starflux, /dimens)
   npixstar = dims[0]
   if (ndim EQ 1) then nstar = 1 $
    else nstar = dims[1]

   icol = where(zans.tcolumn NE -1, ncol)
   starflux = starflux[*,zans.tcolumn[icol]]

   ;----------
   ; Add more eigen-templates that represent polynomial
terms.
```

```
   if (keyword_set(zans.npoly)) then $
    starflux = [ [starflux],
[poly_array(npixstar,zans.npoly)] ]

   ;----------
   ; Construct the synthetic spectrum as a linear combination
of templates

   synflux = starflux # zans.theta[0:ncol-1+zans.npoly]

   ;----------
   ; Apply redshift...

   starloglam = starloglam0 + dindgen(npixstar) * stardloglam
   combine1fiber, starloglam + alog10(1+zans.z), synflux, $
    newloglam=objloglam, newflux=newflux

   return, newflux
end
;------------------------------------------------------------
-----------------
```