QFitsView DPUser

Setting Up QFitsView DPUSER Library

1. Place all libraries (files with “lib_*.dpuser” name) in a convenient location (e.g. “/Users/*user*/DPUser/Functions”)
2. Place “startup.dpuser” in “/Users/*user*/DPUser/”
3. Create a directory under root “/dpuserlib”
4. Make a file in that directory “startup.dpuser” - this will be automatically run when you start QFitsView.
5. The file consists of a single line:
   `@/Users/user/DPUser/startup.dpuser`
   This runs all libraries to make the functions available to QFitsView - you will see a whole bunch of “Stored function…” and “Stored procedure…” plus “Finished General Functions”

Global Variables

```
c           =     299792458.0
pi          =     3.14159
e           =     2.71828
naxis1      =     256
naxis2      =     256
plotdevice  =     /XSERVE
method      =     0
tmpmem      =     20971520
```

Data types - "cube" is 3D, "image" is 2D, "spectrum" is 1D, "data" is 1, 2 or 3D

Libraries

lib_wcs.dpuser - Transform to and from World Coordinate Systems and Pixels
function get_WCS_data, inbuff, axis - return array [CRVAL, CRPIX, CDELT] for *axis* (1,2 or 3)
procedure set_WCS_data, inbuff, wcs, axis - sets WCS values for *axis* (1,2 or 3)
function cvt_pixel_WCS, pix, cpix, cval, cdelt - convert pixel to WCS coords
function cvt_WCS_pixel, value, cpix, cval, cdelt - convert WCS coord to pixel
function cvt_WCS_pixel_data, inbuff, value, axis - converts pixel to WCS for data for axis
function cvt_pixel_WCS_data, inbuff, value, axis - converts WCS to pixel for data for axis
function WCS_range, inbuff, p1, p2, axis, prnt - returns WCS coordinates as range [*w1,w2*] from pixel values [*p1,p2*] for axis on inbuff, if *prnt*=1, then print range
function pixel_range, inbuff, w1, w2, axis, prnt - returns pixel values as range [*p1,p2*] from WCS co-ordinates [*w1,w2*] for axis on inbuff, if *prnt*=1, then print range
function set_WCS_default, inbuff - checks inbuff has minimal WCS keys set (to 1 by default)
function get_WCS_values, inbuff - create WCS array [1,cv,cd] from inbuff data array

lib_general.dpuser
function indexreform, index, xsize, ysize, zsize - returns 3D co-ords from 1D index, given

dimensions *xsize, ysize, zsize*. Values returned as array.
function lognan, inbuff - set log of data, setting zero and Nan values to Nan
function clipnan, inbuff, low, high - set values outside range [*low..high*] to Nan
function axiscentroids, inbuff, axis - returns centroids of each image row/column, row-*axis*=1, column-*axis*=2; used e.g. for finding centroids of pv diagram
function myhist, inbuff, low, high, bin, norm - create a histogram from *inbuff* data (any dimensions), from low to high values in *bin* bins. Histogram is normalised if *norm*=1. Output x-axis values are set to range.

lib_cube.dpuser - Data cube functions
function cube_trim_xy, cube_in, x1, x2, y1, y2- sets *cube* to zero x<*x1*, x>*x2*, y<*y1*, y>*y2* (cblank cube first)
function cube_trim_wl, cube_in, l1, l2, value, trim- sets *cube* to *value* (usually zero) for w<*w1*, w>*w2* in axis 3 using WCS (cblank cube first). If *trim*=1, then truncate the cube outside the wavelength range.
function cube_spectrum_mask, cube, mask, level - mask *cube* on spectral wavelength with "*mask*" (pixel pairs), set masked pixels to "*level*"
function cube_clip, cube_in, lvl, thresh, mask - clips *cube* <0 and > *lvl* in image (x-y) plane, does dpixapply using threshold, cube is spectrally masked
function cube_clip_y, cube_in, lvl, thresh - as above, but in the x-z plane
function cube_interp_z, cube_in, x1, x2, y1, y2, z1,z2 - interpolate in image plane over rectangle [*x1:x2,y1:y2*] in each of wavelength plane *z1:z2*
function cube_interp_x, cube_in, x1, x2, y1, y2, z1,z2 - interpolate in image plane over rectangle [*y1:y2,z1:z2*] in each of spatial range *x1:x2*
function cube_interp_y, cube_in, x1, x2, y1, y2, z1,z2 - interpolate in image plane over rectangle [*x1:x2,z1:z2*] in each of spatial range *y1:y2*
function cube_interp_xy, cube_in, x1, x2, y1, y2, z1,z2 - as above, but interpolate over wavelength *z1:z2* xy plane
function cube_set_value, cube_in, x1, x2, y1, y2, z, xv, yv - set rectangle [*x1:x2,y1:y2*] at image plane *z* to value at [*xv, yv*]
function cube_pixfix_xy, inbuff, pixfixdata, n - fix cube using cube_interp_.. functions, n sets, *pizfixdata* are [*x1,x2,y1,y2,z1,z2,method*] where method="x"/"y"/"xy"
function cube_single_pixel_fix, inbuff, x, y - cube_interp_xy for all z axis for single spaxel
procedure cube_bit_nan, inbuff,x,y - set spaxel [*x,y*] to 0/0 along whole cube
function cube_clean_dpix, inbuff, divisor - Clean cube *inbuff* by dpixcreate/apply, *divisor* = scale set from maximum of median image
function cube_resize_center, inbuff, xcen, ycen, xsize, ysize - resize cube (*xsize,ysize*) and center on pixel [*xcen,ycen*]
function cube_shift_xy, inbuff, xshift, yshift - sub-pixel shifts cube by [*xshift, yshift*]
function cube_redisp, cube, val_new, delt_new, n_new - change the wavelength dispersion of cube (axis 3) to new range defined by *val*, *delt* and *n* by interpolation.
function cube_symm_flip, cube, lambda, width, part - symmetrically flip cube about wavelength "*lambda*", "*part*"=0 (left) or 1 (right), trims cube to *lambda+-width*
function cube_rotate, inbuff, xcen, ycen, platescale, rot_angle - rotate cube on center [*xcen,ycen*] by *rot_angle*, setting *platescale* in arcsec/pixel.
function cube_centroids, inbuff - get centroids at each wavelength pixel
function cube_cont_slope, inbuff, mask - returns image with continuum slope, masked by wavelength pairs
function cube_spectrum_subtract, inbuff, spectrum - subtract spectrum for each spaxel
function cube_spectrum_divide, cube, spec - divide cube by spectrum (e.g. telluric correction)

function cube_spectrum_multiply, cube, spec - multiply cube by spectrum
function cube_set_pixlayers, inbuff, pixl, pix1, pix2 - set cube layers [*pix1,pix2*] to the values for layer *pixl*
function cube_wavelength_correct, cube, correction - correct wavelength solution at each spaxel by "*correction*" values (in wavelength)
function cube_to_2d, inbuff - Convert data cube to 2d apertures for IRAF
function cube_set_flags_nan, cube, layer - set up flags image for *cube_interp_flags*, from a data *cube* (e.g. a velmap) from l*ayer*. This sets 1 where pixel in "NaN", 0 else.
function cube_interp_flags, cube, flags, xi1, xi2, yi1, yi2, dmax - interpolate over pixels in *cube* where *flags* is set to 1, 0= good values to use for interpolation. [*xi1:xi2, yi1:yi2*] is region to interpolate (*xi1* = 0 - do whole area). *dmax* is maximum distance from "good" pixels.
function cube_deslope, inbuff, mask, flag - deslope *cube* for each spectrum using "spectrum_deslope"
function cube_clean_pixels, inbuff, layer, npix - Remove singleton pixels surronded by Nan's, Opposite of "cube_interp_flags", used to clean up boundaries etc. *npix* is max number of good pixels around each pixel before blanking.
function cube_radial_spectrum, cube, xc, yc, rstep, nstep, ann - Radial spectra of cube, centered [*xc,yc*] radial steps *rstep*, number of steps *nstep*. If *ann*=1, output annular spectra
function cube_rebin, cube, psize - Rebin cube to pixel size psize (arcsec). Uses "interpolate" function. Useful for e.g. KCWI data which has rectangular spaxels on the sky.
function cube_from_image_spectrum, image, spectrum - Creates a cube from an image and spectrum. Wavelength axis of cube is spectrum scaled by image value

lib_image.dpuser - Data image functions
function image_erodenan, inbuff - erode image, pixels set to Nan if any neighbour is Nan
function image_smooth, inbuff, smooth - smooth image with NaN values - *smooth* integer=boxcar, non-integer=gaussian
function image_interp_x, image_in, x1, x2, y1, y2 - as for cube_interp_xy, but for single image
function image_interp_y, image_in, x1, x2, y1, y2 - as for cube_interp_x, but for single image
function image_interp_xy, image_in, x1, x2, y1, y2 - as for cube_interp_y, but for single image
function image_from_profile, profile, xp, yp, xc, yc - create 2D image from 1D profile, *xp, yp* size of output image, [*xc, yc*] - center of rebuilt profile
function image_bfilter, image, order, cutoff - Butterworth filter an image, assume square image, filter order =*order*, *cutoff*=Nyquist cutoff (0-1)
function image_enclosed_flux, inbuff, xc, yc, r, smth - Get enclosed flux within radius *r* from [*xc, yc*] (pixels). If *smth*>0, Gaussian smooth the output
function image_avg, image, x, y, s - average value of image in square aperture [*x,y*] +-*s* pixels
function image_structure, image, psf_image - Structure map = *image*/(*image* x *psf*) x *psf*^T, "x"=convolution, "^T" = transpose
function image_interp_flags, image, flags, xi1, xi2, yi1, yi2, dmax - Interpolate over flagged spaxels, *flags* - 2D data with same x/y axes size as image, with value=1 to be interpolated, value=0 - good pixels, [*xi1:xi2, yi1:yi2*] - co-ordinate range to interpolate over. If not input, then do all spaxels. *dmax* - maximum pixel distance for interpolation (=0 don't test)

function image_cut, inbuff, x, y, a - do twodcut at [*x,y*] angle *a* and reset WCS correctly.

lib_spectrum.dpuser - Spectrum functions
function spectrum_make_disp, val, delt, pix, n - make 1D vector over range defined by WCS *val*, *delt*, *pix*, *n*.
function spectrum_make_disp_data, inbuff, axis - make 1D vector over range defined by WCS values from data axis (1,2 or 3)
function spectrum_make_disp_n, val1, val2, n - make 1D vector over range val1-val2, number of points n
function spectrum_mask, inbuff, mask, value, flag - spectrum *inbuff* set to *value* between pixel pairs in *mask*. Works for 1D or 3D, assuming last axis is spectrum. *flag* =0, mask is in pixels, =1, mask is in wavelength
function spectrum_cont_slope, spec, mask, flag - continuum slope of spectrum *spec*, masked by wavelength pairs/flag (as for "spectrum_mask")
function spectrum_deslope, spec, mask, flag - deslope spectrum, using "spectrum_cont_slope" and mask/flag
function spectrum_polyfit, inbuff, order, mask, flag - fit polynomial of order to a spectrum *inbuff* with mask/flag pixel/wavelength set, returns n x 3 array, 1st row=original data masked, 2nd row=polynomial fit, 3rd row = residual
function spectrum_symm_flip, inbuff, lambda, part - split spectrum *inbuff* at wavelength *lambda*, flip and add, taking left (*part*=0) or right (*part*=1) sections
function spectrum_wave_to_lambda, inbuff - convert wavenumber spectrum *inbuff* to wavelength (nm) with same axis length
function spectrum_make_gauss, inbuff, bi, bs, h, c, w - make spectrum with gaussian from *inbuff* WCS. *bi*, *bs* - base intercept and slope, *h* - height, *c* - center, *w* - FWHM (creates artificial emission line)
function spectrum_redisp, inbuff, data, daxis, zero, norms - re-disperse a spectrum *inbuff* to the same wavelength scale/range as *data* (with wavelength axis *daxis*). If *zero* is >0, all pixels outside of the wavelength range of *inbuff* are set to zero. If *norms* is >0, the output is normalised.
function spectrum_from_xy, inbuff - re-disperse spectrum from 2D x and y bintable to wavelength range and same number of points.
function spectrum_from_data, xdata, ydata, w1, w2, delt - re-disperse spectrum from 2D x and y data to wavelength range (*w1, w2*) and same number of points (from *w1, w2* and *delt*)
function spectrum_interp, inbuff, x1, x2 - Smooth over bad pixels [*x1:x2*]

lib_io.dpuser - Input/output to and from text and fits files
function io_text_FITS_1D, inbuff - converts text, format of "wavelength, data" to spectrum fits data, setting WCS values. Assumes wavelength is evenly spaced.
function io_text_FITS_3D, inbuff, nx, ny, nz, blank - converts fits data *inbuff*, format of "i,j,v1,v2..." to fits data cube size [*nx,ny,nz*]. Default value for resulting cube is *blank* (e.g. 0 or 0/0) - can have missing [i,j].
function io_text_FITS_interp, fname, xstart, xdelta, xnum, xscale, yscale, ignore - converts text from file *fname*, format of "wavelength, data" to spectrum fits data, setting WCS values. The values are interpolated to the range defined by *xstart*, *xdelta* and *xnum*. Wavelength and data value are scaled by *xscale*, *yscale* (default 1). *Ignore* lines at the start are skipped (e.g. column headers).
function io_FITS_text_1D, inbuff, prefix, cutoff - converts spectrum to text, CSV format, line 1 = "*prefix*_Wavelength, *prefix*_Counts". Values below *cutoff* (non-zero) are set to "NaN" (Be aware of QFitsView **Edit > Copy** functionality)

function io_FITS_text_2D, inbuff, prefix - converts image to text, CSV format, line 1 = "*prefix*_Wavelength, *prefix*_Flux_1, *prefix*_Flux_2 …. "
procedure io_FITS2TXT_1D, filename, cutoff - converts 1D FITS to text file *fname* assuming file is in working directory - output is same as input file with ".txt" type
procedure io_FITS2TXT_2D, filename - as above but for image (2D) file
function io_cube_from_xyz, cube, data, n - make a cube from imported *data*, *cube* is template, resized to *n* on axis 3, first 2 values in *data* are x,y co-ords, rest are values along z axis
function io_import_TXT_1D, name - import data from file *name* in text format

lib_masking.dpuser - Masking functions for images and cubes
function mask_from_image, inbuff, level, low - create a mask from data *inbuff*, setting to 1 if > *level*, to *low*(usually 0) if <*level*
function mask_from_image_nan, inbuff - create a mask from data *inbuff*, setting to 1 if value<>Nan
function mask_data, inbuff, level, low - masks data *inbuff*, setting to *low* if < *level*
function mask_data_median, inbuff, level, low - as above, but sets data *inbuff* > *level* to median of data
function mask_circle, inbuff, x, y, r, v, rev - masks image/cube with circle center [*x*,*y*] radius *r*, set masked-out value to *v* (default 0). If *rev*<>0, reverse mask.
function mask_set_nan_min, inbuff, minvalue - set *inbuff* values to *minvalue* if data = Nan, if *minvalue* is zero, use the current minimum value
function mask_cone, inbuff, xc1, yc1, xc2, yc2, pa, beta, mask - mask cone area, equator [*xc1, yc1*], [*xc2, yc2*] (can be same coordinate), centerline angle *pa*, internal full-angle *beta*. If *mask*=0, return the mask, if *mask*=1, return the masked input data
function mask_line, inbuff, x1, y1, x2, y2, side - masks an image on one side of a line [*x1, y1*], [*x2, y2*]. *side* =0 for left, =1 for right side of line

lib_velmap.dpuser - Velocity map (velmap) extension functions
function velmap_std_to_ext, velmpstd, r, cmin, vmethod, vcenter, vx, vy - convert standard QFitsView velmap *velmpstd* to extended form, *r*=instrumental resolution, *cmin*= minimum continuum value - output is xtended velmap format - see below
vmethod=0 - median
vmethod=1 - average
vmethod=2 - flux-weighted average
vmethod=3 - manual (*vcenter* value)
vmethod=4 - pixel ([*vx,vy*] is set to zero)
function velmap_vel_center, velmap, vmethod, vcenter, vx, vy - returns the wavelength value from the *velmap* cube, using the methods as above
function velmap_vel, velmap, vmethod, vcenter, vx, vy - returns the velocity map from the *velmap* cube, using the methods as above
function velmap_vel_set, velmap, vmethod, vcenter, vx, vy - Fix extended *velmap* velocity as per velmap_std_to_ext (re-do extended velmap cube)
function velmap_rescale, inbuff, scale - rescales extended *velmap* flux data (e.g. flux calib change)
function velmap_fix, inbuff, contlo, conthi, flo, fhi, vlo, vhi, wlo, whi, setvalue - clean up velmap *inbuff* (either standard or extended form), setting values out of range to *setvalue*. Value ranges
contlo, conthi - continuum
flo, fhi - flux
vlo, vhi - wavelength

wlo, whi - fwhm
setvalue - value to set where spaxel is out of range (usually 0/0)
function velmap_extcorr, velmap, av, lambda - extinction correct velocity map at wavelength *lambda* (in nm), *av*=extinction A_V
function velmap_extcorr_map, velmap, extmap, lambda - as above, but *extmap* is a map of extinction values
function velmap_fix_interp, velmp, npix - interpolate velmap *velmp* missing values, missing is Nan in continuum (usually after "velmap_fix"). *npix* is interpolation width maximum
function velmap_clean_map_wvt, velmp, map, nregion - Clean up velmap *velmp* based on WVT *map region* number, setting region pixels to NaN
function velmap_mask, velmap - set *velmap* to Nan where continuum=0
procedure velmap_comps, velmap, prefix, hmax - Output velmap components, to the current working directory. *prefix* sets file names, terminated with _Flux/_Flux_Norm/_Vel/_EW/_Sig/_VelHist/_SigHist - histograms produced if *hmax*>0, velocities -hmax -> hmax, dispersion 0->hmax, 50 bins

Standard VELMAP Procedure

- Create QFitsView velmap with wavelength, fwhm estimate
- Examine velmap for continuum, height, wavelength and fwhm "sensible" ranges
- Use "velmap_fix" to clean up velmap
- Use "velmap_fix_interp" to interpolate over NaN values (if required)
- Use "velmap_std_to_ext" to create extended velmap format

Extended VELMAP Format
1 = Continuum
2 = Peak height above continuum
3 = Wavelength
4 = FWHM
5 = e_Continuum
6 = e_Peak
7 = e_Wavelength
8 = e_FWHM
10 = Velocity (zero-point calculated/set by "method" parameter)
11 = Dispersion (sigma) velocity, corrected for spectral resolution
12 = Flux (Peak*FWHM*1.0699)
13 = Equivalent width (flux/continuum)
14 = Support (√V^2+σ^2)
15 = Order vs turbulence (|V/σ|)

lib_chmap.dpuser - Channel and position/velocity map functions
function chmap_create, cube, lambda_cent, lambda_width, cutoff, width_factor, smooth - make a channel map from the *cube*
lambda_cent - estimate of central wavelength
lambda_width - estimate of FWHM
threshold - % of maximum for cutoff
width_factor - wavelength widow (multiple of *lambda_width*)
smooth - integer=boxcar, non-integer=gauss, 0=no smoothing
Returns cube, axis 3 in velocity difference from median. Spaxel values are FLUX (not flux

density) in that channel
function chmap_rebin, inbuff, lnew, velwidth, sm, minval- rebin channel maps into *lnew* bins between *v1* and *v2* (usually symmetric about 0, but not necessarily). *sm* smoothing, integer=boxcar, non-integer=gauss, 0=no smoothing, set output to NaN where < *minval*
procedure chmap_comps, inbuff, dirout, fnameout - splits channel map into components and writes images to *dirout*, named *fname* plus velocity (e.g. "/users/mdurre/data/chmaps/ic630_pa_beta-450.fits", "/users/mdurre/data/chmaps/ic630_pa_beta+100.fits")

Standard CHMAP Procedure

- Create basic channel map using "chmap_create" (usually do not smooth)
- Rebin to required # of channels (e.g. 9 or 16) using "chmap_rebin" (smoothing if required)
- Output individual channel maps using "chmap_comps"

lib_pv.dpuser - Position Velocity Diagram functions
function pv_array, cube, ystart, wslit, nslit, lcent, lwidth - create pv diagram from cube parallel to x axis, *ystart* - y pixel to start, *wslit* - slit width, *nslit* - number of slits, extract over range *lcent-lwidth/lcenter+lwidth*
function pv_single, cube, xc, yc, angle, width, lcent, vwidth, npix, cont_flag - extract single PV plot at *xc/yc/angle/width* - centerered on *lcent vwidth* - velocity width around lcent, rebinned in velocity to *npix* - *cont_flag*=1 subtract continuum (flux) =2 divide continuum (EW) =0 don't remove continuum
function pv_ratio, cube, xc, yc, angle, width, lcent1, lcent2, vwidth, npix - create PV diagram for ratio of 2 lines *lcent1*, *lcent2*
function pv_meddev, image - divide image by median along x axis (useful for EW for PV diagrams)

lib_wvt.dpuser - Weighted Voronoi Tesselation functions
function wvt_cube, cube, sn_target - make wvt cube using noise in each spaxel. Bad pixels where S/N is > 10x brightest pixel S/N
function wvt_cube_mask, cube, l1, l2, mask, cutoff, sn1, sn2 - make WVT cube using 2 S/N ratios, within or without mask. Returns WVT applied to *cube*
l1, l2 - wavelength range to use for signal and noise determination ("quiet" part of spectrum with no emission lines)
mask - if 2D mask, use this. If mask=0, use "cutoff" to determine mask
cutoff - percentage of peak maximum for mask level
sn1, sn2 - S/N ratios for inside/outside mask. If *sn2*=0, just use *sn1* over whole cube
function wvt_sn_mask, cube, l1, l2, mask, cutoff, sn1, sn2 - as above, except returns WVT image data:

1 Signal
2 Noise
3 S/N
4 Mask
5 Signal binned
6 Signal bin map
7 Bin density (1=maximum - smallest bins , 0=minimum - biggest bins)

function wvt_build_from_map_cube, inbuff, wvtmap, prnt - make WVT cube from *inbuff* and *wvtmap*. If *prnt* =1 print diagnostic every 100 regions

function wvt_build_from_map_image, inbuff, wvtmap - make WVT image from *inbuff* and *wvtmap*.
function wvt_velmap, velmp, layer, sn - make WVT velmap from standard or extended velmap *velmp*, *layer* is either 0=continuum, 1=flux, *sn*=S/N target
function wvt_density, wvt_map - make map of region density, i.e. 1/# of pixels in region. *wvt_map* is WVT with /map flag.
function wvt_cube_to_specarray, inbuff, wvt_map, nrm, prnt - convert cube *inbuff* to spectrum array, using *wvt_map* regions. If *nrm* = 1, divide each spectrum in array by the first one. If *prnt* = 1, print running diagnostics
function wvt_specarray_to_cube, inbuff, wvtmap - reverse of “wvt_cube_to_specarray"

lib_astro_general.dpuser - General astrophysics functions
function redshift_data, inbuff, z - redshift data by “z”, assuming last axis is wavelength. WCS values set
function bb_make,t, l1, l2, npix, flag - make black-body function at temparture t, wavelength range *l1* to *l2*, number of pixels *n*, *flag*=0, wavelength in A, =1=> nm, =2 => um
function bb_make_log, t, l1, l2, npix, scale, cutoff - make bb at temp t over log wavelength [l1,l2] (in log meters), creating spectrum length npix, multiply wavelengths by scale (to convert to e.g. nm), set result to Nan where below cutoff. Returns
function bb_div, inbuff, temp - divide spectrum by black-body at temperature t
function extinction_calc, f1, f2, l1, l2, rat, galext, s, flmin1, flmin2 - create an extinction map from 2 emission line maps - f1, f2 are flux maps, l1, l2=wavelengths, rat=expected flux ratio, galext=galactic extinction, smth= smooth pixels, flmin1, flmin2=minimum flux value for each map - calculates the extinction constant (CCM laws for IR and optical)
function extinction_correct, inbuff, av - correct cube for extinction (av=single value for extinction) - wavelength from axis 3
function extinction_correct_map, cube, av - correct cube for extinction (av=extinction map) - wavelength from axis 3.
function extinction_correct_lambda, data, av, lambda - correct value/image for extinction av at wavelength lambda can be used on value or image
function flux_to_mag, inbuff, zpm, ssize, flag - convert flux image *inbuff* to mag, *zpm* is either zero-point magnitude (*flag*=0) or zero-magnitude flux (*flag*=1). *ssize* = pixel size in arcsec to convert to mag/arcsec^2

lib_astro_mapping.dpuser - Astronomy functions (mapping)
function map_compare, inbuff, inbuff2, prefix, exflag - pixel-by-pixel comparison data (from each layer of cube *inbuff*) for export to plot. Outputs x,y,p1,p2.... First line of output text is "descriptor `X` `Y` `"+ *prefix*+"_1` `"+ *prefix*+”_2`" +...- suitable for import into Veusz. *exflag* - 1=generates output text file “prefix”.txt
function map_compare_diagram, inbuff1, inbuff2, min1, max1, min2, max2, nbin, lgaxes - Map diagram density plot. *inbuff1, inbuff2* - value maps, x and y axes. *min1/2, max1/2* - min and maximum values for axes 1/2. *nbin* - no of bins on each axis. *lgaxes* - 1=plot in log space (min,max must be in log values)
function map_compare_pos, inbuff1, inbuff2, inbuff3, inbuff4, x, y, boxsize - get 2 sets of map ratios at position [*x,y*], averaged over *boxsize* x *boxsize* pixels (e.g. excitation ratios at feature position)
function map_basis_distance, basex0, basey0,basex100,basey100, x1, x2, y1, y2, size - creates an image (dimensions *size*, limits (*x1, y1*), (*x2, y2*)) of distance from basis points [*basex0, basey0*] to [*basex100, basey100*] - for use in AGN mixing ratios for contour values.

function map_compare_basis, inbuff1, inbuff2, basex0, basey0, basex100,basey100,lgaxes - plots basis distance (AGN mixing ratio) from basis points [*basex0, basey0*] to [*basex100, basey100*] . *lgaxes* - 1=take log of *inbuff1, inbuff2* before calculation
function map_regime_ir, inbuff1, inbuff2, a1, a2, a3, b1, b2 - create position excitation map. If *a1*=0, use the standard Riffel 2013 excitation regimes. *inbuff1* is H_2/Br_gamma, *inbuff2* is [Fe II]/Pa_beta. Both in log values. Output values are SF=1, AGN=2, LINER=3, TO1=4, TO2=5

lib_astro_spectrum.dpuser - Astronomy functions (spectrum)
spec_fluxdens, inbuff, l1, l2, prflag - flux density (counts/nm) between *l1* and *l2* wavelength; data is spectrum (returns single value). If *prflag*<>0, print results as well.

lib_astro_image.dpuser - Image astrophysics functions
function img_aphot_annular, img, xcen, ycen, r, ib, ob - aperture photometry on image,centered on xcen,ycen; aperture r, background annulus from ib to ob (inner to outer boundary). If ib and ob are zero, set to r and 2*r
function img_apphot_simple, img, xcen, ycen, r, pixsize, scale - simple aperture photometry on image,centered on xcen,ycen; aperture r.

lib_astro_cube.dpuser - Cube astrophysics functions
function cube_apphot, cube, xcen, ycen, r1, r2, mx, my, mr - aperture photometry, centered on xcen,yceb; aperture r1, background annulus r2, mask out circle mx/my/mr (mx>0) - result is spectrumlib_clean.dpuser - Cube cleanup functions (mainly CR and bad pixels)
function cube_apspec, cube, ox, oy, or, bx, by, br - Get star spectrum from aperture using circular aperture and background, plus a mask circle.
Inputs - *cube* is 3D cube of star, used for both telluric and flux calibration. *xc, yc, r1* - center and radius of aperture, *r2* - radius of annulus (*r1->r2*) of background. *mx, my, mr* - center and radius of mask, if not required then mx=0. *mask* - any other mask required (2D fits)
Output is spectrum of aperture less average of background (with mask) - values <0 set to Nan
function cube_fluxdens, inbuff, l1, l2, prflag/function spec_fluxdens, inbuff, l1, l2, prflag - flux density (counts/nm) between *l1* and *l2* wavelength; returns image value. If *prflag*<>0, print results as well
function cube_sky_rem, cube_in, bckgnd_lvl - removes skylines. Takes background pixels as those with median value below *bkgnd_lvl*
function cube_sl_clean, inbuff, skyline_list, width - removes skylines using linelist, array of wavelengths - interpolated over wavelength±*width*
function clean_cube_bp_fix, cube, bp_cube - cleans *cube* based on *bp_cube* using dpixapply over x image slices
function clean_cube_bp_limits, cube, ll, ul - create bad pixel cube for input to “clean_cube_bp”, flagging pixels below *ll* and above *ul* values
function clean_cube_bp, cube, threshold - create bad pixel cube using *threshold* scanning over wavelength slices.

lib_all.dpuser
Runs all libraries