Centro de Astrofísica da Universidade do Porto
Universidade de Lisboa, CAAUL & LOLS
INAF, Osservatorio Astronomico di Trieste
INAF, Osservatorio Astronomico di Brera
Observatory of the University of Geneva
Physics Institute, University of Bern
Instituto de Astrofísica de Canarias
European Southern Observatory

ESPRESSO

Data Analysis Library Design

VLT-TRE-ESP-13520-0104, Issue 1.6

December 2016

Prepared	V. D'Odorico	Name	Date	Signature
Approved	P. Di Marcantonio	Name	Date	Signature
Released	F. Pepe	Name	Date	Signature

Change Record

Issue/Rev.	Date	Section/Page affected	Reason/Remarks
1.0	March 12, 2013	All	FDR version
1.1	October 28, 2013	Sects. 3.1, 3.2, 3.4 Sects. 4.2 Chapter 5 Chapter 8 Sect. 9.2 Chapter 11	Revised after FDR
1.2	March 18, 2014	Sects. 4.1, 4.2, 4.5, 4.11 Sects. 7.1.3, 7.2.1, 7.3.1, 7.3.2, 7.3.3 Chapter 8 Sects. 9.1, 9.2, 9.5, 9.11 Chapter 11	Revised after DAL delivery 0.2.0
1.3	October 18, 2014	Sects. 3.4 Sects. 4.1, 4.2, 4.11, 4.13 Sects. 7.1.3, 7.1.10 Sects. 8.1-8.6, 8.33-8.35 Sects. 9.1, 9.2, 9.11, 9.13 Chapter 11	Revised after DAL delivery 0.3.1
1.4	January 2015		Revised after DAL delivery 0.4.0
1.5	May 2015		Revised after DAL delivery 0.5.0
1.6	December 2016		Revised after DAL delivery 0.8.1

Co-authors:
Guido Cupani, Jonay González Hernández, Christophe Lovis, Sérgio Sousa.

Table of Contents

List of Figures

List of Tables

Chapter 1. Introduction

1.1 Scope of the Document

This document provides a technical description of the data analysis tools provided to ESPRESSO users to reach the scientific results from the reduced data.

1.2 Organization of the Document

This document was built according to the specifications from [AD-1]. For clarity, we have split the chapter on Functional and Workflows Description into two chapters, 4 and 5.
Chapter 1 gives the reference information and some introductory and working remarks. **Chapter 2** briefly describes the instrument science cases with a particular emphasis on the data analysis requirements for each case. In **Chapter 3**, we describe the mathematical details for the non standard and/or more complicated algorithms that are applied in the DAS recipes. **Chapter 4** reports the functional description of all recipes using functional diagrams to present their hierarchical structure and the decomposition in DAS functions. In **chapter 5**, the Reflex workflows for the quasar and star branches of the DAS are detailed, with an overview of the foreseen OCA rules that will be applied. **Chapter 6** gives a brief description of the input data to the DAS. This description is copied from [AD-2], since our input data are the output of the DRS pipeline. The structures of the input data, intermediate and final products of the DAS are presented in **chapter 7**. **Chapter 8** reports all the low-level functions of the DAS library in the order in which they appear in the recipes in chapter 4. The CPL plugins are described with the aim of the algorithm pseudo-code in **chapter 9**. **Chapter 10** is only a reference to the DAS Validation and Test document. In **chapter 11**, we report the foreseen development plan for the DAS. **Appendix A** is the list of defined QC1 parameters.

1.3 Applicable Documents

AD-1	Data Flow for VLT/VLTI Instruments Deliverables Specification	VLT-SPE-ESO-19000-1618	3	01.02.2011
AD-2	Data Reduction Library Design	VLT-TRE-ESP-13520-0102	1	25.03.2013
AD-3	Reflex User Manual	VLT-MAN-ESO-19000-5037	3.2	12.11.2012
AD-4	CPL Reference Manual	VLT-MAN-ESO-19500-2720	6.2	23.11.2012
AD-5	Data Analysis Library Validation and Test	VLT-PLA-ESP-13520-0105	1	25.03.2013
AD-6	ESPRESSO Statement of Work	VLT-SOW-ESO-13520-5059	1	01.02.2011
AD-7	ESPRESSO Technical Specifications	VLT-SPE-ESO-13520-4633	3	01.02.2011
AD-8	ESPRESSO Project Plan	VLT-PLA-ESP-13520-0007	5	25.03.2013
AD-9	ESPRESSO Management Plan	VLT-PLA-ESP-13520-0015	4	25.03.2013

1.4 Reference Documents

RD-1	*ELODIE: A spectrograph for accurate radial velocity measurements* Baranne et al. 1996, A&AS 119, 373
RD-2	*The CORALIE survey for southern extra-solar planets VII. Two short-period Saturnian companions to HD 108147 and HD168746* Pepe et al. 2002, A&A, 388, 632
RD-3	*Dimensionless constants, cosmology, and other dark matters* Tegmark et al. 2006, PhRvD 73, 3505
RD-4	*Search for Time Variation of the Fine Structure Constant* Webb et al. 1999 PhRvL 82, 884
RD-5	*Constraining Variations in the Fine-Structure Constant, Quark Masses and the Strong Interaction* Murphy et al., 2004, LNP 648, 131
RD-6	*A new constraint on the time dependence of the proton-to-electron mass ratio. Analysis of the Q 0347-383 and Q 0405-443 spectra* Ivanchik et al. 2005, A&A 440, 45
RD-7	*Indication of a Cosmological Variation of the Proton-Electron Mass Ratio Based on Laboratory Measurement and Reanalysis of H_2 Spectra* Reinhold et al. 2006, PhRvL 96, 1101
RD-8	*The UVES Large Program for testing fundamental physics I. Bounds on a change in α towards quasar HE 2217-2818* *Molaro et al. 2013, A&A, 555, 68*
RD-9	*Wavelength accuracy of the Keck HIRES spectrograph and measuring changes in the fine structure constant* Griest et al. 2010, ApJ, 708, 158
RD-10	*On the statistical uncertainties associated with line profile fitting* Landman D.A., Roussel-Dupre R., Tanigawa G., 1982, ApJ, 261,732

RD-11	*Exploring variations in the fundamental constants with ELTs: the CODEX spectrograph on OWL* Molaro et al. 2006, IAUS 232, 198
RD-12	*Science with a 16m VLT: the case for variability of fundamental constants* Molaro 2007 arXiv:0712.4390
RD-13	*First stars VII - Lithium in extremely metal poor dwarfs* Bonifacio et al. 2007, A&A, 462, 851
RD-14	*The Observation and Analysis of Stellar Photospheres, 3rd ed.* Gray D. F., Cambridge University Press, 2005
RD-15	*The SOPHIE search for northern extrasolar planets. III. A Jupiter-mass companion around HD 109246* Boisse et al. 2010, A&A 523, 88
RD-16	*A new code for automatic determination of equivalent widths: Automatic Routine for line Equivalent widths in stellar Spectra (ARES)* Sousa et al. 2007, A&A, 469, 783
RD-17	*On the calculation of the Voigt line profile: a single proper integral with a damped sine integrand* Zaghloul 2007, MNRAS 375, 1043
RD-18	*CA II H and K measurements made at Mount Wilson Observatory, 1966-1983* Duncan et al. 1991, ApJS, 76, 383
RD-19	*Rotation, convection, and magnetic activity in lower main-sequence stars* Noyes et al. 1984, ApJ, 279, 763
RD-20	*New Grids of ATLAS9 Model Atmospheres*. Castelli, F. & Kurucz, R. L. 2003, IAUS, 210, 20
RD-21	*The Identification of Absorption Redshift Systems in Quasar Spectra* Aaronson et al. 1975, ApJ, 198, 13
RD-22	*Improved Age Estimation for Solar-Type Dwarfs Using Activity-Rotation Diagnostics* Mamajek & Hillenbrand 2008, ApJ 687, 1264
RD-23	*The optical-ultraviolet continuum of a sample of QSOs* Natali et al. 1998, AJ, 115, 397
RD-24	*A simplex method for function minimization* Nelder, J. A. & Mead, R. 1965, Computer Journal 7, 308
RD-25	*Numerical Recipes: The Art of Scientific Computing* 2007, Cambridge University Press

1.5 Acronyms and Abbreviations

1.5.1 Acronyms

CCF	Cross-Correlation Function
CD	Column Density
COM	Commissioning
CPL	Common Pipeline Library
CRH	Cosmic Ray Hits
DAS	Data Analysis Software
DAL	Data Analysis Library
DRS	Data Reduction Software
EW	Equivalent Width
FWHM	Full Width at Half Maximum

GSL	Gnu Scientific Library
IGM	Intergalactic Medium
OCA	Organization Classification and Association
OD	Optical Depth
PAE	Preliminary Acceptance Europe
PAC	Preliminary Acceptance Chile
PB	Pass-Band
PL	Power Law
RV	Radial Velocity
SED	Spectral Energy Distribution
SNR	Signal-to-Noise Ratio
SOF	Set Of Files
SOP	Set Of Parameters
TBC	To Be Completed
TBD	To Be Defined
TBR	To Be Revised

Chapter 2. Summary of ESPRESSO Science Cases

2.1 Search for Rocky Extrasolar Planets

Since 1995, more than 800 planetary companions have been found to orbit dwarfs of spectral types from F to M and more massive evolved stars. The large majority of the exoplanets have been found through the induced Doppler spectroscopic variations of the primary star, the so-called radial-velocity (RV) technique.
Most of the candidates detected so far are giant gaseous planets similar in nature to Jupiter. Of the several tens of exoplanets known today having a mass below 25 Earth Masses, the vast majority has been discovered thanks to the sub m s^{-1} precision reached by the HARPS spectrograph.
ESPRESSO, with its instrumental precision of 10 cm s^{-1}, will be able to extensively explore the low mass tail of the exoplanet distribution and to find rocky planets within the habitable zone around other stars.
The planetary mass estimated from the Doppler measurements is directly proportional to the amplitude of the reflex motion of the primary star. The measure of very precise RVs is hampered by 3 main limitations:

1. stellar noise due to the observed source itself: stellar pulsations, surface granulation and activity-related jitter. This can be beaten choosing carefully the stars to be observed and following a precise observation strategy;
2. instrumental errors, in particular: errors affecting the measurements (stability or repeatability) and calibration errors (i. e. errors on the wavelength scale). To reduce this source of errors the issue of stability has to be considered a primary concern from the project phase of the instrument, as it is the case for ESPRESSO. The development of the laser frequency comb techniques will likely bring the calibration errors to below the cm s^{-1} limit needed for ESPRESSO;
3. photon noise finally sets the fundamental limit for the attainable precision as the latter scales with the signal-to-noise of the spectra. From this respect, ESPRESSO marks a significant improvement with respect to HARPS with an increase in collecting area of a factor 5.

A detailed description of the analysis of stellar spectra for the determination of the RV can be found in the reference papers [RD-1] and [RD-2].
The RV in the stellar spectra observed with ESPRESSO will be computed automatically at the telescope as part of the spectral data reduction (AD-2). However, the recipe to compute the RV, esp_compu_radvel, is also part of the Star branch of the DAS Reflex Workflow and can be used in offline-mode to refine the first automatic determination (see Sections 4.3 and 9.3). In the DAS recipe, the user can choose the spectral template he/she wants to adopt and the wavelength array is corrected for the computed RV.

2.2 Variability of Fundamental Constants

The Standard Model of particle physics needs some 26 physical parameters for the description of the world, including the masses of known particles, mixing angles and phases and the relative strength of fundamental forces (RD-3). Two of them, namely the fine structure constant α =

e2/ħc and the proton-to-electron mass ratio μ, are of particular interest for astronomy since they are directly related to the strength of fundamental forces and can be measured accurately by astronomical observations of intervening absorption systems towards distant quasars. We emphasize that only astronomical observations can effectively probe the variability of the physical dimensionless constants in space-time.
The fine structure constant is related to the strength of the electromagnetic force and the proton-to-electron mass ratio is related to the ratio between the strong and weak nuclear forces.
Thus, astronomical observations can effectively probe tiny variability of the basic forces of nature in space-time, that if present, would have profound implications for physics and cosmology revealing new physics beyond the Standard Model.
In 1999, observations of spectral lines in distant quasars gave the first hints that the fine-structure constant might change its value over time, being lower in the past by about 6 parts per million (RD-4, RD-5). More recently, also μ= mp / me has been found to vary (RD-6, RD-7). However, recent observations by Molaro et al. (RD-8) reveal no evidence for variations in alpha at the 3ppm level (1 sigma). The debate is still open and demands a high-resolution spectrograph for a definitive clarification.
The measure of the variability is carried out from the comparison of the relative positions of a chosen set of absorption lines determined from the fit of the features with Voigt profiles (see RD-9). The main source of uncertainty are represented by:

1. systematic shifts due to the instrument+telescope (slit positioning): it is the dominant error in the spectrographs in use (for UVES, Δv ~ 800 m/s), in ESPRESSO it will become negligible;
2. uncertainties in the wavelength calibration process (for UVES, Δv ~ 50 m/s): the laser comb calibration foreseen for ESPRESSO will decrease substantially these errors;
3. uncertainties introduced by the barycentric correction (Δv ~ 1 m/s): to control these errors we will adopt the latest ephemerides released from the JPL and use routines to compute the barycentric motion that include also the minor bodies of the solar system;
4. uncertainties introduced by the vacuum correction (Δv ~ 1 m/s): drifts in the refractive index of air inside the spectrograph between the wavelength calibration and quasar exposures will cause miscalibrations. This will not be the case for ESPRESSO since the instrument is thermally stable and there will be the possibility to take a simultaneous wavelength calibration;
5. photon noise sets the fundamental limit for the attainable precision as the latter scales with the signal-to-noise of the spectra: $\sigma_v = 0.7 * \mathrm{sqrt}(FWHM * \Delta v_{pix}) / SNR$, where Δv_{pix} is the size in velocity of the CCD pixel (RD-10). This will be the dominant source of errors in ESPRESSO for this kind of measurements.

VLT ESPRESSO, by means of its extreme stability and wavelength calibration, will provide an increase in the accuracy of the measurement for both these constants by at least 1 order-of-magnitude which is more than enough to clarify the present controversy and to probe the variability into an insofar unexplored regime (RD-11, RD-12).
Some very few objects are sufficiently bright to be observed in the 1-UT mode. For the others, the 4-UT mode will be perfectly appropriate and offers sufficient signal-to-noise ratio to answer definitely this fundamental question.
The quasar spectra that will be collected by ESPRESSO for this science case will be processed through the Quasar Branch of the ESPRESSO DAS Reflex Workflow (see Chapter 5.). The most relevant recipe to measure the variability of fundamental constants is espda_fit_line, which performs the fit of detected absorption lines with Voigt profiles (see Sections 4.13 and 0).

2.3 Chemical Composition of Stars in Local Galaxies

Our understanding of galaxy formation has improved in the last 10 years, mainly thanks to the 8m class telescopes. One fundamental piece of information is the chemical composition for local galaxies. In spite of many successes in this field, the majority of local galaxies still lacks detailed abundance information. In the few cases for which the chemical abundance is known it is generally based on few stars, and for the faintest galaxies on low to intermediate resolution spectra.

Although it is clear that most of the chemical information for local galaxies will have to come largely from the ELT, ESPRESSO will give us the chance to have a glimpse of this. The information gathered by ESPRESSO will be of fundamental importance also for planning and developing suitable instrumentation for the ELT.

A specific issue on which ESPRESSO used in the 4UT configuration should allow to forerun the ELT is the measurement of Li in metal-poor populations in external galaxies (see e.g. RD-13).

The technique to derive chemical composition of stars is based on the comparison of the equivalent width of absorption lines measured in the observed spectrum with synthetic spectra, an introduction to this method can be found in [RD-14].

Star spectra acquired with ESPRESSO for this science case will be processed through the Star branch of the DAS Reflex Workflow. Equivalent widths are measured with the recipe espda_compu_eqwidth (Sections 4.5 and 9.5); the comparison with synthetic spectra can be carried out by the recipe esp_synth_spec (Sections 4.8 and 9.8). The identification and fit of lines is obtained with the recipes espda_create_linelist (Sections 4.10 and 9.10) and espda_fit_line (Sections 4.13 and 0), respectively.

Chapter 3. Mathematical Description

3.1 Continuum level determination in quasar spectra

The continuum level in quasar spectra is determined by the recipe esp_fit_qsocont (see Sections 4.10 and 9.10). The fit of the region redward of the quasar HI Lyman-α emission is quite straightforward due to the low number of absorption lines that can be easily masked. On the other hand, the region blueward of the Lyman-α emission is affected by a large number of absorption lines (increasing with redshift) mainly due to the HI Lyman-α transition. They arise in the intervening matter along the line of sight and are a trace of the fluctuations of the IGM density. These lines are called, in general, Lyman-α forest.
The transmitted flux in the Lyman-α forest is reduced by a factor depending on redshift and on the effective IGM optical depth, τ_{eff}. This effect must be taken into account before fitting a function (e.g. a power-law) to the flux distribution in order to determine the continuum level. The effective optical depth is due to resolved and unresolved lines, and is obtained by integrating the convolution between the column density distribution and the curve of growth between 0 and a given maximum column density $N_{\rm lim}$. The implemented algorithm fits the absorption lines from the strongest to the weakest one, as a consequence the value of $N_{\rm lim}$ is updated to exclude the column density range of the lines that have been already fitted (because the spectral regions of these lines are no longer considered for the power law computation). After fitting the last resolved line, $N_{\rm lim}$ corresponds to the largest unresolved column density and consequently τ_{eff} takes into account only the contribution from unresolved lines.
The correction factor for the flux is computed as follows:

$$\text{factor} := e^{\alpha(1+z)^{\beta}\tau_{\rm eff}} \qquad (3.1)$$

where, α and β parameterize the dependence from redshift and the effective optical depth is obtained as

$$\tau_{\rm eff}\,(N_{lim}) := \frac{\int_0^{N_{lim}} N^{-\gamma} G(N)\mathrm{d}N}{\int_0^{\infty} N^{-\gamma} G(N)\mathrm{d}N}, \qquad (3.2)$$

where, γ is the power law index of the column density distribution and the curve of growth is approximated as

$$G(N) = \begin{cases} 0, & N \geq N_{\rm max} \\ N, & N \leq 10^{14} \\ (10^{14}/14\ln 10)\ln N, & 10^{14} \leq N \leq 10^{18} \\ (9/7)10^{5}\sqrt{N}, & 10^{18} \leq N \leq N_{\rm max} \end{cases} \qquad (3.3)$$

where, $N_{\rm max}$ is the maximum column density measured for an HI Lyman-α transition and can be set by the user.

3.2 Computation of the radial velocity

The ESPRESSO DA implements a cross-correlation module that computes the cross-correlation function (CCF) of a S2D spectrum with respect to a binary template (mask) of a given spectral type. The radial velocity (RV) is then obtained from a Gaussian fit to the CCF (see Sections 4.3 and 9.3). This is the technique that has been successfully used on the ELODIE, CORALIE, HARPS, SOPHIE and HARPS-N spectrographs (see RD-1 and RD-2). One of its main advantages is that CCFs can be computed in an automatic way with only a few line masks at hand. Line masks are simply numerical mask consisting of 1 and 0 value-zones, with the non-zero zones corresponding to the theoretical positions and widths of the stellar absorption lines at zero velocity, and can be created for various spectral types. The main steps of the algorithm are:

1. Compare the global flux distribution in the S2D spectrum to a static flux template that approximately corresponds to the spectral type of the star. The S2D flux is scaled accordingly to match the flux distribution of the template. In this way, spectra of any given star are always brought to the same flux distribution, which ensures that variable atmospheric conditions will not induce systematic effects in the CCF computation.
2. Shift the wavelength scale of the S2D spectrum to the Solar System barycenter using the barycentric correction.
3. Define a uniform radial velocity grid that is approximately centered on the radial velocity of the star.
4. For a given RV value in the grid v, shift the line mask by the corresponding Doppler shift, project the line mask onto the S2D spectrum using a specified line width (about one pixel), and sum the S2D flux that goes through the so-defined mask "holes":

$$CCF(v) = \sum_{l} \sum_{x,o} p_{l,x,o}\ (v)\ f_{x,o}$$

   where, $f_{x,o}$ is the value of the 2D spectrum for the order o at the pixel location x and $p_{l,x,o}$ is the fraction of the l^{th} line of the template which falls into the pixel (x,o) at the velocity v.
   The flux from partial pixels is computed via simple linear interpolation. The sum is actually a weighted sum, using line depths as weights to optimally extract the Doppler information. During this process, the S2D spectrum is locally blaze-corrected to remove any continuum slope around spectral lines. This produces one point of the CCF.
5. Loop over all RV values.
6. Fit a Gaussian profile to the CCF to derive RV, FWHM and contrast (see Figure 3.1).

Note that, by construction, CCFs are simply co-added spectral lines in velocity space, weighted by their depth and continuum flux. As such they can be considered as a "master" spectral line for the star.

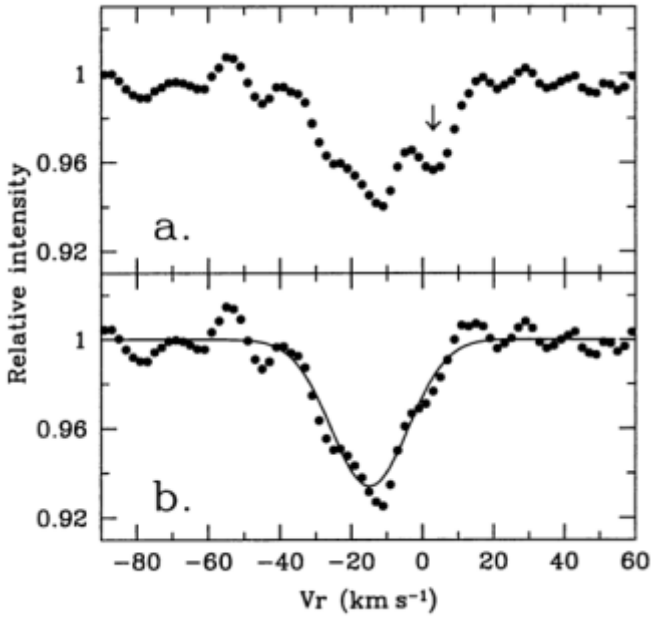


Figure 3.1 - a) CCF of a faint star contaminated by the CCF of the solar spectrum (indicated by the arrow). b) Same CCF after the subtraction of the of the CCF of the sky The line is the Gaussian fit. (Adapted from [RD-1])

Uncertainties on the CCF data points are obtained by propagation of S2D error maps through the cross-correlation stage, which is a simple additive process (fluxes from many spectral lines are co-added). Finally, an estimate of the radial velocity uncertainty is obtained by converting CCF flux errors into RV errors using the measured CCF derivative, as described in the appendix of [RD-15].

3.3 Detection of absorption lines (TBR)

The procedure to determine the presence and location of absorption lines in QSO and stellar spectra is described in detail in [RD-16]. In the following, we report a brief description of the algorithms adopted in the recipes espda_compu_eqwidth (Sections 4.5 and 9.5), esp_fit_qsocont (Sections 4.10 and 9.10) and espda_create_linelist (Sections 4.11 and 9.11).

The position of the line peak or peaks in the case of blended lines is determined using some mathematical properties of the derivatives of a function. It is well known that the zeros of the derivative of a function give us the local minima and maxima. The zeros of the 2nd derivative give us the inflection points, i.e. the points where the function changes its concavity. The local maxima of the 2nd derivative will give us the center of the absorption lines directly. The best way to find these maxima is to use the zeros of the 3rd derivative of the function. This can be seen in the right panel of Figure 3.2. It is clear that the maxima of the second derivative gives us a good estimation of the position of the lines in the original spectra even for cases of strong blended lines (e.g. 5500.6 Å and 5500.8 Å in Figure 3.2). To determine the peak or peaks to fit the spectral line, it is necessary to obtain the first three numerical derivatives of the surroundings of the line profile.

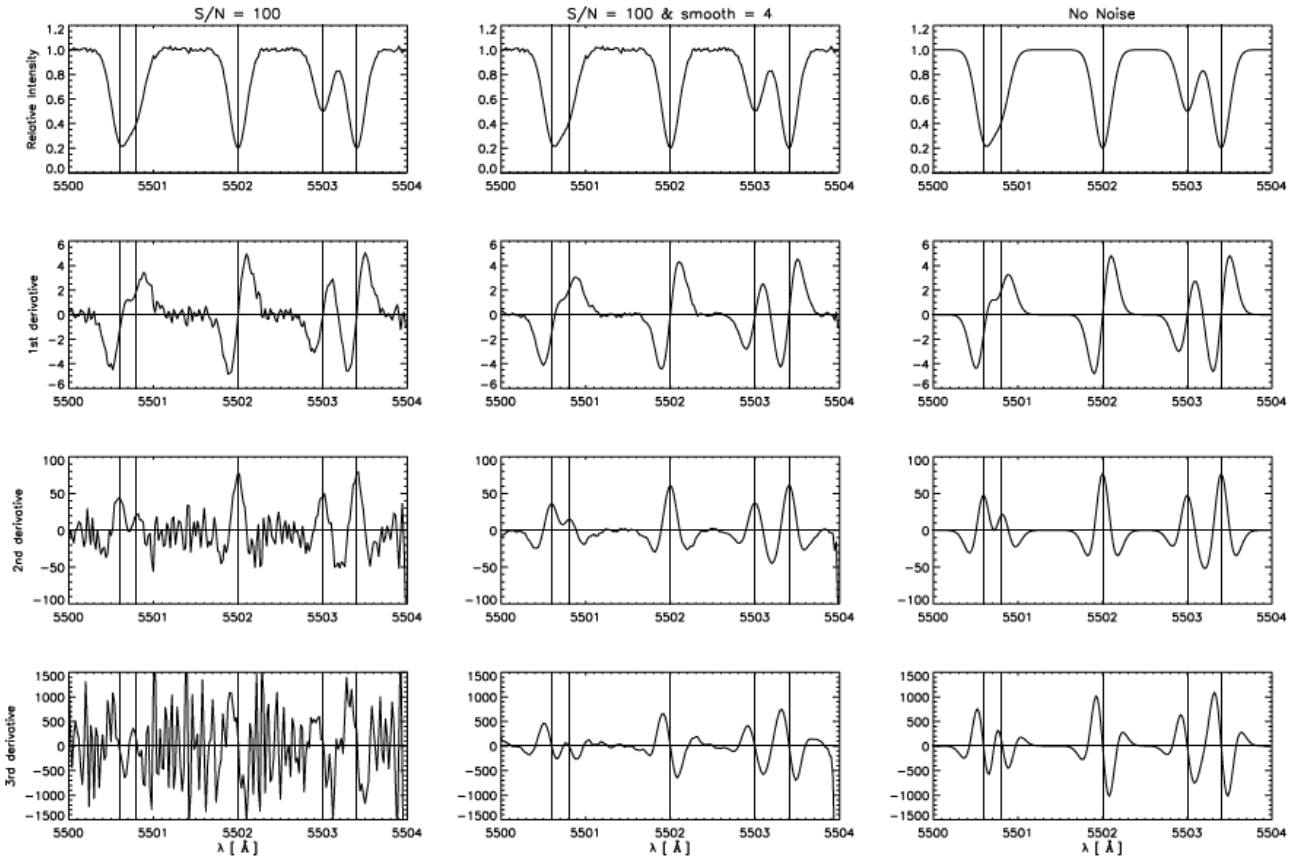


Figure 3.2 – A simulated normalized spectral region and the respective 1st, 2nd, and 3rd derivatives
In the right panel it is easily seen that the local maxima of the 2nd derivative, which can be found using the zeros of the 3rd derivatives, estimate the center of the lines of the simulated spectrum very well. Note that using this we can see extremely blended lines that do not present zeros on the 1st derivative. The left and middle panels show how a smoothing of the flux is useful for eliminating the noise when computing the derivatives (adapted from [RD-16]).

Noise can be a problem, especially when determining numerical derivatives, where the noise propagates very fast (left panel of Figure 3.2). To overcome this problem, we make use of a numerical smoothing applied to the arrays of the derivatives with a boxcar average of a given width eliminating some of the noise. The use of such a smoothing parameter is illustrated in the middle panel of Figure 3.2. The smooth result of an array of values is described by the expression

$$R_i := \begin{cases} \frac{1}{w}\sum_{j=0}^{w-1} A_{i+j-w/2}\,, & i = \frac{w-1}{2}, \ldots, N - \frac{w-1}{2} \\ A_i, & \text{otherwise} \end{cases} \qquad (3.4)$$

where A is the array to be smoothed, w the width of the boxcar to be averaged, N the number of array elements, and R the resulting smoothed array. Even using this numerical trick, if the noise is high the procedure may identify more lines than actually present. To overcome this possible problem, we have introduced another parameter, which is linked to the spectral resolution, it determines the minimal distance between consecutive lines. A typical value for this parameter is 0.1 Å.

3.4 Voigt fit of absorption lines

Absorption lines in quasar and stellar spectra are fitted with Voigt profiles using the recipe

espda_fit_line (Sections 4.13 and 0).
The profile of a spectral line is well approximated by the Voigt function V, which is the convolution of a Gaussian profile and a Lorentzian profile. The Gaussian profile describes the Doppler broadening in the medium where the line is produced, while the Lorentzian profile is a combination of the natural width of the line and the effect of particle collisions in the medium. In the case of an absorption line, the optical depth produced by transition at frequency ν_0 is

$$\tau_\nu = N\frac{\sqrt{\pi}e^2}{m_e c}\frac{f}{\Delta\nu_b}V(a,u), \qquad a \overset{\text{def}}{\equiv} \frac{\Gamma}{4\pi\Delta\nu_b}, \quad u \overset{\text{def}}{\equiv} \frac{\nu-\nu_0}{\Delta\nu_b}, \tag{3.5}$$

where N is the column density of the medium, e is the electron charge, m_e is the electron mass, c is the speed of light, f is the oscillator strength of the transition, Γ is the transition damping constant, and $\Delta\nu_b$ is the Doppler broadening due to thermal velocities and turbulence in the medium. The Voigt function is defined as follows:

$$V(a,u) \overset{\text{def}}{\equiv} \frac{a}{\pi}\int_{-\infty}^{+\infty}\frac{\exp(-y^2)}{a^2+(u-y)^2}\,\mathrm{d}y. \tag{3.6}$$

Except for special values of a and u, an analytical solution of this integral does not exist in literature. A suitable way to assess this problem has been described in [RD-17], where the following representation of the Voigt function as a single proper integral with a damped sine integrand is adopted:

$$V(a,u) = \exp(a^2)\mathrm{erfc}(a)\exp(-u^2)\cos(2au) + \frac{2}{\sqrt{\pi}}\int_0^u \exp[-(u^2-y^2)]\sin[2a(u-y)]\,\mathrm{d}y. \tag{3.7}$$

The computation of this proper integral can be carried out by the standard GSL routine for numerical integration, *gsl_integration*. The term $\exp(a^2)\mathrm{erfc}(a)$ may be subject to an overflow/underflow limitation when a is large (order of 10). In this case, one can use the following approximation, accurate to the 16th decimal digit up to a = 200 with n = 7 [RD-17]:

$$\exp(a^2)\mathrm{erfc}(a) \approx \frac{1}{\sqrt{\pi}a}\sum_{i=0}^{n}\frac{(-1)^i(2i)!}{i!\,(2a)^{2i}}. \tag{3.8}$$

The integral representation above can be also used to tabulate the Voigt function on a grid of (a,u) values. The tabulated values can be used to interpolate (through the bilinear interpolation formulae) the function at any value of a and u, significantly reducing the computation time.
The actual fitting of spectral lines is a minimization procedure in a multidimensional space. Each line is defined by four parameters: the redshift z, the column density N, the thermal Doppler broadening b, and the turbulence broadening b_{tur}. We have

$$z \overset{\text{def}}{\equiv} \frac{\nu_0-\nu_{\mathrm{obs}}}{\nu_{\mathrm{obs}}}, \quad b \overset{\text{def}}{\equiv} \sqrt{\frac{2kT}{m}}, \quad b^2 + b_{\mathrm{tur}}^2 = c^2\left(\frac{\Delta\nu_b}{\nu_0}\right)^2. \tag{3.9}$$

where ν_{obs} is the observed central frequency of the line, k is the Boltzmann constant, T is the medium temperature, and m is the mass of the particles. The observed line profile I_ν is the

convolution between the intrinsic profile and the instrumental profile $\Phi_{\delta\nu}$; if the continuum level is $I_{\nu,0}$, we have

$$I_\nu(\mathbf{x}) = \Phi_{\delta\nu} \otimes \quad I_{\nu,0}\exp[-\tau_\nu(\mathbf{x})], \quad \mathbf{x} \stackrel{\text{def}}{=} (z, N, b, b_{\text{tur}}). \qquad (3.10)$$

For L lines, $\mathbf{x}$ is a vector in a space with up to $4L$ dimensions. If some of the line parameters are constrained, the number of dimensions decreases to $M = 4L - C$, where C is the number of constraints. Two kinds of constraints are allowed:

1. parameters which are kept fixed at a given value;
2. parameters which are common to two or more lines (common parameters are allowed to vary, but they are always kept equal to each other).
3. thermal Doppler parameters of ions of different elements linked to be at the same temperature (Doppler parameters vary with the square root of the ratio of the element masses, see Eq. 3.9).

The function to be minimized is the reduced chi-square function $\chi^2(\mathbf{x})$ defined as follows:

$$\chi^2(\mathbf{x}) \stackrel{\text{def}}{=} \frac{1}{R - M - 1}\sum_{r=1}^{R}\frac{[I_r(\mathbf{x}) - F_r]^2}{E_r}, \qquad (3.11)$$

where R is the number of data points along a wavelength grid, F_r and E_r are the flux and flux error values at these points, and I_r is the line profile computed at the same points.

The reduced chi-square is minimized using the standard GSL tool for multidimensional minimization, *gsl_multimin*. The routine proceeds from a guess choice of parameters $\mathbf{x}_0$ using the Nelder-Mead simplex method [RD-24] to find the "downhill" direction from $\mathbf{x}_0$ in the parameter space (other algorithms will be also tested for speed and robustness).

In practice, a vector $\mathbf{x}_{M,0}$ suitable for minimization is extracted from $\mathbf{x}_0$ according to the adopted constraints. Components of $\mathbf{x}_0$ with a constraint of the first kind are not included in $\mathbf{x}_{M,0}$; components of $\mathbf{x}_0$ with a mutual constraint of the second kind are included as a single component in $\mathbf{x}_{M,0}$. When the downhill direction is found, $\mathbf{x}_{M,0}$ is updated into $\mathbf{x}_{M,1}$ and its component are used to update the original vector $\mathbf{x}_0$ into $\mathbf{x}_1$. Components of $\mathbf{x}_0$ with a constraint of the first kind are copied as they are into $\mathbf{x}_1$; all components of $\mathbf{x}_0$ with a mutual constraint of the second kind are replaced in $\mathbf{x}_1$ by the corresponding value from $\mathbf{x}_{M,1}$.

The procedure is iterated until a given threshold for the reduced chi-square is reached and the final value $\mathbf{x}_n$ is taken as best-fit solution for the line profiles.

The line fitting procedure is quite robust in the presence of one or a few pixels affected by CRHs or bad pixels, due to the significant number of pixels sampling the line profile (in a single frame, or even better in the case of the coaddition of several fames). Based on previous experience, results of Voigt profile fitting are robust for spectra with singal-to-noise ratio larger than ~10.

Chapter 4. *Functional Description

The ESPRESSO DAS comprises a total of 13 recipes. The ESPRESSO DAS is split into four branches: one for the reduction of quasar spectra and three for the reduction of star spectra. Some of the recipes are common to more branches while others are specific to the treated spectra (see Table 4.1). Each DAS branch will be managed by a Reflex workflow (see Chapter 5). All the recipes will be available off-line at the telescope on a dedicated workstation and every user will have the possibility to install the whole package on his/her computer.

Two different recipes perform the computation of the radial velocity in stellar spectra. The recipe espda_compu_radvel (Sections 4.3 and 9.3) is aimed at performing very accurate radial velocity (RV) computation (at the sub-m/s precision) of typically solar-metallicity, main-sequence, late F-, G-, K-, and early M-type stars. This recipe performs the cross-correlation of the observed 2D stellar spectrum order-by-order with a mask containing accurate wavelength positions and relative intensities of stellar lines. It will be available "off-line" as part of the DA package but also "on-line" at the end of the data reduction process run at the telescope. Astronomers interested in the measurement of the radial velocity will have the possibility to have this parameter computed automatically immediately after their observation at the telescope.

The recipe espda_rv_synth (Sections 4.9 and 9.9) is also aimed at performing the RV computation but with a significantly lower precision (below km/s precision depending on the stellar rotation and signal-to-noise of the spectrum) to be able to compare any possible stellar spectrum with synthetic spectra. This recipe performs the cross-correlation of the observed 1D stellar spectrum with a synthetic spectrum computed with a set of given stellar parameters and metallicity. This recipe is expected to work with late F-, G-, K-, and early M-type stars, for giant, subgiant or dwarf stars, and for metal-poor or metal-rich stars.

Operation	Branch	Name
Coaddition of spectra	star, quasar	espda_coadd_spec
Creation of a spectral mask	star, quasar	espda_mask_spec
Computation of the radial velocity	star	espda_compu_radvel
Computation of the stellar activity indexes	star	esp_compu_rhk
Measurement of the EW of spectral lines	star	espda_compu_eqwidth
Computation of the stellar parameters	star	espda_compu_starpar
Continuum fitting of single orders	star	espda_fit_starcont
Comparison with a synthetic spectrum	star	espda_synth_spec
Computation of the radial velocity using a synth. spec.	star	espda_rv_synth
Continuum fitting of a 1D spectrum	quasar	espda_fit_qsocont
Creation of a list of absorption lines	star, quasar	espda_create_linelist
Identification of the absorption systems	quasar	espda_iden_syst
Voigt-profile fitting of absorption lines	star, quasar	espda_fit_line

Table 4.1 – List of recipes.

The following sections describe each recipe specifying the processing steps both in words and with a graphical flow chart. The corresponding DAS functions are described in detail in Chapter 8. It is worth noting that only the *high level functions* are described here; the functions performing simple operations and file manipulation are implicit.

4.1 *Co-addition of spectra (espda_coadd_spec)

This recipe is used to combine different observations of the same object, in order to maximize the SNR. Input spectra are obtained from the DR pipeline as multiextension FITS frames (TBD). The recipe works with spectra whose orders have not been merged (S2D; see Sec. 6.1 , 6.3 and 7.1.2). These spectra are concatenated into a single spectrum (SPEC) in which all pixels keep their wavelength and the information on the order and spectrum of origin. In this way, the subsequent recipes can access the original information extracted by the DRS avoiding the issues related to rebinning (e.g. correlation between noise values in adjacent pixels). For inspectional and plotting use, the recipe gives in output also a coadded, rebinned one-dimensional spectrum (SPEC) with the fixed wavelength grid defined in the DR pipeline.
The recipe can work also with 1D merged spectra obtained in case with other instruments, the output product will be a coadded, rebinned spectrum.
Stellar spectra are corrected for radial velocity to the rest frame.
Input spectra are optionally equalized to the same flux level before coaddition or concatenation, to prevent incorrect weighting of flux values that could be due to slit losses and/or incorrect flux calibration. By default, the first input spectrum is adopted as a reference. Equalization is performed by rescaling the flux in each order to the average flux of the reference spectrum. Optionally, the recipe can eliminate the pixels with deviating flux values via kappa-sigma clipping.
The detailed processing steps are reported in Figure 4.1.

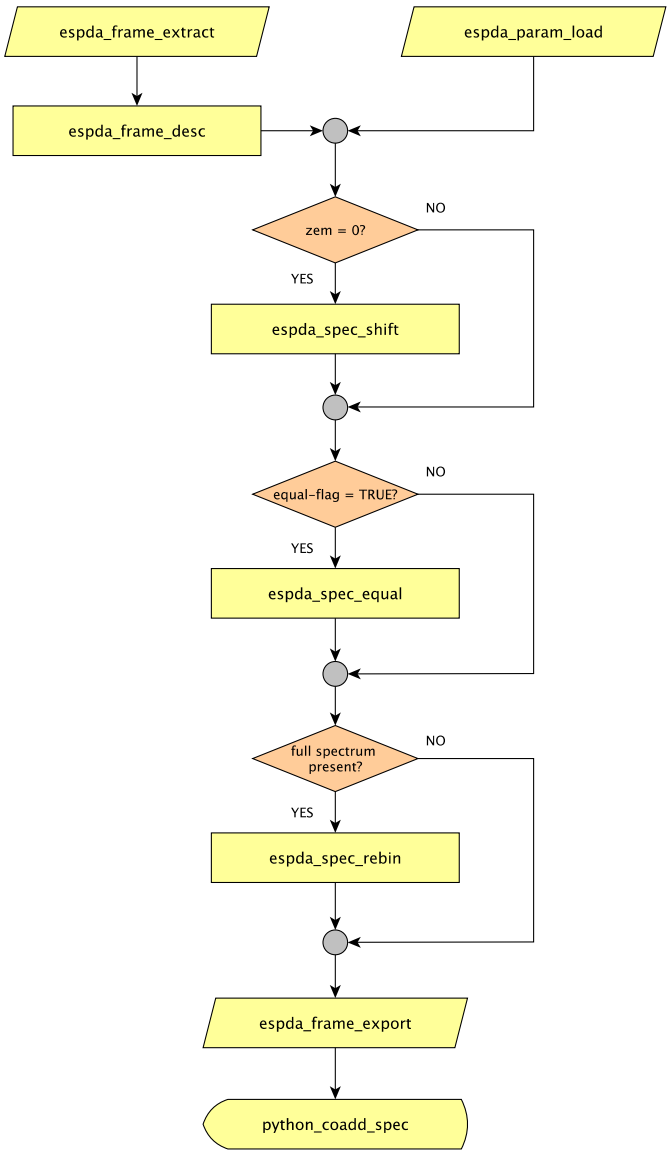


Figure 4.1 - Flowchart of the recipe espda_coadd_spec.

4.2 *Creation and application of a spectral mask (espda_mask_spec)

This recipe is used interactively to create a mask of selected spectral regions, and to apply it to a spectrum. Masks are useful to restrict the analysis to a subset of the available wavelength coverage. The recipe allows the user to interactively create a mask. In the output spectrum the masked wavelengths are deleted.
The processing steps are described in Figure 4.2.

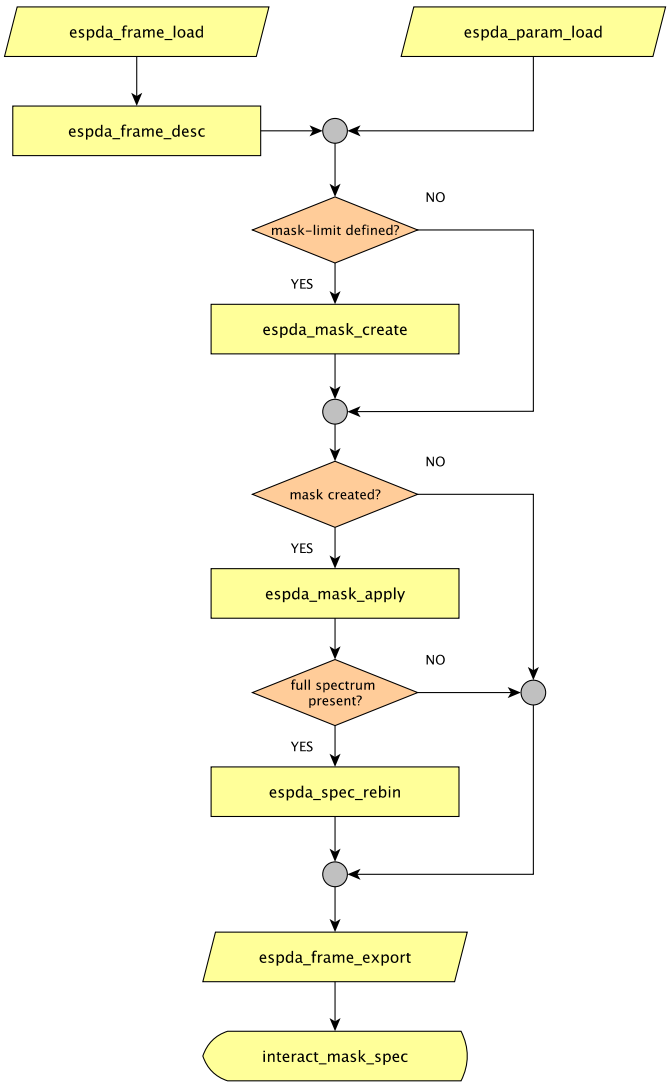


Figure 4.2 - Flowchart of the recipe espda_mask_spec.

4.3 Computation of the radial velocity (esp_compu_radvel) TBR

This recipe computes the cross-correlation function (CCF) of the S2D spectrum using a given line mask, and then fits a Gaussian function to the CCF to obtain CCF parameters including radial velocity. The processing steps (Figure 4.3) are described below.

1. Search and remove residual cosmic hits on S2D spectrum by positive sigma-clipping (esp_filter_cosmics)
2. Correct flux distribution of S2D spectrum to match the given flux template (esp_correct_flux)
3. Shift wavelength scale to the barycentric reference frame (esp_shift_wavelength_scale)
4. Perform cross-correlation of the S2D spectrum using the given CCF template, fit a gaussian function to the CCF and derive CCF parameters and errors including radial velocity (esp_compute_ccf)
5. Compute CCF bisector (RV vs. line depth) and bisector velocity span (esp_compute_bis)
6. Shift wavelength scale to the rest frame of the star (esp_shift_wavelength_scale)

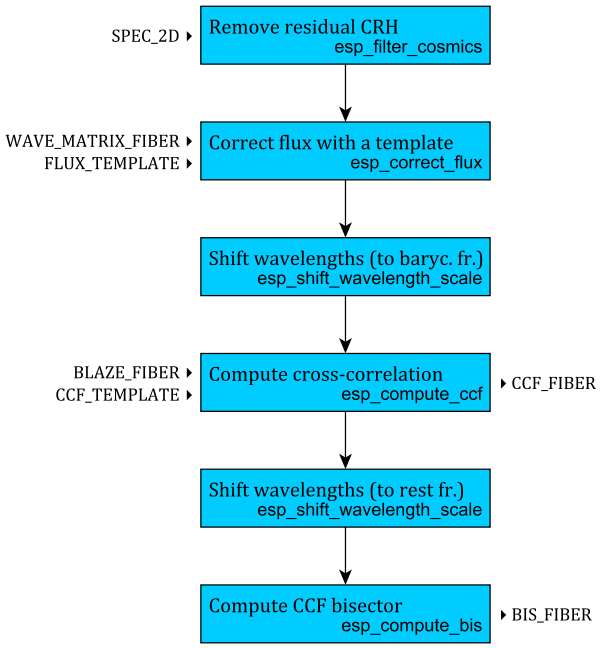


Figure 4.3 - Flowchart of the recipe esp_compu_radvel

4.4 Computation of stellar activity indexes (esp_compu_rhk) TBR

This recipe computes the Ca II H&K activity index, both as S-index on the Mt Wilson scale and log(R'HK) as defined in [RD-18] and [RD-19]. The processing steps (Figure 4.4) are described below.

1. Search and remove residual cosmic hits on S2D spectrum by positive sigma-clipping (esp_filter_cosmics)
2. Remove blaze function from S2D spectrum (esp_correct_blaze)
3. Measure integrated fluxes and errors in the four Mt Wilson passbands H, K, R and V (esp_measure_flux)
4. Compute Mt Wilson CaII H&K S-index (esp_compute_s_index)
5. Compute R'HK index from S-index and stellar B-V color (esp_compute_rhk_index)
6. Compute estimates of stellar rotation period and age from R'HK index (esp_compute_prot_age)

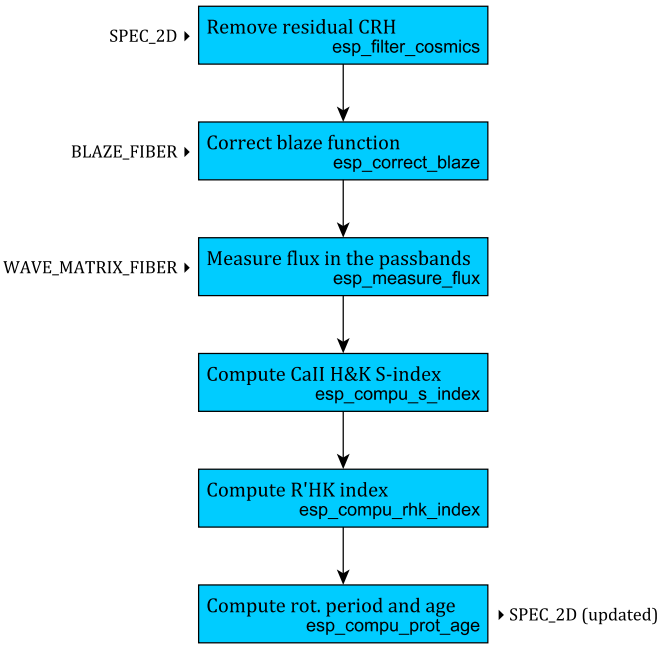


Figure 4.4 - Flowchart of the recipe esp_compu_rhk

4.5 Measurement of the EW of spectral lines (espda_compu_eqwidth) TBR

This recipe computes the equivalent widths of the absorption spectral lines present in an input list. The processing steps (Figure 4.5) are the following.
For each individual line in the input list:

1. Select the local spectral region around the line to be analyzed (espda_select_chunk).
2. Determine the continuum level in the selected region and normalize to it (espda_fit_lcont)
3. Detect the absorption lines that are blended with the selected line (if any) (espda_det_line)
4. Fit the detected lines with gaussian profiles (espda_fit_ngauss)
5. If Reflex is used, plot the Gaussian fit on the local normalized spectrum (EspdaPlotSpecLine.py).
6. Store the result.

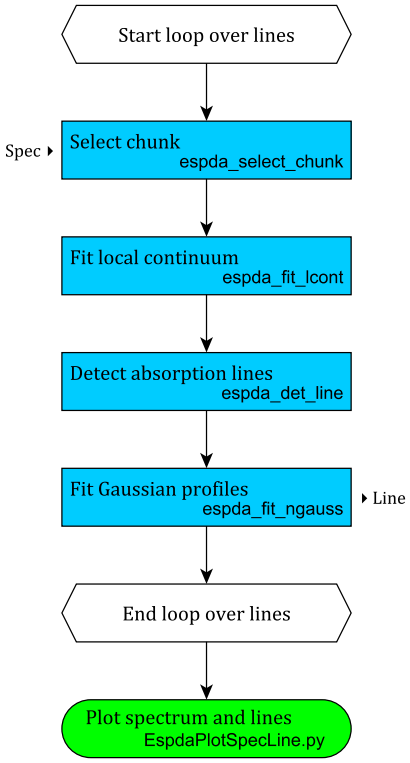


Figure 4.5 - Flowchart of the recipe espda_compu_eqwidth

4.6 Computation of the stellar parameters (espda_compu_starpar) TBR

This recipe estimates the effective temperature and iron abundances for solar type stars.
The processing steps (Figure 4.6) are described in the following.

1. Estimate the effective temperature by comparing the line ratios from the list of lines with EWs (computed by espda_compu_eqwidth) and the line-ratio calibration table (espda_calib_teff)
2. Estimate the [Fe/H] ratio using the estimated temperature, the list of iron lines with measured EW, and an input list of calibrations for iron lines (espda_calib_feh)

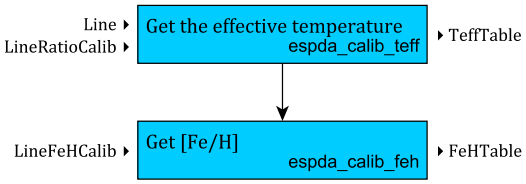


Figure 4.6 - Flowchart of the recipe espda_compu_starpar

4.7 Fit of the continuum level of single orders (espda_fit_starcont) TBR

This recipe computes the continuum order by order by fitting a cubic spline or a Legendre polynomial and produces the merged 1D normalized spectrum corrected for barycentric velocity

(and/or corrected for radial velocity in case accurate radial velocities for individual spectra are available). The recipe uses as input the table with concatenated fluxes with wavelengths corrected to barycentric reference and/or radial velocity corrected, i.e. the output of the recipe espda_coadd_spec.
The processing steps (Figure 4.7) are described below.

1. Read the input table and the input parameters of the recipe
2. Loop over spectral orders
    a. First, mask possible negative values to avoid a wrong initial polynomial fit, mask values equal to 1 means useful data point, mask values equal to 0 means rejected data point (espda_cont_ord)
    b. Fit the continuum of the spectral order using the input (default) parameters, by evaluating the fitting function in unmasked (wavelength, flux) points, errors of fitting procedure only provided for unmasked data points (espda_cont_ord)
    c. Evaluate the fitting function at unmasked wavelength points and divide the order fluxes by the corresponding fitting values
    d. Evaluate if more points should be rejected/masked in next iteration and redo the process until no more points are rejected or the maximum number of iterations is reached
    e. Write the result in the output table with concatenated normalized fluxes
3. Finish the loop over spectral orders
4. Merge all the orders into a single 1D normalized spectrum and (optionally) filter deviating flux points (espda_rebin_flux)
5. Plot final 1D merged spectrum

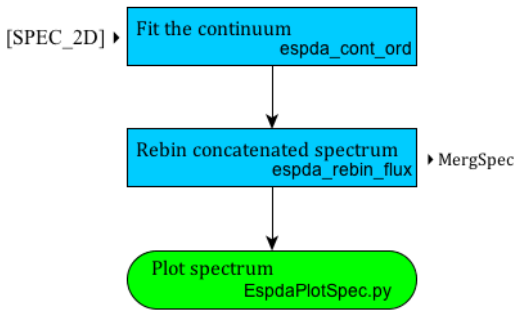


Figure 4.7 - Flowchart of the recipe espda_fit_starcont

4.8 Comparison with a synthetic spectrum (espda_synth_spec) TBR

This recipe compares the observed 1D stellar spectrum normalized to the continuum with three synthetic spectra for a given set of stellar parameters (effective temperature, Teff, and surface gravity, log(g)) and metallicity, [Fe/H], and broadening parameters (namely rotational, Vrot, and macroturbulent, Vmacro, velocities). The whole grid of synthetic spectra, created using Kurucz model atmospheres (see RD-20), will cover Teff values from 3,500 to 30,000 K , log(g) values from 0 to 5 dex and [Fe/H] values from -5 to 1 dex. The first set of stellar parameters should be the preferable one, and the second and the third sets of parameters should provide upper and

lower bounds to be compared with the observed spectrum. This recipe also corrects the radial velocity (RV) of the star using the RV value given in a keyword of the header of the observed spectrum. This value could be the input guess value given by the observer or the RV determination performed by the recipe esp_rv_synth.
The processing steps (Figure 4.8) are described below.

1. Read the input table and the input parameters of the recipe
1. Correct the wavelength of the observed 1D spectrum for the RV of the star using the value given in a keyword on the header of the observed spectrum.
2. Loop on the interpolation of the three synthetic spectra:
    a. Select 6 grid points around each one of the three set of parameters (Teff, log, [Fe/H]) and perform an interpolation (esp_interp_synth) with a given interpolation function previously defined (esp_funct_interp)
    b. Convolve the three synthetic spectra with the macroturbulent, rotational and instrumental profile (esp_convol_synth), defined in a given function (esp_convol_kernel)
3. Finish the loop on the interpolation of the three synthetic spectra.
4. Plot final 1D merged spectrum together with the three synthetic spectra and also the residual plot of the differences (observed – synthetic)

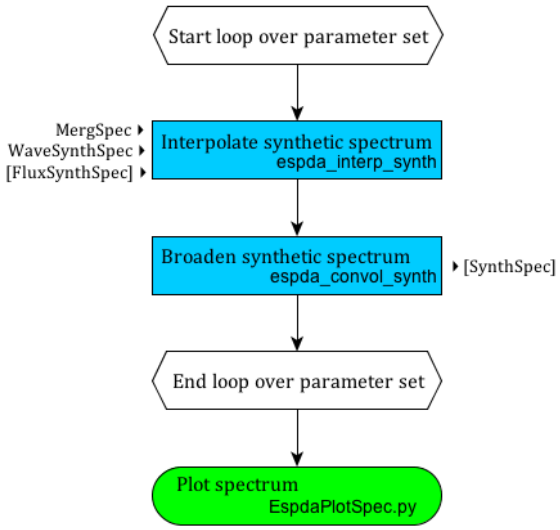


Figure 4.8 - Flowchart of the recipe esp_synth_spec

4.9 Radial velocity computation using a synthetic spectrum (espda_rv_synth) TBR

This recipe derives the radial velocity of the star by cross-correlating the observed 1D stellar spectrum normalized to the continuum with a synthetic spectrum already interpolated and convolved, in fact the first/preferable synthetic spectrum, which is the output of the recipe esp_synth_spec.
The processing steps (Figure 4.9) are described below.

1. Read the input table and the input parameters of the recipe
2. Plot the spectrum to create a mask (espda_create_mask)
3. Mask the regions that will not contribute to the cross-correlation profile both in the observed and synthetic spectrum (espd_apply_mask)

4. Cross-correlate the masked observed spectrum with the masked synthetic spectrum (esp_cc_synth)
5. Fit a Gaussian to the cross-correlation function (esp_fit_ccf)
6. Write the RV determination in the corresponding keyword of the header of the observed 1D spectrum

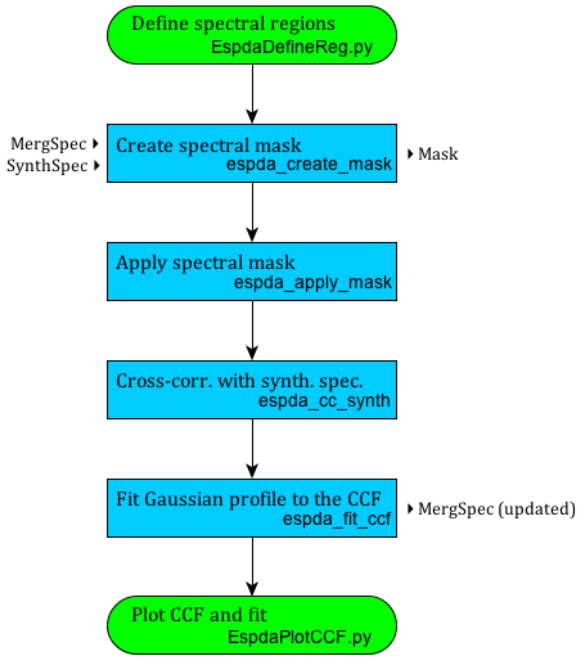


Figure 4.9 - Flowchart of the recipe esp_rv_synth

4.10 *Creation of a list of absorption lines (espda_create_linelist)

This recipe is used to create a list of absorption lines.
In the stellar case, the spectrum is splitted in small regions, which are locally normalized to the continuum. The absorption lines are found using the derivatives of the normalized flux (see 3.3) and are identified by comparison with a table of reference atomic and ionic transitions.
In the quasar case, the spectrum is undersampled to smooth noise fluctuations and then is scanned with a moving window. The presence of a line is verified by the comparison of the mean and the median of the flux and the center of the line is assigned to the pixel with the lower value in the window. It may happen that the same line is identified more than once, in this case duplicates are deleted.
In both the stellar and quasar case the user can add interactively lines that have not been detected automatically.
The processing steps are shown in Figure 4.10.

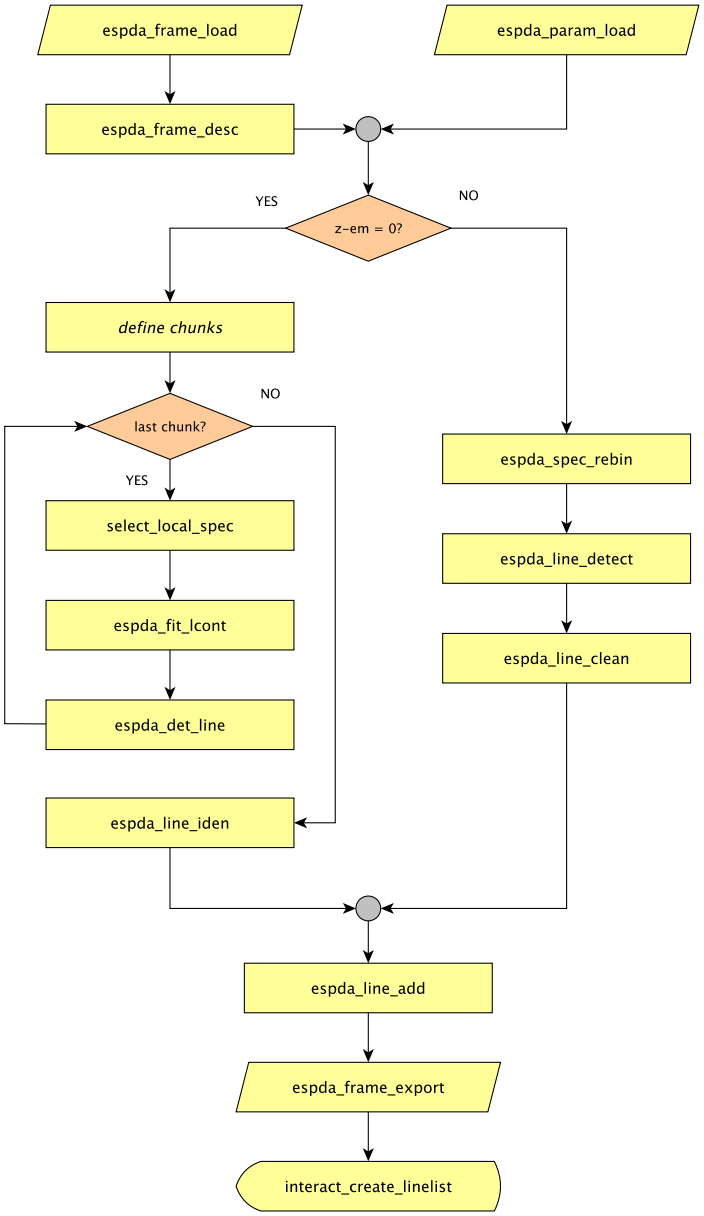


Figure 4.10 - Flowchart of the recipe espda_create_linelist.

4.11 *Fit of continuum level of a QSO spectrum (espda_fit_qsocont)

This recipe fits the continuum component of a quasar spectrum and normalizes the flux to the fitted continuum. We call "continuum" the convolution of the non-thermal component of the spectrum (usually approximated by a power law) and the strong, Doppler-widened emission lines. The continuum is fitted iteratively, starting from a zeroth-order solution, which is used as a reference level to detect the absorption lines in the spectrum. Lines are then detected and removed and the continuum is then re-fitted.
The procedure to determine the continuum level is run on the rebinned spectrum. Only the final result is applied to the full spectrum (containing all the original pixels of all the orders of all the exposures) and validated with a χ^2 test.
The procedure is slightly different for the *blue* and the *red* region of the spectrum, longwards and shortwards of the HI Lyman-α emission, respectively.

- In the *red* region, only sparse absorption lines are found due to ionic transitions of chemical elements heavier than He, and the continuum is easily interpolated between the lines. In this case, a C-Spline fit of the flux is adopted as continuum and all lines are fitted simultaneously.
- The *blue* region is typically crowded by a complex pattern of absorption lines, mainly due to the Lyman-α transition in neutral hydrogen, commonly called the Lyman-α forest. In this case, the flux is initially corrected for the HI optical depth of the forest, to avoid bias, and fitted with a C-spline. Absorption lines are fitted with Voigt profiles and removed starting from the strongest ones, in an iterative process. At each iteration, the correction factor applied initially is recomputed removing the contribution of the fitted lines (see 3.1).

The processing steps (Figure 4.11) are described below.

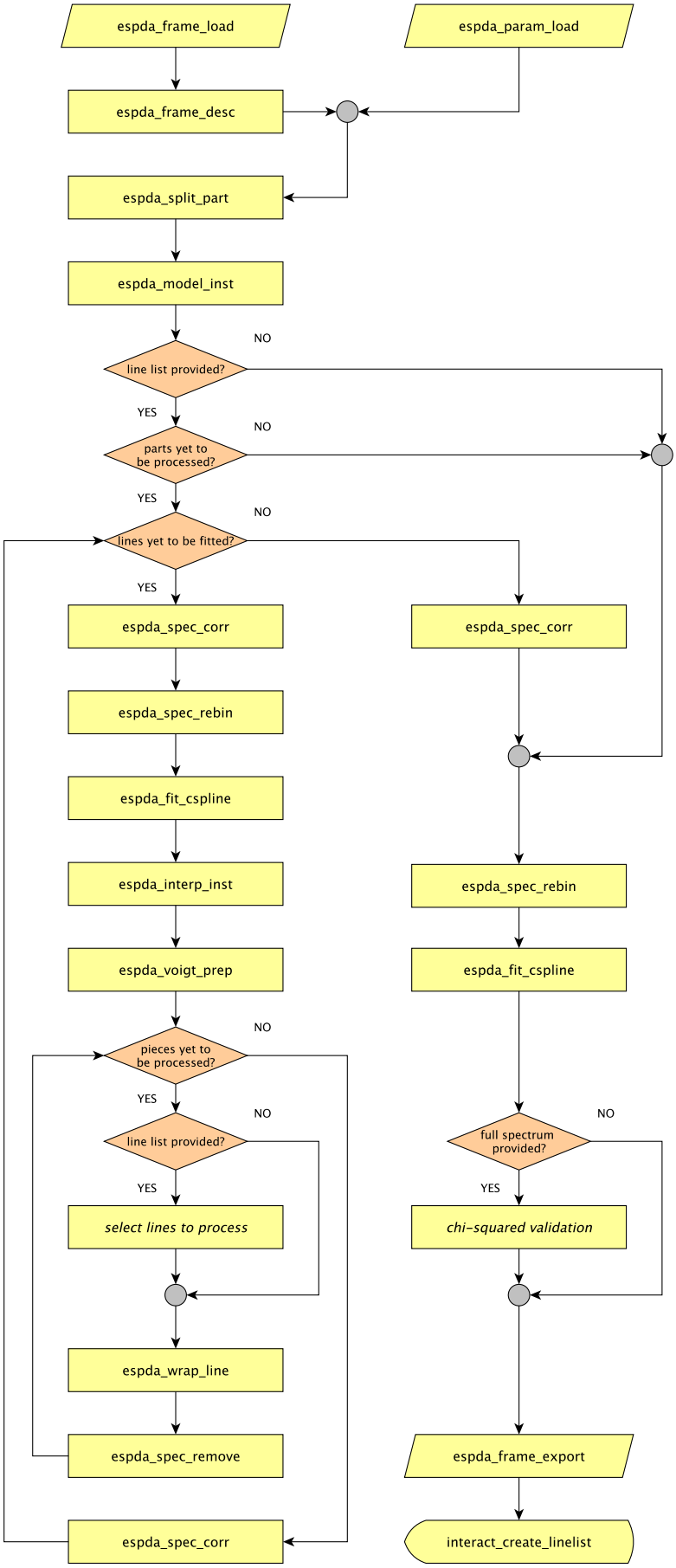


Figure 4.11 - Flowchart of the recipe espda_fit_qsocont.

4.12 *Identification of the absorption systems (espda_iden_syst)

This recipe identifies detected absorption lines with known atomic or ionic transitions and groups them into absorption systems. An absorption system is formed by a group of absorption lines, corresponding to different atomic transitions observed at the "same" redshift.
Two execution stages are foreseen for this recipe, the *identification* and the *refinement* stage.

- In the *identification* stage, the recipe identifies the absorption systems by looking for redshift coincidences. The approach is similar to the one described in [RD-21]. The list of redshifts is computed by dividing the wavelengths of the lines detected in the spectrum by a list of the laboratory wavelengths of the most commonly observed transitions in quasar spectra (e.g. NV 1239, 1243; SiII 1260, 1304, 1526; OI 1302; CII 1334; SiIV 1393, 1402; CIV 1548, 1551; FeII 2344, 2382, 2586, 2600; MgII 2796, 2803; CaII 3935, 3970; NaI 5891, 5897 Å). If two or more coincident redshifts are found, a candidate system is defined and the average redshift is computed. Candidate systems are then checked against a set of selection rules and are either rejected or confirmed. Selection rules will be of two kinds: existence rules, which cause the candidate system to be rejected when violated (e.g. presence of a sufficient number of lines, presence of the Lyman-α line if in the wavelength range, etc.), and acceptance rules, which cause a single line to be rejected when violated (e.g. consistency between the two members of a doublet, etc.).
- In the *refinement* stage, the recipe adds new transitions to the confirmed systems. A larger reference list of laboratory wavelengths for ionic transitions is used at this stage, to identify (in principle) all remaining lines. The recipe checks the regions of the spectrum where lines are expected to be, given the redshift of the input absorption systems, and if lines are found in these regions they are added to the systems and checked again for consistency.

In Reflex, a Python actor is used to plot the results and to let the user confirm or reject the newly added line by visual inspection.
The processing steps are reported in Figure 4.12.

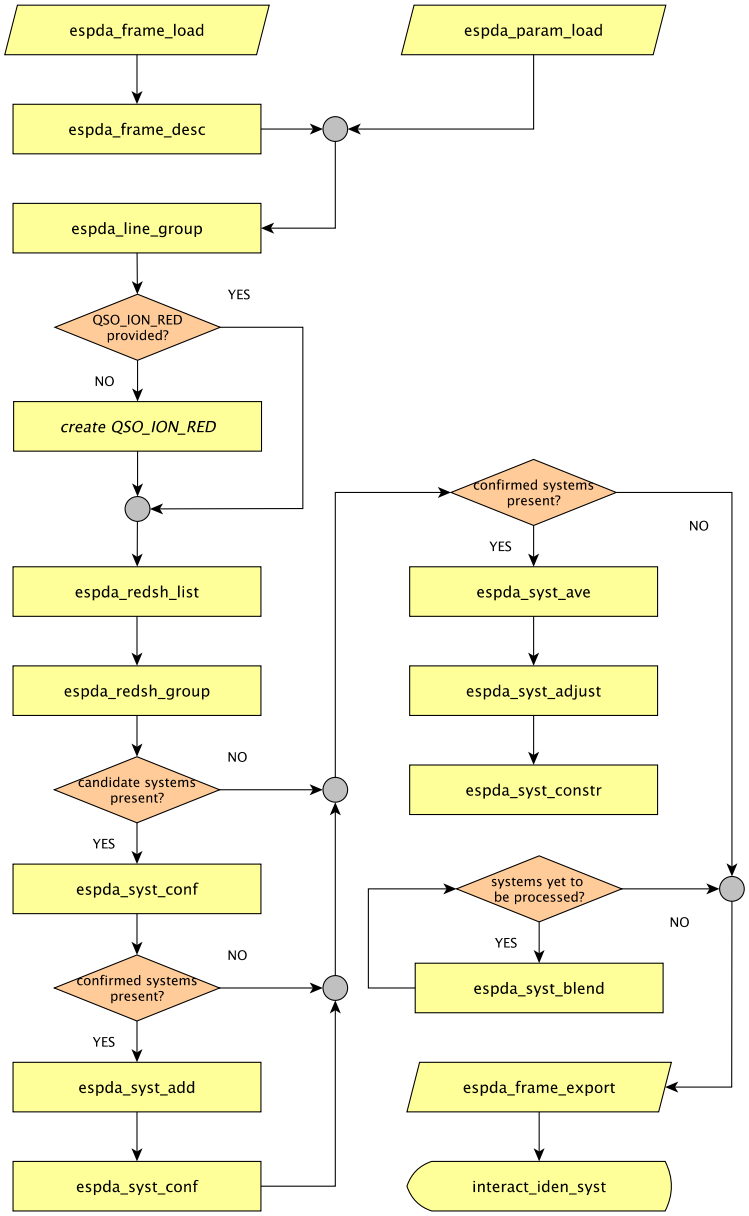


Figure 4.12 - Flowchart of the recipe espda_iden_syst.

4.13 *Voigt-profile fitting of absorption lines (espda_fit_line)

This recipe fits a Voigt profile to an identified absorption system. The collective profile of the lines is determined by minimizing the reduced χ^2 between the normalized flux of the spectrum and the sum of the individual Voigt profiles of the lines, computed as a function of the line parameters (redshift z, column density N, thermal broadening b, and turbulence broadening $b_{\rm tur}$). The user is allowed to put constraints on these parameters, through a Python actor, to keep them fixed during minimization, or to make them common to two or more lines (common parameters are allowed to vary, but they are always kept equal to each other). The user may also add components to improve the quality of the fit. Lines are grouped and fitted together if they are sufficiently close to each other or if they share one or more common parameters. Only the spectral regions around lines are used to compute the reduced χ^2.
When all groups of lines have been fitted, the output parameters are used to compute an overall best-fit profile for the whole spectrum. A Voigt-fitted spectrum and a line list with best-fit

parameters are produced.
The processing steps are described in Figure 4.13.

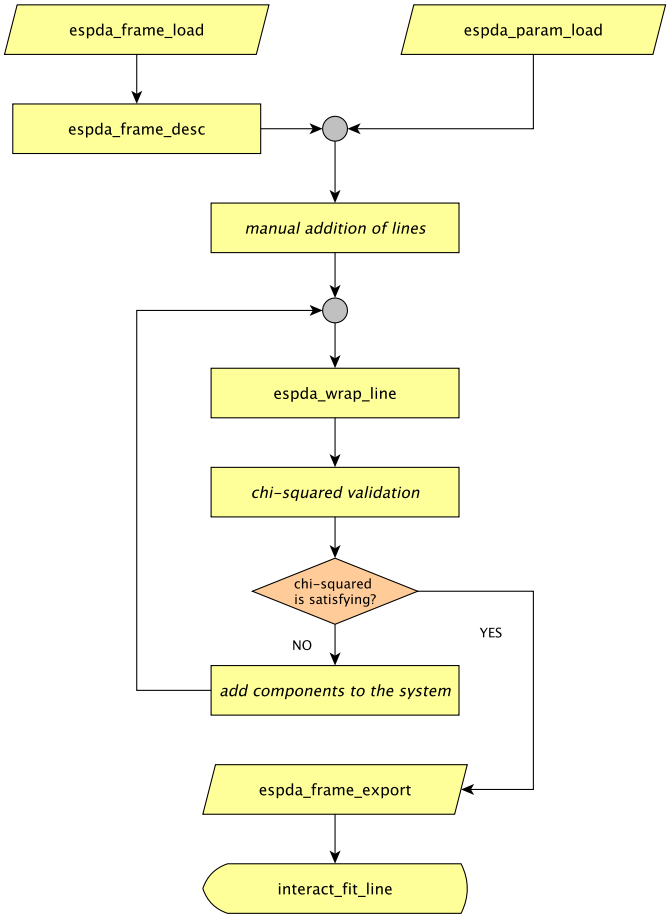


Figure 4.13 - Flowchart of the recipe espda_fit_line.

Chapter 5. Reflex Workflow description

The ESO Recipe Flexible Execution Workbench (Reflex) [AD-3] will provide the primary user interface to the DAS Library. Reflex is an environment, which allows an easy and flexible way to execute VLT pipelines. It is built using the Kepler workflow engine[1], which itself makes use of the Ptolemy II framework[2]. Within Reflex, input data are organized into consistent data sets and processed by "actors", which take care of the recipe execution and save the output according to the prescriptions of the user.
The Reflex formalism will be specifically adapted to the requirements of the DAS Library. The main difference between DA and DR is that the former does not require the injection of new raw data (e.g. a calibration frame or a set of scientific frames) at each step; all operations are triggered simultaneously by the DR output data, and executed in sequence. Another difference is the higher user interactivity required by DA, which is reflected in the widespread use of Python scripts to display the results, set up the parameters and iterate the recipes.
In the DAS, we have conceived four different workflows: one for the analysis of quasar spectra and three for the analysis of star spectra. Figure 5.1 is a screenshot of the current working version of the QSO branch workflow, while Figure 5.2 is a preview of how the star branch I workflow will appear. All workflows are split into three sections:

- Data Organization and Selection: this section includes the standard actors *DataOrganizer* and *DatasetChooser* which retrieve the input frames and organize them into data sets to be analysed;
- Data Processing: this section is the core of the workflows and controls the recipes execution through a set of recipe actors and Python actors;
- Data Storage: this section includes the standard actors *DataFilter*, *ProductRenamer*, and *ProductExplorer*, which collect and save the processed frames in the appropriate location.

In the following sections, we discuss the details of each workflow.

5.1 *Recipe execution

The recipes of both DA workflows are triggered either by the DR output data, or by the output of previous recipes; no raw calibration is required. For this reason, the concept of "purpose" used by Reflex to categorize frames is not relevant for the DA workflows and the OCA rules are consequently simple.
Data are transmitted through the pipe connections displayed in Figure 5.1 and Figure 5.2 in the form of SOF files, which are managed by the standard actors *SofSplitter* and *SofCombiner*. Recipes are contained in composite actors, which include a *RecipeExecutor* and a *RecipeLooper* (when needed), optionally supported by Python actors to to display the input/ouptut data and tune the parameters (see Section 5.3).
The user may choose which actors have to be executed by changing the parameter run_mode in each *RecipeExecutor*. The user may also control complex tasks through the "Task selection" panel in the workflow canvas. Each task enables a consistent cascade of recipes to perform a specific part of the analysis (e.g., continuum fit of a QSO spectrum, determination of the activity indexes from a star spectrum, etc.), skipping all the unnecessary steps.

---

[1] https://kepler-project.org.
[2] http://ptolemy.eecs.berkeley.edu/ptolemyII.

A list of the recipe actors is provided in Table 5.1 and Table 5.2, along with the corresponding recipes and tags of their input/output frames. All the QSO branch recipe actors are currently working, except for actor *Identify systems*; for the time being, actor *Determine continuum* contains only a preliminary implementation of recipe esp_fit_qsocont.

Table 5.1 - QSO branch recipe actors

Actor	Recipe	Input tags	Output tags
Coadd exposures	espda_coadd_spec	SPEC_2D BLAZE_FIBER WAVE_MATRIX_FIBER SPEC_1D RES_MAP_FIBER	MERG_SPEC ORD_SPEC REBIN_SPEC EQ_SPEC
Apply mask	espda_mask_spec	MERG_SPEC ORD_SPEC REBIN_SPEC EQ_SPEC	MERG_SPEC_MASK ORD_SPEC_MASK REBIN_SPEC_MASK EQ_SPEC_MASK MASK
List abs. lines	espda_create_linelist	MERG_SPEC REBIN_SPEC	DET_LINE ID_LFIT NOID_LINE
Determine continuum	esp_fit_qsocont	REBIN_SPEC_MASK EQ_SPEC_MASK DET_LINE	REBIN_SPEC_NORM EQ_SPEC_NORM DET_LINE_NORM
Identify systems	esp_iden_syst	QSO_ATMP_SHORT QSO_ATMP_LONG EQ_SPEC_NORM	SYST_IDEN
Fit Voigt profiles	espda_fit_line	REBIN_SPEC_NORM EQ_SPEC_NORM DET_LINE_NORM SYST_IDEN VOIGT_FUNC QSO_ATMP	REBIN_SPEC_VOIGT VOIGT_LFIT

Table 5.2 - Star branch recipe actors (TBR)

Actor	Recipe	Input tags	Output tags
RadVel	esp_compu_radvel	SPEC_2D BLAZE_FIBER WAVE_MATRIX_FIBER FLUX_TEMPLATE CCF_TEMPLATE	CCF_FIBER BIS_FIBER
ActivityIndexes	esp_compu_rhk	SPEC_2D BLAZE_FIBER WAVE_MATRIX_FIBER	SPEC_2D_ACTIVIND
Coadd exposures	espda_coadd_spec	SPEC_2D BLAZE_FIBER WAVE_MATRIX_FIBER SPEC_1D RES_MAP_FIBER	MERG_SPEC ORD_SPEC REBIN_SPEC EQ_SPEC

Apply mask	espda_mask_spec	ORD_SPEC	ORD_SPEC_MASK MASK
Determine continuum	espda_fit_starcont	ORD_SPEC_MASK	ORD_SPEC_NORM
SyntheticSpectra	esp_synth_spec	ORD_SPEC_NORM WAVE_SYNTH_SPEC FLUX_SYNTH_SPEC	ORD_SPEC_SYNTH
SynthRadVel	esp_rv_synth	ORD_SPEC_SYNTH	ORD_SPEC_RVSYNTH
List abs. lines	espda_create_linelist	ORD_SPEC_RVSYNTH STAR_ATMP	STAR_LINE_DET
Fit Voigt profiles	espda_fit_line	ORD_SPEC_RVSYNTH STAR_LINE_DET	STAR_LINE_FIT
Compute eq. width	espda_compu_eqwidth	ORD_SPEC_MASK	LINE_EQWIDTH
Compute stellar parameters	espda_compu_starpar	LINE_EQWIDTH LINE_RATIO_CALIB LINE_FEH_CALIB	LINE_RATIO_PAR LINE_TEFF_PAR LINE_FEH_PAR

ESPRESSO DAS WORKFLOW - Quasar Branch (v. 0.8.0)

DIRECTORIES

Input directories:

- ROOT_DATA_DIR: ROOT_DATA_PATH_TO_REPLACE
- RAW_DATA_DIR: $ROOT_DATA_DIR/reflex_input/espda
- CALIB_DATA_DIR: CALIB_DATA_PATH_TO_REPLACE/espda-0.6.1

Working Directories:

- BOOKKEEPING_DIR: $ROOT_DATA_DIR/reflex_book_keeping/espda
- LOGS_DIR: $ROOT_DATA_DIR/reflex_logs/espda
- TMP_PRODUCTS_DIR: $ROOT_DATA_DIR/reflex_tmp_products/espda
- BOOKKEEPING_DB: $BOOKKEEPING_DIR/bookkeeping.db

Output directories:

- END_PRODUCTS_DIR: $ROOT_DATA_DIR/reflex_end_products/espda/

PARAMETERS

NB: Parameters in yellow boxes are temporary, only for the UVES/XSH test datasets.

General parameters:

RecipeFailureMode: Ask	In case of recipe failure: 'Continue', 'Stop', 'Ask'
EraseDirs: false	Erase working directories: 'true' or 'false'
FITS_VIEWER: fv	Fits viewer used for inspection: 'fv'
ProductExplorerEnabled: True	Enable Product Explorer: 'True'
ProductExplorerMode: Triggered	Show Product Explorer window: 'Triggered'
SelectDatasetMethod: Interactive	Dataset selection: 'All'/'New', 'Reduced', 'Failed', 'Interactive'
GlobalPlotInteractivity: true	Toggle interactive windows: 'true' or 'false' (not working)

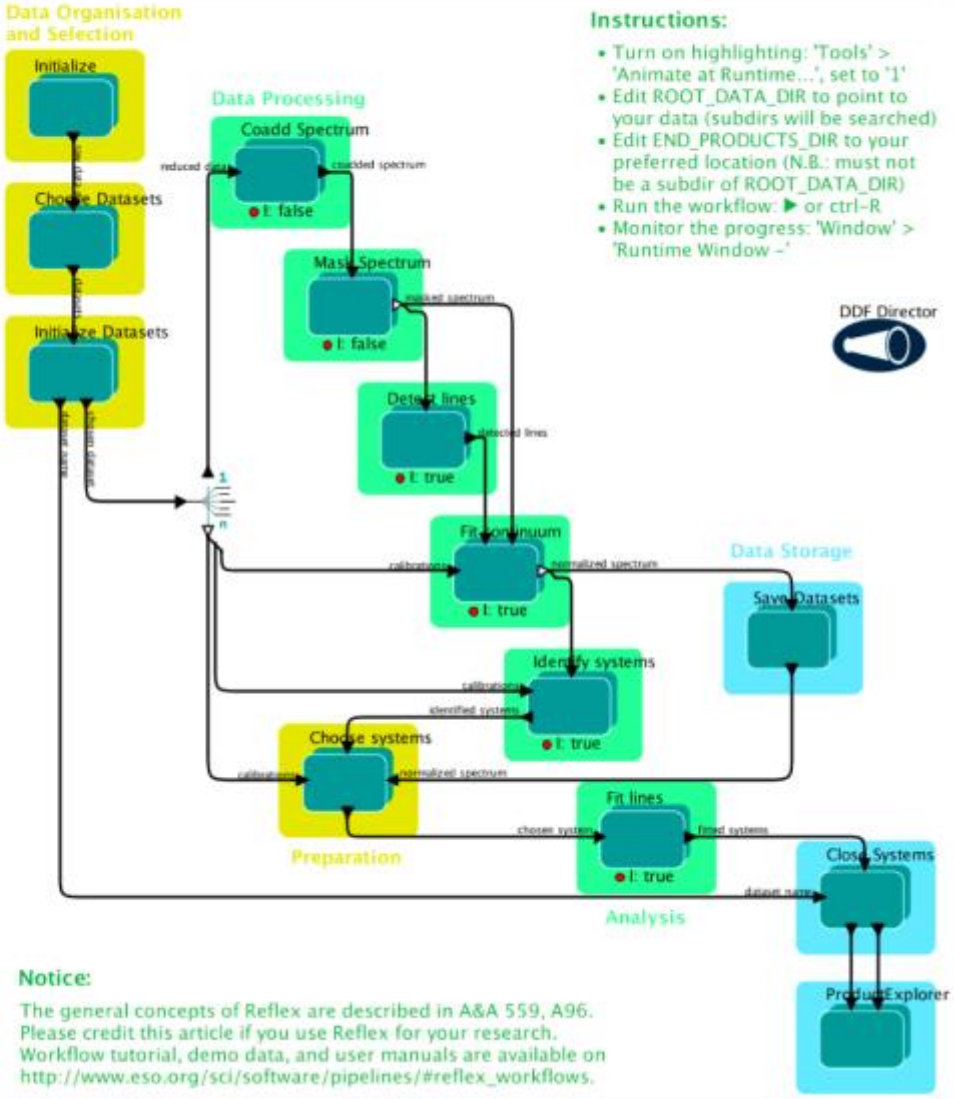
INSTR: UVES	Instrument (UVES/XSH/HARPS)
WAVEL_MIN: 485	Minimum wavelength
WAVEL_MAX: 510	Maximum wavelength

Auxiliary parameters (do not change):

- GLOBAL_TIMESTAMP: 2017-01-16T17:10:47
- END_PRODUCTS_SUBDIR: J0407-4410_flux_reformat/2017-01-16T17:10:47
- ESORexArgs: --suppress-prefix=TRUE
- N_SELECTED_DATASETS: 2
- N_SELECTED_SYSTEMS: 1

VEL_STEP_UVES: 1600	VEL_STEP_XSH: 20000	VEL_STEP_HARPS: 560	GLOB_VEL_STEP: 1600
RESOL_UVES: 60000	RESOL_XSH: 4350	RESOL_HARPS: 115000	GLOB_RESOL: 60000

CASCADE



Notice:

The general concepts of Reflex are described in A&A 559, A96.
Please credit this article if you use Reflex for your research.
Workflow tutorial, demo data, and user manuals are available on
http://www.eso.org/sci/software/pipelines/#reflex_workflows.

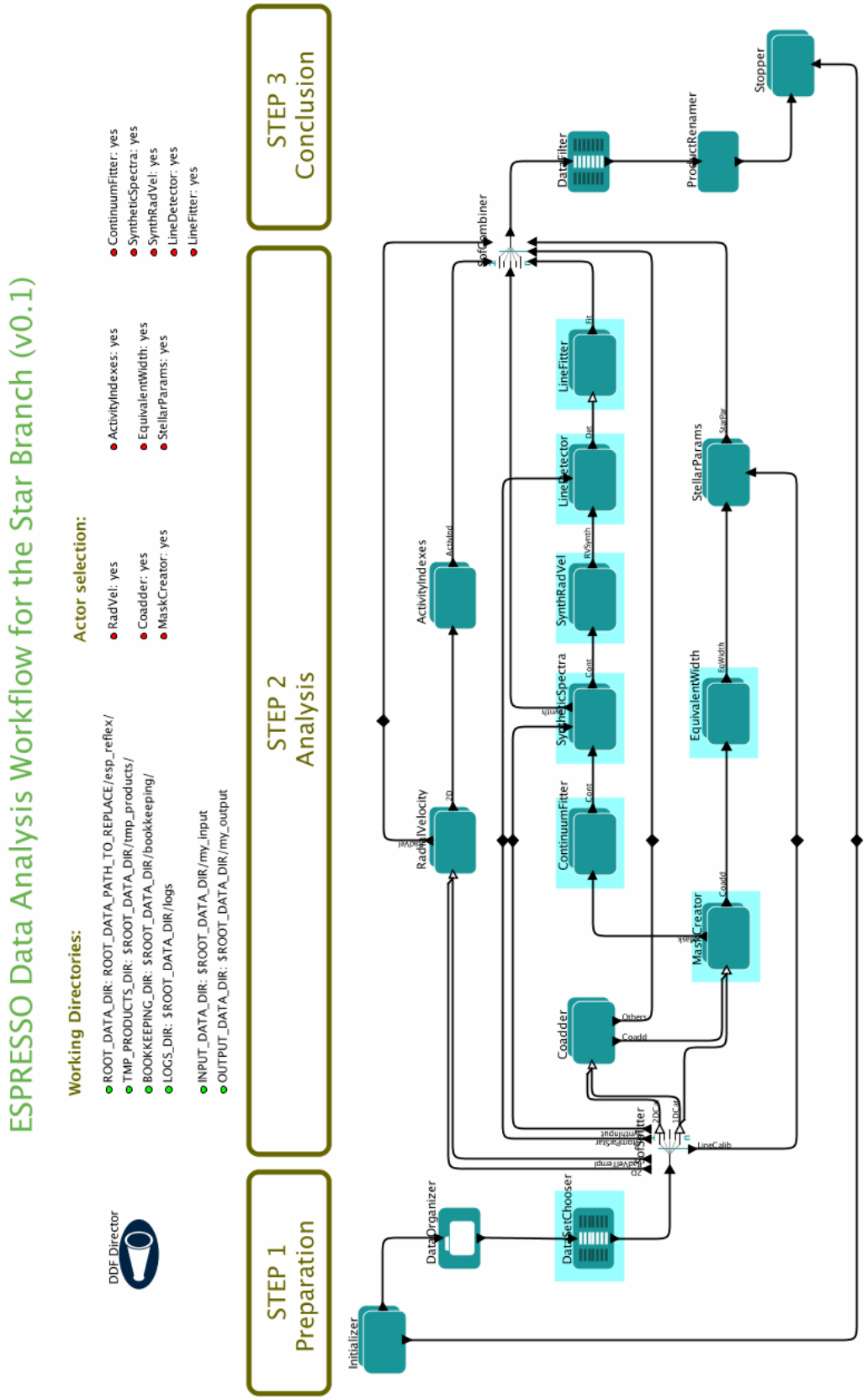


Figure 5.1 - Reflex workflow for the quasar branch

Figure 5.2 - Reflex workflow for the star branch

5.2 User interaction

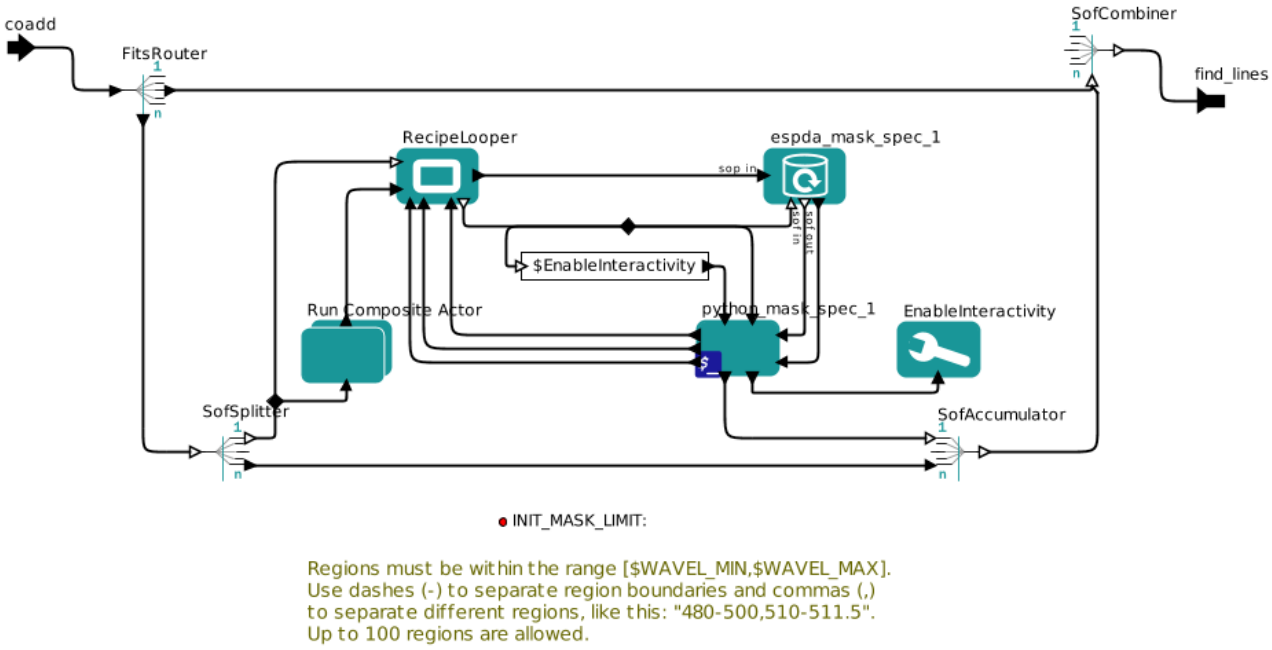
User interaction in the DA workflows is managed by Python scripts based on the standard Reflex Python Library provided by ESO. Those scripts are called by Python actors embedded within the interactive recipe actors. After the recipe is executed, the user has the option to inspect the results and to tune the recipe parameters before iterating the execution. The parameter can be tuned in three ways:

1. by manually editing a text box with the parameter value (*manual mode*);
2. by clicking on the plot (*click mode*);
3. by editing a table (*table mode*).

The click mode is used by recipes which need in input a set of wavelengths (i.e. parameter *mask-limit* in espda_mask_spec, containing the boundaries of the spectral region to be masked; parameter *add* in espda_create_linelist and esp_fit_line, containing the positions of the absorption lines to be added to the line list): instead of typing the values in a text box, the user may select the positions directly on the plot, and the information is automatically translated by Python into a proper value for the corresponding parameter. The table mode is used by recipes which need in input a large set of initial guess values (i.e. parameter *par-edit* in esp_fit_line, encoding the starting values of $(z, N, b, b_{\mathrm{tur}})$ for the absorption lines to be used in the Voigt profile fit): also in this case, the values typed by the user in the table are automatically translated by Python into a proper value for the corresponding recipe parameter.

An example of how interactive Python module will be embedded in the workflows is provided in Figure 5.3 for recipe actor *Apply mask*[3] of the QSO branch workflow. This composite actor contains, in addition to the *RecipeExecutor* (espda_mask_spec_1) and the *RecipeLooper*, a Python actor (python_mask_spec_1), which plots the spectrum and allows the user to define the masked regions in click mode.

A description of the interactive Python scripts for the DA workflows is provided in Table 5.3. A preliminary version of some QSO branch scripts has already been coded.



[3] The figure is meant to provide only a qualitative, not rigorous description of the actor.

Figure 5.3 – Example of recipe actor with embedded Python actors

Table 5.3 - Interactive Python modules

Script	General description
espda_coadd_spec_interactive.py	A spectrum is plotted, showing the coadded spectrum superimposed to the original spectra. The SNR spectrum is also shown.
espda_mask_spec_interactive.py	A coadded spectrum is plotted. The user may define a number of spectral regions via mouse clicks.
espda_create_linelist_interactive.py	A spectrum is plotted. The masked regions are greyed out. Vertical bars highlighting the positions of the detected absorption lines are also plotted. The user may add lines by clicking on the plot.
espda_fit_qsocont_interact.py	A spectrum is plotted. The masked spectral regions are greyed out. Vertical bars highlighting the positions of the detected absorption lines are also plotted. The continuum component is overplotted in red. The normalized spectrum is plotted in a separate panel. (In the current version, the user may add continuum points by clicking on the plot; the recipe computes the continuum by fitting a spline to the points.)
espda_iden_syst_interact.py	A velocity plot of an absorption system is produced. Lines are aligned vertically around the *y*-axis defined by the average redshift of the system.
espda_fit_line_interact.py	A table appears with the guess values of line parameters needed to the Voigt profile fit. The user may edit the table and click OK. A new window appear and a normalized spectrum is plotted. The masked spectral regions are greyed out. Vertical bars highlighting the positions of the detected absorption lines are also plotted. The information from the table is passed to the recipe, which is iterated. (The recipe does not perform any fit of the absoption lines until parameter *fit-flag* is set to 1; when the fit is performed, the resulting Voigt profile is shown, superimposed to the normalized spectrum.)
espda_rv_synth_interact.py	A cross-correlation function is plotted. The Gaussian fit is overplotted in red.

Chapter 6. Input Data Description

Data Analysis is carried out on the 2D and 1D spectra obtained as output of the Data Reduction Pipeline. In this chapter, we describe the data products per science spectrum produced by the DRS that will be the input of the DA workflows, specifying which CPL plugin will use them. Description is taken from the DR Library Design document [AD-2].

6.1 S2D_FIBER

Description:
These are the extracted S2D spectra in [spectral order - extracted pixel] space (one per fiber). They represent the output of the spectrum extraction algorithms that collapse the raw spectrum along the cross-dispersion direction (without resampling). The flux is given in detected photoelectrons.

Generated by: Data reduction workflow

Used by: espda_coadd_spec, esp_compu_radvel, esp_compu_rhk, espda_fit_starcont

Format: FITS image, S2D format [spectral order - extracted pixel space]. First HDU contains the data values, second HDU is the error map and third HDU is the quality map.

6.2 S1D_FIBER

Description:
These are the de-blazed, rebinned and merged S1D spectra (one per fiber). The FITS header contains two keywords defining the linear wavelength scale of the spectrum (starting wavelength and wavelength step). This S1D wavelength grid will be the same for all ESPRESSO spectra. The flux is given in photoelectrons.

Generated by: Data reduction workflow

Used by: espda_coadd_spec, espda_mask_spec, espda_fit_qsocont

Format: FITS image, S1D format (1D frame with uniform wavelength step). First HDU contains the data values, second HDU is the error map and third HDU is the quality map.

6.3 S2D_SKYSUB_FIBER

Description:
This is the sky-subtracted S2D science spectrum in [spectral order - extracted pixel] space. It is generated by subtracting the scaled and resampled sky spectrum (reference fiber) from the science spectrum. The flux is given in detected photoelectrons.

Generated by: Data reduction workflow

Used by: espda_coadd_spec, esp_compu_radvel, esp_compu_rhk, espda_fit_starcont

Format: FITS image, S2D format [spectral order - extracted pixel space]. First HDU contains the data values, second HDU is the error map and third HDU is the quality map.

6.4 S1D_SKYSUB_FIBER

Description:
This is the sky-subtracted S1D science spectrum. It is generated by rebinning and merging the sky subtracted S2D science spectrum. The flux is given in photoelectrons.

Generated by: Data reduction workflow

Used by: espda_coadd_spec, espda_mask_spec, espda_fit_qsocont

Format: FITS image, S1D format (1D frame with uniform wavelength step). First HDU contains the data values, second HDU is the error map and third HDU is the quality map.

6.5 S1D_FLUXCAL_FIBER

Description:
This is the flux-calibrated S1D science spectrum. It is generated by converting the S1D flux into physical units and correcting for the instrumental response. The flux is given in erg/s/cm2/A.

Generated by: Data reduction workflow

Used by: espda_coadd_spec, espda_mask_spec, espda_fit_qsocont

Format: FITS image, S1D format (1D frame with uniform wavelength step). First HDU contains the data values, second HDU is the error map and third HDU is the quality map.

6.6 S1D_SKYSUB_FLUXCAL_FIBER

Description:
This is the sky-subtracted, flux-calibrated S1D science spectrum. It is generated by rebinning and merging the sky-subtracted S2D science spectrum, followed by conversion to physical units and instrumental response correction. The flux is given in erg/s/cm2/A.

Generated by: Data reduction workflow

Used by: espda_coadd_spec, espda_mask_spec, espda_fit_qsocont

Format: FITS image, S1D format (1D frame with uniform wavelength step). First HDU contains the data values, second HDU is the error map and third HDU is the quality map.

6.7 BLAZE_FIBER

Description:
These are the extracted echelle grating blaze functions (one per fiber) in S2D format. They are considered error-free in the sense that all errors incurred during the CAL_FLAT processing are conveyed through the FLAT_FIBER data products.

Generated by: Data reduction workflow

Used by: espda_coadd_spec, esp_compu_radvel, esp_compu_rhk, espda_fit_starcont

Format: FITS image, S2D format (spectral order - extracted pixel space). Single HDU containing the data values.

6.8 WAVE_MATRIX_FIBER

Description:
These are the wavelength calibration arrays (one per fiber) in S2D format, with the wavelength of each extracted pixel stored as data value.

Generated by: Data reduction workflow

Used by: espda_coadd_spec, esp_compu_radvel, esp_compu_rhk, espda_fit_starcont

Format: FITS image, S2D format (spectral order - extracted pixel space). Single HDU containing the wavelengths of the extracted pixels.

6.9 RES_MAP_FIBER

Description:
These are the resolution maps (one per fiber) providing the measured spectral resolution along orders in S2D format. The LINE_TABLE_FIBER data products are used to compute spectral resolution from the FWHM of ThAr or Comb lines. A polynomial fit is performed on the individual measurements to generate a smoothed resolution curve in S2D format.

Generated by: Data reduction workflow

Used by: espda_coadd_spec, espda_mask_spec, espda_fit_qsocont

Format: FITS image, S2D format (spectral order - extracted pixel space). Single HDU containing the data values.

Chapter 7. Data Analysis Library Data Structures TBR

In this chapter, we describe the data format for three different kind of data used by the analysis workflows:

- **Input data**, which are both products of the DR pipeline (described in Chapter 6) and FITS tables with calibration and laboratory data.
- **Intermediate data products** created by the DA functions and temporarily stored during the processing.
- **Final data products** generated by the recipes. They are given in output to the user and contain all the relevant scientific information.

7.1 Input Data Structures

7.1.1 1D merged spectrum (S1D)

General description
This frame contains a science spectrum produced by the DR pipeline in 1D format (i.e. as a merged array).

Structure
Multiextension FITS 1D frame with uniform wavelength step. First HDU contains the data values, second HDU is the error map and third HDU is the quality map.

Data with this structure

Name	Function	Description
SPEC_1D	espda_extract_frame	1D spectrum frame

The different products of the DR pipeline *S1D_FIBER*, *S1D_SKYSUB_FIBER*, *S1D_FLUXCAL_FIBER*, or *S1D_SKYSUB_FLUXCAL_FIBER* are collectively renamed *SPEC_1D* in the context of DA functions.

7.1.2 2D order spectrum (S2D)

General description
This frame contains a science spectrum produced by the DR pipeline in 2D format (i.e. as an order matrix).

Structure
Multiextension FITS frame in the [spectral order - extracted pixel] space. First HDU contains the data values, second HDU is the error map and third HDU is the quality map.

Data with this structure

Name	Function	Description
SPEC_2D	espda_extract_frame esp_correct_flux esp_compu_ccf	2D spectrum frame

	esp_measure_flux	
BLAZE_FIBER	espda_extract_frame esp_compu_ccf	2D blaze frame
WAVE_MATRIX_FIBER	espda_extract_frame esp_shift_wavelength_scale esp_compu_ccf esp_measure_flux	2D wavelength frame

The different products of the DR pipeline *S2D_FIBER*, *S2D_SKYSUB_FIBER* are collectively renamed *SPEC_2D* in the context of DA functions.

7.1.3 List of atomic transitions (ATMP)

General description

This table contains a list of spectral lines corresponding to known atomic transitions. Three such lists are provided within the DAS package. The user may want to create his/her own lists according to the structure below.

Structure

Col	Name	Type	Unit	Description
1	TRANSITION	string	-	Name of the atomic transition
2	LAMBDA	double	nm	Rest frame wavelength in vacuum
3	OSCS	double	–	Oscillator strength
4	DAMP	double	s^{-1}	Natural damping constant
5	MASS	double	g	Mass of the atom

Data with this structure

Name	Function	Description
QSOAtmpShort	esp_list_z	Short list of atomic and ionic transitions for quasar spectra
QSOAtmpLong	esp_list_z esp_add_atom	Long list of atomic and ionic transitions for quasar spectra
QSOAtmp	espda_fit_voigt	Reference list of QSO atomic transitions
StarAtmp	espda_crea_linelsit	Reference list of stellar atomic transitions

7.1.4 Line ratio Calibration Table (LRCT)

General description

This FITS table contains the line-ratios and respective calibrations including the limits of the parameter space where it can be applied.

Structure

Col	Name	Type	Unit	Description
1	ID_FIN	Int	-	Id of final line ratio
2	ID_IN	Int	-	Initial Id of the line ratio
3	STDDEV_FIN	double	-	Final standard deviation for each calibration
4	NSTAR	Int	-	Number of stars used for the calibration of each line ratio (out of a total of 451)
5	WAVE1	double	nm	Wavelength of the first line of the ratio
6	WAVE2	double	nm	Wavelength of the second line of the ratio
7	ELEMENT1	string	-	Chemical element of the first line of the ratio
8	ELEMENT2	string	-	Chemical element of the second line of the ratio

9	C0	double	-	fitting coefficient for each line-ratio calibration
10	C1	double	-	fitting coefficient for each line-ratio calibration
11	C2	double	-	fitting coefficient for each line-ratio calibration
12	C3	double	-	fitting coefficient for each line-ratio calibration
13	TEFF_LOW	double	K	Lower limit in effective temperature of the stars used to calibrate each line ratio
14	TEFF_UP	double	K	Upper limit in effective temperature of the stars used to calibrate each line ratio
15	RATIO_LOW	double	-	Lower limit in ratio value of the stars used to calibrate each line ratio
16	RATIO_UP	double	-	Upper limit in ratio value of the stars used to calibrate each line ratio
17	RATIO_MEAN	double	-	Mean value of each ratio
18	FIT_TYPE	integer	-	Function used for the calibration: the numbers 1 and 2 correspond to the relation "T eff vs. r" – 3 and 4 correspond to the relation "T eff vs. 1/r", and the numbers 5 and 6 correspond to the relations "T eff vs. log(r)". The numbers 2, 4, and 6 correspond to 3rd order polynomial functions and the numbers 1, 3, 5 correspond to linear functions.

Data with this structure

Name	Function	Description
LineRatioCalib	espda_calib_teff	Table with line ratio calibrations

7.1.5 Line FeH calibration Table (LFCT)

General description

This FITS table contains the Fe lines and respective calibrations including the limits of the parameter space where it can be applied.

Structure

Col	Name	Type	Unit	Description
1	ID	Int	-	ID number of the line
2	STDDEV_FIN	double	-	final standard deviation for each calibration
3	NSTAR	Int	-	the number of stars used for line calibration (out of a total of 451)
4	WAVE	double	nm	wavelength of the calibrated line
5	C0	double	-	fitting coefficients for each line
6	C1	double	-	fitting coefficients for each line
7	C2	double	-	fitting coefficients for each line
8	C3	double	-	fitting coefficients for each line
9	C4	double	-	fitting coefficients for each line
10	TEFF_LOW	double	K	Lower limit in effective temperature of the stars used to calibrate each line
11	TEFF_UP	double	K	Upper limit in effective temperature of the stars used to calibrate each line

12	FEH_LOW	double	-	Lower limit in [Fe/H] value of the stars used to calibrate each line
13	FEH_UP	double	-	Upper limit in [Fe/H] value of the stars used to calibrate each line
14	EW_LOW	double	nm	Lower limit in the EW value of the stars used to calibrate each line
15	EW_UP	double	nm	Upper limit in the EW value of the stars used to calibrate each line

Data with this structure

Name	Function	Description
LineFeHCalib	espda_calib_feh	Table with Fe line calibrations

7.1.6 Flux Template (FLTPL)

General description

This is the table containing observed spectral energy distributions of a sample of reference stars with different spectral types spanning late-F to early-M. Suitable stars are solar-metallicity dwarf stars of spectral types F to M, observed at high SNR and at low airmass. To build the flux template, the S2D flux of the star is summed in each spectral order and normalized to one at an arbitrary wavelength (e.g. 550 nm). The flux template is then simply the integrated normalized flux in each spectral order.

Structure

FITS table with one column per specral type (e.g. G2, K0, M2). The number of rows is equal to the number of spectral orders in the S2D spectra.

Col	Name	Type	Unit	Description
1	G0	double		Flux distribution for spectral type G0
2	G2	double		Flux distribution for spectral type G2

Data with this structure

Name	Function	Description
FLUX_TEMPLATE	esp_correct_flux	See description above

7.1.7 CCF Template (CCFTPL)

General description

These are the line masks used by the cross-correlation process (one file per mask). They consist of a list of laboratory wavelengths for the spectral lines of interest, along with their relative depth (contrast) with respect to the continuum. Depths are used as weights in the CCF process since Doppler precision is proportional to line depth. Note that the properties of stellar spectra change relatively slowly with spectral type, so that the use of one mask per main spectral type (G, K, M) does not significantly degrade performances.

Structure

FITS table with two columns: laboratory wavelength (A) and contrast of the spectral line (%). The number of rows is equal to the number of spectral lines in the mask.

Col	Name	Type	Unit	Description
1	WAVEL	double	A	Laboratory wavelength
2	CONTRAST	double	%	Line contrast

Data with this structure

Name	Function	Description
CCF_TEMPLATE_G2	esp_compute_ccf	G2 mask
CCF_TEMPLATE_K5	esp_compute_ccf	K5 mask
CCF_TEMPLATE_M2	esp_compute_ccf	M2 mask

7.1.8 Wavelength grid for synthetic stellar spectrum (WSYNT)

General description
FITS table with the common wavelength grid for the synthetic spectra

Structure

Col	Name	Type	Unit	Description
1	WAVE	double	nm	Wavelength of all 1D synthetic spectra

Data with this structure

Name	Function	Description
WaveSynthSpec	esp_interp_synth	Wavelength grid for 1D synthetic spectra

7.1.9 Flux for synthetic stellar spectrum (FSYNT)

General description
FITS table with the flux for a given the synthetic spectra

Structure

Col	Name	Type	Unit	Description
1	SFLUX	double		Flux of 1D synthetic spectra

Data with this structure

Name	Function	Description
FluxSynthSpec	esp_interp_synth	Flux for 1D synthetic spectra

7.1.10 Tabulated Voigt function (VGTF)

General description
This table contains the values of the Voigt function computed on a grid of (a, u) values (see Section 3.4). It has no particular structure; the cell of the table at column i and row j is simply

the value of the Voigt function, $V(a_i, u_j)$, where a_i and u_j are computed from a proper sampling of the allowed intervals for the two parameters on which the function is defined. One such table is provided within the DAS package. The user may want to create his/her own table.

7.2 Intermediate Data Structure

7.2.1 Spectral mask (MASK)

General description
This table contains the boundaries of a set of spectral regions to be masked. It can be created interactively, clicking on the plot of the spectrum, or it can be provided by the user.

Structure
Table with two columns. The number of rows corresponds to the number of masked regions.

Col	Name	Type	Unit	Description
1	*MREGSTART*	double	nm	Start wavelength of the masked region
2	*MREGEND*	double	nm	End wavelength of the masked region

Data with this structure

Name	Function	Description
Mask	espda_create_mask espda_apply_mask	Table with start and end wavelengths of the masked regions

7.2.2 List of redshifts (REDS)

General description
This table is created by the recipe esp_iden_syst from the list of detected lines in a quasar spectrum and the list of atomic transitions in table *AtomParShort* (see Section 7.1.3). The table is updated as the absorption systems are identified.

Structure
FITS table. Variable number of rows.

Col	Name	Type	Unit	Description
1	*WAVEL*	double	nm	Wavelength
2	*WAVEL_ERR*	double	nm	Formal error on *WAVEL*
3	*PEAK*	double	–	Normalized peak flux
4	*PEAK_ERR*	double	–	Formal error on *PEAK*
5	*EW*	double	nm	EW
6	*EW_ERR*	double	nm	Formal error on *EW*
7	*FWHM*	double	nm	FWHM
8	*FWHM_ERR*	double	nm	Formal error on *FWHM*
9	*LINEID*	string	–	Name of the atomic transition
10	*LINERANK*	string	–	Ranking of the line identification
11	*LINEZ*	double	–	Individual redshift of the line
12	*LINEZ_ERR*	double	–	Formal error on *LINEZ*

13	*SYST_ID*	integer	–	Identification number of the system
14	*SYST_RANK*	string	–	Ranking of the system identification

Data with this structure

Name	Function	Description
Redshift	esp_list_z esp_group_z esp_sele_syst esp_add_atom	Generic redshift list

7.3 Final Data Products

7.3.1 *List of lines (LINE)

General description

A list of lines (LINE) is a CPL table with information about absorption lines. Three types of lists are used: basic line list (LINE_BASIC), line list prepared for Voigt-profile fitting (LINE_VOIGT), and line list with information about the absorption systems (LINE_SYST).
A list of redshifts is a special list of lines (LINE_SYST) produced by matching a list of lines with Voigt profiles (LINE_VOIGT) with a list of ionic transitions (ION). It contains all possible redshift values for the lines (*SYSTID* column is not filled).
A list of absorption systems is a special list of lines (LINE_SYST) produced from a list of redshift by applying specific rules for existence or acceptance. It contains lines associated with absorption systems (*SYSTID* column is filled).

Structure

The columns of LINE_BASIC are:

Name	Type	Description
PEAK	double	Peak value of the flux density [erg cm^{-2} s^{-1} nm^{-1}].
WAVEL	double	Wavelength [nm].

LINE_VOIGT has all the columns of LINE_BASIC plus the following additional columns:

Name	Type	Description
WAVELERR	double	Error on wavelength [nm].
WAVELMIN	double	Minimum of the fitted wavelength range [nm].
WAVELMAX	double	Maximum of the fitted wavelength range [nm].
LINEID	string	ID.
GRPID	integer	ID of the group.
LINEZ	double	Redshift.
LINEZERR	double	Error on redshift.
LINEZMIN	double	Minimum redshift for fitting.
LINEZMAX	double	Maximum redshift for fitting.
LINEZCON	string	Constraint on redshift for fitting.
COLDEN	double	Column density [cm^{-2}, logarithm].
COLDENERR	double	Error on column density [cm^{-2}, logarithm].
COLDENMIN	double	Minimum column density for fitting [cm^{-2}, logarithm].
COLDENMAX	double	Maximum column density for fitting [cm^{-2}, logarithm].
COLDENCON	string	Constraint on column density for fitting [cm^{-2}, logarithm].

THERB	double	Thermal broadening [km s^{-1}].
THERBERR	double	Error on thermal broadening [km s^{-1}].
THERBMIN	double	Minimum thermal broadening for fitting [km s^{-1}].
THERBMAX	double	Maximum thermal broadening for fitting [km s^{-1}].
THERBCON	string	Constraint on thermal broadening for fitting [km s^{-1}].
TURBB	double	Turbulence broadening [km s^{-1}].
TURBBERR	double	Error on turbulence broadening [km s^{-1}].
TURBBMIN	double	Minimum turbulence broadening for fitting [km s^{-1}].
TURBBMAX	double	Maximum turbulence broadening for fitting [km s^{-1}].
TURBBCON	string	Constraint on turbulence broadening for fitting [km s^{-1}].
USE	string	Flag to use the line in fitting.
CONT	double	Fitted flux density of the emission continuum [erg cm^{-2} s^{-1} nm^{-1}].
NORM	double	Flux density normalized to the emission continuum (*PEAK*/*CONT*).

LINE_SYST has all the columns of LINE_VOIGT plus the following additional columns:

Name	Type	Description
SYSTID	string	System ID.
ORDER	integer	Sort order.

Functions handling this structure
LINE_BASIC:

7.3.2 *Spectrum (SPEC)

General description
A spectrum (SPEC) is a CPL table with information about the flux density as a function of wavelength. The rows of the table are also called pixels, especially when they are referred to non-rebinned spectra, where each row corresponds one-to-one to a pixel of the detector. Four types of spectra are used: basic spectrum (SPEC_BASIC), spectrum with information about flux RMS or "rebinned spectrum" (SPEC_RMS), spectrum with information about orders and exposures or "full spectrum" (SPEC_FULL), fitted spectrum (SPEC_FIT).

Structure
The columns of SPEC_BASIC are:

Name	Type	Description
WAVEL	double	Wavelength [nm].
MASK	boolean	Flag for masked rows (deprecated).
PIXSIZE	double	Pixel size [nm].
FLUX	double	Flux density [erg cm^{-2} s^{-1} nm^{-1}].
FLUXERR	double	Error on flux density [erg cm^{-2} s^{-1} nm^{-1}].

SPEC_RMS has all the columns of SPEC_BASIC plus the following additional column:

Name	Type	Description
FLUXRMS	double	RMS of flux density [erg cm^{-2} s^{-1} nm^{-1}].

SPEC_FULL has all the columns of SPEC_BASIC plus the following additional columns:

Name	Type	Description
SPECID	integer	ID of the exposure.

ORDID	integer	ID of the order.
EQFACT	double	Equalization factor.

SPEC_FIT has all the columns of SPEC_RMS and SPEC_FULL plus the following additional columns:

Name	Type	Description
FIT	double	Fitted flux density of the absorption lines [erg cm^{-2} s^{-1} nm^{-1}].
CONT	double	Fitted flux density of the emission continuum [erg cm^{-2} s^{-1} nm^{-1}].
NORM	double	Flux density normalized to the emission continuum (*FLUX*/*CONT*).
NORMERR	double	Error on normalized flux density (*FLUXERR*/*CONT*).

Functions handling this structure
TBC

7.3.3 Summary file (SUMM)

General description
Text file containing a summary of the performed operations in human-readable format

Structure
TBC

Data with this structure

Name	Function	Description
Summary	esp_sele_syst	Generic summary

7.3.4 Teff Table (TEFF)

General description
This FITS table contains the result obtained for the estimation of the effective temperature for solar-type stars using EW line-ratio calibrations.

Structure

Col	Name	Type	Unit	Description
1	TEFF	double	(K)	Effective Temperature
2	STD_TEFF	double	(K)	Standard deviation of the nCalib calibrations results
3	N_CALIB	integer		Total number of calibrations used
4	N_CALIB_IND	integer		Number of independent line-ratios

Data with this structure

Name	Function	Description
TeffTable	espda_calib_teff espda_calib_feh	Table with Teff, Teff error estimates and number of used line ratios

7.3.5 FeH Table (FEH)

General description

This FITS table contains the result obtained for the estimation of the [Fe/H] for solar-type stars using EW Fe line calibrations.

Structure

Col	Name	Type	Unit	Description
1	FEH	double	dex	[Fe/H]
2	STD_FEH	double	dex	Standard deviation of the individual [Fe/H]
3	N_CALIB	int		Total number of used calibrations

Data with this structure

Name	Function	Description
FeHTable	espda_calib_feh	Table with Fe/H, error estimates and number of used line calibrations

7.3.6 Cross-Correlation Function (CCF)

General description

These are the cross-correlation functions (CCF) in spectral order - radial velocity space (one per fiber). The FITS header contains two keywords defining the linear RV scale of the CCF (starting RV and RV step in km/s). The CCF flux is given in detected photoelectrons. The FITS frame contains one additional line, which contains the summed CCF over all spectral orders. This is the final CCF that is used to compute the final RV and CCF parameters.

Structure

FITS image, pseudo-S2D format (spectral order - radial velocity space). First HDU contains the data values, second HDU is the error map, third HDU is the quality map.

Data with this structure

Name	Function	Description
CCF_A	esp_compute_ccf	CCF fiber A
CCF_B	esp_compute_ccf	CCF fiber B

7.3.7 CCF Bisector (BIS)

General description

These are the CCF bisectors (one per fiber), which provide the radial velocity of the mid-points of the CCF as a function of CCF depth, from the continuum level down to the CCF bottom. The FITS header contains two keywords defining the normalized CCF depth scale (starting depth and step). The bisector values are given in km/s.

Structure

FITS image, pseudo-S1D format (1D frame with uniform step in CCF depth). First HDU contains the data values, second HDU is the error map, third HDU is the quality map.

Data with this structure

Name	Function	Description
BIS_A	esp_compute_bis	CCF bisector fiber A
BIS_B	esp_compute_bis	CCF bisector fiber B

7.3.8 Interpolated synthetic spectrum (SYNT)

General description
FITS table with the 1D interpolated synthetic spectrum

Structure

Col	Name	Type	Unit	Description
1	WAVE	double	nm	Wavelength array of the synthetic spectrum
2	FLUX	double	-	Normalized flux of the synthetic spectrum

Data with this structure

Name	Function	Description
SynthSpec	esp_interp_synth esp_rebin_lin	Synthetic 1D spectrum

Chapter 8. Data Analysis Library Functions

In the following sections, all the functions appearing in the functional diagrams of Chapter 4 are described. The main functions called by other functions are also described here, we refer to them as *secondary functions*.

8.1 Function espda_check_name

Working remarks
None.

Purpose
Check the names of the frames.

Calling recipes
This is a secondary function.

Arguments

Name	Type	Description
set	CPL frame set	A frame set.
cfg	CPL recipe configuration	Recipe configuration object.
cfg_tag	string	Configuration tag.

Return
CPL error code.

General description
This function scans a frame set *set* and checks if the frames with the expected tags exist. The tags are taken from the recipe configuration object *cfg*, defined with its configuration tag *cfg_tag*.

Mathematical description
None.

Error/warning conditions
If *set*, *cfg*, or *cfg_tag* are NULL, an error is issued. If a required frame does not exist, an error is issued. If an optional frame does not exist and verbosity level is greater than 0, a warning is issued.

Unit test
Error conditions are checked.

8.2 Function espda_check_num

Working remarks
None.

Purpose
Check the number of the frames.

Calling recipes
This is a secondary function.

Arguments

Name	Type	Description
set	CPL frame set	A frame set.
cfg	CPL recipe configuration	Recipe configuration object.
cfg_tag	string	Configuration tag.

Return
CPL error code.

General description
This function scans a frame set *set* and checks the number of the frames with the expected tags. The tags are taken from the recipe configuration object *cfg*, defined with its configuration tag *cfg_tag*.

Mathematical description
None.

Error/warning conditions
If *set*, *cfg*, or *cfg_tag* are NULL, an error is issued. If there are too few required frames with a given tag, an error is issued. If there are no optional frames with a given tag and verbosity level is greater than 0, a warning is issued.

Unit test
Error conditions are checked.

8.3 Function espda_check_tag

Working remarks
None.

Purpose
Check the tags of the frames.

Calling recipes
This is a secondary function.

Arguments

Name	Type	Description
set	CPL frame set	A frame set.
cfg	CPL recipe configuration	Recipe configuration object.
cfg_tag	string	Configuration tag.

Return

CPL error code.

General description
This function scans a frame set *set* and checks the tags of the frames. The tags are taken from the recipe configuration object *cfg*, defined with its configuration tag *cfg_tag*.

Mathematical description
None.

Error/warning conditions
If *set*, *cfg*, or *cfg_tag* are NULL, an error is issued. If a tag is wrong the frame is not used. In this case, if verbosity level is greater than 0 a warning is issued, suggesting a possible substitution for the wrong tag. The substitution is based on a three-way comparison between the wrong tag and the expected tags, and may not always be accurate.

Unit test
Error conditions are checked.

8.4 Function espda_dfs_catg (TBC)

8.5 Function espda_dfs_groups (TBC)

8.6 Function espda_dfs_license (TBC)

8.7 Function espda_fit_cspline

Working remarks
None.

Purpose
Fit a cubic spline to a spectrum.

Calling recipes
espda_fit_qsocont (Section 4.11 and 9.11), espda_fit_line (Section 4.13 and 9.13).

Arguments

Name	Type	Description
ref	CPL table	A reference spectrum (SPEC_FIT).
col	string	Column with the fit.
spec	CPL table	Spectrum to which the fit is applied (SPEC_FIT).
line	CPL table	List of lines to which the fit is applied (LINE_VOIGT; may be NULL).

Return
CPL error code.

General description

This function takes a reference spectrum *ref*, fits a cubic spline on its flux, and applies the fitting curve to another spectrum *spec* and (optionally) to a list of lines *line*. *ref* is typically an undersampled version of *spec*. The fit is saved as a separate column *col* in *ref*, *spec*, and (optionally) *line*. Wavelengths not strictly contained in the range of *ref* are flagged as invalid and not fitted.

Mathematical description
The cubic spline is computed by the GSL package gsl_spline. If *ref* has only two rows, a two-point interpolation on flux values is performed; if it has only one row, the flux value is taken as fit value.

Error conditions and handling
If the *ref* or *spec* are NULL, an error is issued. If *col* does not exist, an error is issued. If *line* is NULL and verbosity level is greater than 0, a warning is issued.

Unit test
This test creates a mock reference spectrum consisting of a sine wave of given angular frequency (sampled at 10 wavelengths across the ESPRESSO range). A cubic spline is fitted on the reference spectrum and applied to another sine-wave spectrum (sampled at 100 wavelengths across the ESPRESSO range) and to a list of 10 lines with peak values distributed also as a sine wave. The difference between the sine wave and the fitted cubic spline is checked to be within the tolerance in both the spectrum and the line list. Error conditions are also checked.

8.8 Function espda_fit_voigt

Working remarks
None.

Purpose
Fit Voigt profiles to absorption lines.

Calling recipes
This is a secondary function

Arguments

Name	Type	Description
spec_over	CPL table	Spectrum over-chunk (SPEC_FIT).
spec_chunk	CPL_table	Spectrum chunk (SPEC_FIT).
line_chunk	CPL table	List of lines in the spectrum chunk (LINE_VOIGT).
prof_chunk	CPL table	Instrument profile in the spectrum chunk.
ion	CPL table	List of ionic transitions (ION).
voigt_func	CPL table	Tabulated Voigt function (VOIGT).
rchisq	CPL array	Reduced chi-square from fitting.
iter_arr	CPL array	Number of fitting iterations.
iter_max	integer	Maximum number of iterations.
spec_out	CPL table	Fitted spectrum chunk (SPEC_FIT; must be NULL).
line_out	CPL table	List of fitted lines (LINE_VOIGT; must be NULL).

Return
CPL error code.

General description
This function takes a spectrum chunk *spec_chunk* and a list of lines *line_chunk*, fits the lines with Voigt profiles, and produces a fitted spectrum *spec_out* and a list of lines with fitted Voigt parameters *line_out*. The Voigt fitting is based on a previous identification of the lines that are to be fitted and takes the wavelength, the oscillator strength, and the damping parameter of the identified lines from a list of ionic transitions *ion*. The spectral chunk is convolved with the instrument profile *prof_chunk* (a slightly-larger over-chunk *spec_over* is used instead of *spec_chunk* to avoid border effects). An optional table *voigt_func* with previously-computed values of the Voigt function can be used to speed up the computation. The fitting procedure is terminated when either the algorithm converges or a maximum number of iterations is reached *iter_max*.
The function behaves differently depending on external variables. Four modes are available:

1. standard fit on a rebinned spectrum (*grp_max* ≠ "MAX", *cycle_flag* = "rebinned"): fitting is performed and results are saved in *spec_out* and *line_out*;
2. skipped fit on a non-rebinned spectrum (*grp_max* ≠ "MAX", *cycle_flag* ≠ "rebinned"): fitting is skipped and results from a previous fitting of the rebinned spectrum are saved in *spec_out* and *line_out*;
3. last run on a rebinned spectrum (*grp_max* = "MAX", *cycle_flag* = "rebinned"): fitting is skipped; the rebinned *spec_out* is copied from *spec_chunk* and a reduced chi-square is computed by applying on *spec_chunk* the profile obtained the Voigt parameters in *line_chunk*;
4. last run on a non-rebinned spectrum (*grp_max* = "MAX", *cycle_flag* ≠ "rebinned"): same as 3, but on a non-rebinned *spec_chunk* instead.

The values of reduced chi-square from fitting and the number of fitting iterations are saved in the 4-element arrays *rchisq* and *iter_arr*, respectively, depending on the chosen mode (values from mode 1 are saved in the first element, etc.).

Mathematical description
The fitting is performed by the GSL package gsl_multifit_nlin using the Levenberg-Marquardt algorithm. For details on the convolution, see espda_spec_conv.

Error/warning conditions
If *spec_over*, *spec_chunk*, *line_chunk*, *prof_chunk*, or *ion* are NULL, an error is issued. If *voigt_func* is NULL and verbosity level is greater than 0, a warning is issued.

Unit test
Error conditions are checked. (TBC)

8.9 Function espda_frame_adjust

Working remarks
None.

Purpose
Adjust the number of frames.

Calling recipes
This is a secondary function.

Arguments

Name	Type	Description
set	CPL frame set	Frame set.
tag	string	Frame tag.
max	integer	Maximum number of frames allowed for the given tag.

Return
The number of frames adjusted to the maximum.

General description
This function counts how many frames in a frame set *set* have the tag *tag* and adjust the number of frames to be used to a maximum allowed value *max*.

Mathematical description
None.

Error/warning conditions
If *set* or *tag* are NULL, an error is issued. If the number of frames with the given tag exceeds *max* and the verbosity level is greater than 0, a warning is issued.

Unit test
Error conditions are checked.

8.10 Function espda_frame_desc

Working remarks
This function may be demoted to a macro in a future release.

Purpose
Read a frame descriptor.

Calling recipes
espda_coadd_spec (Section 4.1 and 9.1), espda_mask_spec (Section 4.2 and 9.2), espda_create_linelist (Section 4.10 and 9.10), espda_fit_qsocont (Section 4.11 and 9.11), espda_iden_syst (Section 4.12 and 9.12), espda_fit_line (Section 4.13 and 9.13).

Arguments

Name	Type	Description
set	CPL frame set	Frame set.
tag	string	Frame tag.
name	string	Name of the descriptor
...	argument list	Pointer to the descriptor

Return
The number of frames adjusted to the maximum.

General description
This function reads a descriptor *name* from a frame with tag *tag* in a frame set *set*. A pointer to the descriptor is issued and may be dereferenced to an integer, a double, or a string variable.

Mathematical description
None.

Error/warning conditions
If *set*, *tag*, or *name* are NULL, an error is issued. If the descriptor does not exist and the verbosity level is greater than 0, a warning is issued.

Unit test
Error conditions are checked.

8.11 Function espda_frame_export

Working remarks
This function may be demoted to a macro in a future release.

Purpose
Export a table into a frame.

Calling recipes
espda_coadd_spec (Section 4.1 and 9.1), espda_mask_spec (Section 4.2 and 9.2), espda_create_linelist (Section 4.10 and 9.10), espda_fit_qsocont (Section 4.11 and 9.11), espda_iden_syst (Section 4.12 and 9.12), espda_fit_line (Section 4.13 and 9.13).

Arguments

Name	Type	Description
set	CPL frame set	Frame set.
parlist	CPL parameter list	List of parameters.
table	CPL table	Table to be exported (may be NULL).
proplist	CPL property list	Property list with QC parameters
recipe	string	Name of the recipe.
tag	string	Tag of the frame.
name	string	Filename of the frame.

Return
CPL error code.

General description
This function exports a table *table* into a FITS frame with tag *tag* andfilename *name*. The user must provide the original frame set *set*, the list of parameters *parlist* used by the recipe, and the name of the recipe itself *recipe*, as required by cpl_dfs_save_table. The descriptors in the property list *proplist* are appended to the frame header, together with the DAS PRO.TYPE descriptor ("ANALYZED") and PRO.CATG descriptor (equal to *tag*).

Mathematical description

None.

Error/warning conditions
If *set*, *parlist*, *proplist*, *recipe*, *tag*, or *name* are NULL, an error is issued. If *table* is NULL and the verbosity level is greater than 0, a warning is issued.

Unit test
Error conditions are checked.

8.12 Function espda_frame_extract (TBC)

8.13 Function espda_frame_import (TBC)

8.14 Function espda_frame_load

Working remarks
None.

Purpose
Load a frame from a frame set.

Calling recipes
espda_mask_spec (Section 4.2 and 9.2), espda_create_linelist (Section 4.10 and 9.10), espda_fit_qsocont (Section 4.11 and 9.11), espda_iden_syst (Section 4.12 and 9.12), espda_fit_line (Section 4.13 and 9.13).

Arguments

Name	Type	Description
set	CPL frame set	Frame set.
cfg	CPL recipe configuration	Recipe configuration object.
cfg_tag	string	Configuration tag.
pos	integer	Position of the tag in the recipe configuration object.

Return
A table containing the frame.

General description
This function scans a frame set *set*, finds the only frame with tag *tag* and extracts it into a table. The tags are taken from the recipe configuration object *cfg*, defined with its configuration tag *cfg_tag*. The position *pos* of the tag ranges from -1 (corresponding to the configuration tag) to *N*, with *N* the number of tags in the recipe configuration object. The function is not suitable to scan frame sets containing multiple frames with the same tag.

Mathematical description
None.

Error/warning conditions

If *set*, *cfg*, or *cfg_tag* are NULL, an error is issued. If *set* contains multiple frames with the same *tag* and the verbosity level is greater than 0, a warning is issued.

Unit test
Error conditions are checked.

8.15 Function espda_frame_size

Working remarks
None.

Purpose
Find the size of frames.

Calling recipes
This is a secondary function.

Arguments

Name	Type	Description
set	CPL frame set	Frame set.

Return
The size of the frames.

General description
This function scans a frame set *set* and returns the size of the frames. If the frames have different sizes, the smallest size is returned.

Mathematical description
None.

Error/warning conditions
If *set* is NULL, an error is issued. If the size of a frame exceeds the returned size and the verbosity level is greater than 0, a warning is issued.

Unit test
Error conditions are checked.

8.16 Function espda_interp_bilin

Working remarks
None.

Purpose
Perform bi-linear interpolation.

Calling recipes
This is a secondary function.

Arguments

Name	Type	Description
x_a	integer	1st abscissa.
x_b	integer	2nd abscissa.
y_a	integer	1st ordinate.
y_b	integer	2nd ordinate.
v_aa	double	Image of (*x_a*, *y_a*).
v_ba	double	Image of (*x_b*, *y_a*)
v_ab	double	Image of (*x_a*, *y_b*).
v_bb	double	Image of (*x_b*, *y_b*),
x_0	double	Interpolation abscissa.
y_0	double	Interpolation ordinate.

Return
The image of (*x_0*, *y_0*).

General description
This function computes the bi-linear interpolation across four points (*x_a*, *y_a*, *v_aa*), (*x_b*, *y_a*, *v_ba*), (*x_a*, *y_b*, *v_ab*), and (*x_b*, *y_b*, *v_bb*) in a 3-d space, given the interpolation position (*x_0*, *y_0*, *v_0*). It is meant to be used to compute the value of a tabulated 2-d function at an arbitrary position.

Mathematical description
The linear interpolation formula is

$$v\_0 = \frac{v\_aa(x\_b - x\_0)(y\_b - y\_0) + v\_ba(x\_0 - x\_a)(y\_b - y\_0) + v\_ab(x\_b - x\_0)(y\_0 - y\_a) + v\_bb(x\_0 - x\_a)(y\_0 - y\_a)}{(x\_b - x\_a)(y\_b - y\_a)}.$$

It reduces to a linear interpolation when *x_a* = *y_b* or *y_a* = *y_b*.

Error/warning conditions
None.

Unit test
This function computes the bi-linear interpolation across four points (*x_a*, *y_a*, *v_aa*), (*x_b*, *y_a*, *v_ba*), (*x_a*, *y_b*, *v_ab*), and (*x_b*, *y_b*, *v_bb*) in a 3-d space, given the interpolation position (*x_0*, *y_0*, *v_0*). *v_aa*, *v_ba*, *v_ab*, and *v_bb* are defined by a 2-d function *test_func* as *v_aa* = *test_func*(*x_a*, *y_a*), *v_ba* = *test_func*(*x_b*, *y_a*), *v_ab* = *test_func*(*x_a*, *y_b*), *v_bb* = *test_func*(*x_b*, *y_b*). The difference between *v_0* and the expected value is checked to be within the globally-defined tolerance *TOL*.

8.17 Function espda_interp_inst (TBC)

8.18 Function espda_interp_lin

Working remarks
None.

Purpose
Perform linear interpolation.

Calling recipes
This is a secondary function.

Arguments

Name	Type	Description
x_a	integer	1st abscissa.
x_b	integer	2nd abscissa.
v_a	double	Image of *x_a*.
v_b	double	Image of *x_b*,
x_0	double	Interpolation abscissa.

Return
The image of *x_0*.

General description
This function computes the linear interpolation across two points (*x_a*, *v_a*) and (*x_b*, *v_b*) in a 2-d space, given the interpolation position *x_0*. It is used to compute the value of a tabulated 1-d function at an arbitrary position.

Mathematical description
The linear interpolation formula is

$$v\_0 = v\_a + (v\_b - v\_a)\frac{x\_0 - x\_a}{x\_b - x\_a}.$$

It reduces to *v_0* = *v_a* = *v_b* when *x_a* = *x_b*.

Error/warning conditions
None.

Unit test
This test computes the linear interpolation across two points (*x_a*, *v_a*) and (*x_b*, *v_b*) in a 2-d space, given the interpolation position *x_0*. *v_a* and *v_b* are defined by a 1-d function *test_func* as *v_a* = *test_func*(*x_a*), *v_b* = *test_func*(*x_b*). The difference between *v_0* and the expected value is checked to be within the globally-defined tolerance *TOL*.

8.19 Function espda_line_add

Working remarks
None.

Purpose
Add lines to a line list.

Calling recipes
espda_create_linelist (Section 4.10 and 9.10).

Arguments

Name	Type	Description
line	CPL table	List of lines (LINE BASIC; may be NULL).
add	string	Information about the lines to be added.

Return
None.

General description
This function adds manually-selected absorption lines to a line list *line*. The information about the lines to be added is passed through a string parameter *add* consisting in a comma-separated list of wavelength-peak values, which are appended to the *WAVEL* and *PEAK* columns of *line*.

Mathematical description
None.

Error/warning conditions
If *line* or *add* are null and the verbosity level is greater than 0, a warning is issued.

Unit test
This test creates a mock list of just one line and adds to it a set of *add_size* lines, with wavelengths *add_wavel* and peak values *add_peak*. The values in the columns *WAVEL* and *PEAK* in the output line list are checked to be consistent with the expected values within the globally-defined tolerance *TOL*.

8.20 Function espda_line_clean

Working remarks
None.

Purpose
Clean duplicates from a line list.

Calling recipes
espda_create_linelist (Section 4.10 and 9.10).

Arguments

Name	Type	Description
line	CPL table	List of lines (LINE BASIC).
vel_step	double	Velocity step to define duplicates [m s^{-1}].

Return
CPL error code.

General description
This function finds duplicate entries in a line list *line* and remove them. Duplicates are defined as lines separated by less than 1.01 times the velocity step *vel_step*.

Mathematical description
None.

Error/warning conditions
If *line* is null, an error is issued.

Unit test
This test creates a list of three lines: two with the exact same wavelength and peak value and one with a slightly different parameters. A velocity step *vel_step* is used to clean the line list from duplicates. Depending on the value of *vel_step*, the third line may or may not be cleaned. In all cases, the results of cleaning are checking to be consistent with what expected within the globally-defined tolerance *TOL*. Error conditions are also checked.

8.21 Function espda_line_detect

Working remarks
None.

Purpose
Detect absorption lines.

Calling recipes
espda_create_linelist (Section 4.10 and 9.10).

Arguments

Name	Type	Description
spec	CPL table	Spectrum (SPEC BASIC).
width	double	Width of the spectrum chunks [nm].
overlap	double	Overlap of the spectrum chunks.
thres	double	Threshold for line detection.

Return
A list of lines (LINE_BASIC).

General description
This function splits a spectrum *spec* into chunks, checks the flux statistics in chunks and detects an absorption line whenever the average flux in a chunk is significantly lower than the median flux. Chunks have a width *width* and overlap each other by a fraction *overlap* of *width*. The difference between the mean and the median is considered significant if it exceeds a threshold *thres*. The line is placed at the wavelength of lowest flux.

Mathematical description
The number of chunks is obtained as

$$N_{\text{chunks}} = \frac{\lambda_{\max} - \lambda_{\min}}{width \times (1 - overlap)},$$

where $\lambda_{\min}$ and $\lambda_{\max}$ are the minimum and maximum wavelength of the spectrum. The range of the i-th chunk is defined as

$$C_i = [\lambda_{\min} + (i-1) \times width \times overlap, \lambda_{\min} + i \times width \times overlap].$$

Error/warning conditions
If *spec* is null, an error is issued.

Unit test
This test creates a mock spectrum with a number *line_size* of lines. Lines have a Gaussian profile as defined by their center wavelengths *cen*, standard deviations *stdev*, and peak values *peak*. The number of lines detected is checked to be equal to *line_size* and the values in the columns *WAVEL* and *PEAK* of the output line list are checked to be consistent with *cen* and *peak* within an appropriate tolerance. Error conditions are also checked.
The tolerance on the wavelength is equal to one pixel size. The tolerance on the peak value is

$$tol_{peak} = G[0; \max(stdev), 1.0] - G[\Delta\lambda; \max(stdev), 1.0 - \max(peak)],$$

where $G(\lambda;\ \sigma, A)$ is a Gaussian function of the wavelength λ with standard deviation σ and peak value A, and $\Delta\lambda$ is the pixel size.

8.22 Function espda_line_fill

Working remarks
None.

Purpose
Fill columns with Voigt parameters.

Calling recipes
This is a secondary function.

Arguments

Name	Type	Description
line_in	CPL table	List of lines (LINE_BASIC).
ion	CPL table	List of ionic transitions (ION).
hwidth	double	Half-width of the allowed wavelength range [nm].
z_em	double	Emission redshift of the quasar.
par_range	double	Ranges for the Voigt parameters.
line_out	CPL table	A new filled list of lines (LINE_VOIGT).

Return
CPL error code.

General description
This function takes a line list *line_in* and creates a duplicate line list *line_out* filled with guess Voigt parameters for fitting. The additional columns in *line_out* are filled as follows:

1. *LINEID*: Ly_a for lines in the Lyman-alpha forest, CIV_1548 for lines in the CIV forest, and MgII_2803 for lines with greater wavelength (*z_em* is used to compute the regions);
2. *WAVELMIN*: *WAVEL* – *hwidth*;
3. *WAVELMAX*:*WAVEL* + *hwidth*;
4. *LINEZ*: computed using the reference list of ionic transitions *ion*;
5. *LINEZMIN*: @a LINEZ - @a par_range[0];

6. *LINEZMAX*: @a LINEZ + @a par_range[0];
7. *LINEZCON*: V;
8. *COLDEN*: modeled on peak value for Ly_a lines, 13.0 otherwise;
9. *COLDENMIN*: *par_range*[1];
10. *COLDENMAX*: *par_range* [2];
11. *COLDENCON*: V;
12. *THERB*: 20.0 for Ly_a lines, 10.0 otherwise;
13. *THERBMIN*: *par_range* [3];
14. *THERBMAX*: *par_range* [4];
15. *THERBCON*: V;
16. *THERB*: 0.0;
17. *THERBMIN*: *par_range* [5];
18. *THERBMAX*: *par_range* [6];
19. *THERBCON*: F;
20. *USE*: Y.

Mathematical description
For details on the modeling of the column density, see espda_model_colden.

Error/warning conditions
If *line_in* or *ion* are null, an error is issued.

Unit test
This tests creates a mock list of lines with length *size* and wavelengths *wavel*. Peak values for the lines cannot be freely chosen and are set to 0.5 because the column density is modeled from them and it is not easily computable without calling the modeling function, espda_model_colden (we aimed to keep the unit test of the modeling function separate from this one). The list is filled with Voigt parameters, depending on the emission redshift *z_em*. Parameters are defined with ranges given by *width* and *par_range*. The output list is checked to be consistent with what expected within the globally defined tolerance *TOL* (the tolerance on column *COLDEN* is higher – 1e-4 – because the reference value is fixed and not obtained from espda_model_colden). Error conditions are also checked.

8.23 Function espda_line_group

Working remarks
This function previously was used also to set up the redshift of the lines for Voigt profile fitting. This is probably not advisable, because the redshift range is updated at each iteration in the Ly-alpha forest. The relevant code is still present and commented, though, because it may be re-enforced in the future.

Purpose
Define groups in a line list.

Calling recipes
espda_iden_syst (Section 4.12 and 9.12).

Arguments

Name	Type	Description
line	CPL table	List of lines (LINE_VOIGT).

Return
CPL error code.

General description
This function scans a list of lines *line* and groups the lines that must be fitted together. Lines are grouped if their wavelength ranges available for fitting (defined by columns *WAVELMIN* and *WAVELMAX*) overlap, or if specific constraints on their Voigt parameters (defined in columns *LINEZCON*, *COLDENCON*, *THERBCON*, and *TURBBCON*) have been set. Groups of lines are defined by incremental integer values in column *GRPID*.

Mathematical description
None.

Error/warning conditions
If *line* is null, an error is issued.

Unit test
This tests creates a mock list of 11 lines with the following parameters:

#	*LINEID*	*WAVEL*	*WAVELMIN*	*WAVELMAX*	*LINEZCON*	*COLDENCON*	*THERBCON*	*TURBBCON*
0	Ly_a	405.00	404.60	405.40	V	V	V	F
1	CIV_1548	515.78	515.28	516.28	V	V	V	F
2	CIV_1550	516.64	516.14	517.14	V	V	V	F
3	Ly_a	408.00	407.60	408.40	C1	V	V	F
4	Ly_a	409.00	408.60	409.40	C1	V	V	F
5	Ly_a	410.00	409.60	410.40	V	C2	V	F
6	Ly_a	411.00	410.60	411.40	V	C2	V	F
7	Ly_a	412.00	411.60	412.40	V	V	C3	F
8	Ly_a	413.00	412.60	413.40	V	V	C3	F
9	Ly_a	414.00	413.60	414.40	V	V	V	C4
10	Ly_a	415.00	414.60	415.40	V	V	V	C4

Lines 1, and 2 are expected to be grouped because their wavelength ranges overlap. Lines 3 and 4, 5 and 6, 7 and 8, and 9 and 10 are expected to be grouped because they have a constraint in one of the Voigt parameters. Lines are grouped and the results are checked to be equal to the expectations. Error conditions are also checked.

8.24 Function espda_line_iden

Working remarks
None.

Purpose
Identify lines in a stellar spectrum.

Calling recipes

espda_create_linelist (Section 4.10 and 9.10).

Arguments

Name	Type	Description
line_in	CPL table	List of lines (LINE_BASIC).
ion	CPL table	List of ionic transitions (ION).
wdiff	double	Wavelength difference for matching lines [nm].
line_id	CPL table	List of identified lines (LINE_BASIC).
line_rej	CPL table	List of non-identified lines (LINE_BASIC).
id_num	integer	Number of identified lines.
rej_num	integer	Number of non-identified lines.

Return
CPL error code.

General description
This function takes a list of absorption lines *line* from a stellar spectrum, compares it with a list of ionic transitions *ion* and find the correspondences within a given wavelength difference *wdiff*. If a correspondence is found, the line has been identified. A list of identified lines *line_id* and a list of non-identified *line_rej* lines are produced. Lines IDs and wavelengths are copied from the input list to the list of identified lines. The number of identified lines *id_num* and the number of non-identified lines *rej_num* are also computed.

Mathematical description
None.

Error/warning conditions
If *line* or *ion* are null, an error is issued.

Unit test
This test create a mock list with *size* lines and matches it with a mock list of ionic transitions. The identification of the lines is performed as many times as the number of lines in the list, changing the value of the wavelength difference used for matching in a way that one line is identified the first time, two the second time, etc. The number of identified lines and the number of rejected lines are checked to be equal to what expected. Error conditions and handling are also checked.
The wavelength of the *l*-th line in the list is defined as

$$\lambda_l = \lambda_{\min} + \frac{l+1}{size-2}(\lambda_{\max} - \lambda_{\min})$$

while the wavelength of the *l*-th ionic transition in the list is defined as

$$\lambda_l = \lambda_{\min} + \frac{l+1}{size-2}(\lambda_{\max} - \lambda_{\min}) + 10^{-2} l(-1)^{-l},$$

with $\lambda_{\min}$ and $\lambda_{\max}$ the minimum and the maximum wavelength available for ESPRESSO, respectively. In this way, at the iteration *i* it is possible to identify *i* lines by using the following wavelength difference to match lines with ionic transitions:

$$wdiff = 10^{-2}(i+1) - 5 \times 10^{-3}.$$

8.25 Function espda_line_ion

Working remarks
None.

Purpose
Extract information about ionic transitions.

Calling recipes
This is a secondary function.

Arguments

Name	Type	Description
line_in	CPL table	List of lines (LINE_BASIC).
ion	CPL table	List of ionic transitions (ION).
ion_wavel	double	Wavelengths of the ions [nm].
ion_oscs	double	Oscillator strengths of the ions.
ion_damp	double	Damping parameters of the ions.

Return
CPL error code.

General description
This function takes a list *line* of identified absorption lines and for each line it extracts from a list of ionic transitions *ion* the wavelength *ion_wavel*, the oscillator strength *ion_oscs*, and the damping parameter *ion_damp* of the corresponding ion. These parameters are given in output as arrays with the same length of *line*.

Mathematical description
None.

Error/warning conditions
If *line* or *ion* are null, an error is issued.

Unit test
This test creates a mock list with *size* lines having IDs *id*. A mock list of three ionic transitions with IDs Ly_a, CIV_1548, and CIV_1550 is also created (for this reason the choice of *id* is restricted to these ions). The information about the ionic transitions corresponding to the lines is extracted from the latter list and checked to be consistent with what expected within the globally-defined tolerance *TOL*. The size of the output arrays is checked to be equal to *size*. Error conditions are also checked.

8.26 Function espda_mask_apply

Working remarks
None.

Purpose
Apply a spectral mask.

Calling recipes
espda_mask_spec (Section 4.2 and 9.2).

Arguments

Name	Type	Description
spec	CPL table	Spectrum (may be NULL).
mask	CPL table	Spectral mask (may be NULL).
row_num	integer	Number of masked rows.

Return
CPL error code.

General description
This function applies a spectral mask *mask* to a spectrum *spec*. Within the masked regions the spectrum wavelengths are flagged as invalid. If *spec* contains multiple exposures, only those specified in the mask are actually masked. The number of masked rows *row_num* is also computed. The application of a mask is additive, i.e. it preserves any other mask previously applied.

Mathematical description
None.

Error/warning conditions
If *spec* or *mask* are NULL and the verbosity level is greater than 0, a warning is issued.

Unit test
This test creates a mock spectrum with *nexp* exposures of size *size*. A mock mask with *nreg* spectral region with wavelength ranges defined by *mreg_start* and *mreg_end* is applied to the spectrum. The masked spectrum is checked not to contain information in the masked regions.

8.27 Function espda_mask_create

Working remarks
None.

Purpose
Create a spectral mask.

Calling recipes
espda_mask_spec (Section 4.2 and 9.2).

Arguments

Name	Type	Description
exp	integer	IDs of the exposures (may be NULL).
lim	double	Limits of the regions (may be NULL).

Return
A spectral mask (MASK).

General description
This function creates a spectral mask from two vectors *exp* and *lim*, respectively containing the IDs of the exposures to be masked and the wavelengths limits of the region to be masked.

Mathematical description
None.

Error/warning conditions
If *exp* and *lim* do not have the same even number of elements, an error is issued. If *exp* or *lim* are NULL and the verbosity level is greater than 0, a warning is issued.

Unit test
This test creates a mock spectral mask from two vectors *exp* and *lim*. The values in the *EXP* column of the mas is checked to be equal to what expected, while the values in the *MREGSTART* and *MREGEND* column are checked to be consistent with what expected within the globally-defined tolerance *TOL*. Error conditions are also checked.

8.28 Function espda_model_colden

Working remarks
The function is provisional. Better modelizations will be implemented in the future.

Purpose
Model the column density of a Lyman-alpha line.

Calling recipes
This is a secondary function.

Arguments

Name	Type	Description
peak	double	Peak value of the density flux [erg cm^-2 s^-1 nm^-1].
cont	double	Continuum values at peak [erg cm^-2 s^-1 nm^-1].

Return
The approximate column density [cm^-2, logarithm].

General description
This function estimates the column density of a Ly-alpha line from the continuum-normalized flux at line peak *peak*/*cont*.

Mathematical description
The column density is computed from a 5th degree polynomial approximation of the curve of growth of HI, assuming that the Doppler broadening is 20 km/s:

$$\log(N_{\mathrm{HI}}) \approx 14.096 - 4.6251f + 18.657f^2 - 46.229f^3 + 53.301f^4 - 23.442f^5$$

with $f = peak/cont$.

Error/warning conditions

None.

Unit test
This test models the column densities of *num* lines with peak values *peak* and continuum values *cont*. The results are checked to be consistent with the reference column densities @a colden_ref within the globally defined tolerance TOL.

8.29 Function espda_model_inst

Working remarks
The function is provisional. It will be adapted to the actual instrument profile when it is available.

Purpose
Model the instrument profile.

Calling recipes
espda_fit_qsocont (Section 4.11 and 9.11).

Arguments

Name	Type	Description
size	double	Size of the instrument profile.
resol	double	Resolution of the instrument.
kappa	double	Number of FWHM used to truncate the gaussian.

Return
The instrument profile (INST_PROF).

General description
This function models the instrument profile as a gaussian with size *size*, resolution *resol*, and peak equal to one. It is saved in column PROF of the output table. The gaussian is centered on the median row of the table and its FWHM is *size*/*kappa*. Columns *VEL* and *VELRED* are also computed.

Mathematical description
The Gaussian profile is defined at the *i*-th pixel as

$$G[i] = \exp\left\{-\frac{1}{2}\left[\frac{i - 1/2\,(size - 1)}{size} \times \frac{kappa \times \mathrm{FWHM}}{\sigma}\right]^2\right\}$$

with i ranging from 0 to $size - 1$. Here σ is the standard deviation and $\mathrm{FWHM} = c/resol$ is the full-width at half-maximum, c being the speed of light. FWHM/σ is fixed for a Gaussian profile and is approximately equal to 2.345820045. *VEL* and *VELRED* are computed as follows:

$$VEL = \frac{i - 1/2\ (size - 1)}{size} \times kappa \times \mathrm{FWHM},$$

$$VELRED = \frac{i - 1/2\ (size - 1)}{size} \times \frac{kappa}{resol}.$$

Error/warning conditions
None.

Unit test
This test models the instrument profile and checks that the number of rows with *PROF* higher than half of the peak value are consistent with what expected (under the assumption that the profile size is *kappa* times the FWHM) within a 1-pixel tolerance.

8.30 Function espda_model_transm (TBC)

8.31 Function espda_model_voigt (TBC)

8.32 Function espda_param_extract

Working remarks
None.

Purpose
Extract a string parameter into an array.

Calling recipes
espda_mask_spec (Section 4.2 and 9.2), espda_fit_line (Section 4.13 and 9.13).

Arguments

Name	Type	Description
param	string	String parameter.
sep	string	Separator to parse string entries.
start	integer	Starting string entry.
step	integer	Step to parse string entries.
type	CPL type	Type of the array.

Return
An array with the string entries.

General description
This function parses a string parameter *param* consisting in a list of entries identified by the separator *sep*. The parsing starts from entry *start* and uses a step *step*. An array of type *type* is produced from the extracted entries.

Mathematical description
None.

Error/warning conditions
If *param* or *sep* are NULL, an error is issued.

Unit test

This test creates three mock string parameters with *size* entries. The entries are respectively of type integer, double, and string and are defined by *int_ref*, *double_ref*, and *string_ref*. The string parameters are parsed and the results are checked to be equal to what expected (for integer and string entries) or to be consistent with what expected within the globally-defined tolerance *TOL*. Error conditions are also checked.

8.33 Function espda_param_load

Working remarks
None.

Purpose
Load a parameter from a parameter list.

Calling recipes
espda_mask_spec (Section 4.2 and 9.2), espda_create_linelist (Section 4.10 and 9.10), espda_fit_qsocont (Section 4.11 and 9.11), espda_iden_syst (Section 4.12 and 9.12), espda_fit_line (Section 4.13 and 9.13).

Arguments

Name	Type	Description
list	string	Parameter list.
name	string	Name of the parameter
...	argument list	Pointer to the parameter value and optional range.

Return
CPL error code.

General description
This function scans a parameter list *list* and finds the parameter *name*. A pointer to the parameter value is issued and may be dereferenced to an integer, a double, or a string variable. If the parameter is an integer or a double, the range must be specified after the pointer as a pair of values (minimum and maximum respectively). If the parameter value falls outside this range, it is changed to the nearest extreme of the range itself.

Mathematical description
None.

Error/warning conditions
If *list* or *name* are NULL, an error is issued. If an integer or double parameter has been changed to be within the allowed range and the verbosity level is greater than 0, a warning is issued.

Unit test
This test creates a mock parameter list with three parameters, respectively of type integer, double, and string. Values and ranges of the integer and double parameters are given by *value_int*, *min_int*, *max_int*, *value_double*, *min_double*, and *max_double*. The parameters are loaded and the results are checked to be equal to what expected (for the integer parameter) or consistent with what expected within the globally defined tolerance *TOL* (for the double parameter). Error conditions are also checked.

8.34 Function espda_param_replace

Working remarks
None.

Purpose
Replace a pattern in a string parameter.

Calling recipes
espda_mask_spec (Section 4.2 and 9.2), espda_fit_line (Section 4.13 and 9.13).

Arguments

Name	Type	Description
param	string	String parameter.
find	string	Pattern to find.
repl	string	Replacement.
out	string	String parameter with the pattern replaced.

Return
CPL error code.

General description
This function finds a pattern *find* in a parameter *param*, and replaces it with a replacement *repl*. A parameter *out* with the pattern replaced is produced.

Mathematical description
None.

Error/warning conditions
If *param*, *find*, or *repl* are NULL, an error is issued.

Unit test
This test creates a mock string parameter and replaces a pattern in it. The result is checked to be equal to what expected. Error conditions are also checked.

8.35 Function espda_print_covar

Working remarks
None.

Purpose
Print the covariance matrix.

Calling recipes
This is a secondary function

Arguments

Name	Type	Description
covar	GSL matrix	Covariance matrix.

par_vary	GSL vector	Varying parameters.

Return
CPL error code.

General description
This function prints the covariance matrix *covar* (computed during the chi-square minimization of a Voigt profile based on the varying parameters *par_vary*) as a table in the standard output. Each table entry contains both the actual value and the normalized value (in parentheses) of one matrix element.

Mathematical description
The covariance matrix is computed by the GSL package gsl_multifit. *covar* is a $M \times M$ matrix and *par_vary* is a M-d vector, where $M = 4L - C$, with L the number of lines (each fitted with 4 Voigt parameters) and C the number of constraint among parameters.

Error/warning conditions
If *covar* or *par_vary* are NULL, an error is issued.

Unit test
Error conditions are checked. A mock covariance matrix is printed together with a reference for visual inspection.

8.36 Function espda_print_line

Working remarks
None.

Purpose
Print the result of Voigt-profile fitting.

Calling recipes
This is a secondary function.

Arguments

Name	Type	Description
line	CPL table	List of lines (LINE_VOIGT).

Return
CPL error code.

General description
This function prints a table in the standard output listing the Voigt parameters from a list of fitted lines *line*. Parameters are listed with their fitting error and are taken from columns *WAVEL* and *WAVELERR*, *LINEZ* and *LINEZERR*, *COLDEN* and *COLDENERR*, *THERB* and *THERBERR*, and *TURBB* and *TURBBERR* of *line*.

Mathematical description
None.

Error/warning conditions
If *line* is NULL, an error is issued.

Unit test
Error conditions are checked. A mock list of lines is printed together with a reference for visual inspection.

8.37 Function espda_print_par

Working remarks
None.

Purpose
Print the Voigt parameters.

Calling recipes
This is a secondary function.

Arguments

Name	Type	Description
solverx	GSL vector	Solver vector.
par_vary	GSL vector	Varying parameters.
par_lim	CPL table	Range of varying parameters.

Return
CPL error code.

General description
This function prints in the standard output two vectors:

1. *par_vary*, which contains the initial varying parameters to define a Voigt profile;
2. *solverx*, which contains the fitting parameters computed during chi-square minimization.

Depending on the globally-defined transformation mode TRANSF_MODE, both vectors may be transformed to correctly enforce ranges for parameters, as defined by the columns *MIN* and *MAX* of *par_lim* (see function espda_voigt_transf). In this case, the transformed values are also printed in parentheses.

Mathematical description
The solver vector is computed by the GSL package gsl_multifit. Both *solverx* and *par_vary* are *M*-d vectors, where $M = 4L - C$, with L the number of lines (each fitted with 4 Voigt parameters) and C the number of constraint among parameters.

Error/warning conditions
If *solverx*, *par_vary*, or *par_lim* are NULL, an error is issued.

Unit test
Error conditions are checked. A mock solver vector is printed together with a reference for visual inspection.

8.38 Function espda_print_syst

Working remarks
None.

Purpose
Print the list of absorption systems.

Calling recipes
This is a secondary function..

Arguments

Name	Type	Description
line	CPL table	List of lines (LINE_VOIGT).
syst_num	integer	Number of systems

Return
CPL error code.

General description
This function prints a table in the standard output listing *syst_num* absorption systems from a list of identified lines *line*. The columns of the printed table are:

1. ID: from column *SYSTID*;
2. Redshift: average of *LINEZ* values for all the lines in the system;
3. Lines: number of lines in the system;
4. Shared: number of lines that have been associated to at list another system in addition to the one considered.

Mathematical description
None.

Error/warning conditions
If *line* is NULL, an error is issued.

Unit test
Error conditions are checked. A mock list of sys is printed together with a reference for visual inspection.

8.39 Function espda_redsh_group

Working remarks
None.

Purpose
Group redshift values into candidate systems.

Calling recipes
espda_iden_syst (Section 4.12 and 9.12).

Arguments

Name	Type	Description
redsh	CPL table	List of redshifts (LINE_SYST).
grp_width	double	Width of the window used for grouping [nm].
grp_num	integer	Minimum number of lines in a group.
syst_num	integer	Number of candidate systems.

Return
The list of candidate systems (LINE_SYST).

General description
This functions scans a list of redshifts *redsh*, groups the values that match within a given window with width *grp_width* and checks that the number of distinct transitions in the group are above a given threshold *grp_num*. If so, the group is considered a candidate absorption system. A list of *syst_num* candidate systems is produced and its *SYSTID* column is filled with the system IDs.

Mathematical description
Lines are sorted by ascending *WAVEL*. A tentative group with *grp_num* lines is created starting from the first line in the list. The average *LINEZ* of this group, $\bar{z}$, is tested against the following conditions:

$$\left|\frac{1+\bar{z}}{1+z_{\text{first}}}-1\right| < \frac{grp\_width}{\lambda_{\text{first}}},$$

$$\left|\frac{1+\bar{z}}{1+z_{\text{last}}}-1\right| < \frac{grp\_width}{\lambda_{\text{last}}},$$

where, λ_{first} and z_{first}, λ_{last} and z_{last} are the *WAVEL* and *LINEZ* of the first, last line of the group, respectively. If the group passes the test, it is regarded as a candidate system. In this case new lines at greater wavelength are added one by one to the "tail" of the system until the conditions are met.
When the systems fails the test after adding one line it is regarded as complete. A new tentative group of *grp_num* lines is created starting from the second line in the list. This new group undergoes the same treatment. If the group is selected as a new candidate, it is tested against an additional condition, to determine if it is actually independent from the previous one:

$$\left|\frac{1+\overline{z_{\text{old}}}}{1+\bar{z}}-1\right| < \frac{grp\_width}{\bar{\lambda}},$$

where $\bar{\lambda}$ is the average *WAVEL* of the new system and $\overline{z_{\text{old}}}$ is the average *LINEZ* of the previous one. If the condition is met, the two systems are regarded as independent, otherwise they are merged into one.
The procedure is iterated until the whole redshift list has been scanned.

Error/warning conditions
If *redsh* is NULL, an error is issued. If no candidate system is found and the verbosity level is greater than 0, a warning is issued.

Unit test
This test creates a mock list of redshifts with *size* lines having IDs *line_id*, redshifts *line_z*, and

wavelengths *wavel*. Redshift are grouped using a window with width *grp_width* (for the purpose of this tests, the minimum number of lines in a group is 1, meaning that also isolated lines are given a system ID). The resulting system IDs are checked to be equal to a reference *syst_id_ref*.

8.40 Function espda_redsh_list

Working remarks
None.

Purpose
Create a list of redshifts.

Calling recipes
espda_iden_syst (Section 4.12 and 9.12).

Arguments

Name	Type	Description
line	CPL table	List of lines (LINE_SYST).
ion	CPL table	List of ionic transitions (ION).
z_em	double	Quasar emission redshift.

Return
The list of redshifts (LINE_SYST).

General description
This function matches a list of lines *line* and a list of ionic transitions *ion* and creates a list with all possible redshift values for the lines (i.e. not exceeding the quasar emission redshift *z_em*). The list of ionic transitions is trimmed beforehand to remove transitions outside the available wavelength range of the instrument.

Mathematical description
If *line* has N rows, and M ionic transitions from *ion* are considered, the list of redshifts has in principle $N \times M$ rows in total, computed as follows:

$$z_{m,n} = \frac{\lambda_n}{\lambda_m} - 1,$$

where, λ_n are taken from the *WAVEL* column in *line*, λ_m are taken from the *WAVEL* column in *ion*, and $z_{m,n}$ are saved in the *LINEZ* column in the redshift list, n spanning from 1 to N and m spanning from 1 to M.
All values of $z_{m,n} > z\_em$ are removed.
For details on the trimming of *ion*, see espda_redsh_trim.

Error/warning conditions
If *line* or *ion* are NULL, an error is issued.

Unit test
This test creates a mock list of *line_size* lines with wavelengths *line_wavel* and a mock list of *ion_size* ionic transitions with wavelengths *ion_wavel*. A redshift list is created from them. The

size of the redshift list is checked to be equal to the reference size *ref_size* and the redshifts are checked to be consistent with the reference redshifts *ref_line_z* within the tolerance *tol*, taking into account that all redshifts greater than the quasar emission redshift *z_em* are not included in the redshift list. Error conditions are also checked.

8.41 Function espda_redsh_trim

Working remarks
None.

Purpose
Trim a list of ionic transitions.

Calling recipes
espda_iden_syst (Section 4.12 and 0).

Arguments

Name	Type	Description
ion	CPL table	List of ionic transitions (ION).
z_em	double	Quasar emission redshift.

Return
The trimmed list of ionic transitions (ION).

General description
This function takes a list of ionic transitions *ion* and remove all lines that cannot fall in the usable wavelength range of the spectrum (based on the quasar emission redshift *z_em*).

Mathematical description
All transitions with $WAVEL < WAVEL\_START/(1 + z\_corr)$ are removed, where *z_corr* is a maximum redshift defined from *z_em* by taking into account an uncertainty $ZEM\_UNC = 10^3$ km s^{-1} in its value:

$$z\_corr = \frac{z\_corr\ (2c + ZEM\_UNC) + 2ZEM\_UNC}{2c - ZEM\_UNC},$$

c being the speed of light. Also, all transitions with $WAVEL > WAVEL\_END$ are removed.

Error/warning conditions
If *ion* is NULL, an error is issued.

Unit test
This test creates a mock list of *ion_size* ionic transitions with wavelengths *ion_wavel*. The list is trimmed usiìng the emission redshift *z_em*. The size of the trimmed list is checked to be equal to the reference size *ref_size* and the wavelengths are checked to be consistent with the reference wavelengths *ref_wavel* within the globally defined tolerance *TOL*. Error conditions are also checked.

8.42 Function espda_spec_ave

Working remarks
None.

Purpose
Average a spectrum.

Calling recipes
This is a secondary function..

Arguments

Name	Type	Description
spec	CPL table	Spectrum (SPEC).
data_col	string	Column with the data to be averaged.
dataerr_col	string	Column with the errors (may be NULL).
pixfrac_col	string	Column with the pixel fraction (may be NULL).
row_start	integer	Starting row.
row_end	integer	Ending row.
data_ave	double	Average (optionally weighted).
dataerr_ave	double	Error on the average.
datarms	double	RMS of the considered data.

Return
CPL error code.

General description
This function computes the average *data_ave*, its error *dataerr_ave*, and the RMS *datarms* of the data from a column *data_col* of a spectrum *spec*. The average is optionally weighted by the errors *dataerr_col* and/or by the pixel fraction covered by each data point *pixfrac_col*. Only the row range *row_start* and *row_end* is considered.

Mathematical description
The weight of the row r of *spec* is defined as

$$w_r = \frac{f_r}{\sigma_r^2},$$

where f_r is taken from column *pixfrac_col* (if provided, otherwise it is 1) and σ_r is taken from column *dataerr_col* (if provided, otherwise it is 1). *data_ave*, *dataerr_ave*, and *datarms* are thus defined as

$$data\_ave = \frac{\sum_r d_r w_r}{\sum_r w_r},$$

$$dataerr\_ave = \frac{\sqrt{\sum_r \sigma_r^2 w_r^2}}{\sum_r w_r},$$

$$datarms = \frac{\left(\sum_r d_r^2 w_r - \sum_r w_r \times data\_ave^2\right) \sum_r w_r^2 / (\sum_r w_r)^2}{\sum_r w_r - \sum_r w_r / (row\_end - row\_start)},$$

where, d_r is taken from column *pixfrac* and r spans from *row_start* to *row_end*.

Error/warning conditions
If *spec* or *data_col* are NULL, an error is issued. If *dataerr_col* or *pixfrac_col* are NULL and the verbosity level is greater than 0, a warning is issued. If $\sum_r w_r \leq 0$ or $\sum_r \sigma_r^2 w_r^2 < 0$ and the verbosity level it greater than 1, a warning is also issued.

Unit test
This test creates a mock spectrum of size *size* and flux density equal to 1, with a random Gaussian noise depending on some exposure time *exptime*, some read-out noise *ron*, and some background flux density *bkg*. The spectrum is averaged and the results are checked to be consistent with what expected within the tolerance *tol*. Error conditions are also checked.
The flux density in the row r of *spec* is defined as

$$I_r = 1 + Z_r \frac{\sqrt{(1 + bkg) \times exptime + ron}}{exptime}$$

where Z_r is a random variable with a Gaussian distribution and r spans from 0 to *size*. The error on I_r is

$$\sigma_r = \frac{\sqrt{(I_r + bkg) \times exptime + ron}}{exptime}$$

The average of I_r is expected to be consistent with 1, while the average of σ_r is expected to be consistent with $\sqrt{(1 + bkg) \times exptime + ron}/exptime$.

8.43 Function espda_spec_clean (TBC)

8.44 Function espda_spec_conv

Working remarks
None.

Purpose
Convolve a spectrum chunk with the instrument profile.

Calling recipes
This is a secondary function.

Arguments

Name	Type	Description
prof_chunk	CPL table	Instrument profile in the spectrum chunk (INST_PROF).
wavel_over	CPL array	Wavelengths of the spectrum over-chunk [nm].
fit_in	CPL array	Voigt profile in the spectrum over-chunk.

Return
The convolved Voigt profile in the chunk.

General description
This function computes the pixel-by-pixel convolution of a Voigt profile *fit_in* fitted on a spectrum chunk with the instrument profile *prof_chunk*. The convolution uses a slightly larger over-chunk, with wavelengths *wavel_over*, to avoid border effects.

Mathematical description
The convolution of the Voigt profile I and the instrument profile Φ at the row r of the spectrum chunk is defined as

$$I_{r,\mathrm{conv}} = \sum_{s} \Phi_{r,s} I_s,$$

where, r runs within the size of the chunk, while s runs within the width of the instrument profile. In practice, the convolution iscomputed at each row of the chunk within a wavelength range defined by the columns *WAVELMIN* and *WAVELMAX* of *prof_chunk*, while Φ is an array entry from column *PROFNORM* of *prof_chunk*. If the range of the chunk is $[\lambda_{\min}, \lambda_{\max}]$, the range of the over-chunk is $[\lambda_{\min} - \Delta\lambda_{\min}, \lambda_{\max} + \Delta\lambda_{\max}]$, where $\Delta\lambda_{\min}$ ($\Delta\lambda_{\max}$) is the absolute difference between $\lambda_{\min}$ ($\lambda_{\max}$) and the first (last) entry of *WAVELMIN* (*WAVELMAX*) in *prof_chunk*.

Error/warning conditions
If *prof_chunk*, *wavel_over*, or *fit_in* are NULL, an error is issued.

Unit test
Error conditions are checked. (TBC)

8.45 Function espda_spec_corr (TBC)

8.46 Function espda_spec_equal

Working remarks
None.

Purpose
Equalize the flux in spectral orders and exposures.

Calling recipes
espda_coadd_spec (Section 4.1 and 9.1).

Arguments

Name	Type	Description
fspec	CPL table	Full spectrum (SPEC_FULL).
ref	integer	ID number of the reference exposure.

Return

The equalized full spectrum (SPEC_FULL).

General description

This function equalizes the flux in a spectrum *fspec* containing different exposures of the same object. The exposures are equalized order by order to one reference exposure *ref*. A pictorial representation of the equalization is given in Figure 8.1.

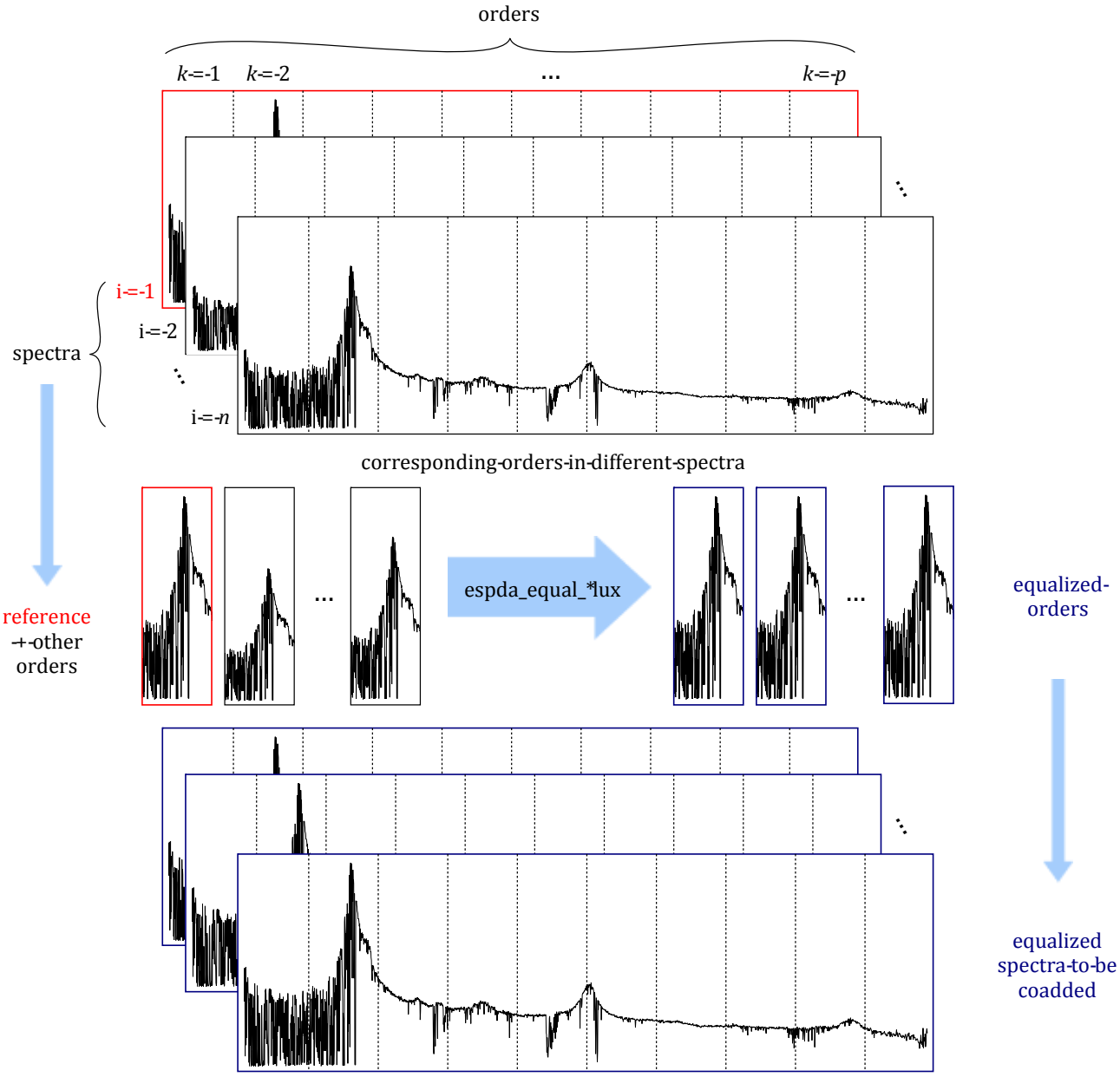


Figure 8.1 - Pictorial representation of the sprectrum equalization.

Mathematical description

The exposures are selected using column *SPECID* of *fspec*. In each exposure, the spectral orders are selected using column *ORDID*. For each order *o* and exposure $e \neq ref$, the equalization factor

$$f_{e,o} = \frac{\bar{I}_{e,o}}{\bar{I}_{ref,o}}$$

is computed, $\bar{I}$ being the error-weighted average of the flux density. The flux density and its error σ are then equalized as follows:

$$I_{e,o}^{\mathrm{eq}} = f_{e,o} I_{e,o}, \quad \sigma_{e,o}^{\mathrm{eq}} = f_{e,o} \sigma_{e,o}.$$

For details on the averaging, see espda_spec_ave.

Error/warning conditions
If *spec* is NULL, an error is issued.

Unit test
This test creates a mock spectrum with *spec_num* exposures, each one with *ord_num* orders and with a size *size*. The flux density in the orders is a function of the wavelength (specifically, a power law) shifted by a factor computed from *shift* and different for each order and exposure. The spectrum is equalized to the reference exposure *ref*. The result is checked to be consistent with what expected within the globally-defined tolerance *TOL*. Error conditions are also checked. The flux density in the order *o* of exposure *e* is

$$I_{e,o}(\lambda) \propto \lambda^{-2} \frac{o}{10/shift + e},$$

λ being the wavelength, and its error is conventionally defined as $\sigma_{e,o} = \sqrt{I_{e,o}}$. The expected equalized flux density is

$$I_{e,o}^{\mathrm{eq}}(\lambda) \propto \lambda^{-2} \frac{o}{10/shift + ref}.$$

8.47 Function espda_spec_fill (TBC)

8.48 Function espda_spec_rebin

Working remarks
kappa may be changed to a double variable.

Purpose
Rebin the flux in a spectrum.

Calling recipes
espda_coadd_spec (Section 4.1 and 9.1), espda_mask_spec (Section 4.2 and 9.2), espda_create_linelist (Section 4.10 and 9.10), espda_fit_qsocont (Section 4.11 and 9.11).

Arguments

Name	Type	Description
fspec	CPL table	Full spectrum (SPEC_FULL).
kappa	integer	Number of standard deviations to clip flux outliers.
wavel_start	double	Starting wavelength [nm].
wavel_end	double	Ending wavelength [nm].

vel_step	double	Velocity step [m s^{-1}].

Return
A rebinned spectrum (SPEC_RMS).

General description
This function rebins the flux from a spectrum *fspec* into a new spectrum having a fixed wavelength grid with a range defined by *wavel_start* and *wavel_end* and a velocity step *vel_step*. The contributions to a row of the rebinned spectrum are computed from the pixels of *fspec* which overlap the row itself. The flux densities from column *FLUX* of *fspec* are weighted by their errors from column *FLUXERR* and by the overlap computed from column *PIXSIZE*; the average, error on average and RMS are saved in columns *FLUX*, *FLUXERR*, and *FLUXRMS* of the rebinned spectrum. Flux outliers are rejected via kappa-sigma clipping. A pictorial representation of the rebinning is given in Figure 8.1.

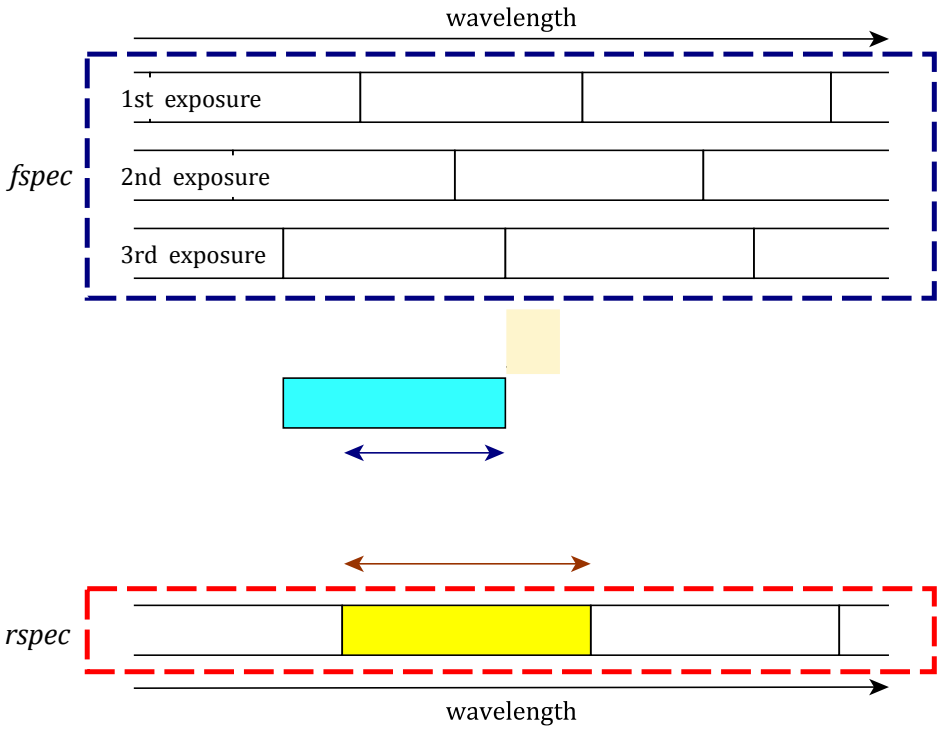


Figure 8.2 - Rebinning of exposure

A spectrum *fspec* (blue dashed square), containing three exposures, is rebinned into a spectrum *rspec* (red dashed square). Each row of the output table (in yellow) takes the contribution from many different rows of the input table (yellow-shaded region), opportunely weighted. The overlap between the pixels of *fspec* and the rows of *rspec* is used in weighting. For example, the ovelap associated to the input row highlighted in cyan corresponds to the ratio between the span of the blue arrow and the span of the red arrow.

Mathematical description
The overlap $f(\lambda)$ between the pixels p of *fspec* and the rows r of the rebinned spectrum is

$$f_r = \frac{\min\left[\lambda_p + \Delta\lambda_p/2\,,\lambda_r + \Delta\lambda_r/2\right] - \max\left[\lambda_p - \Delta\lambda_p/2\,,\lambda_r - \Delta\lambda_r/2\right]}{\Delta\lambda_r},$$

where λ is taken from column *WAVEL* and $\Delta\lambda$ from column *PIXSIZE*. This value is saved in a

temporary column *PIXFRAC* and is used in averaging.
For details on the averaging, see espda_spec_ave.
For details on the clipping of outliers, see espda_spec_clean.

Error/warning conditions
If *spec* is NULL, an error is issued.

Unit test
Error conditions are checked. (TBC)

8.49 Function espda_spec_remove

Working remarks
Currently the function is called before column *NORM* of *spec* is filled, so the normalized flux is computed by dividing column *FLUX* by column *CONT*.

Purpose
Remove lines from a spectrum.

Calling recipes
espda_fit_qsocont (Section 4.11 and 9.11).

Arguments

Name	Type	Description
spec	CPL table	Fitted spectrum (SPEC_FIT).
peak_thres	double	Threshold for truncating line peaks.

Return
The spectrum with lines removed (SPEC_FIT).

General description
This function takes a spectrum *spec* and removes the fitted spectral lines from it by dividing the normalized flux in column *NORM* by the fitted profile in column *FIT*. Results are saved in column *FLUX* of the newly created spectrum. The flux density below *peak_thres* is truncated to avoid large fluctuations in division when *FIT* approaches 0.

Mathematical description
None.

Error/warning conditions
If *spec* is NULL, an error is issued.

Unit test
This test creates a mock spectrum of size *size* and flux density equal to 1, with a Gaussian absorption line of peak *peak* and a random Gaussian noise depending on some exposure time *exptime*, some read-out noise *ron*, and some background flux density *bkg*. The line is removed from the spectrum truncating the flux density below *peak_thres*. The results are checked to be consistent with what expected within the tolerance *tol*. Error conditions are also checked.
The flux density in the row *r* of *spec* is computed as

$$I_r = 1 + Z_r \frac{\sqrt{[G_r(peak) + bkg] \times exptime + ron}}{exptime}$$

where $G_r(peak)$ is a Gaussian profile centered on the median row, with standard deviation equal to 0.1 times *size* and peak *peak*, Z_r is a random variable with a Gaussian distribution and r spans from 0 to *size*. The flux density after the line removal is expected to be consistent with 1 within a tolerance which is taken (conservatively) as 3 times the standard deviation: $tol = 3\sqrt{(1 + bkg) \times exptime + ron}/exptime$.

8.50 Function espda_spec_shift

Working remarks
None.

Purpose
Shift the wavelengths to the rest frame.

Calling recipes
espda_coadd_spec (Section 4.1 and 9.1).

Arguments

Name	Type	Description
spec	CPL table	Full spectrum (SPEC_FULL).
rv	double	Radial velocities of the exposures [km s^{-1}].

Return
CPL error code.

General description
This function shifts the wavelengths of a stellar spectrum *spec* to the rest frame, using the radial velocities *rv* for the different exposures in the spectrum.

Mathematical description
For each exposure e, the shifted wavelengths λ^{RF} are computed from the original wavelengths λ as follows:

$$\lambda_e^{\mathrm{RF}} = \lambda_e \left(1 + \frac{rv_e}{c}\right)$$

c being the speed of light.

Error/warning conditions
If *spec* or *rv* are NULL or the number of elements in *rv* is not equal to the number of exposures, an error is issued.

Unit test
This test creates a mock spectrum with *num* exposures of size *size*. The wavelengths are shifted by the radial velocities *rv*. The results are checked to be consistent with what expected within the globally defined tolerance *TOL*. Error conditions are also checked.

8.51 Function espda_split_group

Working remarks
This function may be deprecated in a later release and merged with espda_split_chunk.

Purpose
Split a line group from a spectrum.

Calling recipes
This is a secondary function.

Arguments

Name	Type	Description
fspec_in	CPL table	Full spectrum (SPEC_FULL; may be NULL).
rspec_in	CPL table	Rebinned spectrum.
line_in	CPL table	List of lines (LINE_VOIGT).
grp_sel	integer	Group to be selected.
grp_list	CPL array	List of line groups.
fpec_out	CPL table	Full spectrum of the line group (SPEC_FULL).
rspec_out	CPL table	Rebinned spectrum of the line group (SPEC_RMS).
line_out	CPL table	List of lines in the group (LINE_VOIGT).

Return
CPL error code.

General description
This function takes a spectrum and extracts the spectral regions corresponding to a group of lines *grp_sel*. A group is a set of lines that must be fitted together because their individual fitting ranges (as defined by columns *WAVELMIN*, *WAVELMAX*) overlap with each other. The function works simultaneously on a full spectrum *fspec_in* and on a rebinned spectrum *rspec_in*, producing the corresponding spectra *fspec_out*, *rspec_out* for the line group (if *fspec_in* is not provided, *fspec_out* is not produced). Similarly, it extracts from a line list *line_in* the list of lines in the group *line_out*. The lines are selected using the column *GRPID*, while the spectral region are selected using the columns *WAVELMIN*, *WAVELMAX* in *line_out*. A list of *grp_list* with all the line groups in *line_in* is also produced. If *grp_sel* is 0, all groups are selected.

Mathematical description
None.

Error/warning conditions
If *rspec_in* or *line_in* are NULL, an error is issued. If *fspec_in* is NULL and the verbosity level is greater than 0, a warning is issued.

Unit test
TBC.

8.52 Function espda_split_chunk

Working remarks

This function may be expanded in a later release to include espda_split_group.

Purpose
Split a spectrum into chunks and over-chunks.

Calling recipes
This is a secondary function.

Arguments

Name	Type	Description
spec	CPL table	Spectrum (SPEC_RMS or SPEC_FULL).
prof	CPL table	Instrument profile (INST_PROF; may be NULL).
line_chunk	CPL table	List of lines in the chunk (LINE_VOIGT).
grp_flag	string	Flag to identify if a group of lines is the last
spec_over	CPL table	Spectrum over-chunk (SPEC_RMS or SPEC_FULL).
spec_chunk	CPL table	Spectrum chunk (SPEC_RMS or SPEC_FULL).
prof_chunk	CPL table	Instrument profile in the spectrum chunk (INST_PROF).

Return
CPL error code.

General description
This function extracts from a spectrum *spec* a chunk *spec_chunk* suitable for fitting the lines from a list *line_chunk*. A chunk contains all the spectral regions (corresponding to one or more line groups) that must be fitted together because their individual fitting ranges (as defined by columns *WAVELMIN*, *WAVELMAX*) overlap with each other, or because there are some constraints between their Voigt profiles (as defined by columns *LINEZCON*, *COLDENCON*, *THERBCON*, *TURBBCON*). If the instrument profile *prof* is provided, a slightly larger over-chunk *spec_over* is produced together with the instrument profile in the chunk *prof_chunk*, to avoid border effects in convolution; otherwise *spec_over* is equal to *spec_chunk* and *prof_chunk* is null.
The function runs in two modes, depending on the external variable *grp_mode*: if *grp_mode* is equal to "OVER" or *grp_flag* is equal to "MAX", chunks are selected as described; otherwise, the whole spectrum across the lines of *line_chunk* (this is suitable for the last iteration of the continuum fitting algorithm).

Mathematical description
For details on the definition of the over-chunks, see espda_spec_conv.

Error/warning conditions
If *spec* or *line_chunk* are NULL, an error is issued. If *prof* is NULL and the verbosity level is greater than 0, a warning is issued.

Unit test
TBC.

8.53 Function espda_split_part

Working remarks
None.

Purpose
Split the blue and the red part.

Calling recipes
espda_fit_qsocont (Section 4.11 and 9.11).

Arguments

Name	Type	Description
table	CPL table	Spectrum (SPEC_BASIC) or list of lines (LINE_BASIC).
lya_zem	double	Redshifted wavelength of the Ly-alpha emission line.
table_blue	CPL table	Blue part of the spectrum (SPEC_BASIC or LINE_BASIC).
table_red	CPL table	Red part of the spectrum (SPEC_BASIC or LINE_BASIC).

Return
CPL error code.

General description
This function splits a spectrum or line list *table* into its blue part *table_blue* and its red part *table_red*, respectively defined as the regions bluewards and redwards from the redshifted wavelength *lya_zem* of the Ly-alpha emission line of the quasar.

Mathematical description
None.

Error/warning conditions
If *table* is NULL, an error is issued.

Unit test
TBC.

8.54 Function espda_struct_copy

Working remarks
None.

Purpose
Copy a table entry.

Calling recipes
This is a secondary function.

Arguments

Name	Type	Description
table_1	CPL table	Source table.
col_1	string	Name of the source column.
pos_1	integer	Source position.
table_2	CPL table	Target table.
col_2	string	Name of the target column.

pos_2	integer	Target position.

Return
CPL error code.

General description
This function copies the entry at position *pos_1* in column *col_1* of table *table_1* to the position *pos_2* in column *col_2* of table *table_2*. Entry values of type integer, double, and string are allowed; other types are disregarded.

Mathematical description
None.

Error/warning conditions
If *table_1*, *col_1*, *table_2*, or *col_2* are NULL, an error is issued. If the entry has a wrong type and the verbosity level is greater than 0, a warning is issued.

Unit test
TBC.

8.55 Function espda_struct_delete

Working remarks
None.

Purpose
Delete a structure.

Calling recipes
This is a secondary function.

Arguments

Name	Type	Description
obj	string	Structure type.
max	int	Number of structures to delete.
...	argument list	Names of the structures.

Return
CPL error code.

General description
This function deletes a set of *max* structure of the same type *obj*. The names of the structures must be provided as an argument list. Structures of type CPL array, CPL table, and CPL vector are allowed; other types are disregarded.

Mathematical description
None.

Error/warning conditions
If the structure has a wrong type and the verbosity level is greater than 0, a warning is issued.

Unit test
TBC.

8.56 Function espda_struct_erase

Working remarks
None.

Purpose
Select and erase rows from a table.

Calling recipes
This is a secondary function.

Arguments

Name	Type	Description
table	CPL table	Table.
max	int	Number of selection instructions.
...	argument list	Selection instructions.

Return
CPL error code.

General description
This function selects rows from a table *table* according to a set of *max* instructions. The selected rows are then erased.
For details on the selection instructions, see espda_struct_select.

Mathematical description
None.

Error/warning conditions
If *table* is NULL, an error is issued.

Unit test
TBC.

8.57 Function espda_struct_extract

Working remarks
None.

Purpose
Select and extract rows from a table.

Calling recipes

espda_fit_line (Section 4.13 and 9.13).

Arguments

Name	Type	Description
table	CPL table	Table.
max	int	Number of selection instructions.
...	argument list	Selection instructions.

Return
The extracted table.

General description
This function selects rows from a table *table* according to a set of *max* instructions. The selected rows are then extracted to form a new table.
For details on the selection instructions, see espda_struct_select.

Mathematical description
None.

Error/warning conditions
If *table* is NULL, an error is issued.

Unit test
TBC.

8.58 Function espda_struct_init

Working remarks
None.

Purpose
Initialize a table column.

Calling recipes
This is a secondary function.

Arguments

Name	Type	Description
table	CPL table	Table.
col	string	Name of the column.
type	CPL type	Type of the column.

Return
CPL error code.

General description
This function initializes column *col* of table *table* with void values. Entry values of type integer, double, and string are allowed; other types are disregarded. Void values are globally defined as *INV_INT*, *INV_DOUBLE*, and *INV_STRING* respectively.

Mathematical description
None.

Error/warning conditions
If *table* or *col* are NULL, an error is issued. If the entry has a wrong type and the verbosity level is greater than 0, a warning is issued.

Unit test
TBC.

8.59 Function espda_struct_select

Working remarks
This function is not meant to be called directly, but only through espda_struct_erase or espda_struct_extract.

Purpose
Select rows from a table.

Calling recipes
This is a secondary function.

Arguments

Name	Type	Description
table	CPL table	Table.
max	int	Number of selection instructions.
ap	argument list	Selection instructions.

Return
CPL error code.

General description
This function selects rows from a table *table* according to a set of *max* instructions specified by the argument list *ap*. The behavior depends on *max*:

- If *max* is 0, *ap* must be a couple of integer values *start* and *count*. In this case, the window of *count* rows starting from row *start* is selected.
- If *max* is greater than 0, *ap* must be a set of *max* conditions. Each condition is a triplet of values: a string *col*, a CPL table select operator *op* and a double *value*. All rows of *col* where *value* satisfy the constraint set by *op* are selected. If more than one condition is provided, each new selection acts on the unselected rows.

Mathematical description
None.

Error/warning conditions
If *table* is NULL, an error is issued.

Unit test

TBC.

8.60 Function espda_struct_sort

Working remarks
None.

Purpose
Sorts a table.

Calling recipes
This is a secondary function.

Arguments

Name	Type	Description
table	CPL table	Table.
max	int	Number of sorting instructions.
...	argument list	Sorting instructions.

Return
CPL error code.

General description
This function sorts a table *table* according to a set of *max* instructions specified by the argument list *ap*. Each instruction is a couple of values: a string *col* and an boolean *ord*. If *ord* is FALSE (TRUE), *table* is sorted by column *col* in ascending (descending) order. If more than one instruction is provided, each new instruction is applied only to the rows that have equal values in the column specified by the previous instruction.

Mathematical description
None.

Error/warning conditions
If *table* is NULL, an error is issued.

Unit test
TBC.

8.61 Function espda_syst_accept

Working remarks
Acceptance rules have not been implemented yet.

Purpose
Apply acceptance rules.

Calling recipes
This is a secondary function.

Arguments

Name	Type	Description
syst	CPL table	List of absorption systems (LINE_SYST).
syst_id	string	ID of the selected system.
row_start	integer	Starting row of the selected system.
row_num	integer	Number of rows of the selected system.
disc_flag	integer	Flag to notify that lines have been discarded.

Return
CPL error code.

General description
This function evaluates an absorption system from a list *syst* against a set of acceptance rules and discards all lines that do not comply. The system is defined by its ID *syst_id* and by the starting row *row_start* and number of rows *row_num* in *syst* (as obtained when *syst* is sorted by ascending *SYSTID*). If lines are discarded, the flag *disc_flag* is set to 1.

Mathematical description
TBC.

Error/warning conditions
If *syst* is NULL, an error is issued. If lines are discarded and the verbosity level is greater than 0, a warning is issued.

Unit test
TBC.

8.62 Function espda_syst_add

Working remarks
None.

Purpose
Add lines to absorption systems.

Calling recipes
espda_iden_syst (Section 4.12 and 9.12).

Arguments

Name	Type	Description
syst	CPL table	List of absorption systems (LINE_SYST).
noid	CPL table	List of non-identified lines (LINE_SYST).
ion	CPL table	List of ionic transitions (ION).
syst_num	integer	Number of systems.
add_hwidth	double	Half-width of the window to define coincidences.

Return
CPL error code.

General description
This function scans a list of non-identified lines *noid*, computes the possible redshifts of the lines by comparison with a list of ionic transitions *ion*, and find coincidences between these redshifts and those of a list *syst* of *syst_num* absorption systems, within a window of half-width *add_hwidth*. The lines with coincident redshifts are added to *syst* and removed from *noid*.

Mathematical description
The redshift z of a line is said to be coincident with the redshift $\bar{z}$ of a system if

$$|z - \bar{z}| < \frac{add\_width}{\lambda_{\text{trans}}},$$

where, λ_{trans} is the wavelength of the transitions associated to the line (from column *LINEID*).
For details on redshift computation, see espda_redsh_list.

Error/warning conditions
If *syst*, *noid*, or *ion* are NULL, an error is issued.

Unit test
TBC.

8.63 Function espda_syst_adjust

Working remarks
None.

Purpose
Adjust number of components in multiplets.

Calling recipes
espda_iden_syst (Section 4.12 and 9.12).

Arguments

Name	Type	Description
syst	CPL table	List of absorption systems (LINE_SYST).
syst_num	integer	Number of systems.

Return
CPL error code.

General description
This function scans a list *syst* of *syst_num* absorption systems, detects multiplet transitions and adjusts the number of components for all multiplet members (i.e., lines are added until all the multiplet members have as many components as the richest member). The multiplets currently considered are NV $\lambda\lambda$1238-1242 (doublet), SiII $\lambda\lambda$1260-1526 (triplet), SiIV $\lambda\lambda$1393-1402 (doublet), CIV $\lambda\lambda$1548-1550 (doublet), FeII $\lambda\lambda$2344-2600 (quadruplet), MgII $\lambda\lambda$2796-2803 (doublet), CaII $\lambda\lambda$3934-3969 (doublet), and NaI $\lambda\lambda$5891-5897 (doublet).

Mathematical description

None.

Error/warning conditions
If *syst* is NULL, an error is issued.

Unit test
TBC.

8.64 Function espda_syst_adv

Working remarks
None.

Purpose
Advance the system ID.

Calling recipes
This is a secondary function.

Arguments

Name	Type	Description
syst_id	string	ID of an absorption system.

Return
CPL error code.

General description
This function advances the ID of an absorption system by one step. The ID consists of a couple of characters (defined through the external variables *syst_id1*, *syst_id2*) that together count from a to z and from aa to zz. The function also updates the string *syst_id* obtained by juxtaposing *syst_id1* and *syst_id2*.

Mathematical description
None.

Error/warning conditions
TBC.

Unit test
TBC.

8.65 Function espda_syst_ave

Working remarks
None.

Purpose
Compute the average redshift of systems.

Calling recipes
espda_iden_syst (Section 4.12 and 9.12).

Arguments

Name	Type	Description
syst	CPL table	A list of absorption systems (LINE_SYST).
syst_num	integer	Number of systems.

Return
The average redshifts of systems.

General description
This function scans a list *syst* of *syst_num* absorption systems and computes the redshift of each one by averaging the redshift of the lines in the system.

Mathematical description
The average is computed by espda_spec_ave using *LINEZ* as *data_col*. For details on the averaging, see espda_spec_ave.

Error/warning conditions
If *syst* is NULL, an error is issued.

Unit test
TBC.

8.66 Function espda_syst_blend

Working remarks
None.

Purpose
Associate blended lines to an absorption system.

Calling recipes
espda_iden_syst (Section 4.12 and 9.12).

Arguments

Name	Type	Description
syst	CPL table	List of absorption systems (LINE_SYST).
syst_id	string	ID of the system.

Return
The system with the associated blended lines.

General description
This function selects an absorption system with ID *syst_id* from a list *syst* and finds all the lines that, though not belonging to the system, are blended with its lines. A line with *SYSTID* different from *syst_id* and *GRPID* equal to one or more line of the system is considered a blended line (because the wavelength range used to fit it overlaps the wavelength range used to fit one or

more lines of the system). An expanded system with the associated blended lines is produced, and *syst_id* is advanced for iteration purpose.

Mathematical description
None.

Error/warning conditions
If *syst* is NULL, an error is issued.

Unit test
TBC.

8.67 Function espda_syst_conf

Working remarks
None.

Purpose
Confirm the absorption systems.

Calling recipes
espda_iden_syst (Section 4.12 and 9.12).

Arguments

Name	Type	Description
syst	CPL table	List of absorption systems (LINE_SYST).
line	CPL table	List of lines where the systems were identified (LINE).
comp_num	integer	Minimum number of components.
cand_num	integer	Number of candidate systems
syst_num	integer	Number of confirmed systems.
noid	CPL table	List of non-identified lines (LINE_SYST).

Return
CPL error code.

General description
This function scans a list of lines *syst* where *cand_num* candidate absorption systems were already identified, and applies the existence and acceptance rules (depending on the minimum number of components *comp_num* to confirm a system). The lines that do not pass the test are removed from *syst* and the number of confirmed system *syst_num* is computed. Concurrently, the lines that do pass the test are removed from the list *line* where the systems were originally identified, to produce a list *noid* of non-identified lines.

Mathematical description
None.

Error/warning conditions
If *syst* or *line* are NULL, an error is issued. If *syst_num* is 0 and the verbosity level is greater than 0, a warning is issued.

Unit test
TBC.

8.68 Function espda_syst_constr

Working remarks
None.

Purpose
Put constraints on multiplet components.

Calling recipes
espda_iden_syst (Section 4.12 and 9.12).

Arguments

Name	Type	Description
syst	CPL table	List of absorption systems (LINE_SYST).
ion	CPL table	List of ionic transitions (ION).

Return
CPL error code.

General description
This function takes a list of absorption systems *syst*, detects the lines identified as components of a multiplet, and set constraints on the value of their Voigt parameters, so that each multiplet member can be fitted with the same redshift *z*, column density *N*, and thermal broadening *b* of the matching members. A list of ionic transitions *ion* is used to identify the multiplet components (all lines whose *LINEID* contains one of the tags from column *MULTID* of *ion* is considered as a member of a multiplet). The constraints are set with a standard syntax: *mult_ID-par_symbol-count*, where the *mult_ID* is the multiplet ID as taken from column *MULTID* of *ion*, *par_symbol* is the parameter symbol (either "z", "N", or "b" for redshift, column density, and thermal broadening respectively), and *count* is a 3-digit counter increased for each new multiplet component.

Mathematical description
None.

Error/warning conditions
If *syst* or *ion* are NULL, an error is issued.

Unit test
TBC.

8.69 Function espda_syst_dec

Working remarks
None.

Purpose
Decrease the system ID.

Calling recipes
This is a secondary function.

Arguments
None.

Return
CPL error code.

General description
This function decreases the ID of an absorption system by one step. The ID consists of a couple of characters (defined through the external variables *syst_id1*, *syst_id2*) that together count from a to z and from aa to zz.

Mathematical description
None.

Error/warning conditions
TBC.

Unit test
TBC.

8.70 Function espda_syst_exist

Working remarks
Existence rules have not been completely implemented yet.

Purpose
Apply existence rules.

Calling recipes
This is a secondary function.

Arguments

Name	Type	Description
syst	CPL table	List of absorption systems (LINE_SYST).
syst_id	string	ID of the selected system.
comp_num	integer	Minimum number of components.
row_start	integer	Starting row of the selected system.
row_num	integer	Number of rows of the selected system.
exist_flag	integer	Flag to notify that the candidate exists.

Return
CPL error code.

General description
This function evaluates an absorption system from a list *syst* against a set of existence rules and discards all systems that do not comply. The system is defined by its ID *syst_id* and by the starting row *row_start* and number of rows *row_num* in *syst* (as obtained when *syst* is sorted by ascending *SYSTID*). The rules are:

- The system must have more than *comp_num* components.

If the system is discarded, the flag *exist_flag* is set to 0.

Mathematical description
TBC.

Error/warning conditions
If *syst* is NULL, an error is issued. If the system is discarded and the verbosity level is greater than 0, a warning is issued.

Unit test
TBC.

8.71 Function espda_syst_find

Working remarks
None.

Purpose
Apply existence rules.

Calling recipes
This is a secondary function.

Arguments

Name	Type	Description
syst	CPL table	List of absorption systems (LINE_SYST).
syst_id	string	ID of the selected system.
row_start	integer	Starting row of the selected system.
row_num	integer	Number of rows of the selected system.

Return
CPL error code.

General description
This function scans a list of absorption systems *syst* previously sorted by *SYSTID* and finds the rows corresponding to the system with ID *syst_id*. The starting row *row_start* and the number of rows *row_num* are issued.

Mathematical description
None.

Error/warning conditions
If *syst* is NULL, an error is issued.

Unit test
TBC.

8.72 Function espda_syst_update

Working remarks
None.

Purpose
Update an absorption system.

Calling recipes
This is a secondary function.

Arguments

Name	Type	Description
syst	CPL table	List of absorption systems (LINE_SYST).
syst_id	string	ID of the selected system.
line_id	string	ID of the line
line_z	double	Redshift of the line.
edit	string	Operation to perform.
row_start	integer	Starting row of the selected system.
row_num	integer	Number of rows of the selected system.

Return
CPL error code.

General description
This function selects an absorption system from a list *syst* and updates it, either adding or removing a line. The system is defined by its ID *syst_id* and by the starting row *row_start* and number of rows *row_num* in *syst* (as obtained when *syst* is sorted by ascending *SYSTID*). The behavior depends on *edit*, which describes the operation to perform:

- if *edit* is "add", the line is added and the corresponding row in *syst* is updated with the system ID *syst_id*, the line ID *line_id*, and the line redshift *line_z* (respectively copied into columns *SYSTID*, *LINEID*, and *LINEZ*);
- if *edit* is "remove", the value of *SYSTID* in the corresponding row is changed to the globally-defined void string INV_STRING.

Mathematical description
None.

Error/warning conditions
If *syst* is NULL, an error is issued.

Unit test

TBC.

8.73 Function espda_voigt_create

Working remarks
None.

Purpose
Create data structures with Voigt parameters.

Calling recipes
This is a secondary function.

Arguments

Name	Type	Description
line	CPL table	List of lines (LINE_SYST).
par_pos	CPL array	Position of varying and fixed parameters (LINE).
par_fix	CPL array	Fixed parameters.
par_vary	GSL vector	Varying parameters.
par_lim	CPL table	Limits on parameters.
vary_num	integer	Number of varying parameters.

Return
CPL error code.

General description
This function takes a list *line* containing L lines and copies their Voigt parameters into data structures suitable for fitting. The Voigt parameters are taken from columns *LINEZ*, *COLDEN*, *THERB*, and *TURBB* of *line*; their range and constraints are taken for each parameter from columns **MIN*/**MAX* and **CON*. The following structures are produced:

- *par_pos*: a $4L$-dimensional integer array to relate the positions of the Voigt parameters in *line* to their positions in *par_fix* and *par_vary* (see below). The rules are arcane; this parameter is not meant to be handled by the user.
- *par_fix*: a F-dimensional double array with parameters to be kept fixed. These are the F Voigt parameters with **CON* equal to "F".
- *par_vary*: a $(4L - F - C)$-dimensional double array with the parameters allowed to vary. These are the Voigt parameters with **CON* not equal to "V". Their number *vary_num* is also issued. C is the number of mutual constraints between parameters; if two parameters are constrained to be equal in the fitting procedure, they are counted as one parameter in *par_vary*.
- *par_lim*: a $4L$-row table with minimum and maximum value for each Voigt parameter. The values are saved in columns *MIN*, *MAX* and duplicate in columns *MIN_FIX*, *MAX_FIX* (*MIN_VARY*, *MAX_VARY*) for fixed (varying) parameters.

Depending on the value of the global variable *TRANSF_MODE*, a transformation may be applied to the parameters to enforce the prescribed ranges. For details on the transformation, see espda_voigt_transf.

Mathematical description
None.

Error/warning conditions
If *line* is NULL, an error is issued.

Unit test
TBC.

8.74 Function espda_voigt_integ

Working remarks
This function is meant to be called only by the gsl_integration package.

Purpose
Compute the integrand of the Voigt function.

Calling recipes
This is a secondary function.

Arguments

Name	Type	Description
x	double	Argument of the Voigt function.
p	double	Parameters of the Voigt function.

Return
The integrand.

General description
This function computes the integrand which appears in the representation of the Voigt function described in [RD-17], as a function of the argument x and the parameters p.

Mathematical description
The integrand is

$$I(x; a, u) = \exp[-(u^2 - x^2)] \sin[2a(u - x)],$$

where a, u are the two parameters in p.

Error/warning conditions
None.

Unit test
TBC.

8.75 Function espda_voigt_merge (TBC)

8.76 Function espda_voigt_prep (TBC)

8.77 Function espda_voigt_trans (TBC)

8.78 Function espda_wrap_check (TBC)

8.79 Function espda_wrap_line (TBC)

8.80 Function esp_filter_cosmics (TBR)

Working remarks
None

Purpose
Clean residual cosmic hits on S2D spectrum using positive sigma-clipping within a window

Calling recipes
esp_compu_radvel (Sections 4.3 and 9.3)
esp_compu_rhk (Sections 4.4 and 9.4)

Input data

Name	Format	Quantity	Description
SPEC_2D	S2D	1	

Input Parameters

Name	Type	Default	Description
filter-cosm-ksig	double	4.5	Kappa parameter for the sigma-clipping
filter-cosm-winsize	int	128	Size of the window in pixels

Output data

Name	Format	Quantity	Description
SPEC_2D	S2D	1	

QC1 outputs

Name	Type	Description
QC.FILTER.NCOS	integer	Number of cosmic hits found

Other output
None

General description
This function searches and removes residual cosmic hits on S2D spectrum using positive sigma clipping within a window.

Mathematical description
1. Each S2D spectral order is split into sub-windows (chunks) of size *filter-cosm-winsize* pixels
2. In each window the data values that are *filter-cosm-ksig* times the data dispersion above the mean are flagged as cosmic hits and replaced by the mean value of the chunk
3. Loop over all orders and chunks and count cosmic hits

Error/warning conditions
TBC

Unit test
Verify the function using a synthetic spectrum with superposed some outliers (e.g. in form of

very narrow, large amplitude Gaussians) to simulate CRH and see if they are corrected.

8.81 Function esp_correct_flux (TBR)

Working remarks
None

Purpose
Correct S2D spectrum flux to match a given distribution

Calling recipes
esp_compu_radvel (Sections 4.3 and 9.3)

Parameters
None

Input data

Name	Format	Quantity	Description
SPEC_2D	S2D	1	
WAVE_MATRIX_FIBER	S2D	1	

Other input

Name	Format	Quantity	Description
FLUX_TEMPLATE	FLTPL	1	

Output frames

Name	Format	Quantity	Description
SPEC_2D	S2D	1	

QC1 outputs
None

Other output
None

General description
This function corrects the flux distribution of the given S2D spectrum to match the given flux template, which should have a spectral type close to the one of the target.

Mathematical description
1. Sum the flux in all S2D spectral orders
2. Normalize the fluxes to 1.0 at 550 nm (considering several pixels in the proximity of this wavelength, and avoiding flagged bad pixels and cosmic hits)
3. Choose the best available flux template based on the target spectral type
4. Divide the obtained fluxes by the selected flux template to generate a correction factor for each order
5. Fit a low-order polynomial through the correction factors vs. central wavelength of each

order
6. Compute the correction factor for each pixel of the S2D spectrum using the wavelength calibration
7. Divide the S2D spectrum by the correction factor

Error/warning conditions
TBD

Unit test
TBD

8.82 Function esp_compute_ccf (TBR)

Working remarks
None

Purpose
Compute the cross-correlation function between S2D spectrum and CCF template

Calling recipes
esp_compu_radvel (Sections 4.3 and 9.3)

Parameters

Name	Type	Default	Description
ccf-rv-cen	double	TBC	Central RV (km/s)
ccf-rv-step	double	TBC	RV step (km/s)
ccf-rv-window	double	TBC	CCF window width (km/s)
ccf-mask-width	double	TBC	Width of mask holes (km/s)
ccf-order-range	double	TBC	Spectral orders to consider (array?)

Input data

Name	Format	Quantity	Description
SPEC_2D	S2D	1	
BLAZE_FIBER	S2D	1	
WAVE_MATRIX_FIBER	S2D	1	

Other input

Name	Format	Quantity	Description
CCF_TEMPLATE	CCFTPL	1	

Output frames

Name	Format	Quantity	Description
CCF_FIBER	CCF	1	

QC1 outputs

Name	Type	Description
QC.CCF.RV	double	Radial velocity (km/s)
QC.CCF.FWHM	double	CCF FWHM (km/s)

QC.CCF.CONTRAST	double	CCF contrast (%)
QC.CCF.NOISE	double	Photon noise on RV (km/s)

Other output
None

General description
This function computes the cross-correlation of the S2D spectrum using the given CCF template.

Mathematical description
1. Create a linear RV table using input parameters;
2. Loop over each RV value:
    a. Shift the CCF mask according to the RV value;
    b. Project the shifted CCF mask over 2D spectrum using input mask width;
    c. Correct blaze function locally;
    d. Sum the transmitted flux (flux in partial pixels is linearly interpolated);
    e. Compute error on transmitted flux by propagation of S2D errors;
3. Sum order-by-order CCFs into final CCF, propagating the errors.
4. Fit a Gaussian model to the CCF and derive RV, FWHM and contrast
5. Estimate uncertainty on RV from method given in [RD-15].

Quality assessment
The radial velocity precision has to be better than 10cm/s, local accuracy better than 10m/s (AD-7).

Error/warning conditions
TBD

Unit test
TBD

8.83 Function esp_compu_bis (TBR)

Working remarks
None

Purpose
Compute the CCF bisector

Calling recipes
esp_compu_radvel (Sections 4.3 and 9.3)

Parameters

Name	Type	Default	Description
bis-step	double	TBC	Step in CCF depth

Input data

Name	Format	Quantity	Description

CCF_FIBER	CCF	1	

Other input
None

Output frames

Name	Format	Quantity	Description
BIS_FIBER	BIS	1	CCF bisector

QC1 outputs

Name	Type	Description
QC.BIS.SPAN	double	Bisector velocity span (km/s)

Other output
None

General description
This function computes the CCF bisector (RV vs. line depth) and bisector velocity span.

Mathematical description
1. Create a linear line depth grid using the input parameter;
2. Loop over each line depth value:
    a. Interpolate CCF on blue and red sides using cubic splines;
    b. Compute CCF mid-point (i.e. bisector value);
3. Compute velocity span by subtraction of upper and lower bisector points.

Quality assessment
The function is applied to proper test frames. Output is evaluated against expected values.

Error/warning conditions
TBD

Unit test
A simulated CCF is created with known asymmetry and processed by the function. The output bisector and bisector span are checked against the known input values.

8.84 Function esp_correct_blaze (TBR)

Working remarks
None

Purpose
Divide the S2D spectrum by the blaze function

Calling recipes
esp_compu_rhk (Sections 4.4 and 9.4)

Parameters
None

Input data

Name	Format	Quantity	Description
SPEC_2D	S2D	1	
BLAZE_FIBER	S2D	1	

Other input
None

Output frames

Name	Format	Quantity	Description
SPEC_2D	S2D	1	

QC1 outputs
None

Other output
None

General description
This function corrects the S2D spectrum for the spectrograph blaze function.

Mathematical description
The S2D spectrum is simply divided by the blaze function (which is also in S2D format).

Quality assessment
None

Error/warning conditions
TBD

Unit test
TBD

8.85 Function esp_measure_flux (TBR)

Working remarks
None

Purpose
Compute the flux density in a given passband

Calling recipes
esp_compu_rhk (Sections 4.4. and 9.4)

Parameters

Name	Type	Default	Description
pbflux-wlen-start	double		Start wavelength (nm)

pbflux-wlen-end	double		End wavelength (nm)

Input data

Name	Format	Quantity	Description
SPEC_2D	S2D	1	
WAVE_MATRIX_FIBER	S2D	1	

Other input
None

Output frames
None

QC1 outputs
None

Other output

Name	Type	Description
pbflux	double	Integrated flux in the PB
pbflux-err	double	Formal error on *pb-flux*

General description
This function measures the flux density and error in a given passband.

Mathematical description
The flux density is obtained by simply summing the flux of all pixels within the passband and dividing it by the passband width in wavelength units. Errors are propagated from the S2D errors in the usual way.

Quality assessment
TBC

Error/warning conditions
TBC

Unit test
A synthetic spectrum is generated and the flux is computed in a given passband, the test verifies that the value si as expected.

8.86 Function esp_compu_s_index (TBR)

Working remarks
None

Purpose
Compute the Mt Wilson S-index

Calling recipes
esp_compu_rhk (Sections 4.4. and 9.4)

Parameters

Name	Type	Default	Description
H	double		Flux in H bandpass
H-err	double		Formal error on H
K	double		Flux in K bandpass
K-err	double		Formal error on K
R	double		Flux in R bandpass
R-err	double		Formal error on R
V	double		Flux in V bandpass
V-err	double		Formal error on V

Input data
None

Other input
None

Output frames
None

QC1 outputs
None

Other output

Name	Type	Description
s	double	Mt Wilson S index (unitless)
s-err	double	Formal error on *s* (unitless)

General description
This function computes the Mt Wilson CaII H&K S-index according to [RD-18].

Mathematical description
The S-index is defined as S=(H+K)/(R+V). Errors are propagated using the standard formula.

Quality assessment
TBC

Error/warning conditions
TBC

Unit test
Using typical values of H, K , R and V computes the S-index and verifies that the result is correct.

8.87 Function esp_compu_rhk_index (TBR)

Working remarks
None

Purpose
Compute the R'HK activity index

Calling recipes
esp_compu_rhk (Sections 4.4. and 9.4)

Parameters

Name	Type	Default	Description
s	double		Mt Wilson S-index
s-err	double		Formal error on s
bv	double		B-V color of the star

Input data
None

Other input
None

Output frames
None

QC1 outputs
None

Other output

Name	Type	Description
rhk	double	R'HK index
rhk-err	double	Formal error on *rhk*

General description
This function computes the R'HK index from the S-index and the stellar B-V color according to [RD-19].

Mathematical description
See the above reference for the detailed calibration formulae. Errors are propagated in the usual way.

Quality assessment
The error in R'HK has to be smaller than 0.2 for a SNR of 100 (AD-7)

Error/warning conditions
TBD

Unit test
Using typical values the S-index computes the R'HK index and verifies that the result is correct.

8.88 Function esp_compu_prot_age (TBR)

Working remarks
None

Purpose
Compute the star rotation period and age from the activity-rotation-age calibration

Calling recipes
esp_compu_rhk (Sections 4.4. and 9.4)

Parameters

Name	Type	Default	Description
rhk	double		R'HK index
rhk-err	double		Formal error on *rhk*

Input data
None

Other input
None

Output frames
None

QC1 outputs
None

Other output

Name	Type	Description
prot	double	Rotation period
prot-err	double	Formal error on prot
age	double	Age
age-err	double	Formal error on age

General description
This function computes estimates of stellar rotation period and age from R'HK activity index, following [RD-19].

Mathematical description
See the above reference for the detailed calibration formulae. Errors are propagated in the usual way.

Quality assessment
TBD

Error/warning conditions
TBD

Unit test

Using typical values the R'HK index computes the rotation period and the age and verifies that the result is correct.

8.89 Function espda_select_chunk (TBR)

Working remarks
Currently called select_local_spec.

Purpose
Select a chunk of a spectrum.

Calling recipes
espda_compu_eqwidth (Sections 4.5 and 9.5)
espda_create_linelist (Sections 4.11 and 9.11)

Input data

Name	Format	Quantity	Description
MergSpec	SPEC	1	Table with merged spectrum

Input parameters

Name	Type	Description
hwidth	double	Half-width of the chunk (Angstrom)
wavel	double	Central wavelength of the chunk (Angstrom)

Output data

Name	Format	Quantity	Description
Chunk	SPEC	1	Table with spectral chunk

Output parameters
None

General description
TBC

Mathematical description
TBC

Quality assessment
TBC

Error/warning conditions
TBC

Unit test
TBD

8.90 Function espda_fit_lcont (TBR)

Working remarks
Currently called esp_fit_lcont.

Purpose
Fit the continuum in a region close to a line

Calling recipes
espda_compu_eqwidth (Sections 4.5 and 9.5)
espda_create_linelist (Sections 4.11 and 9.11)

Input data

Name	Format	Quantity	Description
Chunk	SPEC	1	Table with spectral chunk

Input Parameters

Name	Type	Description
cont-rejt	double	Rejection threshold for the continuum determination. The value will depend on the signal-to-noise
cont-iter	integer	Number of iterations to fit the continuum

Output data

Name	Format	Quantity	Description
Chunk	SPEC	1	Table with spectral chunk (with updated column *CONT*)

Output parameters
None

General description
This function receives a local region of a spectrum *Chunk* and fits iteratively the spectrum flux until it reaches the continuum. It returns the coefficients of the fit and the updated array with the fitted continuum which is copied in the column *Chunk:CONT*.

Mathematical description
Performs an iterative fit (5 times) to find the points that correspond to the continuum. In each iteration, the points are fit to a 2^{nd} order polynomial and normalized accordingly. The points with a value lower than *cont-rejt* are rejected.

Quality assessment
The accuracy in the continuum determination has to be down to the SNR level up to SNR=2000.

Error/warning conditions
None

Unit test
Use a synthetic spectrum with a typical shape to which a simulated line is added. Noise, extra lines and maybe CRH can also be added. The function fits the local continuum and the result is compared with the know shape. The accuracy is evaluated by computing the RMS pixel by pixel

from the difference of the computed and input continua.

8.91 Function espda_det_line (TBR)

Working remarks
Currently called esp_det_line.

Purpose
Detect absorption lines in a spectrum after normalizing it.

Calling Recipes
espda_compu_eqwidth (Section 4.5 and 9.5)
esp_fit_qsocont (Section 4.10 and 9.10)
espda_create_linelist (Section 4.10 and 9.11)

Input data

Name	Format	Quantity	Description
Chunk	SPEC	1	Table with spectral chunk

Input Parameters

Name	Type	Description
thres	double	Normalized threshold for line acceptance
resol	double	Minimum accepted separation between two lines (in nm)
smwidth	integer	Number of pixels in the boxcar to be averaged

Output data

Name	Format	Quantity	Description
LineChunk	LINE	1	List of absorption lines found in a spectral chunk

Output parameters
None

General description
A list of absorption lines *LineChunk* is extracted from a spectral region *Chunk*. Only pixels of the spectrum where *Chunk:MASK* is 0 are considered. Absorption lines are detected by identifying the local maxima of the 2nd derivative (i.e. the zeros of the 3rd derivative) of the normalized flux, obtained dividing *Chunk:FLUX* by *Chunk:CONT*. Only lines deeper than a given threshold *thres* are accepted. If two lines are closer than *resol* they are taken as a single line. The wavelengths of the accepted lines and their formal errors are saved in columns *LineChunk:WAVEL* and *LineChunk:WAVEL_ERR*. The normalized peak values and their formal errors are saved in *LineChunk:PEAK* and *LineChunk:PEAK_ERR*. The other columns of *LineChunk* (*MU*, *MU_ERR*, *SIGMA*, *SIGMA_ERR*, *EW*, *EW_ERR*, *FWHM*, and *FWHM_ERR*) are left empty and will be updated by the following functions and recipes.

Mathematical description
See Section 3.3 for a description of the line detection algorithm.

Quality assessment

TBD

Error/warning conditions
Inconsistent or badly formatted input: the process is interrupted and an error message is issued

Unit test
Use a synthetic spectrum with a typical shape to which a simulated line is added. Noise, extra lines and maybe CRH can also be added. The function is run on this spectrum and the result are compared with the expected ones.

8.92 Function espda_fit_ngauss (TBR)

Working remarks
Currently called esp_fit_ngauss.

Purpose
Fit a list of absorption lines with *n* Gaussian profiles and compute the equivalent width of the lines.

Calling Recipes
espda_compu_eqwidth (Section 4.5 and 9.5)
esp_fit_qsocont (Section 4.10 and 9.10)

Input data

Name	Format	Quantity	Description
Chunk	SPEC	1	Table with spectral chunk
LineChunk	LINE	1	List of absorption lines found in a spectral chunk

Input Parameters

Name	Type	Description
width	double	Width of the interval used for fitting (nm)

Output data

Name	Format	Quantity	Description
LineChunk	LINE	1	List of absorption lines (with updated column *EW*)

Output Parameters
None

General description
A set of *n* Gaussian profiles is fitted to the lines listed in the table *LineChunk*. Each line is fitted in a wavelength interval of the input spectrum *Chunk* defined by *width*. Only pixels where *Chunk:MASK* is 0 are considered. The results of the fit are used to update some columns of *LineChunk* (*MU*, *MU_ERR*, *SIGMA*, *SIGMA_ERR*, *EW*, *EW_ERR*, *FWHM*, and *FWHM_ERR*). *LineChunk* is then sorted by ascending *EW*. *Chunk:MASK_LINE* is also updated to exclude the regions where lines have been detected.

Mathematical description

The Gaussian profiles are computed with a Levenberg-Marquardt minimization algorithm, implemented by GSL routine *gsl_multifit_fdfsolver_lmder* (TBC).

1. For each line *l* in table *LineChunk*, a region of width *width* is selected in the spectrum table *Chunk*, centered on the wavelength of the line, $(LineChunk:WAVEL)_l$

2. A Gaussian profile is fitted to the normalized spectrum in this region, excluding the pixels for which *Chunk:MASK*=1

$$G_l = \frac{1 - (LineChunk:PEAK)_l}{\sigma_l\sqrt{2\pi}} \exp\left\{-\frac{[(Chunk:FLUX)_r - \mu_l]^2}{2\sigma_l^2}\right\}. \qquad (8.21)$$

3. If M(x) is the mean value of the parameter *x* and S(x) its formal error obtained by the routine, the columns of *LineChunk* will be updated as follows, for each line *l*:

$$(LineChunk:MU)_l := M(\mu_l),$$
$$(LineChunk:MU\_ERR)_l := S(\mu_l);$$

$$(LineChunk:SIGMA)_l := M(\sigma_l),$$
$$(LineChunk:SIGMA\_ERR)_l := S(\sigma_l);$$

4. The equivalent width and its error are computed for each line *l*:

$$(LineChunk:EW)_l := width - [1 - (LineChunk:PEAK)_l] \times \mathrm{erf}\left[\frac{width}{2^{3/2}M(\sigma_l)}\right], \qquad (8.22)$$

$$(LineChunk:EW\_ERR)_l := \sqrt{(LineChunk:PEAK\_ERR)_l^2\left\{\mathrm{erf}\left[\frac{width}{2^{3/2}M(\sigma_l)}\right]\right\}^2 + [S(\sigma_l)]^2\left\{\frac{(LineChunk:PEAK)_l \times width}{M(\sigma_l)^2\sqrt{2\pi}}\exp\left[\frac{width}{2^{3/2}M(\sigma_l)}\right]\right\}^2} \qquad (8.23)$$

5. The FWHM and its error are computed for each line *l*:

$$(LineChunk:FWHM)_l := 2\sqrt{2\ln 2}\, M(\sigma_l), \qquad (8.24)$$

$$(LineChunk:FWHM\_ERR)_l := 2\sqrt{2\ln 2}\, S(\sigma_l). \qquad (8.25)$$

6. For each line *l* an interval equal to twice its FWHM will be masked. This mask is used in the determination of the continuum level in quasar spectra. *Chunk:MASK_LINE* will be updated as follows:

$$|(Chunk:WAVE)_r - (LineChunk:WAVEL)_l| < (LineChunk:FWHM)_l$$
$$\Rightarrow (Chunk:MASK\_LINE)_r := \text{TRUE}.$$

Quality assessment

The accuracy in the determination of the equivalent widths has to be photon noise limited, i.e. the error must be proportional to $1/\mathrm{SNR}$ up to SNR=2000 (AD-7).

Error/warning conditions

Inconsistent or badly formatted input: the process is interrupted and an error message is issued

Unit test
Use a synthetic spectrum with a typical shape to which a simulated line with known Gaussian parameters is added. Noise, extra lines and maybe CRH can also be added. The function is run on this spectrum and the results are compared with the expected ones.

8.93 Function espda_calib_teff (TBR)

Working remarks
None

Purpose
Compute the value of the effective temperature

Calling recipes
espda_compu_starpar (Section 4.6 and 9.6)

Input data

Name	Format	Quantity	Description
LineRatioCalib	LRCT	1	

Input parameters
None

Output data

Name	Format	Quantity	Description
TeffTable	TEFF	1	

Output parameters
None

General description
This function computes the value of the effective temperature based on the computed line ratios.

Mathematical description
The result is obtained as a weighted mean of all the individual line-ratio calibrations with the removal of 2 sigma outliers.

Quality assessment
The error is obtained from the standard deviation of all line ratios used.

Error/warning conditions
TBC

Unit test
The function is fed with the EWs of a known star and the computed effective temperature is compared with the known one.

8.94 Function espda_calib_feh (TBR)

Working remarks
None

Purpose
Compute the value of the [Fe/H] ratio

Calling recipes
espda_compu_starpar (Section 4.6 and 9.6)

Input data

Name	Format	Quantity	Description
LineListEW	LINE	1	
TeffTable	TEFF	1	
LineFeHcalib	LFCT	1	

Input parameters
None

Output data

Name	Format	Quantity	Description
FeHTable	FEH	1	

Output parameters
None

General description
This function computes the final value for [Fe/H] calibrations.

Mathematical description
The result is obtained as a weighted mean of all the individual line-ratio calibrations with the removal of 2 sigma outliers.

Quality assessment
The error is obtained from the standard deviation of all line ratios used.

Error/warning conditions
TBC

Unit test
The function is fed with the EWs and the temperature of a known star and the computed [Fe/H] is compared with the known one.

8.95 Function espda_cont_ord (TBR)

Working remarks
None

Purpose
Fit the continuum to a spectral order

Calling recipes
espda_fit_starcont (Sections 4.7 and 9.7)

Input data

Name	Format	Quantity	Description
OrdSpec	SPEC	1	

Input parameters

Name	Type	Default	Description
lrejt	double	0.5	Multiplication factor to determine the low rejection threshold
hrejt	double	2.5	Multiplication factor to determine the high rejection threshold
niter	integer	5	Number of iteration to fit the continuum
forder	integer	1	Order of the fitting function

Output data

Name	Format	Quantity	Description
OrdSpec	SPEC	1	

Output parameters
None

General description
This function performs the fit of the continuum of a stellar spectrum order by order, by iterating the procedure of definition of the continuum, rejecting the flux outliers. The flux of a spectral order in table *RebinSpec* is fitted using only the unmasked regions, i.e. considering only the rows where *OrdSpec:MASK* is FALSE. The output is used to update columns *OrdSpec: CONT* and *OrdSpec:CONT_ERR*.

Mathematical description
The function works as follows:

1. Iterating the fit of the continuum, typically 10 iterations
2. In each iteration the stellar flux is fitted and the standard deviation of the difference between the flux points and the fitting function is computed
3. The outliers in flux points are determined using the higher and lower rejection parameters and excluded in the next iteration
4. Once the number of iterations are done, finally the stellar flux is divided by fitting function value of the continuum position

Quality assessment
The accuracy in the continuum determination has to be down to the SNR level up to SNR=2000 (AD-7)

Error/warning conditions
Inconsistent or badly formatted input: the process is interrupted and an error message is issued

Unit test
Use a synthetic spectrum with a typical shape to which simulated lines, noise, and maybe CRH have been added. The function fits the continuum and the result is compared with the know shape. The accuracy is evaluated by computing the RMS pixel by pixel from the difference of the computed and input continua.

8.96 Function esp_interp_synth (TBR)

Working remarks
None

Purpose
Interpolate on the grid of synthetic spectra

Calling recipes
esp_synth_spec (Sections 4.8 and 9.8)

Input data

Name	Format	Quantity	Description
WaveSynthSpec	WSYNT	1	
[FluxSynthSpec]	FSYNT	6	

Input parameters

Name	Type	Default	Description
Teff-synth	double		Teff value of the synthetic spectrum
Logg-synth	double		Log (*g*) value of the synthetic spectrum
FeH-synth	double		[Fe/H] value of the synthetic spectrum

Output data

Name	Format	Quantity	Description
SynthSpec	SYNT	1	

Output parameters
None

General description
This function interpolates six synthetic spectra for a given set of three stellar parameters and metallicity using a tri-linear/cubic spline interpolation procedure.

Mathematical description
The function search for the six synthetic spectra used to perform the interpolation and then perform the interpolation using a tri-linear or a cubic spline function.

Quality assessment
The function is applied to proper test frames.

Error/warning conditions
The set of stellar parameters and metallicity fall outside of the grid of synthetic spectra: the process is interrupted and an error message is issued

Unit test
Use a given set of stellar parameters and metallicity, run the function and plot the computed interpolated synthetic spectrum with two models of the 6 synthetic models and check the result.

8.97 Function esp_convol_synth (TBR)

Working remarks
None

Purpose
Convolution of the synthetic spectra with the broadening profile

Calling recipes
esp_synth_spec (Sections 4.8 and 9.8)

Input data

Name	Format	Quantity	Description
LSynthSpec	LSPEC	1	*REBIN.LOG.VSTEP* : Pixel scale in velocity *REBIN.LOG.CWAVE* : Central wavelength in logarithmic scale

Input parameters

Name	Type	Default	Description
Vrot-synth	double		Vrot value of the synthetic spectrum
Vmac-synth	double		Vmac value of the synthetic spectrum

Output data

Name	Format	Quantity	Description
LSynthSpec	LSPEC	1	

Output parameters
None

General description
This function broadens the synthetic 1D spectrum, using a convolution of the synthetic 1D spectrum with the rotation profile, the macroturbulence profile and the instrumental profile.

Mathematical description
The normalized flux points of the synthetic spectrum *LSynthSpec:LFLUX* with the wavelength in logarithmic scale *LSynthSpec:LWAVE* are first convolved with macroturbulent profile, then with the rotational profile and finally with the instrumental profile.

Quality assessment
The function is applied to proper test frames.

Error/warning conditions
Not applicable.

Unit test
Use a given set of broadening parameters, run the function and plot the computed broadened synthetic spectrum with two models of the 6 synthetic models and check the result.

8.98 Function esp_cc_synth (TBR)

Working remarks
None

Purpose
Compute the cross-correlation function between the observed and synthetic 1D spectra

Calling recipes
esp_rv_synth (Sections 4.8 and 9.8)

Input data

Name	Format	Quantity	Description
Lspec or *LSpecSynth*	LSPEC	1	

Input parameters

Name	Type	Default	Description
ccf-rv-synth-cen	double	0.	Central RV (km/s)
ccf-rv-synth-window	double	350.	CCF window width (km/s)

Output data

Name	Format	Quantity	Description
CCFSynth	CCF	1	RV.SYNTH: radial velocity shift of the observed spectrum (km/s)

Output parameters
None

General description
This function computes the cross-correlation of the observed 1D spectrum with the synthetic 1D spectrum.

Mathematical description
1. Subtract 1 to the normalized fluxes of the observed and synthetic 1D spectra
2. Loop over each pixel/RV value:
    a. Shift the synthetic spectrum;
    b. Multiply the observed spectrum by the synthetic spectrum;
    c. Integrate/sum all multiplied fluxes for all wavelength points;
    d. Multiply the result by the step in RV estimated from the logarithmic wavelength scale of the input frames

Quality assessment
None

Error/warning conditions
None

Unit test
Generate a synthetic spectrum, broaden it with different broadening parameters, shift it, perform the cross-correlation and check the result.

8.99 Function esp_fit_ccf (TBR)

Working remarks
None

Purpose
Fit a single Gaussian to the cross-correlation function

Calling recipes
esp_rv_synth (Sections 4.9 and 9.9)

Input data

Name	Format	Quantity	Description
CCFSynth	CCF	1	RV.SYNTH: radial velocity shift of the observed spectrum (km/s) RVERR.SYNTH: error on radial velocity shift of the observed spectrum (km/s)

Input parameters
Radial velocity value with its error

Output data
None

Output parameters
None

General description
Fit of a Gaussian function to the CCF to derive the centroid

Mathematical description
The cross-correlation function *CCFSynth* is fitted with a Gaussian function and the radial velocity and its error are derived from its centroid. They are saved in the header of the frame *CCFsynth* in the keywords RV.SYNTH and RVERR.SYNTH

Quality assessment
None

Error/warning conditions

None

Unit test
Use the CCF generated in the unit test of function esp_cc_synth, run this function and check the result.

8.100 Function esp_fit_pl (TBR)

Working remarks
None

Purpose
Fit a power law in selected regions of a QSO spectrum.

Calling Recipes
esp_fit_qsocont (Sections 4.10 and 9.10)

Input data

Name	Format	Quantity	Description
Spec	SPEC	1	• *CORR.OPTDEPTH.FACTOR*: factor to correct flux for the optical depth

Input parameters
TBD

Output data

Name	Format	Quantity	Description
Spec	SPEC	1	• *FIT.PL.INDEX*: power-law index • *FIT.PL.INDEX.ERR*: formal error on *FIT.PL.INDEX* • *FIT.PL.NORM*: power-law normalization • *FIT.PL.NORM.ERR*: formal error on *FIT.PL.NORM*

Output parameters
None

General description
A power-law function is fitted to the flux of the spectrum *Spec*, considering only the unmasked pixels, where *MASK* and *MASK_LINE* are both FALSE. The flux is corrected for the effect of the optical depth using keyword *CORR.OPTDEPTH.FACTOR* (see Section 8.36). The power-law index and normalization, together with their formal errors, are saved as keywords of the spectrum *Spec*. The columns *Spec:CONT* and *Spec:CONT_ERR* are updated with the value of the power law computed as a function of *Spec:WAVE*.

Mathematical description
The power-law function is computed using the GSL routine *gsl_fit_linear*, which performs a least-square fitting to a straight line model to the dataset

$$\{\log[(Spec{:}WAVE)_r], \log[(Spec{:}FLUX)_r \times Spec{:}CORR.OPTDEPTH.FACTOR]\},$$

obtained extracting only the pixels r where $(Spec{:}MASK)_r$ = FALSE and $(Spec{:}MASK\_LINE)_r$ =

FALSE.

Quality assessment
TBD

Error/warning conditions
Inconsistent or badly formatted input: the process is interrupted and an error message is issued

Unit test
Use a synthetic spectrum with a known power law shape to which noise, absorption lines, bad pixels and CRH have been added. The function is run on this spectrum and the result is compared with the original continuum. The accuracy is evaluated by computing the RMS pixel by pixel from the difference of the computed and input continua.

Chapter 9. Data Analysis CPL plugin

In order to be used as part of the DFS pipeline the recipes have to be implemented using the CPL library's plugin interface. A pipeline recipe implementation using the plugin interface results in a dynamically loadable software module, which can be executed by different front-end applications (e.g. esorex or Reflex).
The 13 CPL plugins in the DAS are described in the following sections.
In the pseudo-codes, we have not written explicitly the operations of input reading and QC parameters writing.

9.1 CPL plugin for espda_coadd_spec

Purpose
Combine different spectra of the same object.

Input data

Description	Format	Quantity	Tag
1D merged spectrum	S1D	0 to 50	SPEC_1D
2D order spectrum of the flux	S2D	0 to 50	SPEC_FLUX_2D
2D order spectrum of the flux error	S2D	0 to 50	SPEC_FLUXERR_2D
2D order spectrum of the wavelength	S2D	0 to 50	WAVE_MATRIX_FIBER
2D order spectrum of the blaze function	S2D	0 to 50	BLAZE_FIBER
Resolution map	RES_MAP_FIBER	1	RES_MAP_FIBER

Note: SPEC_FLUX_2D must be present if SPEC_1D is not present. SPEC_FLUXERR_2D and WAVE_MATRIX_FIBER should be present in the same number as SPEC_FLUX_2D. BLAZE_FIBER and RES_MAP_FIBER are currently not used.

Relevant FITS keywords

Name	Frames	Type	Description
ESO.DAS.Z_EM	SPEC_1D, SPEC_FLUX_2D	double	Quasar emission redshift.

Input Parameters

Name	Type	Default	Range	Description
equal-flag	boolean	TRUE	TRUE, FALSE	Flag to equalize spectra.
equal-ref	integer	1	[0, number of exposures]	ID number of the reference exposure to be used in equalization.
kappa	integer	3	[3, 5]	Threshold for kappa-sigma clipping to be used in rebinning.
wavel-min	double	380.0	[0.0, 1.0e5]	Minimum wavelength [nm].
wavel-max	double	780.0	[0.0, 1.0e5]	Maximum wavelength [nm].
vel-step	double	5000	[3.0e2, 1.0e5]	Velocity step [m/s].

Algorithm pseudo-code

- Extract spectra from frames into tables (espda_frame_extract) [*The order spectra are put*

together into a full spectrum.]

- IF a full spectrum is provided THEN
  - Use the full spectrum as fiducial spectrum
- ELSE
  - Use the merged spectrum as fiducial spectrum
- ENDIF
- Read ESO.DAS.Z_EM from the fiducial spectrum (espda_frame_desc)
- IF ESO.DAS.Z_EM is 0 THEN
  - Shift the wavelengths to the rest frame (espda_spec_shift)
- ENDIF
- IF *equal-flag* is TRUE THEN
  - Equalize the flux in spectral orders and exposures (espda_spec_equal)
- ENDIF
- IF a full spectrum is provided THEN
  - Rebin the flux in the full spectrum (espda_spec_rebin)
- ENDIF
- Export the products from tables into frames (espda_frame_export) [*If no rebinned spectrum was produced the merged spectrum is exported as rebinned spectrum.*]

Output data

Description	Format	Quantity	Tag
Original full spectrum	SPEC_FULL	0 or 1	FSPEC_ORIG
Original merged spectrum	SPEC_RMS	0 or 1	RSPEC_ORIG
Full spectrum, optionally equalised	SPEC_FULL	0 or 1	FSPEC_PRE
Rebinned spectrum, optionally equalised	SPEC_RMS	1	RSPEC_PRE

Note: FSPEC_ORIG is produced only if SPEC_FLUX_2D, SPEC_FLUXERR_2D, and WAVE_MATRIX_FIBER are present.

QC1 parameters

Name	Frames	Type	Description
ESO.DAS.Z_EM	All	double	Quasar emission redshift.

Error conditions and handling

If the CPL error state changes during the execution of a function, the recipe is terminated.

9.2 CPL plugin for espda_mask_spec

Purpose

Create a spectral mask and apply it to a spectrum.

Input data

Description	Format	Quantity	Tag
Full spectrum	SPEC_FULL	0 or 1	FSPEC_PRE
Rebinned spectrum	SPEC_RMS	0 or 1	RSPEC_PRE

Note: RSPEC_PRE must be present if FSPEC_PRE is not present.

Relevant FITS keywords

None.

Input Parameters

Name	Type	Default	Range	Description
mask-limit	string	0.0-0.0	None	Limits of the regions to be masked [nm].
mask-exp	double	0	[0, total number of exposures]	Number of the exposure to be masked, 0 meaning all exposures.

Algorithm pseudo-code

- Load input frames (espda_frame_load)
- IF *mask-limit* is not empty THEN
  - Create the spectral mask (espda_mask_create)
- ENDIF
- IF a mask has been created THEN
  - Apply the spectral mask (espda_mask_apply)
  - IF a full spectrum is provided THEN
    - Rebin the flux in the masked full spectrum (espda_spec_rebin)
  - ENDIF
- ENDIF
- Export the products from tables into frames (espda_frame_export)

Output data

Description	Format	Quantity	Tag
Masked full spectrum	SPEC_FULL	0 or 1	FSPEC
Masked rebinned spectrum	SPEC_RMS	0 or 1	RSPEC
Mask	MASK	1	MASK

Note: FSPEC and RSPEC are produced in the same number as FSPEC_PRE and RSPEC_PRE, respectively.

QC1 parameters

Name	Frames	Type	Description
ESO.QC.MASK.NROW	FSPEC, RSPEC	integer	Number of masked rows.

Error conditions and handling

An error message is issued whenever a function is not terminated with success, and the recipe is also terminated.

9.3 CPL plugin for esp_compu_radvel (TBR)

Purpose

Compute the radial velocity for a stellar spectrum

Input data

Name	Format	Quantity	Tag
SPEC_2D	S2D	1	
BLAZE_FIBER	S2D	1	
WAVE_MATRIX_FIBER	S2D	1	
FLUX_TEMPLATE	FLTPL	1	
CCF_TEMPLATE	CCFTPL	1	

Relevant FITS keywords
None

Input Parameters

Name	Type	Default	Description
filter-cosm-ksig	double	4.5	Kappa parameter for the sigma-clipping
filter-cosm-winsize	integer	128	Size of the window in pixels
ccf-rv-cen	double	TBC	Central RV (km/s)
ccf-rv-step	double	TBC	RV step (km/s)
ccf-rv-window	double	TBC	CCF window width (km/s)
ccf-mask-width	double	TBC	Width of mask holes (km/s)
ccf-order-range	double	TBC	Spectral orders to consider (array?)
bis-step	double	TBC	Step in CCF depth

Algorithm pseudo-code
FilteredSPEC2D := esp_filter_cosmics(SPEC2D;filter-cosmics-par)
CorrSPEC2D := esp_correct_flux(FilteredSPEC2D,WAVE_MATRIX,FLUX_TEMPLATE)
berv := read FITS keyword from SPEC2D frame
Bary_WAVE_MATRIX := esp_shift_wavelength_scale(WAVE_MATRIX;berv)
CCF := esp_compute_ccf(CorrSPEC2D,BLAZE,Bary_WAVE_MATRIX,CCF_TEMPLATE;ccf-par)
BIS := esp_compute_bis(CCF;bis-par)
rv := read FITS keyword from CCF frame
Stellar_WAVE_MATRIX := esp_shift_wavelength_scale(Bary_WAVE_MATRIX;rv)

Output data

Name	Format	Quantity	Tag
SPEC_2D	S2D	1	
WAVE_MATRIX_FIBER	S2D	1	
CCF_FIBER	CCF	1	
BIS_FIBER	BIS	1	

QC1 parameters

Name	Type	Description
QC.FILTER.NCOS	integer	Number of cosmic hits found
QC.CCF.RV	double	Radial velocity (km/s)
QC.CCF.FWHM	double	CCF FWHM (km/s)
QC.CCF.CONTRAST	double	CCF contrast (%)
QC.CCF.NOISE	double	Photon noise on RV (km/s)
QC.BIS.SPAN	double	Bisector velocity span (km/s)

Error conditions and handling
Inconsistent or badly formatted input: the process is interrupted and an error message is issued

9.4 CPL plugin for esp_compu_rhk (TBR)

Purpose
Compute the activity indexes S and R'HK from a stellar spectrum

Input data

Name	Format	Quantity	Tag
SPEC_2D	S2D	1	
BLAZE_FIBER	S2D	1	
WAVE_MATRIX_FIBER	S2D	1	

Relevant FITS keywords
None

Input Parameters

Name	Type	Default	Description
filter-cosm-ksig	double	4.5	Kappa parameter for the sigma-clipping
filter-cosm-winsize	int	128	Size of the window in pixels
pbflux-wlen-start	double		Start wavelength (nm)
pbflux-wlen-end	double		End wavelength (nm)
bv	double		B-V color of the star

Algorithm pseudo-code
FilteredSPEC2D := esp_filter_cosmics(SPEC2D;filter-cosmics-par)
DeblazedSPEC2D := esp_correct_blaze(FilteredSPEC2D,BLAZE_FIBER)
H,H-err := esp_measure_flux(DeblazedSPEC2D,WAVE_MATRIX_FIBER;H-band-par)
K,K-err := esp_measure_flux(DeblazedSPEC2D,WAVE_MATRIX_FIBER;K-band-par)
R,R-err := esp_measure_flux(DeblazedSPEC2D,WAVE_MATRIX_FIBER;R-band-par)
V,V-err := esp_measure_flux(DeblazedSPEC2D,WAVE_MATRIX_FIBER;V-band-par)
S,S-err := esp_compute_s_index(H,H-err,K,K-err,R,R-err,V,V-err)
rhk,rhk-err := esp_compute_rhk_index(S,S-err,bv)
prot,prot-err,age,age-err := esp_compute_prot_age(rhk,rhk-err)

Output data

Name	Format	Quantity	Tag
SPEC_2D	S2D	1	

Other output (as FITS keywords in the SPEC_2D frame)

Name	Type	Description
s	double	Mt Wilson S index (unitless)
s-err	double	Formal error on *s* (unitless)
rhk	double	R'HK index
rhk-err	double	Formal error on *rhk*
prot	double	Rotation period
prot-err	double	Formal error on *prot*
age	double	Age
age-err	double	Formal error on *age*

QC1 parameters
None

Error conditions and handling
Inconsistent or badly formatted input: the process is interrupted and an error message is issued

9.5 CPL plugin for espda_compu_eqwidth (TBR)

Purpose: Compute the equivalent widths for a list of absorption lines.

Input data

Description	Format	Quantity	Tag
Full spectrum	SPEC_FULL	0 or 1	FSPEC_PRE
Rebinned spectrum	SPEC_RMS	0 or 1	RSPEC_PRE
Line list	LINE	0 or 1	DET_LINE

Note: RSPEC_PRE must be present if FSPEC_PRE is not present.

Relevant FITS keywords

None

Input Parameters

Name	Type	Default	Description
ew-ll-begin	double	380.0	Starting wavelength of the spectrum (nm)
ew-ll-end	double	990.0	Ending wavelength of the spectrum (nm)
ew-space	double	0.3	width of the chunks in which the spectrum is divided (nm)
ew-miniline	double	2.0	Minimum accepted value for an EW.
cont-rejt	double	-1	Rejection threshold for the continuum determination. The value will depend on the signal-to-noise
cont-iter	integer	5	Number of iteration to fit the continuum
det-thres	double	0.02	Normalized threshold for line acceptance
det-resol	double	0.01	Minimum accepted separation between two lines (in nm)
det-smwidth	integer	4	Number of pixels in the boxcar to be averaged
fit-width	double	1.0	Width of the interval used for fitting (nm)

Algorithm pseudo-code

```
do-iterations i
 if (Line(i) is in [ew-ll-begin,ew-ll-end] and in spectralLimits)
   reg_spec := espda_select_chunk(SPEC_1D, ew-space);
   reg_spec_cont := espda_fit_lcont(reg_spec, cont-par);
   line_det := espda_det_line(reg_spec_cont, det-line-par);
   reg_line_ew := espda_fit_ngauss(reg_spec_cont,line_det);
   if line-ew > ew-miniline
     copy reg_line_ew Line:EW
     total-lines++;
   end-if
 end-if
end
```

Output data

Name	Format	Quantity	Tag
LineListEW	LINE	1	

Other outputs

Name	Type	Description
[cont-res]	double	2nd order polynomial coefficient of the continuum fit

QC1 parameters

Name	Type	Description
QC.DET.NLINE	integer	Number of detected lines
QC.FIT.NLINE	integer	Number of fitted lines
[QC.FIT.NGAUSS.STAT.l]	integer	Status from the Gaussian fit

Error conditions and handling
TBD

9.6 CPL plugin for espda_compu_starpar (TBR)

Purpose: Estimate the effective temperature and [Fe/H] for solar-type star spectrum.

Input data

Name	Format	Quantity	Tag
LineListEW	LINE	1	
LineRatioCalib	LRCT	1	
LineFeHCalib	LFCT	1	

Relevant FITS keywords
None

Input Parameters
None

Algorithm pseudo-code
TeffTable := esp_calib_teff(LineRatioCalib, calib-teff-par);
FehTable := esp_calib_feh(LineListEW, TeffTable,LineFeHCalib);

Output data

Name	Format	Quantity	Tag
TeffTable	TEFF	1	TEFF_TABLE
FehTable	FEH	1	FEH_TABLE

QC1 parameters

Name	Type	Description
QC.NRATIO	integer	Number of computed line ratios
QC.TEFF.ERROR	double	Error obtained from the standard deviation of all line ratios used
QC.FEH.NRATIO	integer	Number of line ratios used for the computation of [Fe/H]

QC.FEH.ERROR	double	Error obtained from the standard deviation of all line ratios used

Error conditions and handling
TBD

9.7 CPL plugin for espda_fit_starcont (TBR)

Purpose
Fit the stellar continuum of 2D spectra order by order

Input data

Name	Format	Quantity	Tag
OrdSpec or *EqSpec*	SPEC	1	ORD_SPEC EQ_SPEC

Relevant FITS keywords
None

Input Parameters

Name	Type	Default	Description
cont-lrejt	double	0.5	Multiplication factor to determine the low rejection threshold
cont-hrejt	double	2.5	Multiplication factor to determine the high rejection threshold
cont-niter	integer	5	Number of iteration to fit the continuum
cont-forder	integer	1	Order of the fitting function
rebin-concat-filter	boolean	TRUE	Flag to filter deviating flux values
rebin-kappa	integer	3	Number of sigma for clipping. If equal to -1 the clipping is not carried out

Algorithm pseudo-code

```
for (i = 1 to m, i++)
  if (MaskOrd in input)
    OrdSpec(i) := espda_apply_mask(OrdSpec(i),MaskOrd;apply-mask-par)
  Endif
  OrdSpec(i)  := espda_cont_ord(OrdSpec(i);cont-ord-par)
endfor
RebinSpec := espda_rebin_flux(OrdSpec;rebin-concat-par)
```

Output data

Name	Format	Quantity	Tag
RebinSpec	SPEC	1	REBIN_SPEC

QC1 parameters

Name	Type	Description
[QC.ORDER.MEAN.IN.i]	double	Mean flux of the input orders (i = 1 … m)

[QC.ORDER.MEAN.OUT.i]	double	Mean flux of the output orders (i = 1 … m)

Error conditions and handling
Inconsistent or badly formatted input: the process is interrupted and an error message is issued

9.8 CPL plugin for espda_synth_spec (TBR)

Purpose
Intepolate three synthetic spectra within a grid and compare them with the observed spectrum

Input data

Name	Format	Quantity	Tag
Spec	SPEC	1	
WaveSynthSpec	SPEC	1	
[FluxSynthSpec]	SPEC	6	

Relevant FITS keywords
None

Input Parameters

Name	Type	Default	Description
Teff1-synth	double		Teff value of the 1st synthetic spectrum
Logg1-synth	double		Log (g) value of the 1st synthetic spectrum
FeH1-synth	double		[Fe/H] value of the 1st synthetic spectrum
Teff2-synth	double		Teff value of the 2nd synthetic spectrum
Logg2-synth	double		Log (g) value of the 2nd synthetic spectrum
FeH2-synth	double		[Fe/H] value of the 2nd synthetic spectrum
Teff3-synth	double		Teff value of the 3rd synthetic spectrum
Logg3-synth	double		Log (g) value of the 3rd synthetic spectrum
FeH3-synth	double		[Fe/H] value of the 3rd synthetic spectrum
Vrot1-synth	double		Vrot value of the 1st synthetic spectrum
Vmac1-synth	double		Vmac value of the 1st synthetic spectrum
Vrot2-synth	double		Vrot value of the 2nd synthetic spectrum
Vmac2-synth	double		Vmac value of the 2nd synthetic spectrum
Vrot3-synth	double		Vrot value of the 3rd synthetic spectrum
Vmac3-synth	double		Vmac value of the 3rd synthetic spectrum
Lamb0-synth	double	6470	Initial wavelength for χ^2 computation and plot
Lamb1-synth	double	6730	Final wavelength for χ^2 computation and plot

Algorithm pseudo-code

```
for (i = 1 to 3, i++)
  SynthSpec  := esp_interp_synth(WaveSynthSpec,[FluxSynthSpec]; interp-synth-par)
  LSynthSpec := esp_rebin_log(SynthSpec)
  LSynthSpec(i) := esp_convol_synth(LSynthSpec(i); convol-synth-par)
      SynthSpec := esp_rebin_lin(Spec,LSynthSpec)
endfor
```

Output data

Name	Format	Quantity

[SynthSpec]	SPEC	3

QC1 parameters

Name	Type	Description
QC.INTERP.SYNTH.NMOD	integer	Number of models used in the interpolation
QC.INTERP.SYNTH.STATUS	integer	Status of the interpolation procedure
QC.INTEGFLUX.IN	double	Integrated flux of the input spectrum
QC.INTEGFLUX.LOG	double	Integrated flux of the log spectrum
QC.INTEGFLUX.OUT	double	Integrated flux of the output spectrum

Error conditions and handling
Inconsistent or badly formatted input: the process is interrupted and an error message is issued

9.9 CPL plugin for espda_rv_synth (TBR)

Purpose
Compute the radial velocity for a stellar spectrum from the cross-correlation with a synthetic spectrum

Input data

Name	Format	Quantity
Spec	SPEC	1
SynthSpec	SPEC	1

Relevant FITS keywords
None

Input Parameters

Name	Type	Default	Description
ccf-rv-synth-cen	double	0.	Central RV (km/s)
ccf-rv-synth-window	double	350.	CCF window width (km/s)

Algorithm pseudo-code

```
LSpec := esp_rebin_log(Spec)
LSynthSpec := esp_rebin_log(SynthSpec)
if (Mask in input)
   LSpec := esp_apply_mask(LSpec,Mask;apply-mask-par)
Endif
CCFSynth := esp_cc_synth(LSpec,LSpecSynth;cc-synth-par)
RV.SYNTH := esp_fit_ccf(CCFSynth;fit_ccf-par)
rvs := read FITS keyword RV.SYNTH from CCFSynth frame
for (i = 1 to p, i++)
      Spec:WAVE(i)=Spec:WAVE(i)*SQRT((1+(rvs/c))/(1.-(rvs/c)))
endfor
```

Output data

Name	Format	Quantity
CCFSynth	CCF	1

QC1 parameters

Name	Type	Description
QC.CCF.SYNTH.RV	double	Radial velocity (km/s)
QC.CCF.SYNTH.FWHM	double	CCF FWHM (km/s)

Error conditions and handling
Inconsistent or badly formatted input: the process is interrupted and an error message is issued.

9.10 CPL plugin for espda_create_linelist

Purpose
Create a list of absorption lines found in a spectrum.

Input data

Description	Format	Quantity	Tag
Rebinned spectrum	SPEC_RMS	1	RSPEC
List of ionic transitions	ION	0 or 1	STAR_ION

Relevant FITS keywords

Name	Frames	Type	Description
ESO.DAS.Z_EM	SPEC_1D, SPEC_FLUX_2D	double	Quasar emission redshift.

Input Parameters

Name	Type	Def.	Range	Description
hwidth	double	1.0	[1.0e–2, 3.0]	Half width of the spectral chunks (for stellar spectra) or half width of undersampling (for QSO spectra).
overlap	double	0.5	[0.0, 0.7]	Relative size of the overlapping region between adjacent chunks
iter	integer	3	[1, 1e2]	Number of iterations to fit the local continuum.
thres	double	0.02	[1.0e–10, 1.0]	Normalized threshold for line acceptance.
resol	double	0.01	[1.0e–3, 1.0e–1]	Minimum allowed separation between two lines [nm].
smwidth	integer	20	[1, 1e2]	Number of pixels in the boxcar to be averaged to fit the continuum.
wdiff	double	0.02	[1.0e–3, 1.0e–1]	Maximum allowed difference between observed and laboratory wavelengths [nm].
add	string	[0]	None	Wavelengths of additional, manually-selected lines [nm].
kappa	integer	3	[3, 5]	Threshold for kappa-sigma clipping to be used in rebinning.
vel-step	double	5.0e3	[3.0e2, 1.0e10]	Velocity step for undersampling [m/s].

Note: iter and *wdiff* apply only to stellar spectra. *kappa* applies only to quasar spectra.

Algorithm pseudo-code

- Load input frames (espda_frame_load)
- Read ESO.DAS.Z_EM (espda_frame_desc)
- IF ESO.DAS.Z_EM is 0 THEN

    - Define spectrum chunks
    - FOR all chunks in the spectrum DO
        - Select the local spectrum (select_local_spec)
        - Perform a local normalization of the spectrum region (espda_fit_lcont)
        - Detect absorption lines (espda_det_line)
    - ENDFOR
    - Identify lines (espda_line_iden)
- ELSE
    - Undersample the spectrum (espda_spec_rebin)
    - Detect absorption lines (espda_line_detect)
    - Clean duplicate lines (espda_line_clean)
- ENDIF
- Add manually-selected lines to the line list (espda_line_add)
- Export the products from tables into frames (espda_frame_export)

Output data

Description	Format	Quantity	Tag
List of absorption lines	LINE_BASIC	1	FLINE_PRE
List of identified absorption lines	LINE_BASIC	0 or 1	FLINE_IDEN
List of non-identified absorption lines	LINE_BASIC	0 or 1	FLINE_REJ
Undersampled spectrum	SPEC_RMS	0 or 1	RSPEC_SMOOTH

Note: FLINE_IDEN and FLINE_REJ are produced only if STAR_ION is provided. RSPEC_SMOOTH is produced only if ESO.DAS.Z_EM is not 0.

QC1 parameters

Name	Frames	Type	Description
ESO.QC.DET.LINE_NUM	All	integer	Number of identified lines.

Error conditions and handling

If the CPL error state changes during the execution of a function, the recipe is terminated.

9.11 *CPL plugin for espda_fit_qsocont

Purpose

Fit the emission continuum in a QSO spectrum.

Input data

Description	Format	Quantity	Tag
Full spectrum	SPEC_FULL	0 or 1	FSPEC
Rebinned spectrum	SPEC_RMS	1	RSPEC
List of absorption lines	LINE_BASIC	0 or 1	FLINE_PRE
List of ionic transitions	ION	1	QSO_ION
Table with the Voigt function	VOIGT	0 or 1	VOIGT_FUNC

Note: VOIGT_FUNC is recommended (it speeds up the computation).

Relevant FITS keywords

Name	Frames	Type	Description

ESO.DAS.Z_EM	SPEC_1D, SPEC_FLUX_2D	double	Quasar emission redshift.

Input Parameters

Name	Type	Default	Range	Description
hwidth	double	0.1	[2.5e–2, 1.0e1]	Half width for line grouping [nm].
par-range	string	[1.0e-4, 10.0, 17.0, 2.0, 50.0, 0.0, 0.0]	None	Range of line parameters (redshift maximum variation, min column density, max column density, min thermal line width, max thermal line width, min turbulent line width, max turbulent line width)
peak-thres	double	3	[1.0e–2, 1.0]	Relative threshold to cut line peaks during normalization.
vel-sampl-blue	double	0.02	[1.0e2, 1.0e4]	Velocity sampling to fit a spline to the blue part [km s^{-1}].
vel-sampl-red	double	0.01	[1.0e2, 1.0e4]	Velocity sampling to fit a spline to the red part [km s^{-1}].
model-resol	double	6.0e4	[1.0e3, 1.0e6]	Resolution of the instrument model.
iter-max-temp	integer	5.0e3	[–1, 1e3]	Maximum number of lines fitted, –1 meaning all lines.

Algorithm pseudo-code

- Load input frames (espda_frame_load)
- Read ESO.DAS.Z_EM (espda_frame_desc)
- Split the blue and the red part based on the ESO.DAS.Z_EM value (espda_split_part)
- Model the instrument profile (espda_model_inst)
- IF the list of lines is provided THEN
    - FOR the blue and the red part DO
        - WHILE there are lines yet to be fitted DO
            - Correct the spectral flux for HI absorption (espda_spec_corr)
            - Define knots for spline and pieces (espda_spec_rebin) [*Knots are defined by undersampling the rebinned spectrum*]
            - Fit a cubic spline to the rebinned spectrum (espda_fit_cspline)
            - Resample the instrument profile on the spline-fitted spectrum (espda_interp_inst)
            - Prepare the line list for Voigt profile fitting (espda_voigt_prep)
            - FOR all pieces DO
                - FOR the blue part DO
                    - Select lines to process [*One more line is selected at each iteration, from the strongest to the weakest one*]
                - ENDDO
                - Fit absorption lines with Voigt profiles (espda_wrap_line)
                - Remove lines from the spline-fitted spectrum (espda_spec_remove)
            - ENDDO
            - Correct the flux back to the original level (espda_spec_corr)
        - ENDWHILE
        - Correct the spectral flux for the residual HI absorption in the line-removed spectrum (espda_spec_corr)

    - ENDDO
- ENDIF
- Define the continuum (espda_spec_rebin) [*The continuum is defined by oversampling the spline fitted to the line removed spectrum*]
- Fit a cubic spline to the rebinned and non-rebinned spectra (espda_fit_cspline)
- IF the full spectrum is provided THEN
    - Validate the continuum on the non-rebinned spectrum with a chi-squared test
- ENDIF
- Export the products from tables into frames (espda_frame_export)

Output data

Description	Format	Quantity	Tag
Full spectrum with continuum	SPEC_FIT	0 or 1	FSPEC_CONT
Full spectrum with lines removed	SPEC_FIT	0 or 1	FSPEC_REM
Rebinned spectrum with continuum	SPEC_FIT	1	RSPEC_CONT
Rebinned spectrum with lines removed	SPEC_FIT	1	RSPEC_REM
List of removed absorption lines	LINE_VOIGT	0 or 1	FLINE_REM

Note: FSPEC_CONT and FSPEC_REM are produced only if FSPEC is provided. FLINE_REM is produced only if FLINE_PRE is provided.

QC1 parameters

Name	Frames	Type	Description
ESO.QC.CONT.RCHISQ	FSPEC_CONT	double	Reduced chi-squared of continuum fitting.
ESO.QC.LINE.RCHISQ	FSPEC_LINE	double	Reduced chi-squared of line fitting.
ESO.DAS.Z_EM	FLINE_REM	double	Quasar emission redshift.
ESO.QC.COLDEN.THRES	FLINE_REM	double	Lower limit of completeness in the line column density distribution used for fitting.

Error conditions and handling

If the CPL error state changes during the execution of a function, the recipe is terminated.

9.12 CPL plugin for espda_iden_syst

Purpose

Identify absorption systems in a spectrum.

Input data

Description	Format	Quantity	Tag
List of absorption lines	LINE_VOIGT	1	FLINE_PRE
List of ionic transitions	ION	1	QSO_ION
Reduced list of ionic transitions	ION	0 or 1	QSO_ION_RED

Relevant FITS keywords

Name	Frames	Type	Description
ESO.DAS.Z_EM	SPEC_1D, SPEC_FLUX_2D	double	Quasar emission redshift.

Input Parameters

Name	Type	Def.	Range	Description

grp-width	double	0.08	[1.0e–2, 1.0]	Width for redshift grouping [nm].
grp-num	integer	4	[1, 10]	Minimum number of lines to define a group.
add-hwidth	double	0.2	[0.0, 1.0]	Half width for redshift grouping [nm].

Algorithm pseudo-code

- Load input frames (espda_frame_load)
- Read ESO.DAS.Z_EM (espda_frame_desc)
- Define groups of close lines in the line list (espda_line_group)
- IF QSO_ION_RED is not provided THEN
  - Create a short list of ionic transitions
- ENDIF
- Create a list of redshifts using QSO_ION_RED (espda_redsh_list)
- Group redshift values into candidate systems (espda_redsh_group)
- IF there are candidate systems THEN
  - Confirm the candidate systems (espda_syst_conf)
  - IF there are confirmed systems THEN
    - Add lines to the confirmed systems using QSO_ION (espda_syst_add)
    - Confirm the systems with added lines (espda_syst_conf)
  - ENDIF
- ENDIF
- IF there are confirmed systems THEN
  - Compute the average redshift of systems (espda_syst_ave)
  - Adjust the number of components in multiplets (espda_syst_adjust)
  - Put constraints on multiplet components (espda_syst_constr)
  - FOR all candidate systems DO
    - Associate blended lines to the candidate system (espda_syst_blend)
  - ENDDO
- ENDIF
- Export the products from tables into frames (espda_frame_export)

Output data

Description	Format	Quantity	Tag
Individual absorption systems	LINE_SYST	0 to N	SLINE_IDEN
List of identified lines	LINE_SYST	0 or 1	FLINE_IDEN
List of non-identified lines	LINE_VOIGT	0 or 1	FLINE_REJ
Reduced list of ionic transitions	ION	0 or 1	QSO_ION_RED

Note: QSO_ION_RED is produced only if QSO_ION_RED is not provided in input.

QC1 parameters

Name	Frames	Type	Description
ESO.QC.SYSTID	SLINE_IDEN	string	ID of the absorption system.
ESO.QC.SYSTZ	SLINE_IDEN	double	Redshift of the absorption system.

Error conditions and handling

If the CPL error state changes during the execution of a function, the recipe is terminated.

9.13 *CPL plugin for espda_fit_line

Purpose
Fit an absorption system with Voigt profiles.

Input data

Description	Format	Quantity	Tag
Full spectrum with continuum	SPEC_FIT	1	FSPEC_CONT
Rebinned spectrum with continuum	SPEC_FIT	1	RSPEC_CONT
Individual absorption system	LINE_SYST	0 or 1	SLINE_IDEN
List of identified lines	LINE_SYST	0 or 1	FLINE_IDEN
List of non-identified lines	LINE_VOIGT	0 or 1	FLINE_REJ
List of ionic transitions	ION	1	QSO_ION
Table with the Voigt function	VOIGT	0 or 1	VOIGT_FUNC

Note: FLINE_IDEN is not considered if SLINE_IDEN is provided. FLINE_REJ is not considered if either SLINE_IDEN or FLINE_IDEN are provided. VOIGT_FUNC is recommended (it speeds up the computation).

Relevant FITS keywords

Name	Frames	Type	Description
ESO.DAS.Z_EM	SPEC_1D, SPEC_FLUX_2D	double	Quasar emission redshift.

Input Parameters

Name	Type	Default	Range	Description
hwidth	double	0.1	[2.5e–2, 1.0e1]	Half width for line grouping [nm].
par-range	string	[1.0e-4, 10.0, 17.0, 2.0, 50.0, 0.0, 0.0]	None	Range of line parameters (redshift maximum variation, min column density, max column density, min thermal line width, max thermal line width, min turbulent line width, max turbulent line width)
edit-name	string	VOID	None	Name of the file with instructions to edit the line parameters.
add	string	0	None	Wavelengths of additional, manually-selected lines.
fit-flag	boolean	TRUE	[TRUE, FALSE]	Flag to fit the lines.
iter-max	integer	1000	[1e3, 1e5]	Maximum number of iterations.
grp-sel	integer	0	[0, number of groups]	Group of lines selected for fitting, 0 meaning all groups.
model-resol	double	6.0e4	[1.0e3, 1.0e6]	Resolution of the instrument model.

Algorithm pseudo-code

- Load input frames (espda_frame_load)
- Load input parameters (espda_param_load)
- Read ESO.DAS.Z_EM (espda_frame_desc)
- Add manually-selected lines to the line list
- WHILE the chi-squared is not satisfying DO
    - Fit absorption lines with Voigt profiles (espda_wrap_line)
    - Validate the line fit on the non-rebinned spectrum with a chi-squared test

    - Add components to the system
- ENDWHILE
- Export the products from tables into frames (espda_frame_export)

Output data

Description	Format	Quantity	Tag
Full spectrum with Voigt guess	SPEC_FIT	0 or 1	FSPEC_GUESS
Rebinned spectrum with Voigt guess	SPEC_FIT	0 or 1	RSPEC_GUESS
List of Voigt-guessed lines	LINE_SYST or LINE_VOIGT	0 or 1	SLINE_GUESS
Full spectrum with Voigt fit	SPEC_FIT	0 or 1	FSPEC_VOIGT
Rebinned spectrum with Voigt fit	SPEC_FIT	0 or 1	RSPEC_VOIGT
List of Voigt-fitted lines	LINE_SYST or LINE_VOIGT	0 or 1	SLINE_VOIGT

Note: *_GUESS (*_VOIGT) frames are produced only if *fit-flag* is FALSE (TRUE).

QC1 parameters

Name	Frames	Type	Description
ESO.QC.LINE.RCHISQ	All	double	Reduced chi-squared of line fitting.
ESO.QC.SYSTID	All	string	ID of the absorption system.
ESO.QC.SYSTZ	All	double	Redshift of the absorption system.

Error conditions and handling

If the CPL error state changes during the execution of a function, the recipe is terminated.

Chapter 10. Validation and Tests

The scientific validation strategy of the implemented algorithms, CPL recipes and Reflex workflows is described in a separate document [AD-5].

Chapter 11. Development Plan

The development of the ESPRESSO Data Analysis Library (DAL) functions will follow the major milestones of the instrument and those described in Chapter 5 of [AD-1]. Note that, in general, the documents and software due at PAE, COM1, COM2 and PAC will be released 4 weeks in advance with respect to the reported dates.
After FDR, we foresee the following development plan (dates are only indicative):

- **May 2013 – Feb 2017** DAL coding and testing.
  Eight partial releases of the DAL are foreseen (see Table 11.1).
  Each delivery of the DAL will include the corresponding test data, an updated version of the DAL Design document and of the DAL Validation and Test document.
- **April 2017** PAE. Release of the DAL v1.0, and of the DAL Test and Validation document v2.0
- **end 2017** COM 1. Release of the DAL v1.1 and of the DA Reflex Workflows v0.5
- **mid 2018** COM 2. Release of the DAL v1.2 and of the DA Reflex Workflows v1.
- **end 2018** PAC. Release of the DAL v1.x and DAL Design document v2.

Version	Recipes coded (with their functions and tests)	Milestones
DAL v0.1.0	espda_mask_spec	November 15, 2013
DAL v0.2.0	espda_compu_eqwidth, espda_coadd_spec, espda_create_linelist	February 14, 2014
DAL v0.3	espda_fit_line	May 16, 2014
DAL v0.3.1	espda_compu_starpar	July 15, 2014
DAL v0.4	QSO workflow, espda_fit_starcont	November 15, 2014
DAL v0.5	espda_fit_qsocont, espda_synth_spec	May 14, 2015
DAL v0.6	espda_iden_syst, espda_rv_synth, star workflow I	October 26, 2015
DAL v0.7	espda_compu_radvel	March 31, 2016
DAL v0.8		
DAL v1.0		PAE
DAL v1.1		COM1
DAL v1.2		COM2
DAL v1.3		PAC

Table 11.1 - DAL Development Plan

Appendix A: QC1 Parameters (TBR)

QC1 parameters of recipe espda_coadd_spec

Name	Type	Description
QC.OBJ.Z_EM	double	Emission redshift (0 for stellar spectra)

QC1 parameters of recipe espda_mask_spec

Name	Type	Description
QC.MASK.ROW.NUM	integer	Number of masked rows

QC1 parameters of recipe esp_compu_radvel

Name	Type	Description
QC.FILTER.NCOS	integer	Number of cosmic hits found
QC.CCF.RV	double	Radial velocity (km/s)
QC.CCF.FWHM	double	CCF FWHM (km/s)
QC.CCF.CONTRAST	double	CCF contrast (%)
QC.CCF.NOISE	double	Photon noise on RV (km/s)
QC.BIS.SPAN	double	Bisector velocity span (km/s)

QC1 parameters of recipe esp_compu_rhk
None

QC1 parameters of recipe espda_compu_eqwidth

Name	Type	Description
QC.DET.NLINE	integer	Number of detected lines
QC.FIT.NLINE	integer	Number of fitted lines
[QC.FIT.NGAUSS.STAT.l]	integer	Status from the Gaussian fit

QC1 parameters of recipe espda_compu_starpar

Name	Type	Description
QC.NRATIO	integer	Number of computed line ratios
QC.TEFF.ERROR	double	Error obtained from the standard deviation of all line ratios used
QC.FEH.NRATIO	integer	Number of line ratios used for the computation of [Fe/H]
QC.FEH.ERROR	double	Error obtained from the standard deviation of all line

		ratios used

QC1 parameters of recipe esp_fit_starcont

Name	Type	Description
[QC.ORDER.MEAN.IN.i]	double	Mean flux of the input orders (i = 1 … m)
[QC.ORDER.MEAN.OUT.i]	double	Mean flux of the output orders (i = 1 … m)

QC1 parameters of recipe esp_synth_spec

Name	Type	Description
QC.INTERP.SYNTH.NMOD	integer	Number of models used in the interpolation
QC.INTERP.SYNTH.STATUS	integer	Status of the interpolation procedure
QC.INTEGFLUX.IN	double	Integrated flux of the input spectrum
QC.INTEGFLUX.LOG	double	Integrated flux of the log spectrum
QC.INTEGFLUX.OUT	double	Integrated flux of the output spectrum

QC1 parameters of recipe esp_rv_synth

Name	Type	Description
QC.CCF.SYNTH.RV	double	Radial velocity (km/s)
QC.CCF.SYNTH.FWHM	double	CCF FWHM (km/s)

QC1 parameters of recipe esp_fit_qsocont

Name	Type	Description
QC.PL.NPIX	integer	Number of pixels in the regions used for the pl fit
QC.DET.NLINE	integer	Number of detected lines
QC.FIT.NLINE	integer	Number of fitted lines
[QC.FIT.NGAUSS.STAT.l]	integer	Status from the Gaussian fit
QC.PL.INDEX	double	Power-law index
QC.PL.INDEX.ERR	double	Formal error on *QC.FIT.PL.INDEX*
QC.PL.NORM	double	Power-law normalization
QC.PL.NORM.ERR	double	Formal error on *QC.FIT.PL.NORM*

QC1 parameters of recipe espda_create_linelist

Name	Type	Description
QC.DET.LINE.NUM	integer	Number of detected lines
QC.IDEN.STARLINE.NUM	integer	Number of identified lines

QC1 parameters of recipe esp_iden_syst

Name	Type	Description
QC.LIST.NZ	integer	Number of computed redshifts
[QC.COIN.NZ]	integer	Number of lines assigned to a candidate system
[QC.MULT.NZ]	integer	Number of lines with multiple assignments
QC.NCAND	integer	Number of candidate systems
QC.NSYST	integer	Number of selected systems
QC.QSO.NNONIDEN	integer	Number of lines non-identified by esp_sele_syst
QC.ADD.NZ	integer	Number of lines added to a selected system
QC.ADD.NNONIDEN	integer	Number of lines non-identified by esp_add_atom

QC1 parameters of recipe espda_fit_line

None