

Spectropython_English

Manual of Capabilities

What the 192 scripts in the collection can do – arranged by task, from observation planning to time-series analysis

As of: 2026-10-05

Author: Lothar Schanne

Contents

1 Introduction.....	4
1.1 Working environment.....	4
2 The processing chain at a glance.....	5
3 Preparing observations and obtaining data.....	6
3.1 Planning an observing night.....	6
3.2 Querying object and line data.....	6
3.3 Line database.....	6
4 Reducing raw data.....	7
4.1 Master frames and visual inspection.....	7
4.2 Echelle pipeline (echelle spectrographs).....	7
4.3 Slit spectrograph and slitless spectrograph.....	8
4.4 NRES spectra (Las Cumbres Observatory).....	8
5 Managing and checking spectrum series.....	9
5.1 Headers and time stamps.....	9
5.2 Inventory and quality control.....	9
5.3 Renaming and converting files.....	9
6 Preparing and correcting spectra.....	10
6.1 Continuum normalisation.....	10
6.2 Barycentric and telluric correction.....	10
6.3 Correcting the radial velocity.....	10
6.4 Cropping, averaging, rebinning, smoothing.....	11
6.5 Difference spectra.....	11
6.6 Adding noise.....	11
7 Measuring lines and spectra.....	12
7.1 Equivalent width, line depth, position and width.....	12
7.2 Profile fitting.....	12
7.3 Measuring radial velocity.....	13
7.4 Bisectors and line asymmetry.....	13
7.5 Rotation and macroturbulence.....	13
7.6 Temperature and electron pressure.....	14
8 Displaying spectra.....	15
8.1 Viewing individual spectra.....	15
8.2 Multiple spectra and time series.....	15
8.3 Line profiles in velocity space.....	15
8.4 Plotting measurement tables.....	16

9 Model spectra and convolution.....	17
10 Binary stars and light curves.....	18
10.1 Period, phase and orbit.....	18
10.2 Component spectra.....	18
10.3 AAVSO light curves.....	18
11 Time-series analysis (TSA).....	19
11.1 Preparing data.....	19
11.2 Displaying measurement series.....	19
11.3 Generating test data.....	19
11.4 Period analysis.....	20
11.5 Time-frequency analysis.....	20
11.6 Noise and stochastic models.....	21
11.7 Comparing lines with one another.....	21
11.8 Causality.....	22
11.9 Nonlinear dynamics.....	22
11.10 Statistics.....	23
11.11 Systematic analysis chain.....	23
12 Typical workflows.....	25
12.1 Reducing an observing night with the echelle spectrograph.....	25
12.2 Integrating a new echelle spectrograph.....	25
12.3 From the spectrum series to the line time series.....	25
12.4 Investigating periods and layer structure of a pulsating atmosphere.....	25
12.5 Determining the orbit of a spectroscopic binary.....	26
12.6 Following emission lines of Be stars and reporting them to BeSS.....	26
12.7 Checking the robustness of a result.....	26
13 Conventions, formats and dependencies.....	27
13.1 Data formats.....	27
13.2 Header entries of the correction scripts.....	27
13.3 Required packages.....	27
13.4 Shared modules.....	27
Appendix A - Script index.....	29

1 Introduction

The Spectropython_English collection contains 192 Python scripts and modules for amateur spectroscopy: from planning an observing night, through the reduction of raw CCD exposures, to the statistical and nonlinear analysis of long measurement series. This manual describes the collection by task: for each question it shows which scripts are available, what they do, what inputs they expect and what they output.

It complements the existing documents:

- Uebersicht_Ordner_und_Skripte.docx – directory of all scripts by folder (as of 2026-10-04).
- <script_name>_Documentation.docx – detailed description of an individual script with all its parameters; each one is located in the folder of its script.
- Echelle_Pipeline_Documentation.docx – complete description of the Echelle reduction.
- TSA/90_Documentation – method descriptions for the time-series analysis.

For operating details (parameter blocks, dialogs, column names), the documentation of the respective script is authoritative. This manual is meant for orientation: what is possible, and where do I start?

1.1 Working environment

- Python 3.13 (environment astro-env) on Linux, developed and run in Spyder with the Qt5 backend.
- Most scripts ask for files and parameters through Qt5 dialogs and do without input(); they run with F5 in Spyder. Many can additionally be controlled from the command line or through a configuration block at the top of the script.
- Shared modules (line_lists.py, rv_tools.py, observatories.py) are located in the folder Building_Blocks and must be on the PYTHONPATH (Spyder: PYTHONPATH manager).
- Original files remain unchanged. Results are saved as new files with a name suffix or in subfolders; scripts that modify headers create backup copies (*.bak) on request.

2 The processing chain at a glance

Together, the folders form a continuous processing chain. The following table assigns the work steps to the chapters of this manual.

Step	What happens	Chapter / folder
1 Prepare	Observability, object data, archive data, line wavelengths	3 – astroplanner, Catalogue_Queries, Format_Conversion
2 Reduce	Master frames, extraction, wavelength calibration, normalisation, merging of the orders	4 – Reduce, NRES
3 Check the series	Headers, time stamps, inventory, quality control, removal of cosmic rays	5 – Header, Series, SNR
4 Process	Continuum normalisation, barycentric and telluric correction, RV zero point, cropping, averaging, smoothing, difference spectra	6 – Normalisation, BC, Drying, RV, Series, Rebinning...
5 Measure	EW, line depth, FWHM, RV, profile fits, bisector, $v \sin i$, asymmetry, temperature	7 – EW, Line_Parameter_Determination, Line_Fitting, CC, RV, TSA/60
6 Display	Viewers, comparison plots and time-series plots, dynamic spectra, animations	8 – Spectra_Plots, Velocity_Space_Plots, Plots, ascii
7 Model	Synthetic spectra, convolution, rotational broadening	9 – Theoretical_Spectra, Convolution
8 Binary stars	Period, phase, orbital elements, component spectra, light curves	10 – binaries_..., Light_Curves
9 Time series	Periods, time-frequency, line comparison, lags, causality, nonlinear dynamics, statistics	11 – TSA

Chapter 12 describes typical workflows that combine several of these steps; Chapter 13 summarises conventions, file formats and required packages. Appendix A lists all scripts with a reference to the chapter.

3 Preparing observations and obtaining data

3.1 Planning an observing night

astroplanner_20260924.py shows, for an observing site and a night, the altitude curve of one or more objects in local time, with airmass, twilight phases, Moon altitude, Moon illumination and Moon distance. It calculates culmination as well as rising and setting above a minimum altitude, and outputs the usable window (object high enough and Sun below the chosen twilight limit). Optionally it creates an annual overview of the usable dark hours. Objects are specified by name (Sesame, with a local cache) or by coordinates.

3.2 Querying object and line data

Script	Capabilities
simbad_query_a.py	Object profile from SIMBAD for one or more objects: coordinates, proper motion, parallax, RV, spectral type, magnitudes U...K and G, all identifiers; measurement tables (Teff/log g/[Fe/H], v sin i, variability, diameters and others); optional Excel export with one sheet per table.
nist_line_wavelengths.py	Line wavelengths from the NIST Atomic Spectra Database for one or more ions and a wavelength range, in air or vacuum, with an intensity threshold; output as Excel and as a text list for your own line lists.
eso_query.py	Search and download of reduced Phase 3 data from the ESO archive (e.g. HARPS, UVES, FEROS, ESPRESSO) by object and search box; overview, selection, CSV table and download, optionally with login.
format_conversions.py	Interactive conversions: JD/MJD ↔ calendar date, RA/DEC sexagesimal ↔ degrees, HJD, BJD_TDB, barycentric RV correction, altitude, azimuth and airmass; the functions can also be imported.

3.3 Line database

Building_Blocks/line_lists.py contains a line database with 228 lines of 21 ions plus telluric lines, with search, filter, selection and export functions. More than 20 scripts access it, for example for line selection by dialog, for line markers in the viewer or for atomic data (E_low, gf) in the temperature determinations. New lines - e.g. from the NIST query - are entered here and are then available in all scripts.

4 Reducing raw data

4.1 Master frames and visual inspection

Script	Capabilities
CreateMasters_CCD_2d_fits.py	Combines several 2D exposures per group (science, dark, calibration light, calibration dark, flat, flat dark) into master frames (mean, median or σ -clipping) and produces the dark-current-corrected masters. Interactive or via the command line.
FITS_2D_Viewer.py	Viewer for 2D FITS: ZScale/MinMax/percentile scaling, linear/square-root/log/asinh stretch, pixel info, row and column profile on click, marking of saturated pixels, paging through the folder, header window with search, PNG export.
fits_to_png.py	Exports all FITS files of a folder (1D, 2D, first plane of cubes) as PNG - useful as a quick image overview of a night (see Chapter 8).

4.2 Echelle pipeline (echelle spectrographs)

The pipeline in the folder Reduce/Echelle leads from master flat, master ThAr and science exposure to the normalised, merged 1D spectrum. It was developed for FLISES ($R \approx 18,000$) and is also used for the instruments of other observers; the associated verified line catalogues are located in the subfolders verifiedVersion_*

Step	Script	What it does
1	Echelle_Extraction.py	Order detection in the flat, tracing by polynomial, fixed order numbers via a reference table, scattered-light correction, extraction with adaptive aperture (sum or optimal extraction) including error spectrum, flat-fielding.
2	ThAr_Calibration.py	Wavelength calibration against the instrument-specific line list: two anchor clicks, echelle model, robust grid search per order, Gaussian centroids, global 2D fit $\lambda \cdot m = P(p, m)$ to detect misassignments, application to ThAr and science.
3	Normalisation_Orders.py	Continuum normalisation of every order: cosmic-ray masking, asymmetric clipping in log flux, symmetric final fit, polynomial or spline.
4	Merge_Orders.py	Merging of the orders onto a common linear grid with $\cos^2$ weighting in the overlaps, optionally S/N-weighted; diagnosis of the continuum offset.
-	Live_Reduction.py	Monitors an input folder during the night and automatically reduces every new science exposure with steps 1-4; calibration with the evening ThAr or an adopted solution.

Tools for calibration and diagnostics:

Script	Capabilities
Echelle_InstrumentSpecific_LineCatalogue_Creation.py	Creates the identification table (batchesLINE format: X, ORDER, WAVEC, IDENT, PEAK) for a new or modified echelle spectrograph from extracted ThAr orders and a general ThAr catalogue; anchors are saved and reused.
ThAr_LineCounter.py	Counts usable, saturated and blended ThAr lines per order and outputs FWHM medians – a check of exposure, focus and resolution.
Diagnosis_LineYield.py	Shows, per order and peak threshold, how many lines each filter step of the assignment costs; clarifies whether a lower threshold helps in orders with few lines.
ThAr_LineCatalogue_Viewer.py	Displays a line catalogue as a synthetic ThAr spectrum; clicking on a line shows wavelength, amplitude, order and pixel position.
ThAr_Image_Show.py	Displays a 2D ThAr exposure and saves it as PNG.

4.3 Slit spectrograph and slitless spectrograph

For classical spectrographs, the folder Reduce/Classical_Spectrograph provides a complete workflow, either as individual scripts or as a graphical user interface. Both use the same calculation functions (spectro_core.py, spectro_workflow.py) and therefore deliver identical results.

Script	Capabilities
2._AlignSpectrumStrip_CCD_Images_fits.py	Rotates the spectrum strip to the horizontal; angle determined automatically from the trace, by two clicks or given as a fixed value; application to further exposures of the night.
3._ExtractionAndWavelengthCalibrationNormalisation_withSlit_...py	Extraction with sky subtraction and cosmic-ray filter, calibration lamp without sky subtraction, lines clicked and centred to sub-pixel accuracy, polynomial $\lambda(\text{pixel})$ with residuals, linear grid, optional normalisation. Parameters are saved as JSON and reused for further exposures of the same night.
3a_ExtractionAndWavelengthCalibrationNormalisation_slitless_...py	The same for slitless spectra; calibration approximately via known lines of the stellar spectrum itself, otherwise in pixels.
Reduce_GUI.py	Graphical interface for the entire workflow: masters, alignment, extraction, wavelength calibration, normalisation, display.

4.4 NRES spectra (Las Cumbres Observatory)

Sequence: NRES_extract.py extracts the 68 orders of the science fibre as 1D FITS at 0.05 Å/pixel → VacToAir.py converts from vacuum to air wavelengths → heliocentricCorrection_NRES.py applies the barycentric correction from the header value BARYCORR → PlotAllOrders.py produces a control plot.

5 Managing and checking spectrum series

Before a series is evaluated, the time stamps, headers and data quality should be checked. The following scripts each work on an entire series of 1D FITS spectra.

5.1 Headers and time stamps

Script	Capabilities
fits_header_check_correct.py	Finds and corrects violations of the FITS standard: umlauts and special characters, angles and numbers with units without quotation marks, lower-case keys and others; reports unclear cases (MJD keys with a full JD, non-ISO-compliant dates). Check report as CSV.
jd_header_check.py	Reads all time keywords, converts them to JD (UTC), compares them with one another and, on request, writes the keyword JD (start or middle of the exposure). HJD/BJD are not silently adopted as JD.
header_print_series_csv.py	Headers of all files as a CSV per file, as an overall table (file × keyword) and as a comparison report: which keywords are constant, vary or are missing.
BESS_fits_header_series.py	Completes the headers of a series in BeSS-compliant form; observer-specific entries are set automatically, missing object-related information is requested.

5.2 Inventory and quality control

Script	Capabilities
TimeSeries_Overview.py	Table per spectrum: time, wavelength range, dispersion, flux minimum/maximum, S/N estimate; plus a summary (time span, nights, gaps) and an overview graphic.
spectra_diagnosis.py	Actual wavelength and JD ranges of the spectra (linear and log-linear dispersion, units) with recommendations for time-series plots.
spectra_quality_control.py	Interactive review: S/N, NaN, zero and negative pixels, spikes, continuum level, warnings; keep or discard by key press, comparison with the series median, log QC_Log.csv, resuming interrupted sessions, discarded files moved to a subfolder.
SNR_Determination.py	S/N of a series with six methods: continuum window, DER_SNR, beta-sigma, moving window, autocorrelation, Poisson. Detects resampled spectra (correlated noise) and corrects the difference methods accordingly.
remove_cosmics.py	Detects cosmic rays and hot pixels in 1D spectra from their width and sharpness and replaces them by interpolation; spectral lines are preserved.

5.3 Renaming and converting files

Script	Capabilities
JD_In_Filename.py	Copies with the JD (and optionally the wavelength range) at the start of the file name - series then sort themselves chronologically.
fits_series_to_csv_and_dat.py	FITS series as CSV (comma) and DAT (tab) with columns WAVE, FLUX. The header is lost in the process.

6 Preparing and correcting spectra

6.1 Continuum normalisation

Script	Capabilities
Universal_Normalisation.py	Normalisation of whole 1D spectra: stiff initial spline with confirmation, then iterative clipping; either single-phase or 1-point normalisation (median). Batch mode for whole directories with log file.
Normalisation_Single_Lines_20260923.py	Normalisation of the surroundings of a line via support points (by click, CSV or list of numbers), flux per support point determined robustly as a percentile; support points can be carried along with the line and then avoid exclusion regions. For time series with identical support points.
support_point_validator_single_lines_20260923.py	Checks whether support points can end up inside lines for RV-variable lines (γ , K, barycentric correction) and suggests replacement points.
Normalisation_Orders.py	Normalisation of individual echelle orders (Chapter 4).

6.2 Barycentric and telluric correction

Script	Capabilities
barycentric_correction_spectra.py	Heliocentric correction of a series; observer location from observatories.py, stellar coordinates online or manual; log CSV. Residual error up to a few tens of m/s compared with a true barycentric correction.
telluric_bc_correction.py	Telluric correction (“drying”) after Hanuschik (2006) followed by barycentric correction with barycorrpy in the range 6500–6620 Å; output <name>_bc_dry.fits.
telluric_drying_hitran.py	Telluric correction without barycentric correction: Hanuschik list, supplemented in the H α gap by HITRAN H ₂ O lines; fit of water column and instrumental FWHM; optionally on the original grid (no resampling).
heliocentricCorrection_NRES.py	Correction of NRES spectra from the header value BARYCORR.
Telluric_Zero_Point.py	Zero-point determination from telluric lines per spectrum, with time series of the offset.

6.3 Correcting the radial velocity

The scripts in the folder RV use the shared module rv_tools.py and document every shift in the header in a uniform way (RVMEAS, RVERR, RVSHIFT, RVSHTOT, RVMETH, HISTORY). Several successive corrections thus remain traceable.

Script	Capabilities
telluricLines_RVCorrection_RBF_...py / ...Regression_...py	Correction of the wavelength zero point on telluric lines (minimum by radial basis function or polynomial, respectively); spectra must not yet be barycentrically corrected.
RVCorrectionByTemplate_1d_spectra_fits_series.py	RV relative to a template (file or iteratively formed series median) by cross-correlation; shifting of all spectra to the template, with selectable wavelength ranges and exclusions.
RVCorrection_1d_spectra_fits_series.py	Shift by a fixed value or by tabulated values from a log CSV - e.g. to transfer a correction determined once to other versions of the same spectra.
spectrum_conversion_barycentric_to_stellar_rest_frame.py	Transfers spectra into the rest frame of a binary component (Chapter 10).

6.4 Cropping, averaging, rebinning, smoothing

Script	Capabilities
Crop_Section_TimeSeries.py	Crops FITS and ASCII spectra to a wavelength range; shows beforehand how many spectra contain the range completely.
Average_Of_fits.py	Mean spectrum from any number of spectra, also with different grids: median, mean or S/N^2 -weighted, with σ -clipping, minimum number of contributing spectra per pixel; output with extensions NCOMB and SIGMA.
newbinning_mean_spectrum_v2.py	Rebinning of a series onto a common grid (linear, spline or flux-conserving), mean spectrum and statistics spectrum (mean, median, scatter, count per pixel).
TimeSeries_Spectra_Smooth.py	Smoothing with Gaussian to a target R, fixed km/s or pixels, boxcar, Savitzky-Golay or median; effective resolution and noise factor are calculated and written to the header.

6.5 Difference spectra

DifferenceSpectra_v2.py forms, for each spectrum of the series, the difference to a reference spectrum (file, mean or median of the series), optionally with a combined plot, a dynamic difference spectrum and a scatter spectrum $\sigma(\lambda)$.

DifferenceSpectra_withPlotAll_v2.py calls the same script with the combined plot. For line sections in velocity space see Chapter 7 (SpectraDifferenceToReferenceSpectrum_Line.py, HalphaVariability.py).

6.6 Adding noise

add_noise.py adds Gaussian noise to spectra - with a fixed σ or to a target S/N, as constant or photon noise; for resampled spectra, with the same correlation structure as the existing noise. This makes it possible to test how RV, EW or the periods found depend on the S/N, and to estimate Monte Carlo errors.

7 Measuring lines and spectra

The measurement scripts generally expect continuum-normalised 1D FITS spectra of a time series and write one row per spectrum to a results table (CSV or Excel) with JD. These tables are the input of the time-series analysis in Chapter 11.

7.1 Equivalent width, line depth, position and width

Script	Capabilities
TimeSeries_EW_LineDepth_LineWavelength_FWHM_MultipleModels.py	Central measurement script: EW, line depth, wavelength of the minimum and FWHM of a line over the series. Integration limits graphically or as numbers, local renormalisation, line model spline, Gaussian, Voigt, split Voigt or automatic choice by BIC; 1σ errors of all quantities.
EW_absorption_emission.py	EW with separate absorption and emission components (EW_abs, EW_emi, EW_net); in addition a noise-cleaned variant that counts only significant regions. Graphical range selection.
TimeSeries_EWs_MultipleRegionsInOneLine.py	EW of several contiguous sub-ranges of a line (e.g. blue wing, core, red wing); limits the same for all or per spectrum, can be saved and loaded.
LineArea_VelocityRange.py	Line area in a freely selectable velocity interval relative to the laboratory wavelength, in Å and km/s, with a sign that is constant over the series.
Tomography_Single_Lines.py	Travel time of RV changes through the atmosphere from individually chosen lines and bisector levels; lag by ICCF (FR/RSS) and phase method; for Deneb, relative to the lowest line.
Tomography_Formation_Depth.py	Lines sorted by formation-height proxy into K masks, joint multi-profile LSD, RV time series per layer and travel time between layers; external series (H α , H β) can be included.

7.2 Profile fitting

Script	Capabilities
FitLine.py	Fit of a line or blend over the series with linear continuum and Gaussian, double Gaussian, Lorentzian, split Lorentzian, Voigt or double Voigt (lmfit); starting values from the data. Result: centre, FWHM, height, area, EW (model and model-free), optionally RV – all with errors.
FitLine_auto_parametrised.py	Like FitLine.py, with tracking of range and starting values from spectrum to spectrum and automatic restart after a poor fit – for Doppler-shifted or strongly variable profiles.
FitLine_PositiveAndNegativeGauss_Version2.py	Emission line with central absorption (Be stars, β Lyr, H α in binaries): emission plus absorption Gaussian with constraints against role swapping; result includes EW of both components, RV of both components, V/R ratio.
FitLine_PositiveAndNegativeVoigt_Version2.py	The same with Voigt components for extended wings; each component can also be calculated as a Gaussian.

Script	Capabilities
HalpHa_Double_Gaussian_Fit.py	H α in Deneb: emission and absorption components, multistart fit, warning for ambiguous solutions; optionally FITS model spectrum and display over the phase.
HalpHa_Profile_Fit_Voigt_Gauss.py	Multi-component fit of the H α profile in velocity space (absorption as Gaussian, Voigt or split Voigt, emission near the laboratory wavelength) with optional masking of telluric lines; errors from the full covariance matrix.
FWHM_of_one_line_series_fits.py	FWHM and resolving power R from lamp or telluric lines: one line over the series (focus stability) or all lines of a spectrum (R(λ) of the spectrograph).

7.3 Measuring radial velocity

Script	Capabilities
RV_Measurement_OneLineInFitsSeries_Spline.py	RV of a line from the position of the minimum (or maximum): exact or smoothing spline, parabola or Gaussian; error by Monte Carlo; optionally barycentrically corrected with HJD and BJD_TDB.
cross_correlation_fits_series_v2.py	RV of the whole series by cross-correlation with a template on a $\ln \lambda$ grid; error after Zucker (2003); exclusion of telluric and interstellar regions; barycentric correction without internet.
cross_correlation_fits_series_single_lines_v2.py	The same, restricted to windows around individual lines – RV of several lines from different atmospheric layers in one run.
Bisector_RV_TimeSeries.py	Bisector RV at a fixed relative line depth with a co-moving velocity window; quality indicators and flag bit field per measurement. For consistent variability analysis, not for absolute RV.

7.4 Bisectors and line asymmetry

Script	Capabilities
Bisector_Analysis.py	Bisector of a line for each spectrum at selectable depth levels (computed in parallel); mean bisector with $\pm 1\sigma$, time series per depth, individual bisectors overlaid and coloured by JD.
Bisector_CCF_Depths.py	Cross-correlation of the bisector RV series of all depth levels: lag matrix between inner and outer line formation.
Spectral_line_asymmetry_interactive.py	Asymmetry by mirroring the red half of the line at the minimum; difference areas (total, positive, negative) and extreme values per spectrum.
SpectraDifferenceToReferenceSpectrum_Line.py	Difference of each spectrum to the median or an external reference spectrum in the range of a line; difference areas in velocity space.
DifferenceAreaPlot.py	Plot of the difference areas from the previous script against JD.
FluxAtOffset_Line.py	Flux at a fixed velocity distance to the left and right of the line minimum as a time series.

Script	Capabilities
HalpVariability.py	Velocity-resolved variability map $\sigma(v)$ and $\sigma(v, t)$ as well as difference integrals in selectable velocity zones as time series.

7.5 Rotation and macroturbulence

MacroTurbulence_Fourier_Method.py determines $v \sin i$ and the macroturbulence ζ from a line with the Fourier method after Gray (2005): first zero of the transform for $v \sin i$, significance test of the side lobe, ζ from the main lobe, independent profile fit in velocity space as a cross-check, and Monte Carlo errors.

MacroTurbulence_Fourier_Method_batch.py applies the same calculation to many spectra and lines from line_lists.py and produces a combined CSV as well as time series $v \sin i(\text{JD})$ and $\zeta(\text{JD})$.

7.6 Temperature and electron pressure

Script	Capabilities
ExcitationTemperature.py	Excitation temperature from the EW ratio of two lines of the same ion (Boltzmann), as a time series and phase-folded.
Boltzmann_Temperature_Fitter.py	$T_{\text{exc}}(\text{JD})$ from a joint Boltzmann fit over lines of freely selectable ions (e.g. Ti II, Fe II, Cr II); atomic data from line_lists.py.
Boltzmann_Saha_Analysis.py	Simultaneous determination of T_{exc} (Boltzmann, Fe I) and electron pressure P_e (Saha, Fe I/Fe II) from the same spectra.
LDR_Calibration_Hot_Stars.py	Calibration of line-depth ratios against T_{eff} on model spectra (own MOOG spectra or BOSZ grid from MAST) for A supergiants.
LDR_Batch_Analysis_from_fits.py	Application of this calibration to a FITS series: T_{eff} time series with errors and individual values per line pair.

8 Displaying spectra

8.1 Viewing individual spectra

Script	Capabilities
1d_fitSpectrum_view.py	Full spectrum with coupled zoom window, header output, paging through the folder with the arrow keys, PNG saving.
1d_fitSpectrum_view_withLineIdentification_WithSubplot.py	Viewer with line markers from line_lists.py (selection by name, element, range), optionally shifted by an RV; right-click identifies the nearest line with its velocity distance.
FITS_2D_Viewer.py	2D images (Chapter 4).
1d_dat_read_plot_save_csv.py, 1d_txt_plot.py	Reads ASCII spectra of any two-column format (separator and encoding detected automatically) and saves them as PNG.
PlotAndCrop_1d_ascii.py	Plots an ASCII spectrum and saves a section as .dat.

8.2 Multiple spectra and time series

Script	Capabilities
Spectra_Comparison.py	Qt interface for overlaying calibrated spectra (FITS image or table, also ESO format): show/hide, offset, normalisation in the section, difference to the mean, colours by JD, line markers, export.
timeseries_Spectra.py	Time series with offset or one graphic per spectrum; JD and wavelength range, labelling with JD or file name, colour gradient by JD.
spectra_stack_plot.py	Several ASCII spectra stacked, in colour or with offset (waterfall), with S/N filter.
comparison_plot_2spectra_ascii.py	Overplot of two ASCII spectra as PDF.
fits_to_png.py	Batch export of all FITS files of a folder as PNG, in parallel, only new files.
AnimationFromPlots.py	Plays an image series as an animation and exports it as MP4 or GIF.

8.3 Line profiles in velocity space

Script	Capabilities
Dynamic_Spectrum.py	Dynamic spectrum of a line: velocity (or wavelength) against JD, colour = flux; residuals to the mean profile, temporal scatter $\sigma(v)$, nightly means, correct display of observing gaps, optionally 3D; matrix as FITS.
fits_series_line_velocity_space_jd_range_v2.py	One line: all profiles overlaid, with offset by JD and as a dynamic spectrum, in a selectable JD range.

Script	Capabilities
fits_series_lines_velocity_space _jd_range_v2.py	Several lines side by side with a common JD axis or in one plot - comparison of different formation layers.

8.4 Plotting measurement tables

PlotsFromTableColumns.py plots columns from CSV, FITS tables (TFTS), Excel or ASCII against one another, also across table boundaries with assignment via a key column such as JD, and calculates correlation and regression. For time series of measured quantities see also Chapter 11.2.

9 Model spectra and convolution

Script	Capabilities
Spectrum_GaussBroadened.py	Preparation of synthetic or tabulated spectra (Kurucz via PyAstronomy, POLLUX, any ASCII tables): vacuum→air, cropping, Gaussian convolution to constant R or constant FWHM, sampling onto an equidistant grid; output as DAT, CSV, FITS.
convol_fits.py / convol_dat.py	Gaussian convolution of FITS or table spectra to a resolving power R (via $\ln \lambda$), a FWHM or σ ; if R0 is given, only the difference is convolved.
Pollux_Ausschnitt_convolve.py	Cropping and convolution of a POLLUX model spectrum.
rotation_broadening_fits.py	Rotational broadening with $v \sin i$ and limb darkening (kernel after Gray), optionally followed by instrumental convolution – model directly comparable with the measurement.
CreateArtificialSpectrum.py	Test spectrum with six lines of known position, broadened to R and with added noise – for testing measurement and calibration procedures.
LDR_Calibration_Hot_Stars.py	Loads and degrades BOSZ or MOOG model spectra for the LDR calibration (Chapter 7).

10 Binary stars and light curves

10.1 Period, phase and orbit

Script	Capabilities
period_analysis.py	Period search with Generalized Lomb-Scargle (with harmonics, FAP after Baluev) and string-length method after Dworetsky; reads text, Excel, ODS, FITS, Parquet, JSON.
phase_plot_known_period_20260924.py	Phase plot with known period and epoch from any table format: Fourier or median model, residuals, σ -clipping on the residuals, estimate of the required period accuracy.
phase_calculation.py	Phase and epoch of a measured-value table for a known ephemeris, with phase uncertainty from σ_P and σ_{T0} .
phase_calculation_from_spectrum_a_given_t0_and_period.py	Phase and epoch per FITS spectrum from the header time stamp; graphic of the phase coverage and dates on which the largest phase gap can be observed.
orbital_elements_from_jd_rv_fit.py	Orbital elements of an SB1 (K , e , ω , $T0$, γ) from RV measurements for a known period; starting values by grid search, errors, $a_1 \sin i$, mass function.
rv_calculation_from_orbital_elements_and_jd.py	Theoretical RV of both components from orbital elements for observation times (with O-C) or a time span (observation planning).

10.2 Component spectra

Script	Capabilities
sum_and_difference_spectrum_...py	Theoretical sum spectrum of a double or triple system per phase (components shifted and weighted with Phoebe fluxes) and difference to the measured spectrum - shows emission, mass streams or discs.
spectrum_conversion_barycentric_to_stellar_rest_frame.py	Shift spectra into the rest frame of a component (RV from a table or orbital elements) and average them in the rest frame.

10.3 AAVSO light curves

`aavso_lightcurve_analysis.py` processes AAVSO downloads and other brightness tables: automatic column detection, upper limits ("fainter than") displayed separately, discrepant observations discarded, means per band and time interval (error-weighted, with σ -clipping), optionally a Lomb-Scargle periodogram and a phase diagram.

11 Time-series analysis (TSA)

The folder TSA contains 86 scripts for analysing time series of measured values – RV, EW, FWHM, line depth, bisector velocities, temperatures – which are predominantly irregularly sampled and have seasonal gaps. The numbers of the subfolders indicate the usual order: preparation (01-03), periods and time-frequency (10-12), line comparison and causality (20-21), nonlinear dynamics (30), statistics (40), systematic method chain (50). The spectral analyses in 60_Spectral_Analysis are described in Chapter 7.

The input consists of tables with a time column (JD) and one or more measured-value columns, mostly as CSV (separator and decimal mark are usually detected automatically) or as a FITS table (TFTS). Many scripts of the line comparison expect files of the format `line_analysis_<line>.csv` or `line_analysis_<line>_filtered.csv`.

11.1 Preparing data

Script	Capabilities
OutlierAnalysis.py	Four-stage outlier detection: S/N filter, LOESS residuals with robust z-score per measurement channel, cross-validation over several channels (e.g. RV, EW, FWHM of the same line), and finally a visual decision per candidate. Output: flagged and cleaned CSV per channel.
MedianLoessFilter.py	Time-based median filter followed by LOESS smoothing for any number of channels; produces the *_filtered.csv files with original, median and LOESS columns.
time_range_binning_and_interpolation.py	Averaging of a variable over time ranges of selectable length (e.g. nightly means) with scatter and count.
parse_stats_to_csv.py	Combines *_stats.txt files into one CSV.

11.2 Displaying measurement series

Script	Capabilities
Measurement_Plot.py	Time series of a measured quantity from one or more tables: combined plot or one per file, error bars, robust normalisation, marker lines, any number of zoom windows.
Measurement_Comparison.py	Many measurement series over a common JD axis in three layouts (overview, combined, panels), with nightly means, σ -clipping and smoothing (Savitzky-Golay, Gaussian, median, LOWESS).
PlotsFromColumns_batch.py	Columns from CSV, FITS, Excel, ASCII plotted against one another; interactive, in batch per file or pairwise over all file combinations; outlier filter, statistics CSV.
LOESS_Comparison_Plots_batch.py	One comparison plot per line of the LOESS-filtered quantities APX, EW, FWHM and RV.

11.3 Generating test data

`synthSignal_GUI.py` generates synthetic time series from freely combinable components – 23 signal shapes, from sine, period change, amplitude and phase modulation, quasi-

periodic and damped oscillations, through transients and flares, to white, red and $1/f^\beta$ noise - with realistic sampling (jitter, observing windows, gaps). This makes it possible to test beforehand whether a method recovers a known signal under the conditions of the real observations.

11.4 Period analysis

Script	Capabilities
GLS.py	Generalized Lomb-Scargle periodogram (Zechmeister & Kürster 2009) for a column of a CSV.
PDM_Period_Analysis.py	Phase Dispersion Minimization for a single file or batch; automatic minima, bin stability analysis, phase plots, top 10 periods per file.
CLEAN_Period_Analysis.py	CLEAN deconvolution (Roberts et al. 1987): separates true periods from aliases of the window function; dirty, clean and window spectrum, phase plots.
ACF.py	Autocorrelation function with confidence band.
tsa_prewhitener_auto_white.py	Detrending, multi-term Lomb-Scargle, iterative prewhitening down to white residuals (Ljung-Box, ACF), bootstrap FAP, MCMC period estimation, window function, optionally GP; export as CSV, JSON, FITS.
SineFit_Period.py	Simultaneous harmonic regression for given periods: amplitudes, phases, residuals.
Phase_Folding_Single_Line.py	Folds several measured quantities of a line (RV, EW, FWHM, depth, asymmetry) onto a common period, with median binning and phase offsets of the extrema - e.g. to demonstrate shock signatures.
Epoch_Screening.py	Sliding GLS or PDM screening in time windows: finds periods that occur only episodically.
amplitude_tracking.py	Tracks the amplitude of a period found over time (moving window, segments, WWZ).

11.5 Time-frequency analysis

Script	Capabilities
wwz.py (with wwz_core.py)	Weighted Wavelet Z-transform (Foster 1996) with measurement errors: single file or batch, significance by χ^2 and against red noise (analytical and Monte Carlo), cone of influence, ridge tracking of the dominant periods, interactive synthesis. The CSV output is the input of all WWZ comparison scripts.
SlidingGLS_Spectrogram.py	Sliding GLS spectrogram as a gap-robust alternative to the WWZ; optional Hann taper and period tracking.
eemd_analysis_final.py	Decomposition into intrinsic modes with EEMD or CEEMDAN, Hilbert assessment, optionally recurrence plots per mode.
Synchrosqueezing_Analysis.py	Synchrosqueezed wavelet transform after GP interpolation; sharper period trajectories than the classical scalogram.

Script	Capabilities
Sliding_SHO_celerite2.py	Interpolation-free cross-check: windowed fit of stochastically driven oscillators (celerite2) with period, quality factor Q and amplitude over time.

11.6 Noise and stochastic models

Script	Capabilities
GPR_Time_Series_Analysis.py	Gaussian process regression with automatic kernel selection (RBF, Matérn, quasi-periodic, with white noise); reconstructed signal with uncertainty band and power spectrum of the kernel.
Structure_Function.py	Structure function of irregularly sampled series with power-law fit – characteristic time scales and noise character.
PSD_plot.py	Power spectral densities of several measured quantities of one line or of one measured quantity of several lines.

11.7 Comparing lines with one another

This area answers whether different lines – and thus different layers of the stellar atmosphere – change together, with a time delay or independently.

Correlation, regression, common influences

Script	Capabilities
Correlation_Analysis.py	Pearson and Spearman matrix between measured quantities from several CSV files after JD matching.
Regression_Analysis.py	Regression line of two measurement series after JD matching with tolerance.
cross_correlation_measurements.py	Pairwise cross-correlation of any measurement series (interpolated CCF and DCF), peak and centroid lag, heatmaps (folder CC).
InfluenceFactor_Analysis.py	Share of common and individual influences: PCA/factor analysis, CCF, spectral coherence and cross-WWZ.
Line_Parameter_Comparison.py	One parameter over many lines: z-standardised heatmap lines × time, Spearman matrix, dendrogram, moving correlation to a reference line.
Line_Comparison_Clustering.py	Clustering of smoothed RV time series of many lines into groups with a common course.
Measures_Line_Comparison.py	Course of any measured quantities of several lines in comparison.
GLS_Line_Comparison.py	GLS periodograms of many lines, their correlation and hierarchical clustering.
TimeDifferences_Extrema_Two_Lines.py	Time differences between local minima and maxima of two series.

Time delay (lag) and depth of formation

Script	Capabilities
Lag_Analysis_Line_Pair.py	Lag between two RV series: overall CCF with subsampling uncertainty and period-dependent lag from the cross-WWZ (dispersive wave propagation).
Lag_All_Line_Pairs_CCF.py	CCF lag for all pairs of a line selection in one table.
Lag_Ranking.py	Order of the lines by depth of formation from the lag matrix (weighted score, win rate, least-squares consistency).
Lag_Reference_Line_Extraction.py	Lags of all lines relative to a reference line as a sorted bar chart with errors.
EW_APX_vs_Lag.py / Eup_Elow_vs_Lag.py	Line strength or excitation potential against the lag - physical test of the ranking.
DCF_CrossCorrelation_Verification.py	Verification of a lag with the DCF (Edelson & Krolik 1988) directly on the original data, with bootstrap error.
CrossCorrelation_Bandpass_Verification.py	Verification by cross-correlation of bandpass-filtered series per period band.

Cross spectra and WWZ comparison

Script	Capabilities
LSSA_CrossSpectrum.py	Segmented LSSA coherence without interpolation with permutation significance.
WaveletCrossBispectrum.py	Wavelet cross-bispectrum and bicoherence: nonlinear coupling between frequencies.
WWZ_Comparison_Multiple_Lines.py	WWZ spectrograms of several lines side by side, as a difference, as a contour overlay and as a cross-WWZ with phase difference and time offset.
WWZ_Line_Comparison.py	Overlaid mean periodograms, spectrogram gallery, difference and ratio maps, dominant periods over time.
WWZ_Similarity_Analysis.py	Similarity of WWZ spectrograms in four dimensions: periods, temporal modulation, phase coherence, overall pattern.
WWZ_Correlation_Neighbours.py	All lines whose spectrogram resembles an anchor line (SSIM or Pearson), plus a PCA/UMAP map of all lines.
WWZ_Line_Grouping.py	Hierarchical grouping of all spectrograms with automatic choice of the number of groups and a representative per group.

11.8 Causality

Script	Capabilities
GrangerCausality.py	Multivariate Granger causality (VAR model, lag order by AIC/BIC) between line parameters.

Script	Capabilities
TransferEntropy.py	Symbolic transfer entropy between RV series with surrogate test; net information flow and causality matrix (e.g. Fe II $\rightarrow$ H β $\rightarrow$ H α).
ConditionalTransferEntropy.py	Conditional transfer entropy (k-NN estimator): separates direct from indirect dependencies.

11.9 Nonlinear dynamics

Script	Capabilities
PhaseSpaceReconstruction.py	Time-delay embedding after Takens: τ from the mutual information, embedding dimension from false nearest neighbours, largest Lyapunov exponent (Rosenstein), correlation dimension D_2 , recurrence plot, 2D/3D display; three variants for irregular sampling.
attractor_autoencoder_interactive.py	Attractor reconstruction with autoencoders; latent coordinates are then available in the Variable Explorer.
lyapunov_analysis.py / NonlinearDynamics.py	Lyapunov exponents or Lyapunov spectrum, Kaplan-Yorke dimension and permutation entropy (global and sliding).
multifractal_analysis.py	Multifractal dimension spectra.
DMD_Analysis.py	Dynamic Mode Decomposition: coherent modes of several measured quantities with frequencies, growth rates and time courses.
RC_and_DistanceMatrix_value space.py (with rqa_metrics.py)	Distance matrix and recurrence plot in value space, RQA measures (RR, DET, LAM, ENTR, TT and others) static and sliding; single file or batch.
PhaseSpace_DistanceMatrix_Analysis.py	Distance matrix and RQA in the reconstructed phase space, periods from the diagonals with significance and alias checks, lifetime of periods; report as RTF.
Value_DistanceMatrix_analysis.py	Comprehensive evaluation of distance matrices: statistics, temporal segmentation, regime changes, state clustering.
TSA_Analysis_Multiple_Methods.py	Distance matrix, autocorrelation, Lomb-Scargle, wavelet, recurrence plot and FFT in one run – quick first overview.
Multi_File_Cross_RQA.py	Cross-recurrence analysis of the same quantity in several lines (e.g. H α , H β , H γ).

11.10 Statistics

Script	Capabilities
histogram_analysis.py	Histograms and statistics of the measured values or of the observation intervals Δ JD (sampling).
Statistics_OneVariable_AllLines.py	Statistics of one variable over all files of a directory in one table.

Script	Capabilities
Extrema_Statistics.py	Determine local extrema, clean up interactively, histograms and ECDF.
Changepoint_Analysis_SiII.py	Changepoint detection (PELT) in the RV series of Si II 6347/6371 with consistency comparison; transferable to other line pairs.

11.11 Systematic analysis chain

The five scripts in 50_Systematic_Analysis apply different method families to the same data one after another, without regularisation, and compute in parallel. They are suitable for securing a result with independent procedures.

Script	Capabilities
01_Periodicity_Analysis.py	GLS, WWZ, PDM, CLEAN, string length, AoV.
02_CrossCorrelation_Lag.py	DCF, ZDCF, cross-WWZ, optionally JAVELIN; lag error by Monte Carlo.
03_Nonlinear_Recurrence.py	RQA, cross and joint recurrence plots, convergent cross mapping, mutual information.
04_Statistical_Dependence.py	Distance correlation, transfer entropy, copula analysis.
05_ModelBased_GP_CARMA.py	Gaussian process and CARMA spectrum (celerite2), sparse Bayesian learning in frequency space.

12 Typical workflows

The following workflows show how the scripts work together. They are suggestions; individual steps may be omitted depending on the data.

12.1 Reducing an observing night with the echelle spectrograph

1. In advance: `astroplanner_20260924.py` for object selection and time window.
2. During the night, optionally `Live_Reduction.py` - each new exposure is reduced immediately and can be assessed.
3. After the night: `CreateMasters_CCD_2d_fits.py` for master flat, master ThAr and darks; visual inspection with `FITS_2D_Viewer.py`.
4. `Echelle_Extraction.py` → `ThAr_Calibration.py` → `Normalisation_Orders.py` → `Merge_Orders.py`.
5. If the calibration shows anomalies: `ThAr_LineCounter.py` and `Diagnosis_LineYield.py`.
6. `jd_header_check.py` and `TimeSeries_Overview.py`, then classification into the series.

12.2 Integrating a new echelle spectrograph

1. Create the master flat and master ThAr of the instrument, extract the orders with `Echelle_Extraction.py` (this creates the order reference).
2. Use `Echelle_InstrumentSpecific_LineCatalogue_Creation.py` to create the identification table from two to three anchor lines.
3. Check the result with `ThAr_LineCatalogue_Viewer.py` and `ThAr_LineCounter.py` and file it as a verified version.
4. From then on, normal reduction with `ThAr_Calibration.py`.

12.3 From the spectrum series to the line time series

1. Check the series: `fits_header_check_correct.py`, `jd_header_check.py`, `spectra_quality_control.py`, `remove_cosmics.py`.
2. Correct the wavelength zero point on telluric lines (`telluricLines_RVCorrection_*`), then correct barycentrically (`barycentric_correction_spectra.py`) and remove telluric lines (`telluric_drying_hitran.py` or `telluric_bc_correction.py`).
3. Normalise the surroundings of the line: `Normalisation_Single_Lines_20260923.py`; check the support points beforehand with `support_point_validator_single_lines_20260923.py`.
4. Measure: `TimeSeries_EW_LineDepth_LineWavelength_FWHM_MultipleModels.py`, `Bisector_RV_TimeSeries.py` or `cross_correlation_fits_series_single_lines_v2.py`.
5. Visual check: `Dynamic_Spectrum.py` and `fits_series_line_velocity_space_jd_range_v2.py`.

12.4 Investigating periods and layer structure of a pulsating atmosphere

1. Clean and smooth the measurement series: `OutlierAnalysis.py`, `MedianLoessFilter.py`; assess the sampling with `histogram_analysis.py` (Δ JD mode).
2. Test the methods beforehand on `synthSignal_GUI.py` data with the same sampling.

3. Periods: GLS.py and CLEAN_Period_Analysis.py, confirmation with PDM_Period_Analysis.py; temporal variability with wwz.py, SlidingGLS_Spectrogram.py and Epoch_Screening.py.
4. Compare lines: WWZ_Comparison_Multiple_Lines.py and WWZ_Line_Grouping.py; lags with Lag_All_Line_Pairs_CCF.py → Lag_Ranking.py → EW_APX_vs_Lag.py / Eup_Elow_vs_Lag.py; verification with DCF_CrossCorrelation_Verification.py.
5. Directed dependencies: TransferEntropy.py, ConditionalTransferEntropy.py, GrangerCausality.py.
6. Physical quantities: Boltzmann_Temperature_Fitter.py, Boltzmann_Saha_Analysis.py, LDR_Batch_Analysis_from_fits.py, MacroTurbulence_Fourier_Method_batch.py.
7. Safeguarding: scripts 01-05 of the systematic analysis chain.

12.5 Determining the orbit of a spectroscopic binary

1. Measure RV: cross_correlation_fits_series_v2.py (template) or RV_Measurement_OneLineInFitsSeries_Spline.py, in each case barycentrically corrected.
2. Search for the period: period_analysis.py; check the result with phase_plot_known_period_20260924.py.
3. Orbital elements: orbital_elements_from_jd_rv_fit.py; checking and planning further observations with rv_calculation_from_orbital_elements_and_jd.py and phase_calculation_from_spectra_given_t0_and_period.py (phase gaps).
4. Optionally separate components: sum_and_difference_spectrum...py and spectrum_conversion_barycentric_to_stellar_rest_frame.py.

12.6 Following emission lines of Be stars and reporting them to BeSS

1. Fit the H α profile: FitLine_PositiveAndNegativeGauss_Version2.py or the Voigt version (EW, V/R, peak separation).
2. Separate EW of absorption and emission: EW_absorption_emission.py.
3. Display: Dynamic_Spectrum.py, timeseries_Spectra.py.
4. For submission: BESS_fits_header_series.py completes the headers.

12.7 Checking the robustness of a result

- Bring spectra to a lower S/N with add_noise.py and repeat the measurement.
- Reproduce time series with synthSignal_GUI.py and apply the analysis to known signals.
- Confirm a lag or a period with a second, independent method (DCF, bandpass CCF, sliding SHO, systematic chain).

13 Conventions, formats and dependencies

13.1 Data formats

- 1D spectra: FITS with a linear wavelength scale (CRVAL1, CDELTA1 or CD1_1, CRPIX1); many scripts also read log-linear scales (DC-FLAG = 1) and table FITS.
- Time stamp: keyword JD preferred; otherwise JD-OBS, JD-MID, BJD, HJD, MJD-OBS or DATE-OBS are evaluated. `jd_header_check.py` standardises this.
- ASCII spectra: two columns WAVE and FLUX (.dat tab-separated, .csv comma-separated).
- Results tables: CSV with a semicolon as separator and a comma as decimal mark; some older line-comparison scripts use a comma and a decimal point. Inputs are mostly detected automatically.
- Graphics: PNG at 600 dpi with a white background, in some cases additionally PDF.

13.2 Header entries of the correction scripts

Keyword	Meaning
RVMEAS / RVERR	Measured RV and error [km/s], positive = redshift.
RVSHIFT / RVSHTOT	Applied shift and sum of all previous shifts [km/s], respectively.
RVMETH	Method: MANUAL, TABLE, CCF, TELL_RBF, TELL_POLY.
VSINI, LIMBDARK, CONV_R	Parameters of the rotational and Gaussian convolution.
ROTANG, ROTMETH	Rotation angle and method when aligning the spectrum strip.
ECHORDER, ORDER_M, CAL_MODE	Order number and calibration mode in the echelle pipeline.
HISTORY	Log of every processing step.

13.3 Required packages

Basic equipment (used by almost all scripts): numpy, scipy, matplotlib, astropy, pandas, PyQt5.

Additionally, depending on the script:

Area	Packages
Fit and statistics	lmfit, statsmodels, scikit-learn, emcee, corner, ruptures, pyvinecopulib
Astronomy	PyAstronomy, astroquery, astroplan, barycorrpy, hapi (HITRAN), keyring (ESO login)
Time series	celerite2, celerite, PyEMD (EMD-signal), ssqueezepy, PyWavelets, javelin (optional)
Miscellaneous	joblib, tqdm, networkx, umap-learn, scikit-image, tensorflow (autoencoder), imageio/imageio-ffmpeg, Pillow, requests; cupy optional for GPU

13.4 Shared modules

Module	Contents
Building_Blocks/line_lists.py	Line database with atomic data; line selection dialogs.
Building_Blocks/rv_tools.py	Reading, Doppler shift, header, log, measurement of telluric lines for the RV scripts.
Building_Blocks/observatories.py	Coordinates of the observing sites; enter new locations here.
Reduce/Classical_Spectrograph/spectro_core.py, spectro_workflow.py	Calculation functions and workflow of the slit-spectrograph reduction.
TSA/11_Time_Frequency/WWZ/wwz_core.py	Calculation core of the WWZ, also importable by other scripts.
TSA/30_.../Value_Space/rqa_metrics.py	RQA measures.

Appendix A - Script index

All 192 Python files of the collection, in the alphabetical order of their original German names, with folder and the chapter in which they are described.

Script	Folder	Chapter
01_Periodicity_Analysis.py	TSA/50_Systematic_Analysis	11
02_CrossCorrelation_Lag.py	TSA/50_Systematic_Analysis	11
03_Nonlinear_Recurrence.py	TSA/50_Systematic_Analysis	11
04_Statistical_Dependence.py	TSA/50_Systematic_Analysis	11
05_ModelBased_GP_CARMA.py	TSA/50_Systematic_Analysis	11
1d_dat_read_plot_save_csv.py	ascii	8
1d_fitSpectrum_view_withLineIdentification_WithSubplot.py	Spectra_Plots	8
1d_fitSpectrum_view.py	Spectra_Plots	8
1d_txt_plot.py	ascii	8
2_AlignSpectrumStrip_CCD_Images_fits.py	Reduce/Classical_Spectrograph	4
3_ExtractionAndWavelengthCalibrationNormalisation_withSlit_CCD_Images_withCalibrationLineOptimisation.py	Reduce/Classical_Spectrograph	4
3a_ExtractionAndWavelengthCalibrationNormalisation_slitless_CCD_Images.py	Reduce/Classical_Spectrograph	4
aavso_lightcurve_analysis.py	Light_Curves	10
ACF.py	TSA/10_Period_Analysis/ACF	11
amplitude_tracking.py	TSA/10_Period_Analysis/PDM	11
AnimationFromPlots.py	Spectra_Plots	8
ExcitationTemperature.py	TSA/60_Spectral_Analysis/Temperature	7
astroplanner_20260924.py	astroplanner	3
attractor_autoencoder_interactive.py	TSA/30_Nonlinear_Dynamics/Phase_Space/attractor_encoder	11
OutlierAnalysis.py	TSA/01_Data_Preparation	11
barycentric_correction_spectra.py	BC	6
BESS_fits_header_series.py	Series/BESS	5
Bisector_Analysis.py	TSA/60_Spectral_Analysis/Bisector	7
Bisector_CCF_Depths.py	TSA/60_Spectral_Analysis/Bisector	7
Bisector_RV_TimeSeries.py	Line_Parameter_Determination	7
Boltzmann_Saha_Analysis.py	TSA/60_Spectral_Analysis/Temperature	7
Boltzmann_Temperature_Fitter.py	TSA/60_Spectral_Analysis/Temperature	7
Changepoint_Analysis_SiII.py	TSA/40_Statistics	11
CLEAN_Period_Analysis.py	TSA/10_Period_Analysis/CLEAN	11
comparison_plot_2spectra_ascii.py	ascii	8
ConditionalTransferEntropy.py	TSA/21_Causality	11
convol_dat.py	Convolution	9
convol_fits.py	Convolution	9

Script	Folder	Chapter
remove_cosmics.py	Series	5
Crop_Section_TimeSeries.py	Series	6
cross_correlation_fits_series_single_lines_v2.py	CC	7
cross_correlation_fits_series_v2.py	CC	7
DCF_CrossCorrelation_Verification.py	TSA/20_Line_Comparison/Cross_Correlation	11
Diagnosis_LineYield.py	Reduce/Echelle	4
DifferenceAreaPlot.py	TSA/60_Spectral_Analysis/Line_Asymmetry	7
DifferenceSpectra_withPlotAll_v2.py	Rebinning_Mean_Spectrum_Difference_Spectra	6
DifferenceSpectra_v2.py	Rebinning_Mean_Spectrum_Difference_Spectra	6
DMD_Analysis.py	TSA/30_Nonlinear_Dynamics/DMD	11
Dynamic_Spectrum.py	Spectra_Plots	8
Echelle_Extraction.py	Reduce/Echelle	4
Echelle_InstrumentSpecific_LineCatalogue_Creation.py	Reduce/Echelle	4
eemd_analysis_final.py	TSA/11_Time_Frequency/EMD	11
InfluenceFactor_Analysis.py	TSA/20_Line_Comparison/Influence_Factors	11
Epoch_Screening.py	TSA/10_Period_Analysis	11
eso_query.py	Catalogue_Queries	3
Eup_Elow_vs_Lag.py	TSA/20_Line_Comparison/Lag	11
EW_absorption_emission.py	EW	7
EW_APX_vs_Lag.py	TSA/20_Line_Comparison/Lag	11
Extrema_Statistics.py	TSA/40_Statistics	11
fits_series_to_csv_and_dat.py	Series	5
FitLine_auto_parametrised.py	Line_Fitting	7
FitLine_PositiveAndNegativeGauss_Version2.py	Line_Fitting	7
FitLine_PositiveAndNegativeVoigt_Version2.py	Line_Fitting	7
FitLine.py	Line_Fitting	7
FITS_2D_Viewer.py	Reduce	4
fits_header_check_correct.py	Header	5
fits_to_png.py	Spectra_Plots	8
fits_series_line_velocity_space_jd_range_v2.py	Velocity_Space_Plots	8
fits_series_lines_velocity_space_jd_range_v2.py	Velocity_Space_Plots	8
FluxAtOffset_Line.py	TSA/60_Spectral_Analysis/Line_Asymmetry	7
format_conversions.py	Format_Conversion	3
FWHM_of_one_line_series_fits.py	Line_Fitting	7
SlidingGLS_Spectrogram.py	TSA/11_Time_Frequency/Sliding_GLS	11
GLS_Line_Comparison.py	TSA/20_Line_Comparison/GLS_Comparison	11
GLS.py	TSA/10_Period_Analysis/GLS	11
GPR_Time_Series_Analysis.py	TSA/12_Noise_Models/GPR	11
GrangerCausality.py	TSA/21_Causality	11
Halpna_Double_Gaussian_Fit.py	Line_Fitting	7

Script	Folder	Chapter
HalpHa_Profile_Fit_Voigt_Gauss.py	Line_Fitting	7
HalpHaVariability.py	TSA/60_Spectral_Analysis/HalpHa	7
header_print_series_csv.py	Header	5
heliocentricCorrection_NRES.py	NRES	4
histogram_analysis.py	TSA/40_Statistics	11
jd_header_check.py	Header	5
JD_In_Filename.py	Series	5
Correlation_Analysis.py	TSA/20_Line_Comparison/Correlation_Regression	11
CrossCorrelation_Bandpass_Verification.py	TSA/20_Line_Comparison/Cross_Correlation	11
cross_correlation_measurements.py	CC	7
CreateArtificialSpectrum.py	Theoretical_Spectra	9
Lag_All_Line_Pairs_CCF.py	TSA/20_Line_Comparison/Lag	11
Lag_Analysis_Line_Pair.py	TSA/20_Line_Comparison/Lag	11
Lag_Ranking.py	TSA/20_Line_Comparison/Lag	11
Lag_Reference_Line_Extraction.py	TSA/20_Line_Comparison/Lag	11
LDR_Batch_Analysis_from_fits.py	TSA/60_Spectral_Analysis/Temperature	7
LDR_Calibration_Hot_Stars.py	TSA/60_Spectral_Analysis/Temperature	7
LineArea_VelocityRange.py	Line_Parameter_Determination	7
line_lists.py	Building_Blocks	13
Line_Parameter_Comparison.py	TSA/20_Line_Comparison/Line_Parameters	11
Line_Comparison_Clustering.py	TSA/20_Line_Comparison/Line_Parameters	11
nist_line_wavelengths.py	Catalogue_Queries	3
Live_Reduction.py	Reduce/Echelle/live_Reduction	4
LOESS_Comparison_Plots_batch.py	TSA/02_Plotting	11
LSSA_CrossSpectrum.py	TSA/20_Line_Comparison/Cross_Spectrum	11
lyapunov_analysis.py	TSA/30_Nonlinear_Dynamics/Lyapunov	11
MacroTurbulence_Fourier_Method_batch.py	Micro_Macro_Turbulence	7
MacroTurbulence_Fourier_Method.py	Micro_Macro_Turbulence	7
CreateMasters_CCD_2d_fits.py	Reduce	4
MedianLoessFilter.py	TSA/01_Data_Preparation	11
Merge_Orders.py	Reduce/Echelle	4
Measures_Line_Comparison.py	TSA/20_Line_Comparison/Line_Parameters	11
Measurement_Comparison.py	TSA/02_Plotting	11
Measurement_Plot.py	TSA/02_Plotting	11
Average_Of_fits.py	Series	6
Multi_File_Cross_RQA.py	TSA/30_Nonlinear_Dynamics/Recurrence/Cross_RQA	11
multifractal_analysis.py	TSA/30_Nonlinear_Dynamics/Multifractal	11
newbinning_mean_spectrum_v2.py	Rebinning_Mean_Spectrum_Difference_Spectra	6
NonlinearDynamics.py	TSA/30_Nonlinear_Dynamics/Lyapunov	11
Normalisation_Single_Lines_20260923.py	Normalisation	6
Normalisation_Orders.py	Reduce/Echelle	4

Script	Folder	Chapter
NRES_extract.py	NRES	4
observatories.py	Building_Blocks	13
orbital_elements_from_jd_rv_fit.py	binaries_Period_Determination_Orbital_Elements	10
parse_stats_to_csv.py	TSA/01_Data_Preparation	11
PDM_Period_Analysis.py	TSA/10_Period_Analysis/PDM	11
period_analysis.py	binaries_Period_Determination_Orbital_Elements	10
phase_calculation_from_spectra_given_t0_and_p eriod.py	binaries_Period_Determination_Orbital_Elements	10
phase_calculation.py	binaries_Period_Determination_Orbital_Elements	10
Phase_Folding_Single_Line.py	TSA/10_Period_Analysis/Phase_Folding	11
phase_plot_known_period_20260924.py	binaries_Period_Determination_Orbital_Elements	10
PhaseSpace_DistanceMatrix_Analysis.py	TSA/30_Nonlinear_Dynamics/Recurrence/ Phase_Space	11
PhaseSpaceReconstruction.py	TSA/30_Nonlinear_Dynamics/Phase_Space	11
PlotAllOrders.py	NRES	4
PlotAndCrop_1d_ascii.py	ascii	6
PlotsFromColumns_batch.py	TSA/02_Plotting	11
PlotsFromTableColumns.py	Plots	8
Pollux_Ausschnitt_convolve.py	Convolution	9
PSD_plot.py	TSA/12_Noise_Models	11
RC_and_DistanceMatrix_valuespace.py	TSA/30_Nonlinear_Dynamics/Recurrence/ Value_Space	11
Reduce_GUI.py	Reduce/Classical_Spectrograph	4
Regression_Analysis.py	TSA/20_Line_Comparison/Correlation_Regression	11
rotation_broadening_fits.py	Convolution	9
rqa_metrics.py	TSA/30_Nonlinear_Dynamics/Recurrence/ Value_Space	11
rv_calculation_from_orbital_elements_and_jd.py	binaries_Period_Determination_Orbital_Elements	10
RV_Measurement_OneLineInFitsSeries_Spline.p y	RV	7
rv_tools.py	Building_Blocks	13
RVCorrection_1d_spectra_fits_series.py	RV	6
RVCorrectionByTemplate_1d_spectra_fits_series .py	RV	6
simbad_query_a.py	Catalogue_Queries	3
SineFit_Period.py	TSA/10_Period_Analysis/Sine_Fit	11
Sliding_SHO_celerite2.py	TSA/11_Time_Frequency/Sliding_SHO	11
SNR_Determination.py	SNR	5
Spectral_line_asymmetry_interactive.py	TSA/60_Spectral_Analysis/Line_Asymmetry	7
spectra_diagnosis.py	Series	5
spectra_quality_control.py	Series	5
spectra_stack_plot.py	ascii	8

Script	Folder	Chapter
Spectra_Comparison.py	Spectra_Plots	8
SpectraDifferenceToReferenceSpectrum_Line.py	TSA/60_Spectral_Analysis/Line_Asymmetry	7
spectro_workflow.py	Reduce/Classical_Spectrograph	4
spectro_core.py	Reduce/Classical_Spectrograph	4
Spectrum_GaussBroadened.py	Theoretical_Spectra	9
Statistics_OneVariable_AllLines.py	TSA/40_Statistics	11
Structure_Function.py	TSA/12_Noise_Models	11
support_point_validator_single_lines_20260923.py	Normalisation	6
sum_and_difference_spectrum_...py	binaries_Period_Determination_Orbital_Elements	10
Synchrosqueezing_Analysis.py	TSA/11_Time_Frequency/Synchrosqueezing	11
synthSignal_GUI.py	TSA/03_Test_Data	11
telluric_bc_correction.py	BC	6
telluric_drying_hitran.py	Drying	6
telluricLines_RVCorrection_RBF_1d_spectra_fits_series.py	RV	6
telluricLines_RVCorrection_Regression_1d_spectra_fits_series.py	RV	6
ThAr_Image_Show.py	Reduce/Echelle	4
ThAr_Calibration.py	Reduce/Echelle	4
ThAr_LineCatalogue_Viewer.py	Reduce/Echelle	4
ThAr_LineCounter.py	Reduce/Echelle	4
timeseries_Spectra.py	Spectra_Plots	8
TransferEntropy.py	TSA/21_Causality	11
TSA_Analysis_Multiple_Methods.py	TSA/30_Nonlinear_Dynamics/Recurrence	11
tsa_prewhitener_auto_white.py	TSA/10_Period_Analysis/Prewhitening	11
spectrum_conversion_barycentric_to_stellar_rest_frame.py	binaries_Period_Determination_Orbital_Elements	10
Universal_Normalisation.py	Normalisation	6
VacToAir.py	NRES	4
add_noise.py	SNR	6
WaveletCrossBispectrum.py	TSA/20_Line_Comparison/Cross_Spectrum	11
Value_DistanceMatrix_analysis.py	TSA/30_Nonlinear_Dynamics/Recurrence/Value_Space	11
WWZ_Similarity_Analysis.py	TSA/20_Line_Comparison/WWZ_Comparison	11
wwz_core.py	TSA/11_Time_Frequency/WWZ	11
WWZ_Correlation_Neighbours.py	TSA/20_Line_Comparison/WWZ_Comparison	11
WWZ_Line_Grouping.py	TSA/20_Line_Comparison/WWZ_Comparison	11
WWZ_Line_Comparison.py	TSA/20_Line_Comparison/WWZ_Comparison	11
WWZ_Comparison_Multiple_Lines.py	TSA/20_Line_Comparison/WWZ_Comparison	11
wwz.py	TSA/11_Time_Frequency/WWZ	11

Script	Folder	Chapter
TimeDifferences_Extrema_TwoLines.py	TSA/20_Line_Comparison/Extrema_Time_Differences	11
time_range_binning_and_interpolation.py	TSA/01_Data_Preparation	11
TimeSeries_EW_LineDepth_LineWavelength_FWHM_MultipleModels.py	Line_Parameter_Determination	7
TimeSeries_EWs_MultipleRegionsInOneLine.py	Line_Parameter_Determination	7
TimeSeries_Spectra_Smooth.py	Series	6
TimeSeries_Overview.py	Series	5
Tomography_Single_Lines.py	Line_Parameter_Determination	7
Tomography_Formation_Depth.py	Line_Parameter_Determination	7
Telluric_Zero_Point.py	Reduce	6