Spektropython_Deutsch

Handbuch der Möglichkeiten

Was die 192 Skripte der Sammlung leisten – geordnet nach Aufgaben, von der Beobachtungsplanung bis zur Zeitreihenanalyse

Stand: 13.10.2026

Autor: Lothar Schanne

Inhalt

1 Einführung....4
1.1 Arbeitsumgebung....4
2 Die Arbeitskette im Überblick....5
3 Beobachtung vorbereiten und Daten beschaffen....6
3.1 Beobachtungsnacht planen....6
3.2 Objekt- und Liniendaten abfragen....6
3.3 Liniendatenbank....6
4 Rohdaten reduzieren....7
4.1 Master-Aufnahmen und Sichtprüfung....7
4.2 Echelle-Pipeline (Echelle-Spektrographen)....7
4.3 Spaltspektrograph und spaltloser Spektrograph....8
4.4 NRES-Spektren (Las Cumbres Observatory)....8
5 Spektrenserien verwalten und prüfen....9
5.1 Header und Zeitangaben....9
5.2 Bestandsaufnahme und Qualitätskontrolle....9
5.3 Dateien umbenennen und umwandeln....10
6 Spektren aufbereiten und korrigieren....11
6.1 Kontinuumsnormierung....11
6.2 Baryzentrische und tellurische Korrektur....11
6.3 Radialgeschwindigkeit korrigieren....11
6.4 Ausschnitt, Mittelung, Rebinning, Glättung....13
6.5 Differenzspektren....13
6.6 Rauschen hinzufügen....13
7 Linien und Spektren messen....14
7.1 Äquivalentweite, Linientiefe, Lage und Breite....14
7.2 Profilanpassung....14
7.3 Radialgeschwindigkeit messen....15
7.4 Bisektoren und Linienasymmetrie....15
7.5 Rotation und Makroturbulenz....15
7.6 Temperatur und Elektronendruck....16
8 Spektren darstellen....17
8.1 Einzelne Spektren ansehen....17
8.2 Mehrere Spektren und Zeitserien....17
8.3 Linienprofile im Geschwindigkeitsraum....17
8.4 Messtabellen plotten....18

9 Modellspektren und Faltung....19
10 Doppelsterne und Lichtkurven....20
10.1 Periode, Phase und Orbit....20
10.2 Komponentenspektren....20
10.3 AAVSO-Lichtkurven....20
11 Zeitreihenanalyse (TSA)....21
11.1 Daten vorbereiten....21
11.2 Messreihen darstellen....21
11.3 Testdaten erzeugen....21
11.4 Periodenanalyse....22
11.5 Zeit-Frequenz-Analyse....22
11.6 Rauschen und stochastische Modelle....23
11.7 Linien miteinander vergleichen....23
11.8 Kausalität....24
11.9 Nichtlineare Dynamik....25
11.10 Statistik....25
11.11 Systematische Analysekette....26
12 Typische Arbeitsabläufe....27
12.1 Eine Beobachtungsnacht mit dem Echelle-Spektrographen reduzieren....27
12.2 Einen neuen Echelle-Spektrographen einbinden....27
12.3 Von der Spektrenserie zur Linien-Zeitreihe....27
12.4 Perioden und Schichtstruktur einer pulsierenden Atmosphäre untersuchen....27
12.5 Bahn eines spektroskopischen Doppelsterns bestimmen....28
12.6 Emissionslinien von Be-Sternen verfolgen und an BeSS melden....28
12.7 Robustheit eines Ergebnisses prüfen....28
13 Konventionen, Formate und Abhängigkeiten....29
13.1 Datenformate....29
13.2 Header-Einträge der Korrekturskripte....29
13.3 Benötigte Pakete....29
13.4 Gemeinsame Module....30
Anhang A – Skriptindex....31

1 Einführung

Die Sammlung Spektropython_Deutsch enthält 192 Python-Skripte und Module für die Amateurspektroskopie: von der Planung einer Beobachtungsnacht über die Reduktion der CCD-Rohaufnahmen bis zur statistischen und nichtlinearen Analyse langer Messreihen. Dieses Handbuch beschreibt die Sammlung nach Aufgaben: Zu jeder Fragestellung wird gezeigt, welche Skripte zur Verfügung stehen, was sie leisten, welche Eingaben sie erwarten und was sie ausgeben.

Es ergänzt die vorhandenen Dokumente:

- Uebersicht_Ordner_und_Skripte.docx – Verzeichnis aller Skripte nach Ordnern (Stand 04.10.2026).
- <Skriptname>_Dokumentation.docx – ausführliche Beschreibung eines einzelnen Skripts mit allen Parametern; liegt jeweils im Ordner des Skripts.
- Echelle_Pipeline_Dokumentation.docx – Gesamtbeschreibung der Echelle-Reduktion.
- TSA/90_Dokumentation – Methodenbeschreibungen zur Zeitreihenanalyse.

Für die Bedienung im Einzelnen (Parameterblöcke, Dialoge, Spaltennamen) ist die jeweilige Skriptdokumentation maßgeblich. Das Handbuch dient der Orientierung: Was ist möglich, und womit fange ich an?

1.1 Arbeitsumgebung

- Python 3.13 (Umgebung astro-env) unter Linux, Entwicklung und Ausführung in Spyder mit Qt5-Backend.
- Die meisten Skripte fragen Dateien und Parameter über Qt5-Dialoge ab und kommen ohne input() aus; sie laufen mit F5 in Spyder. Viele lassen sich zusätzlich über die Kommandozeile oder einen Konfigurationsblock am Skriptanfang steuern.
- Gemeinsam genutzte Module (Linienlisten.py, RV_Werkzeuge.py, Observatorien.py) liegen im Ordner Bausteine und müssen im PYTHONPATH stehen (Spyder: PYTHONPATH-Manager).
- Originaldateien bleiben unverändert. Ergebnisse werden als neue Dateien mit Namenszusatz oder in Unterordnern abgelegt; Skripte, die Header ändern, legen auf Wunsch Sicherungskopien (*.bak) an.

2 Die Arbeitskette im Überblick

Die Ordner bilden zusammen eine durchgehende Verarbeitungskette. Die folgende Tabelle ordnet die Arbeitsschritte den Kapiteln dieses Handbuchs zu.

Schritt	Was geschieht	Kapitel / Ordner
1 Vorbereiten	Beobachtbarkeit, Objektdaten, Archivdaten, Linienwellenlängen	3 – astroplanner, Katalogabfragen, Formatumwandlungen
2 Reduzieren	Master-Frames, Extraktion, Wellenlängenkalibrierung, Normierung, Merge der Ordnungen	4 – Reduce, NRES
3 Serie prüfen	Header, Zeitangaben, Bestandsaufnahme, Qualitätskontrolle, Kosmiks entfernen	5 – Header, Serien, SNR
4 Aufbereiten	Kontinuumsnormierung, baryzentrische und tellurische Korrektur, RV-Nullpunkt, Ausschnitt, Mittelung, Glättung, Differenzspektren	6 – Normierung, BC, Trocknung, RV, Serien, Rebinning…
5 Messen	EW, Linientiefe, FWHM, RV, Profilfits, Bisektor, v sin i, Asymmetrie, Temperatur	7 – EW, LinienparameterBestimmen, Linienfitting, KK, RV, TSA/60
6 Darstellen	Viewer, Vergleichs- und Zeitserienplots, dynamische Spektren, Animationen	8 – Spektrenplots, DarstellungImGeschwindigkeitsraum, Plots, ascii
7 Modellieren	Synthetische Spektren, Faltung, Rotationsverbreiterung	9 – theoretischeSpektren, Convolution
8 Doppelsterne	Periode, Phase, Orbitelemente, Komponentenspektren, Lichtkurven	10 – binaries_…, Lichtkurven
9 Zeitreihen	Perioden, Zeit-Frequenz, Linienvergleich, Lags, Kausalität, nichtlineare Dynamik, Statistik	11 – TSA

Kapitel 12 beschreibt typische Abläufe, die mehrere dieser Schritte verbinden; Kapitel 13 fasst Konventionen, Dateiformate und benötigte Pakete zusammen. Anhang A listet alle Skripte alphabetisch mit Verweis auf das Kapitel.

3 Beobachtung vorbereiten und Daten beschaffen

3.1 Beobachtungsnacht planen

astroplanner_20260924.py zeigt für einen Beobachtungsort und eine Nacht den Höhenverlauf eines oder mehrerer Objekte in Ortszeit, mit Luftmasse, Dämmerungsphasen, Mondhöhe, Mondbeleuchtung und Mondabstand. Es berechnet Kulmination sowie Auf- und Untergang über einer Mindesthöhe und gibt das nutzbare Fenster aus (Objekt hoch genug und Sonne unter der gewählten Dämmerungsgrenze). Optional erstellt es eine Jahresübersicht der nutzbaren dunklen Stunden. Objekte werden per Name (Sesame, mit lokalem Zwischenspeicher) oder mit Koordinaten angegeben.

3.2 Objekt- und Liniendaten abfragen

Skript	Möglichkeiten
Simbadabfrage_a.py	Objektsteckbrief aus SIMBAD für ein oder mehrere Objekte: Koordinaten, Eigenbewegung, Parallaxe, RV, Spektraltyp, Helligkeiten U…K und G, alle Bezeichner; Messtabellen (Teff/log g/[Fe/H], v sin i, Variabilität, Durchmesser u. a.); optional Excel-Export mit einem Blatt je Tabelle.
LinienwellenlängenNIST.py	Linienwellenlängen aus der NIST Atomic Spectra Database für ein oder mehrere Ionen und einen Wellenlängenbereich, in Luft oder Vakuum, mit Intensitätsschwelle; Ausgabe als Excel und als Textliste für die eigenen Linienlisten.
ESO_Abfrage.py	Suche und Download reduzierter Phase-3-Daten aus dem ESO-Archiv (z. B. HARPS, UVES, FEROS, ESPRESSO) nach Objekt und Suchbox; Übersicht, Auswahl, CSV-Tabelle und Download, optional mit Login.
Formatumwandlungen.py	Interaktive Umrechnungen: JD/MJD ↔ Kalenderdatum, RA/DEC sexagesimal ↔ Grad, HJD, BJD_TDB, baryzentrische RV-Korrektur, Höhe, Azimut und Luftmasse; die Funktionen sind auch importierbar.

3.3 Liniendatenbank

Bausteine/Linienlisten.py enthält eine Liniendatenbank mit 228 Linien aus 21 Ionen sowie tellurischen Linien, mit Such-, Filter-, Auswahl- und Exportfunktionen. Über 20 Skripte greifen darauf zu, etwa für die Linienauswahl per Dialog, für Linienmarken im Viewer oder für Atomdaten (E_low, gf) in den Temperaturbestimmungen. Neue Linien – z. B. aus der NIST-Abfrage – werden hier eingetragen und stehen dann in allen Skripten zur Verfügung.

4 Rohdaten reduzieren

4.1 Master-Aufnahmen und Sichtprüfung

Skript	Möglichkeiten
MastersErzeugen_CCD_2d_fits.py	Kombiniert mehrere 2D-Aufnahmen je Gruppe (Science, Dark, Kalibrierlicht, Kalibrier-Dark, Flat, Flat-Dark) zu Master-Aufnahmen (Mittel, Median oder σ-Clipping) und erzeugt die dunkelstromkorrigierten Master. Interaktiv oder per Kommandozeile.
FITS_2D_Viewer.py	Viewer für 2D-FITS: ZScale/MinMax/Perzentil-Skalierung, Stretch linear/Wurzel/Log/Asinh, Pixelinfo, Zeilen- und Spaltenprofil per Klick, Markierung gesättigter Pixel, Blättern durch den Ordner, Header-Fenster mit Suche, PNG-Export.
fits_zu_png.py	Exportiert alle FITS-Dateien eines Ordners (1D, 2D, erste Ebene von Würfeln) als PNG – nützlich als schnelle Bildübersicht einer Nacht (siehe Kapitel 8).

4.2 Echelle-Pipeline (Echelle-Spektrographen)

Die Pipeline im Ordner Reduce/Echelle führt von Master-Flat, Master-ThAr und Science-Aufnahme zum normierten, zusammengeführten 1D-Spektrum. Sie ist für FLISES (R ≈ 18 000) entwickelt und wird auch für die Instrumente weiterer Beobachter verwendet; die zugehörigen geprüften Linienkataloge liegen in den Unterordnern geprüfteVersion_*.

Schritt	Skript	Leistung
1	Echelle_Extraktion.py	Ordnungsdetektion im Flat, Tracing per Polynom, feste Ordnungsnummern über eine Referenztabelle, Streulichtkorrektur, Extraktion mit adaptiver Apertur (Summe oder optimale Extraktion) einschließlich Fehlerspektrum, Flat-Fielding.
2	ThAr_Kalibrierung.py	Wellenlängenkalibrierung gegen die instrumentenspezifische Linienliste: zwei Ankerklicks, Echelle-Modell, robuste Gittersuche je Ordnung, Gauß-Zentroide, globaler 2D-Fit $\lambda \cdot m = P(p, m)$ zur Erkennung von Fehlzuordnungen, Anwendung auf ThAr und Science.
3	Normierung_Ordnungen.py	Kontinuumsnormierung jeder Ordnung: Kosmik-Ausblendung, asymmetrisches Clipping im Log-Fluss, symmetrischer Endfit, Polynom oder Spline.
4	Mergen_Ordnungen.py	Zusammenführen der Ordnungen auf ein gemeinsames lineares Gitter mit $\cos^2$-Gewichtung in den Überlappungen, optional S/N-gewichtet; Diagnose des Kontinuumsversatzes.
–	Live_Reduktion.py	Überwacht während der Nacht einen Eingangsordner und reduziert jede neue Science-Aufnahme automatisch mit den Schritten 1–4; Kalibrierung mit dem Abend-ThAr oder einer übernommenen Lösung.

Werkzeuge zur Kalibrierung und Diagnose:

Skript	Möglichkeiten
Echelle_instrumentenspezifischen_Linienkatalog_Erstellen.py	Erstellt für einen neuen oder geänderten Echelle-Spektrographen die Identifikationstabelle (bachesLINE-Format: X, ORDER, WAVEC, IDENT, PEAK) aus extrahierten ThAr-Ordnungen und einem allgemeinen ThAr-Katalog; Anker werden gespeichert und wiederverwendet.
ThAr_Linienzaehler.py	Zählt je Ordnung brauchbare, gesättigte und verblendete ThAr-Linien und gibt FWHM-Mediane aus – Kontrolle von Belichtung, Fokus und Auflösung.
Diagnose_Linienausbeute.py	Zeigt je Ordnung und Peak-Schwelle, wie viele Linien jeder Filterschritt der Zuordnung kostet; klärt, ob eine niedrigere Schwelle in linienarmen Ordnungen hilft.
ThAr_Linienlkatalog_Viewer.py	Stellt einen Linienkatalog als synthetisches ThAr-Spektrum dar; Klick auf eine Linie zeigt Wellenlänge, Amplitude, Ordnung und Pixelposition.
ThAr_Aufnahme_Zeigen.py	Zeigt eine 2D-ThAr-Aufnahme und speichert sie als PNG.

4.3 Spaltspektrograph und spaltloser Spektrograph

Für klassische Spektrographen steht im Ordner Reduce/klassischerSpektrograph ein durchgängiger Ablauf zur Verfügung, wahlweise als Einzelskripte oder als grafische Oberfläche. Beide nutzen dieselben Rechenfunktionen (spektro_kern.py, spektro_ablauf.py) und liefern deshalb identische Ergebnisse.

Skript	Möglichkeiten
2._Ausrichten des Spektrumstreifens_CCD_Aufnahmen_fits.py	Dreht den Spektrumstreifen waagerecht; Winkel automatisch aus dem Spurverlauf, per zwei Klicks oder fest vorgegeben; Anwendung auf weitere Aufnahmen der Nacht.
3._ExtraktionUndWellenlaengenkalibrationNormierung_mit Spalt_...py	Extraktion mit Himmelsabzug und Kosmikfilter, Kalibrierlampe ohne Himmelsabzug, Linien anklicken und subpixelgenau zentrieren, Polynom λ(Pixel) mit Residuen, lineares Gitter, optional Normierung. Parameter werden als JSON gespeichert und für weitere Aufnahmen derselben Nacht wiederverwendet.
3a_ExtraktionUndWellenlaengenkalibrationNormierung_spaltlos_...py	Dasselbe für spaltlose Spektren; Kalibrierung ungefähr über bekannte Linien des Sternspektrums selbst, sonst in Pixeln.
Reduce_GUI.py	Oberfläche für den gesamten Ablauf: Master, Ausrichten, Extraktion, Wellenlängenkalibrierung, Normierung, Anzeige.

4.4 NRES-Spektren (Las Cumbres Observatory)

Reihenfolge: NRES_auslesen.py extrahiert die 68 Ordnungen der Wissenschaftsfaser als 1D-FITS mit 0,05 Å/Pixel → VacToAir.py rechnet von Vakuum- auf Luftwellenlängen um → heliocentricCorrection_NRES.py wendet die baryzentrische Korrektur aus dem Header-Wert BARYCORR an → PlotAllerOrdnungen.py erzeugt einen Kontrollplot.

5 Spektrenserien verwalten und prüfen

Bevor eine Serie ausgewertet wird, sollten Zeitangaben, Header und Datenqualität geprüft werden. Die folgenden Skripte arbeiten jeweils auf einer ganzen Serie von 1D-FITS-Spektren.

5.1 Header und Zeitangaben

Skript	Möglichkeiten
fits_header_pruefen_korrigieren.py	Findet und korrigiert Verstöße gegen den FITS-Standard: Umlaute und Sonderzeichen, Winkel und Zahlen mit Einheit ohne Anführungszeichen, kleingeschriebene Schlüssel u. a.; meldet unklare Fälle (MJD-Schlüssel mit vollem JD, nicht ISO-konforme Datumsangaben). Prüfbericht als CSV.
JD_Header_Pruefung.py	Liest alle Zeit-Keywords, rechnet sie in JD (UTC) um, vergleicht sie untereinander und trägt auf Wunsch das Keyword JD ein (Beginn oder Mitte der Belichtung). HJD/BJD werden nicht stillschweigend als JD übernommen.
Headerprint_Serie_csv.py	Header aller Dateien als CSV je Datei, als Gesamttabelle (Datei × Keyword) und als Vergleichsbericht: welche Keywords konstant sind, variieren oder fehlen.
BESS_fits_header_Serie.py	Ergänzt die Header einer Serie BeSS-konform; beobachterspezifische Einträge werden automatisch gesetzt, fehlende objektbezogene Angaben abgefragt.

5.2 Bestandsaufnahme und Qualitätskontrolle

Skript	Möglichkeiten
Zeitserie_Uebersicht.py	Tabelle je Spektrum: Zeitpunkt, Wellenlängenbereich, Dispersion, Flussminimum/-maximum, S/N-Schätzung; dazu Zusammenfassung (Zeitraum, Nächte, Lücken) und Übersichtsgrafik.
spektren_diagnose.py	Tatsächliche Wellenlängen- und JD-Bereiche der Spektren (lineare und log-lineare Dispersion, Einheiten) mit Empfehlungen für Zeitreihenplots.
spektren_qualitaetskontrolle.py	Interaktive Durchsicht: S/N, NaN-, Null- und negative Pixel, Spikes, Kontinuumsniveau, Warnhinweise; Behalten oder Verwerfen per Taste, Vergleich mit dem Serienmedian, Protokoll QC_Protokoll.csv, Fortsetzen unterbrochener Sitzungen, verworfene Dateien in einen Unterordner.
SNR_Bestimmung.py	S/N einer Serie mit sechs Methoden: Kontinuumsfenster, DER_SNR, Beta-Sigma, gleitendes Fenster, Autokorrelation, Poisson. Erkennt resampelte Spektren (korreliertes Rauschen) und korrigiert die Differenzmethoden entsprechend.
cosmics_entfernen.py	Erkennt Kosmiken und Hotpixel in 1D-Spektren anhand von Breite und Schärfe und ersetzt sie durch Interpolation; Spektrallinien bleiben erhalten.

5.3 Dateien umbenennen und umwandeln

Skript	Möglichkeiten
JD_Wellenlaenge_in_Dateiname.py	Kopien mit JD (und optional Wellenlängenbereich) am Anfang des Dateinamens – Serien sortieren sich dann zeitlich.
fit_Serie_in_csv_und_dat.py	FITS-Serie als CSV (Komma) und DAT (Tab) mit Spalten WAVE, FLUX. Dabei Verlust des Headers.

6 Spektren aufbereiten und korrigieren

6.1 Kontinuumsnormierung

Skript	Möglichkeiten
Universelle_Normierung.py	Normierung ganzer 1D-Spektren: steifer Anfangsspline mit Bestätigung, danach iteratives Clipping; wahlweise einphasig oder 1-Punkt-Normierung (Median). Batch-Betrieb für ganze Verzeichnisse mit Logdatei.
Normierung_einzelne_Linien_20260923.py	Normierung des Umfelds einer Linie über Stützpunkte (per Klick, CSV oder Zahlenliste), Fluss je Stützpunkt robust als Perzentil; Stützpunkte können mit der Linie mitgeführt werden und weichen dabei Ausschlussbereichen aus. Für Zeitserien mit identischen Stützpunkten.
stuetzpunkt_validator_einzelneLinien_20260923.py	Prüft, ob Stützpunkte bei RV-variablen Linien (γ, K, baryzentrische Korrektur) in Linien geraten können, und schlägt Ersatzpunkte vor.
Normierung_Ordnungen.py	Normierung einzelner Echelle-Ordnungen (Kapitel 4).

6.2 Baryzentrische und tellurische Korrektur

Skript	Möglichkeiten
Baryzentrische_Korrektur_Spektren.py	Heliozentrische Korrektur einer Serie; Beobachterstandort aus Observatorien.py, Sternkoordinaten online oder manuell; Protokoll-CSV. Restfehler bis einige 10 m/s gegenüber einer echten baryzentrischen Korrektur.
telluric_bc_korrektur.py	Tellurische Korrektur (“Trocknung”) nach Hanuschik (2006) und anschließende baryzentrische Korrektur mit barycorrpy im Bereich 6500–6620 Å; Ausgabe <Name>_bc_trocken.fits.
telluric_trocknung_hitran.py	Tellurische Korrektur ohne baryzentrische Korrektur: Hanuschik-Liste, in der Hα-Lücke ergänzt durch HITRAN-H_2O-Linien; Fit von Wassersäule und instrumenteller FWHM; wahlweise auf dem Originalgitter (kein Resampling).
heliocentricCorrection_NRES.py	Korrektur von NRES-Spektren aus dem Header-Wert BARYCORR.

6.3 Radialgeschwindigkeit korrigieren

Die Skripte im Ordner RV nutzen das gemeinsame Modul RV_Werkzeuge.py und dokumentieren jede Verschiebung einheitlich im Header (RVMEAS, RVERR, RVSHIFT, RVSHTOT, RVMETH, HISTORY). Mehrere aufeinanderfolgende Korrekturen bleiben so nachvollziehbar.

Skript	Möglichkeiten
terrLinien_RVKorrektur_RBF_...py / ..._Regression_...py	Korrektur des Wellenlängen-Nullpunkts an tellurischen Linien (Minimum per Radial-Basis-Funktion bzw. Polynom); Spektren dürfen noch nicht baryzentrisch korrigiert sein.

Skript	Möglichkeiten
RVKorrekturPerTemplate_1d_spectra_fits_series.py	RV relativ zu einem Template (Datei oder iterativ gebildeter Serienmedian) per Kreuzkorrelation; Verschiebung aller Spektren auf das Template, mit wählbaren Wellenlängenbereichen und Ausschlüssen.
RVKorrektur_1d_spectra_fits_series.py	Verschiebung um einen festen Wert oder um tabellierte Werte aus einer Protokoll-CSV – z. B. um eine einmal bestimmte Korrektur auf andere Fassungen derselben Spektren zu übertragen.
UmrechnungSpektrumVonBaryzentrischAufSternBezogen.py	Überführt Spektren in das Ruhesystem einer Doppelsternkomponente (Kapitel 10).

6.4 Ausschnitt, Mittelung, Rebinning, Glättung

Skript	Möglichkeiten
Crop_Ausschnitt_Zeitserie.py	Schneidet FITS- und ASCII-Spektren auf einen Wellenlängenbereich zu; zeigt vorab, wie viele Spektren den Bereich vollständig enthalten.
MittelungVon_fits.py	Mittleres Spektrum aus beliebig vielen Spektren, auch mit unterschiedlichem Gitter: Median, Mittel oder S/N^2-gewichtet, mit σ-Clipping, Mindestzahl beitragender Spektren je Pixel; Ausgabe mit Extensionen NCOMB und SIGMA.
newbinning_mittleresSpektrum_v2.py	Rebinning einer Serie auf ein gemeinsames Gitter (linear, Spline oder flusserhaltend), Mittelspektrum und Statistikspektrum (Mittel, Median, Streuung, Anzahl je Pixel).
Zeitserie_Spektren_smoothen.py	Glättung mit Gauß auf Ziel-R, festen km/s oder Pixeln, Boxcar, Savitzky-Golay oder Median; effektive Auflösung und Rauschfaktor werden berechnet und in den Header geschrieben.

6.5 Differenzspektren

Differenzspektren_v2.py bildet für jedes Spektrum der Serie die Differenz zu einem Bezugsspektrum (Datei, Mittel oder Median der Serie), optional mit Sammelplot, dynamischem Differenzspektrum und Streuungsspektrum σ(λ). Differenzspektren_mitPlotAlle_v2.py ruft dasselbe Skript mit Sammelplot auf. Für Linienausschnitte im Geschwindigkeitsraum siehe Kapitel 7 (SpektrenDifferenzZumReferenzspektrum_Linie.py, HalphaVariabilitaet.py).

6.6 Rauschen hinzufügen

verrauschen.py versieht Spektren mit gaußschem Rauschen – mit festem σ oder auf ein Ziel-S/N, als konstantes oder Photonenrauschen, bei resampelten Spektren mit derselben Korrelationsstruktur wie das vorhandene Rauschen. Damit lässt sich prüfen, wie RV, EW oder gefundene Perioden vom S/N abhängen, und es lassen sich Monte-Carlo-Fehler abschätzen.

7 Linien und Spektren messen

Die Messskripte erwarten in der Regel kontinuumsnormierte 1D-FITS-Spektren einer Zeitserie und schreiben je Spektrum eine Zeile in eine Ergebnistabelle (CSV oder Excel) mit JD. Diese Tabellen sind die Eingabe der Zeitreihenanalyse in Kapitel 11.

7.1 Äquivalentweite, Linientiefe, Lage und Breite

Skript	Möglichkeiten
Zeitserie_EW_Linientiefe_Linienwellenlaenge_FWHM_mehrereModelle.py	Zentrales Messskript: EW, Linientiefe, Wellenlänge des Minimums und FWHM einer Linie über die Serie. Integrationsgrenzen grafisch oder als Zahlen, lokale Renormierung, Linienmodell Spline, Gauß, Voigt, Split-Voigt oder automatische Wahl nach BIC; 1σ-Fehler aller Größen.
EW_Absorption_Emission.py	EW mit getrennter Absorptions- und Emissionskomponente (EW_abs, EW_emi, EW_net); zusätzlich eine rauschbereinigte Variante, die nur signifikante Bereiche zählt. Grafische Bereichswahl.
Zeitserie_EWs_mehrereBereicheInEinerLinie.py	EW mehrerer zusammenhängender Teilbereiche einer Linie (z. B. blauer Flügel, Kern, roter Flügel); Grenzen für alle gleich oder je Spektrum, speicher- und ladbar.
Linienflaeche_Geschwindigkeitsbereich.py	Linienfläche in einem frei wählbaren Geschwindigkeitsintervall relativ zur Laborwellenlänge, in Å und km/s, mit über die Serie konstantem Vorzeichen.

7.2 Profilanpassung

Skript	Möglichkeiten
FitLinie.py	Fit einer Linie oder eines Blends über die Serie mit linearem Kontinuum und Gauß, Doppel-Gauß, Lorentz, Split-Lorentz, Voigt oder Doppel-Voigt (lmfit); Startwerte aus den Daten. Ergebnis: Zentrum, FWHM, Höhe, Fläche, EW (Modell und modellfrei), optional RV – alle mit Fehlern.
FitLinie_automatischParametrisiert.py	Wie FitLinie.py, mit Nachführung von Bereich und Startwerten von Spektrum zu Spektrum und automatischem Neustart bei schlechtem Fit – für dopplerverschobene oder stark veränderliche Profile.
FitLinie_PositiverUndNegativerGauss_Version2.py	Emissionslinie mit zentraler Absorption (Be-Sterne, β Lyr, Hα in Doppelsternen): Emissions- plus Absorptions-Gauß mit Nebenbedingungen gegen Rollentausch; Ergebnis u. a. EW beider Komponenten, RV beider Komponenten, V/R-Verhältnis.
FitLinie_PositiverUndNegativerVoigt_Version2.py	Dasselbe mit Voigt-Komponenten für ausgedehnte Flügel; jede Komponente auch als Gauß rechenbar.
Halpha_Doppelgauss_Fit.py	Hα bei Deneb: Emissions- und Absorptionskomponente, Multistart-Fit, Warnung bei mehrdeutigen Lösungen; optional FITS-Modellspektrum und Darstellung über der Phase.

Skript	Möglichkeiten
Halpha_ProfilanpassungVoigt_Gauss.py	Mehrkomponentenfit des Hα-Profils im Geschwindigkeitsraum (Absorption als Gauß, Voigt oder Split-Voigt, Emission nahe Laborwellenlänge) mit optionaler Maskierung tellurischer Linien; Fehler aus der vollen Kovarianzmatrix.
FWHM_einerLinie_Serie_fits.py	FWHM und Auflösungsvermögen R aus Lampen- oder tellurischen Linien: eine Linie über die Serie (Fokusstabilität) oder alle Linien eines Spektrums (R(λ) des Spektrographen).

7.3 Radialgeschwindigkeit messen

Skript	Möglichkeiten
RV_Messung_eineLinieInEinerfitsSerie_Spline.py	RV einer Linie aus der Lage des Minimums (bzw. Maximums): exakter oder glättender Spline, Parabel oder Gauß; Fehler per Monte-Carlo; optional baryzentrisch korrigiert mit HJD und BJD_TDB.
CrossCorrelation_fits_Serie_v2.py	RV der ganzen Serie durch Kreuzkorrelation mit einem Template auf ln-λ-Gitter; Fehler nach Zucker (2003); Ausschluss tellurischer und interstellarer Bereiche; baryzentrische Korrektur ohne Internet.
CrossCorrelation_fits_Serie_einzelneLinien_v2.py	Dasselbe, beschränkt auf Fenster um einzelne Linien – RV mehrerer Linien aus verschiedenen Atmosphärenschichten in einem Lauf.
Bisektor_RV_Zeitreihe.py	Bisektor-RV bei fester relativer Linientiefe mit mitwanderndem Geschwindigkeitsfenster; Qualitätskennzahlen und Flag-Bitfeld je Messung. Für konsistente Variabilitätsanalyse, nicht für absolute RV.

7.4 Bisektoren und Linienasymmetrie

Skript	Möglichkeiten
Bisektor_Analyse.py	Bisektor einer Linie für jedes Spektrum bei wählbaren Tiefenniveaus (parallel gerechnet); mittlerer Bisektor mit ±1σ, Zeitreihen je Tiefe, überlagerte Einzelbisektoren nach JD eingefärbt.
Bisektor_CCF_Tiefen.py	Kreuzkorrelation der Bisektor-RV-Reihen aller Tiefenniveaus: Lag-Matrix zwischen innerer und äußerer Linienbildung.
Spectral_line_asymmetry_interactive.py	Asymmetrie durch Spiegelung der roten Linienhälfte am Minimum; Differenzflächen (gesamt, positiv, negativ) und Extremwerte je Spektrum.
SpektrenDifferenzZumReferenzspektrum_Linie.py	Differenz jedes Spektrums zum Median- oder einem externen Referenzspektrum im Bereich einer Linie; Differenzflächen im Geschwindigkeitsraum.
DifferenzflaechenPlot.py	Plot der Differenzflächen aus dem vorigen Skript über JD.

Skript	Möglichkeiten
FluxBeiOffset_Linie.py	Fluss in festem Geschwindigkeitsabstand links und rechts vom Linienminimum als Zeitreihe.
HalphaVariabilitaet.py	Geschwindigkeitsaufgelöste Variabilitätskarte σ(v) und σ(v, t) sowie Differenzintegrale in wählbaren Geschwindigkeitszonen als Zeitreihen.

7.5 Rotation und Makroturbulenz

Makroturbulenz_Fourier_Methode.py bestimmt v sin i und die Makroturbulenz ζ aus einer Linie mit der Fourier-Methode nach Gray (2005): erste Nullstelle der Transformierten für v sin i, Signifikanztest des Nebenmaximums, ζ aus dem Hauptmaximum, unabhängiger Profilfit im Geschwindigkeitsraum als Gegenprobe und Monte-Carlo-Fehler. Makroturbulenz_Fourier_Methode_batch.py wendet dieselbe Rechnung auf viele Spektren und Linien aus Linienlisten.py an und erzeugt eine Sammel-CSV sowie Zeitreihen v sin i(JD) und ζ(JD).

7.6 Temperatur und Elektronendruck

Skript	Möglichkeiten
Anregungstemperatur.py	Anregungstemperatur aus dem **EW-Verhältnis** zweier Linien desselben Ions (Boltzmann), als Zeitreihe und phasengefaltet.
Boltzmann_Temperatur_Fitter.py	T_exc(JD) aus einem gemeinsamen Boltzmann-Fit über Linien frei wählbarer Ionen (z. B. Ti II, Fe II, Cr II); Atomdaten aus Linienlisten.py.
Boltzmann_Saha_Analyse.py	Gleichzeitige Bestimmung von T_exc (Boltzmann, Fe I) und Elektronendruck P_e (Saha, Fe I/Fe II) aus denselben Spektren.
LDR_Kalibration_Heisse_Sterne.py	Kalibration von **Linientiefenverhältnissen** gegen Teff an Modellspektren (eigene MOOG-Spektren oder BOSZ-Gitter von MAST) für A-Überriesen.
LDR_Batch_Analyse_aus_fits.py	Anwendung dieser Kalibration auf eine FITS-Serie: Teff-Zeitreihe mit Fehlern und Einzelwerte je Linienpaar.

8 Spektren darstellen

8.1 Einzelne Spektren ansehen

Skript	Möglichkeiten
1d_fitSpektrum_ansehen.py	Gesamtspektrum mit gekoppeltem Zoomfenster, Header-Ausgabe, Blättern durch den Ordner mit Pfeiltasten, PNG-Speicherung.
1d_fitSpektrum_ansehen_mitLinienidentifikation_MitSubplot.py	Viewer mit Linienmarken aus Linienlisten.py (Auswahl nach Name, Element, Bereich), optional um eine RV verschoben; Rechtsklick identifiziert die nächstliegende Linie mit Geschwindigkeitsabstand.
FITS_2D_Viewer.py	2D-Aufnahmen (Kapitel 4).
1d_dat_read_plot_csv_speichern.py, 1d_txt_plotten.py	ASCII-Spektren beliebigen zweispaltigen Formats lesen (Trennzeichen und Kodierung automatisch) und als PNG speichern.
PlotAndCrop_1d_ascii.py	ASCII-Spektrum plotten und einen Ausschnitt als .dat speichern.

8.2 Mehrere Spektren und Zeitserien

Skript	Möglichkeiten
Spektren_Vergleich.py	Qt-Oberfläche zum Übereinanderlegen kalibrierter Spektren (FITS-Bild oder -Tabelle, auch ESO-Format): Ein-/Ausblenden, Offset, Normierung im Ausschnitt, Differenz zum Mittel, Farben nach JD, Linienmarken, Export.
timeseries_Spektren.py	Zeitserien mit Offset oder je Spektrum eine Grafik; JD- und Wellenlängenbereich, Beschriftung mit JD oder Dateiname, Farbverlauf nach JD.
spektren_stapelplot.py	Mehrere ASCII-Spektren übereinander, farbig oder mit Offset (Wasserfall), mit S/N-Filter.
comparison_plot_2spectra_ascii.py	Overplot zweier ASCII-Spektren als PDF.
fits_zu_png.py	Batch-Export aller FITS-Dateien eines Ordners als PNG, parallel, nur neue Dateien.
AnimationAusPlots.py	Bildserie als Animation abspielen und als MP4 oder GIF exportieren.

8.3 Linienprofile im Geschwindigkeitsraum

Skript	Möglichkeiten
Dynamisches_Spektrum.py	Dynamisches Spektrum einer Linie: Geschwindigkeit (oder Wellenlänge) gegen JD, Farbe = Fluss; Residuen zum Mittelprofil, zeitliche Streuung σ(v), Nachtmittel, korrekte Darstellung von Beobachtungslücken, optional 3D; Matrix als FITS.

Skript	Möglichkeiten
fitsSerie_Linie_imGeschwindigkeitsraum_JDBereich_v2.py	Eine Linie: alle Profile überlagert, mit Offset nach JD und als dynamisches Spektrum, in einem wählbaren JD-Bereich.
fitsSerie_Linien_imGeschwindigkeitsraum_JDBereich_v2.py	Mehrere Linien nebeneinander mit gemeinsamer JD-Achse oder in einem Plot – Vergleich verschiedener Bildungsschichten.

8.4 Messtabellen plotten

PlotsAusSpaltenVon_Tabellen.py stellt Spalten aus CSV, FITS-Tabellen (TFTS), Excel oder ASCII gegeneinander dar, auch über Tabellengrenzen hinweg mit Zuordnung über eine Schlüsselspalte wie JD, und berechnet Korrelation und Regression. Für Zeitreihen von Messgrößen siehe zusätzlich Kapitel 11.2.

9 Modellspektren und Faltung

Skript	Möglichkeiten
Spektrum_GaussBroadened.py	Aufbereitung synthetischer oder tabellierter Spektren (Kurucz über PyAstronomy, POLLUX, beliebige ASCII-Tabellen): Vakuum→Luft, Ausschnitt, Gauß-Faltung auf konstantes R oder konstante FWHM, Abtastung auf ein äquidistantes Gitter; Ausgabe als DAT, CSV, FITS.
convol_fits.py / convol_dat.py	Gauß-Faltung von FITS- bzw. Tabellenspektren auf ein Auflösungsvermögen R (über ln λ), eine FWHM oder σ; mit Angabe von R0 wird nur die Differenz gefaltet.
Pollux_Ausschnitt_convol.py	Ausschnitt und Faltung eines POLLUX-Modellspektrums.
rotationsverbreiterung_fits.py	Rotationsverbreiterung mit v sin i und Randverdunkelung (Kern nach Gray), optional anschließende Instrumentenfaltung – Modell direkt mit Messung vergleichbar.
KunstspektrumErzeugen.py	Testspektrum mit sechs Linien bekannter Lage, auf R verbreitert und verrauscht – zur Prüfung von Mess- und Kalibrierverfahren.
LDR_Kalibration_Heisse_Sterne.py	Lädt und degradiert BOSZ- oder MOOG-Modellspektren für die LDR-Kalibration (Kapitel 7).

10 Doppelsterne und Lichtkurven

10.1 Periode, Phase und Orbit

Skript	Möglichkeiten
Periodenanalyse.py	Periodensuche mit Generalized Lomb-Scargle (mit Oberwellen, FAP nach Baluev) und Stringlängenmethode nach Dworetsky; liest Text, Excel, ODS, FITS, Parquet, JSON.
Phasenplot_bekannte_Periode_20260924.py	Phasenplot mit bekannter Periode und Epoche aus beliebigen Tabellenformaten: Fourier- oder Medianmodell, Residuen, σ-Clipping auf den Residuen, Abschätzung der nötigen Periodengenauigkeit.
Phasenberechnung.py	Phase und Epoche einer Messwerttabelle bei bekannter Ephemeride, mit Phasenunsicherheit aus σ_P und σ_T0.
Phasenberechnung_ausSpektren_mitvorgegebenenToUndPeriode.py	Phase und Epoche je FITS-Spektrum aus dem Header-Zeitpunkt; Grafik der Phasenabdeckung und Termine, an denen die größte Phasenlücke beobachtbar ist.
Orbitelemente_ausJDundGemessenenRV_optimieren.py	Orbitelemente einer SB1 (K, e, ω, T0, γ) aus RV-Messungen bei bekannter Periode; Startwerte per Gittersuche, Fehler, $a_1 \sin i$, Massenfunktion.
RV_Berechnung_ausOrbitalelementenUndJD.py	Theoretische RV beider Komponenten aus Orbitelementen für Messzeitpunkte (mit O–C) oder einen Zeitraum (Beobachtungsplanung).

10.2 Komponentenspektren

Skript	Möglichkeiten
SummenUndDifferenzspektrum_gemessenMinustheoretisches_JePhase_mitIntegralkorrektur.py	Theoretisches Summenspektrum eines Doppel- oder Dreifachsystems je Phase (Komponenten verschoben und mit Phoebe-Flüssen gewichtet) und Differenz zum gemessenen Spektrum – zeigt Emission, Materieströme oder Scheiben.
UmrechnungSpektrumVonBaryzentrischAufSternBezogen.py	Spektren in das Ruhesystem einer Komponente verschieben (RV aus Tabelle oder Orbitelementen) und im Ruhesystem mitteln.

10.3 AAVSO-Lichtkurven

aavso_lightcurve_analysis.py verarbeitet AAVSO-Downloads und andere Helligkeitstabellen: automatische Spaltenerkennung, Obergrenzen (»fainter than«) getrennt dargestellt, diskrepante Beobachtungen verworfen, Mittelwerte je Band und Zeitintervall (fehlergewichtet, mit σ-Clipping), optional Lomb-Scargle-Periodogramm und Phasendiagramm.

11 Zeitreihenanalyse (TSA)

Der Ordner TSA enthält 86 Skripte zur Analyse von Messwert-Zeitreihen – RV, EW, FWHM, Linientiefe, Bisektorgeschwindigkeiten, Temperaturen –, die überwiegend unregelmäßig abgetastet sind und Saisonlücken haben. Die Nummern der Unterordner geben die übliche Reihenfolge an: vorbereiten (01–03), Perioden und Zeit-Frequenz (10–12), Linien vergleichen und Kausalität (20–21), nichtlineare Dynamik (30), Statistik (40), systematische Methodenkette (50). Die Spektrenanalysen in 60_Spektrenanalyse sind in Kapitel 7 beschrieben.

Eingabe sind Tabellen mit einer Zeitspalte (JD) und einer oder mehreren Messwertspalten, meist als CSV (Trenn- und Dezimalzeichen werden in der Regel automatisch erkannt) oder als FITS-Tabelle (TFTS). Viele Skripte des Linienvergleichs erwarten Dateien im Format line_analysis_<Linie>.csv bzw. line_analysis_<Linie>_gefiltert.csv.

11.1 Daten vorbereiten

Skript	Möglichkeiten
AusreisserAnalyse.py	Vierstufige Ausreißererkennung: S/N-Filter, LOESS-Residuen mit robustem z-Score je Messkanal, Kreuzvalidierung über mehrere Kanäle (z. B. RV, EW, FWHM derselben Linie), abschließend visuelle Entscheidung je Kandidat. Ausgabe: markierte und bereinigte CSV je Kanal.
MedianLoessFilter.py	Zeitbasierter Medianfilter und anschließende LOESS-Glättung für beliebig viele Kanäle; erzeugt die *_gefiltert.csv-Dateien mit Original-, Median- und LOESS-Spalte.
zeitraum_binning_und_interpolation.py	Mittelung einer Variablen auf Zeiträume wählbarer Länge (z. B. Nachtmittel) mit Streuung und Anzahl.
parse_stats_to_csv.py	Fasst *_stats.txt-Dateien in einer CSV zusammen.

11.2 Messreihen darstellen

Skript	Möglichkeiten
Messwerteplot.py	Zeitreihe einer Messgröße aus einer oder mehreren Tabellen: gemeinsamer Plot oder je Datei, Fehlerbalken, robuste Normierung, Markierungslinien, beliebig viele Zoomfenster.
Messwerte_Vergleich.py	Viele Messreihen über gemeinsamer JD-Achse in drei Layouts (Übersicht, kombiniert, Panels), mit Nachtmittel, σ-Clipping und Glättung (Savitzky-Golay, Gauß, Median, LOWESS).
PlotsAusSpalten_batch.py	Spalten aus CSV, FITS, Excel, ASCII gegeneinander; interaktiv, im Batch je Datei oder paarweise über alle Dateikombinationen; Ausreißerfilter, Statistik-CSV.
LOESS_Vergleich_Plots_batch.py	Je Linie ein Vergleichsplot der LOESS-gefilterten Größen APX, EW, FWHM und RV.

11.3 Testdaten erzeugen

synthSignal_GUI.py erzeugt synthetische Zeitreihen aus beliebig kombinierbaren Komponenten – 23 Signalformen von Sinus, Periodenänderung, Amplituden- und Phasenmodulation, quasiperiodischen und gedämpften Schwingungen über Transienten und Flares bis zu weißem, rotem und 1/f^β-Rauschen – mit realistischer Abtastung (Jitter, Beobachtungsfenster, Lücken). Damit lässt sich vorab prüfen, ob eine Methode ein bekanntes Signal unter den Bedingungen der echten Beobachtungen wiederfindet.

11.4 Periodenanalyse

Skript	Möglichkeiten
GLS.py	Generalized-Lomb-Scargle-Periodogramm (Zechmeister & Kürster 2009) für eine Spalte einer CSV.
PDM_Periodenanalyse.py	Phase Dispersion Minimization für Einzeldatei oder Batch; automatische Minima, Bin-Stabilitätsanalyse, Phasenplots, Top-10-Perioden je Datei.
CLEAN_Periodenanalyse.py	CLEAN-Dekonvolution (Roberts et al. 1987): trennt echte Perioden von Aliasen der Fensterfunktion; Dirty-, Clean- und Fensterspektrum, Phasenplots.
AKF.py	Autokorrelationsfunktion mit Konfidenzband.
tsa_prewhitener_auto_white.py	Detrending, mehrgliedriges Lomb-Scargle, iteratives Prewhitening bis zu weißen Residuen (Ljung-Box, ACF), Bootstrap-FAP, MCMC-Periodenschätzung, Fensterfunktion, optional GP; Export als CSV, JSON, FITS.
SinusfitPeriode.py	Simultane harmonische Regression für vorgegebene Perioden: Amplituden, Phasen, Residuen.
Phasenfaltung_eine_Linie.py	Faltet mehrere Messgrößen einer Linie (RV, EW, FWHM, Tiefe, Asymmetrie) auf eine gemeinsame Periode, mit Median-Binning und Phasenversätzen der Extrema – z. B. zum Nachweis von Schocksignaturen.
Epochen_Screening.py	Gleitendes GLS- oder PDM-Screening in Zeitfenstern: findet Perioden, die nur episodisch auftreten.
amplituden_tracking.py	Verfolgt die Amplitude einer gefundenen Periode über die Zeit (gleitendes Fenster, Segmente, WWZ).

11.5 Zeit-Frequenz-Analyse

Skript	Möglichkeiten
wwz.py (mit wwz_kern.py)	Weighted Wavelet Z-Transform (Foster 1996) mit Messfehlern: Einzeldatei oder Batch, Signifikanz per χ^2 und gegen rotes Rauschen (analytisch und Monte-Carlo), Cone of Influence, Ridge-Tracking der dominanten Perioden, interaktive Synthese. Die CSV-Ausgabe ist Eingabe aller WWZ-Vergleichsskripte.
gleitendesGLS_Spektrogramm.py	Gleitendes GLS-Spektrogramm als lückenrobuste Alternative zur WWZ; optional Hann-Taper und Perioden-Tracking.

Skript	Möglichkeiten
eemd_analysis_final.py	Zerlegung in intrinsische Moden mit EEMD oder CEEMDAN, Hilbert-Bewertung, optional Rekurrenzplots je Mode.
Synchrosqueezing_Analyse.py	Synchrosqueezed Wavelet Transform nach GP-Interpolation; schärfere Perioden-Trajektorien als das klassische Skalogramm.
Sliding_SHO_celerite2.py	Interpolationsfreie Gegenprobe: fensterweiser Fit stochastisch getriebener Oszillatoren (celerite2) mit Periode, Gütefaktor Q und Amplitude über die Zeit.

11.6 Rauschen und stochastische Modelle

Skript	Möglichkeiten
GPR_Zeitreihenanalyse.py	Gaussian-Process-Regression mit automatischer Kernelwahl (RBF, Matérn, quasiperiodisch, mit Weißrauschen); rekonstruiertes Signal mit Unsicherheitsband und Leistungsspektrum des Kernels.
Strukturfunktion.py	Strukturfunktion unregelmäßig abgetasteter Reihen mit Potenzgesetz-Fit – charakteristische Zeitskalen und Rauschcharakter.
PSD_plot.py	Leistungsdichtespektren mehrerer Messgrößen einer Linie oder einer Messgröße mehrerer Linien.

11.7 Linien miteinander vergleichen

Dieser Bereich beantwortet, ob sich verschiedene Linien – und damit verschiedene Schichten der Sternatmosphäre – gemeinsam, zeitversetzt oder unabhängig verändern.

Korrelation, Regression, gemeinsame Einflüsse

Skript	Möglichkeiten
Korrelations_Analyse.py	Pearson- und Spearman-Matrix zwischen Messgrößen aus mehreren CSV nach JD-Abgleich.
Regressions_Analyse.py	Regressionsgerade zweier Messreihen nach JD-Abgleich mit Toleranz.
Kreuzkorrelation_Messwerte.py	Paarweise Kreuzkorrelation beliebiger Messreihen (interpolierte CCF und DCF), Peak- und Schwerpunkt-Lag, Heatmaps (Ordner KK).
EinflussfaktorAnalyse.py	Anteil gemeinsamer und individueller Einflüsse: PCA/Faktoranalyse, CCF, spektrale Kohärenz und Kreuz-WWZ.
Linienparameter_Vergleich.py	Ein Parameter über viele Linien: z-standardisierte Heatmap Linien × Zeit, Spearman-Matrix, Dendrogramm, gleitende Korrelation zu einer Referenzlinie.
Linienvergleich_Clustering.py	Clustering geglätteter RV-Zeitreihen vieler Linien zu Gruppen mit gemeinsamem Verlauf.
Messgroessen_Linienvergleich.py	Verlauf beliebiger Messgrößen mehrerer Linien im Vergleich.

Skript	Möglichkeiten
GLS_Linienvergleich.py	GLS-Periodogramme vieler Linien, deren Korrelation und hierarchisches Clustering.
ZeitdifferenzenExtremea_zweierLinien.py	Zeitdifferenzen zwischen lokalen Minima und Maxima zweier Reihen.

Zeitversatz (Lag) und Entstehungstiefe

Skript	Möglichkeiten
Lag_Analyse_Linienpaar.py	Lag zwischen zwei RV-Reihen: Gesamt-CCF mit Subsampling-Unsicherheit und periodenabhängiger Lag aus der Kreuz-WWZ (dispersive Wellenausbreitung).
Lag_Alle_Linienpaare_CCF.py	CCF-Lag für alle Paare einer Linienauswahl in einer Tabelle.
Lag_Ranking.py	Reihenfolge der Linien nach Entstehungstiefe aus der Lag-Matrix (gewichteter Score, Gewinnquote, Least-Squares-Konsistenz).
Lag_Referenzlinie_Extraktion.py	Lags aller Linien relativ zu einer Referenzlinie als sortiertes Balkendiagramm mit Fehlern.
EW_APX_vs_Lag.py / Eup_Elow_vs_Lag.py	Linienstärke bzw. Anregungspotenzial gegen den Lag – physikalische Prüfung des Rankings.
DCF_Kreuzkorrelation_Verifikation.py	Prüfung eines Lags mit der DCF (Edelson & Krolik 1988) direkt auf den Originaldaten, mit Bootstrap-Fehler.
Kreuzkorrelation_Bandpass_Verifikation.py	Prüfung durch Kreuzkorrelation bandpassgefilterter Reihen je Periodenband.

Kreuzspektren und WWZ-Vergleich

Skript	Möglichkeiten
LSSA_Kreuzspektrum.py	Segmentierte LSSA-Kohärenz ohne Interpolation mit Permutations-Signifikanz.
WaveletCrossBispektrum.py	Wavelet-Cross-Bispektrum und Bikoherenz: nichtlineare Kopplung zwischen Frequenzen.
WWZ_Vergleich_mehrere_Linien.py	WWZ-Spektrogramme mehrerer Linien nebeneinander, als Differenz, als Kontur-Overlay und als Kreuz-WWZ mit Phasendifferenz und Zeitversatz.
WWZ_Linienvergleich.py	Überlagerte mittlere Periodogramme, Spektrogramm-Galerie, Differenz- und Ratio-Karten, dominante Perioden über die Zeit.
WWZ_Aehnlichkeitsanalyse.py	Ähnlichkeit von WWZ-Spektrogrammen in vier Dimensionen: Perioden, zeitliche Modulation, Phasenkohärenz, Gesamtmuster.
WWZ_Korrelations_Nachbarn.py	Alle Linien, deren Spektrogramm einer Ankerlinie ähnelt (SSIM oder Pearson), dazu PCA/UMAP-Karte aller Linien.
WWZ_Liniengruppierung.py	Hierarchische Gruppierung aller Spektrogramme mit automatischer Wahl der Gruppenzahl und Repräsentant je Gruppe.

11.8 Kausalität

Skript	Möglichkeiten
GrangerKausalitaet.py	Multivariate Granger-Kausalität (VAR-Modell, Lag-Ordnung nach AIC/BIC) zwischen Linienparametern.
TransferEntropie.py	Symbolische Transferentropie zwischen RV-Reihen mit Surrogat-Test; Netto-Informationsfluss und Kausalitätsmatrix (z. B. Fe II → Hβ → Hα).
ConditionalTransferEntropie.py	Bedingte Transferentropie (k-NN-Schätzer): trennt direkte von indirekten Abhängigkeiten.

11.9 Nichtlineare Dynamik

Skript	Möglichkeiten
PhasenraumRekonstruktion.py	Time-Delay-Embedding nach Takens: τ aus der Mutual Information, Einbettungsdimension aus False Nearest Neighbours, größter Lyapunov-Exponent (Rosenstein), Korrelationsdimension D_2, Rekurrenzplot, 2D/3D-Darstellung; drei Varianten für unregelmäßige Abtastung.
attractor_autoencoder_interactive.py	Attraktorrekonstruktion mit Autoencodern; latente Koordinaten stehen anschließend im Variable Explorer zur Verfügung.
lyapunov_analysis.py / NichtlineareDynamik.py	Lyapunov-Exponenten bzw. Lyapunov-Spektrum, Kaplan-Yorke-Dimension und Permutationsentropie (global und gleitend).
multifractal_analysis.py	Multifraktale Dimensionsspektren.
DMD_Analyse.py	Dynamic Mode Decomposition: kohärente Moden mehrerer Messgrößen mit Frequenzen, Wachstumsraten und Zeitverläufen.
RC_und_Distanzmatrix_werteraum.py (mit rqa_kenngroessen.py)	Distanzmatrix und Rekurrenzplot im Werteraum, RQA-Kenngrößen (RR, DET, LAM, ENTR, TT u. a.) statisch und gleitend; Einzeldatei oder Batch.
Phasenraum_Distanzmatrix_Analyse.py	Distanzmatrix und RQA im rekonstruierten Phasenraum, Perioden aus den Diagonalen mit Signifikanz- und Aliasprüfung, Lebensdauer von Perioden; Bericht als RTF.
Werte_Distanzmatrix_analyse.py	Umfassende Auswertung von Distanzmatrizen: Statistik, zeitliche Segmentierung, Regimewechsel, Zustandsclustering.
TSA_Analyse_mehrereMethoden.py	Distanzmatrix, Autokorrelation, Lomb-Scargle, Wavelet, Rekurrenzplot und FFT in einem Lauf – schneller Erstüberblick.
Multi_File_Cross_RQA.py	Cross-Rekurrenzanalyse derselben Größe in mehreren Linien (z. B. Hα, Hβ, Hγ).

11.10 Statistik

Skript	Möglichkeiten
histogram_analysis.py	Histogramme und Statistik der Messwerte oder der Beobachtungsabstände ΔJD (Abtastung).
Statistik_eineVariable_allerLinien.py	Statistik einer Variablen über alle Dateien eines Verzeichnisses in einer Tabelle.
Extrema_Statistik.py	Lokale Extrema bestimmen, interaktiv bereinigen, Histogramme und ECDF.
ChangepointAnalyse_SiII.py	Changepoint-Erkennung (PELT) in den RV-Reihen von Si II 6347/6371 mit Konsistenzvergleich; auf andere Linienpaare übertragbar.

11.11 Systematische Analysekette

Die fünf Skripte in 50_Systematische_Analyse wenden auf dieselben Daten nacheinander verschiedene Methodenfamilien ohne Regularisierung an und rechnen parallel. Sie eignen sich, um ein Ergebnis mit unabhängigen Verfahren abzusichern.

Skript	Möglichkeiten
01_Periodizitaetsanalyse.py	GLS, WWZ, PDM, CLEAN, String Length, AoV.
02_Kreuzkorrelation_Lag.py	DCF, ZDCF, Kreuz-WWZ, optional JAVELIN; Lag-Fehler per Monte-Carlo.
03_Nichtlinear_Recurrence.py	RQA, Cross- und Joint-Recurrence-Plots, Convergent Cross Mapping, Mutual Information.
04_Statistische_Abhaengigkeit.py	Distanzkorrelation, Transferentropie, Copula-Analyse.
05_Modellbasiert_GP_CARMA.py	Gaussian Process und CARMA-Spektrum (celerite2), Sparse Bayesian Learning im Frequenzraum.

12 Typische Arbeitsabläufe

Die folgenden Abläufe zeigen, wie die Skripte zusammenwirken. Sie sind Vorschläge; einzelne Schritte können je nach Datenlage entfallen.

12.1 Eine Beobachtungsnacht mit dem Echelle-Spektrographen reduzieren

1. Vorab: astroplanner_20260924.py für Objektauswahl und Zeitfenster.
2. Während der Nacht optional Live_Reduktion.py – jede neue Aufnahme wird sofort reduziert und kann beurteilt werden.
3. Nach der Nacht: MastersErzeugen_CCD_2d_fits.py für Master-Flat, Master-ThAr und Darks; Sichtprüfung mit FITS_2D_Viewer.py.
4. Echelle_Extraktion.py → ThAr_Kalibrierung.py → Normierung_Ordnungen.py → Mergen_Ordnungen.py.
5. Bei Auffälligkeiten in der Kalibrierung: ThAr_Linienzaehler.py und Diagnose_Linienausbeute.py.
6. JD_Header_Pruefung.py und Zeitserie_Uebersicht.py, anschließend Einordnung in die Serie.

12.2 Einen neuen Echelle-Spektrographen einbinden

1. Master-Flat und Master-ThAr des Instruments erzeugen, Ordnungen mit Echelle_Extraktion.py extrahieren (legt die Ordnungsreferenz an).
2. Mit Echelle_instrumentenspezifischen_Linienkatalog_Erstellen.py aus zwei bis drei Ankerlinien die Identifikationstabelle erzeugen.
3. Ergebnis mit ThAr_Linienlkatalog_Viewer.py und ThAr_Linienzaehler.py prüfen und als geprüfte Version ablegen.
4. Ab dann normale Reduktion mit ThAr_Kalibrierung.py.

12.3 Von der Spektrenserie zur Linien-Zeitreihe

1. Serie prüfen: fits_header_pruefen_korrigieren.py, JD_Header_Pruefung.py, spektren_qualitaetskontrolle.py, cosmics_entfernen.py.
2. Wellenlängen-Nullpunkt an tellurischen Linien korrigieren (terrLinien_RVKorrektur_*), dann baryzentrisch korrigieren (Baryzentrische_Korrektur_Spektren.py) und tellurische Linien entfernen (telluric_trocknung_hitran.py bzw. telluric_bc_korrektur.py).
3. Linienumfeld normieren: Normierung_einzelne_Linien_20260923.py, Stützpunkte vorher mit stuetzpunkt_validator_einzelneLinien_20260923.py prüfen.
4. Messen: Zeitserie_EW_Linientiefe_Linienwellenlaenge_FWHM_mehrereModelle.py, Bisektor_RV_Zeitreihe.py oder CrossCorrelation_fits_Serie_einzelneLinien_v2.py.
5. Sichtkontrolle: Dynamisches_Spektrum.py und fitsSerie_Linie_imGeschwindigkeitsraum_JDBereich_v2.py.

12.4 Perioden und Schichtstruktur einer pulsierenden Atmosphäre untersuchen

1. Messreihen bereinigen und glätten: AusreisserAnalyse.py, MedianLoessFilter.py; Abtastung mit histogram_analysis.py (Modus ΔJD) beurteilen.
2. Methoden vorab an synthSignal_GUI.py-Daten mit derselben Abtastung testen.

3. Perioden: GLS.py und CLEAN_Periodenanalyse.py, Bestätigung mit PDM_Periodenanalyse.py; zeitliche Veränderlichkeit mit wwz.py, gleitendesGLS_Spektrogramm.py und Epochen_Screening.py.
4. Linien vergleichen: WWZ_Vergleich_mehrere_Linien.py und WWZ_Liniengruppierung.py; Lags mit Lag_Alle_Linienpaare_CCF.py → Lag_Ranking.py → EW_APX_vs_Lag.py / Eup_Elow_vs_Lag.py; Verifikation mit DCF_Kreuzkorrelation_Verifikation.py.
5. Gerichtete Abhängigkeiten: TransferEntropie.py, ConditionalTransferEntropie.py, GrangerKausalitaet.py.
6. Physikalische Größen: Boltzmann_Temperatur_Fitter.py, Boltzmann_Saha_Analyse.py, LDR_Batch_Analyse_aus_fits.py, Makroturbulenz_Fourier_Methode_batch.py.
7. Absicherung: Skripte 01–05 der systematischen Analysekette.

12.5 Bahn eines spektroskopischen Doppelsterns bestimmen

1. RV messen: CrossCorrelation_fits_Serie_v2.py (Template) oder RV_Messung_eineLinieInEinerfitsSerie_Spline.py, jeweils baryzentrisch korrigiert.
2. Periode suchen: Periodenanalyse.py; Ergebnis mit Phasenplot_bekannte_Periode_20260924.py prüfen.
3. Orbitelemente: Orbitelemente_ausJDundGemessenenRV_optimieren.py; Kontrolle und Planung weiterer Beobachtungen mit RV_Berechnung_ausOrbitalelementenUndJD.py und Phasenberechnung_ausSpektren_mitvorgegebenenToUndPeriode.py (Phasenlücken).
4. Optional Komponenten trennen: SummenUndDifferenzspektrum_…py und UmrechnungSpektrumVonBaryzentrischAufSternBezogen.py.

12.6 Emissionslinien von Be-Sternen verfolgen und an BeSS melden

1. Hα-Profil fitten: FitLinie_PositiverUndNegativerGauss_Version2.py oder die Voigt-Fassung (EW, V/R, Peakabstand).
2. Getrennte EW von Absorption und Emission: EW_Absorption_Emission.py.
3. Darstellung: Dynamisches_Spektrum.py, timeseries_Spektren.py.
4. Für die Übermittlung: BESS_fits_header_Serie.py ergänzt die Header.

12.7 Robustheit eines Ergebnisses prüfen

- Spektren mit verrauschen.py auf niedrigeres S/N bringen und die Messung wiederholen.
- Zeitreihen mit synthSignal_GUI.py nachbilden und die Analyse auf bekannte Signale anwenden.
- Einen Lag oder eine Periode mit einer zweiten, unabhängigen Methode bestätigen (DCF, Bandpass-KKF, Sliding SHO, systematische Kette).

13 Konventionen, Formate und Abhängigkeiten

13.1 Datenformate

- 1D-Spektren: FITS mit linearer Wellenlängenskala (CRVAL1, CDELT1 bzw. CD1_1, CRPIX1); viele Skripte lesen auch log-lineare Skalen (DC-FLAG = 1) und Tabellen-FITS.
- Zeitpunkt: Keyword JD bevorzugt; sonst werden JD-OBS, JD-MID, BJD, HJD, MJD-OBS oder DATE-OBS ausgewertet. JD_Header_Pruefung.py vereinheitlicht dies.
- ASCII-Spektren: zwei Spalten WAVE und FLUX (.dat mit Tab, .csv mit Komma).
- Ergebnistabellen: CSV mit Semikolon als Trenner und Komma als Dezimalzeichen; einige ältere Linienvergleichs-Skripte verwenden Komma und Dezimalpunkt. Eingaben werden meist automatisch erkannt.
- Grafiken: PNG mit 600 dpi und weißem Hintergrund, teilweise zusätzlich PDF.

13.2 Header-Einträge der Korrekturskripte

Keyword	Bedeutung
RVMEAS / RVERR	Gemessene RV und Fehler [km/s], positiv = Rotverschiebung.
RVSHIFT / RVSHTOT	Angewandte Verschiebung bzw. Summe aller bisherigen Verschiebungen [km/s].
RVMETH	Verfahren: MANUELL, TABELLE, CCF, TELL_RBF, TELL_POLY.
VSINI, LIMBDARK, CONV_R	Parameter der Rotations- und Gaußfaltung.
ROTANG, ROTMETH	Drehwinkel und Verfahren beim Ausrichten des Spektrumstreifens.
ECHORDER, ORDER_M, CAL_MODE	Ordnungsnummer und Kalibrierungsart in der Echelle-Pipeline.
HISTORY	Protokoll jedes Bearbeitungsschritts.

13.3 Benötigte Pakete

Grundausstattung (von fast allen Skripten genutzt): numpy, scipy, matplotlib, astropy, pandas, PyQt5.

Zusätzlich je nach Skript:

Bereich	Pakete
Fit und Statistik	lmfit, statsmodels, scikit-learn, emcee, corner, ruptures, pyvinecopulib
Astronomie	PyAstronomy, astroquery, astroplan, barycorrpy, hapi (HITRAN), keyring (ESO-Login)
Zeitreihen	celerite2, celerite, PyEMD (EMD-signal), ssqueezepy, PyWavelets, javelin (optional)
Sonstiges	joblib, tqdm, networkx, umap-learn, scikit-image, tensorflow (Autoencoder), imageio/imageio-ffmpeg, Pillow, requests; cupy optional für GPU

13.4 Gemeinsame Module

Modul	Inhalt
Bausteine/Linienlisten.py	Liniendatenbank mit Atomdaten; Linienauswahl-Dialoge.
Bausteine/RV_Werkzeuge.py	Einlesen, Dopplerverschiebung, Header, Protokoll, Messung tellurischer Linien für die RV-Skripte.
Bausteine/Observatorien.py	Koordinaten der Beobachtungsorte; neue Standorte hier eintragen.
Reduce/ klassischerSpektrograph/ spektro_kern.py, spektro_ablauf.py	Rechenfunktionen und Ablauf der Spaltspektrograph-Reduktion.
TSA/11_Zeit-Frequenz/WWZ/ wwz_kern.py	Rechenkern der WWZ, auch von anderen Skripten importierbar.
TSA/30_.../Werteraum/ rqa_kenngroessen.py	RQA-Kenngrößen.

Anhang A – Skriptindex

Alle 192 Python-Dateien der Sammlung alphabetisch, mit Ordner und dem Kapitel, in dem sie beschrieben sind.

Skript	Ordner	Kapitel
01_Periodizitaetsanalyse.py	TSA/50_Systematische_Analyse	11
02_Kreuzkorrelation_Lag.py	TSA/50_Systematische_Analyse	11
03_Nichtlinear_Recurrence.py	TSA/50_Systematische_Analyse	11
04_Statistische_Abhaengigkeit.py	TSA/50_Systematische_Analyse	11
05_Modellbasiert_GP_CARMA.py	TSA/50_Systematische_Analyse	11
1d_dat_read_plot_csv_speichern.py	ascii	8
1d_fitSpektrum_ansehen_mitLinienidentifikation_MitSubplot.py	Spektrenplots	8
1d_fitSpektrum_ansehen.py	Spektrenplots	8
1d_txt_plotten.py	ascii	8
2._Ausrichten des Spektrumstreifens_CCD_Aufnahmen_fits.py	Reduce/klassischerSpektrograph	4
3._ExtraktionUndWellenlaengenkalibrationNormierung_mitSpalt_CCD_Aufnahmen_mitKalibrierlinienoptimierung.py	Reduce/klassischerSpektrograph	4
3a_ExtraktionUndWellenlaengenkalibrationNormierung_spaltlos_CCD_Aufnahmen.py	Reduce/klassischerSpektrograph	4
aavso_lightcurve_analysis.py	Lichtkurven	10
AKF.py	TSA/10_Periodenanalyse/AKF	11
amplituden_tracking.py	TSA/10_Periodenanalyse/PDM	11
AnimationAusPlots.py	Spektrenplots	8
Anregungstemperatur.py	TSA/60_Spektrenanalyse/Temperatur	7
astroplanner_20260924.py	astroplanner	3
attractor_autoencoder_interactive.py	TSA/30_Nichtlineare_Dynamik/Phasenraum/attractor_encoder	11
AusreisserAnalyse.py	TSA/01_Datenvorbereitung	11
Baryzentrische_Korrektur_Spektren.py	BC	6
BESS_fits_header_Serie.py	Serien/BESS	5
Bisektor_Analyse.py	TSA/60_Spektrenanalyse/Bisektor	7
Bisektor_CCF_Tiefen.py	TSA/60_Spektrenanalyse/Bisektor	7
Bisektor_RV_Zeitreihe.py	LinienparameterBestimmen	7
Boltzmann_Saha_Analyse.py	TSA/60_Spektrenanalyse/Temperatur	7
Boltzmann_Temperatur_Fitter.py	TSA/60_Spektrenanalyse/Temperatur	7
ChangepointAnalyse_SiII.py	TSA/40_Statistik	11
CLEAN_Periodenanalyse.py	TSA/10_Periodenanalyse/CLEAN	11
comparison_plot_2spectra_ascii.py	ascii	8
ConditionalTransferEntropie.py	TSA/21_Kausalitaet	11
convol_dat.py	Convolution	9

Skript	Ordner	Kapitel
convol_fits.py	Convolution	9
cosmics_entfernen.py	Serien	5
Crop_Ausschnitt_Zeitserie.py	Serien	6
CrossCorrelation_fits_Serie_einzelneLinien_v2.py	KK	7
CrossCorrelation_fits_Serie_v2.py	KK	7
DCF_Kreuzkorrelation_Verifikation.py	TSA/20_Linienvergleich/Kreuzkorrelation	11
Diagnose_Linienausbeute.py	Reduce/Echelle	4
DifferenzflaechenPlot.py	TSA/60_Spektrenanalyse/Linienasymmetrie	7
Differenzspektren_mitPlotAlle_v2.py	Rebinning_gemitteltesSpektrum_Differenzspektren	6
Differenzspektren_v2.py	Rebinning_gemitteltesSpektrum_Differenzspektren	6
DMD_Analyse.py	TSA/30_Nichtlineare_Dynamik/DMD	11
Dynamisches_Spektrum.py	Spektrenplots	8
Echelle_Extraktion.py	Reduce/Echelle	4
Echelle_instrumentenspezifischen_Linienkatalog_Erstellen.py	Reduce/Echelle	4
eemd_analysis_final.py	TSA/11_Zeit-Frequenz/EMD	11
EinflussfaktorAnalyse.py	TSA/20_Linienvergleich/Einflussfaktoren	11
Epochen_Screening.py	TSA/10_Periodenanalyse	11
ESO_Abfrage.py	Katalogabfragen	3
Eup_Elow_vs_Lag.py	TSA/20_Linienvergleich/Lag	11
EW_Absorption_Emission.py	EW	7
EW_APX_vs_Lag.py	TSA/20_Linienvergleich/Lag	11
Extrema_Statistik.py	TSA/40_Statistik	11
fit_Serie_in_csv_und_dat.py	Serien	5
FitLinie_automatischParametrisiert.py	Linienfitting	7
FitLinie_PositiverUndNegativerGauss_Version2.py	Linienfitting	7
FitLinie_PositiverUndNegativerVoigt_Version2.py	Linienfitting	7
FitLinie.py	Linienfitting	7
FITS_2D_Viewer.py	Reduce	4
fits_header_pruefen_korrigieren.py	Header	5
fits_zu_png.py	Spektrenplots	8
fitsSerie_Linie_imGeschwindigkeitsraum_JDBereich_v2.py	DarstellungImGeschwindigkeitsraum	8
fitsSerie_Linien_imGeschwindigkeitsraum_JDBereich_v2.py	DarstellungImGeschwindigkeitsraum	8
FluxBeiOffset_Linie.py	TSA/60_Spektrenanalyse/Linienasymmetrie	7
Formatumwandlungen.py	Formatumwandlungen	3
FWHM_einerLinie_Serie_fits.py	Linienfitting	7
gleitendesGLS_Spektrogramm.py	TSA/11_Zeit-Frequenz/gleitendes_GLS	11

Skript	Ordner	Kapitel
GLS_Linienvergleich.py	TSA/20_Linienvergleich/GLS_Vergleich	11
GLS.py	TSA/10_Periodenanalyse/GLS	11
GPR_Zeitreihenanalyse.py	TSA/12_Rauschen_Modelle/GPR	11
GrangerKausalitaet.py	TSA/21_Kausalitaet	11
Halpha_Doppelgauss_Fit.py	Linienfitting	7
Halpha_ProfilanpassungVoigt_Gauss.py	Linienfitting	7
HalphaVariabilitaet.py	TSA/60_Spektrenanalyse/Halpha	7
Headerprint_Serie_csv.py	Header	5
heliocentricCorrection_NRES.py	NRES	4
histogram_analysis.py	TSA/40_Statistik	11
JD_Header_Pruefung.py	Header	5
JD_Wellenlaenge_in_Dateiname.py	Serien	5
Korrelations_Analyse.py	TSA/20_Linienvergleich/Korrelation_Regression	11
Kreuzkorrelation_Bandpass_Verifikation.py	TSA/20_Linienvergleich/Kreuzkorrelation	11
Kreuzkorrelation_Messwerte.py	KK	7
KunstspektrumErzeugen.py	theoretischeSpektren	9
Lag_Alle_Linienpaare_CCF.py	TSA/20_Linienvergleich/Lag	11
Lag_Analyse_Linienpaar.py	TSA/20_Linienvergleich/Lag	11
Lag_Ranking.py	TSA/20_Linienvergleich/Lag	11
Lag_Referenzlinie_Extraktion.py	TSA/20_Linienvergleich/Lag	11
LDR_Batch_Analyse_aus_fits.py	TSA/60_Spektrenanalyse/Temperatur	7
LDR_Kalibration_Heisse_Sterne.py	TSA/60_Spektrenanalyse/Temperatur	7
Linienflaeche_Geschwindigkeitsbereich.py	LinienparameterBestimmen	7
Linienlisten.py	Bausteine	13
Linienparameter_Vergleich.py	TSA/20_Linienvergleich/Linienparameter	11
Linienvergleich_Clustering.py	TSA/20_Linienvergleich/Linienparameter	11
LinienwellenlängenNIST.py	Katalogabfragen	3
Live_Reduktion.py	Reduce/Echelle/live_Reduktion	4
LOESS_Vergleich_Plots_batch.py	TSA/02_Darstellung	11
LSSA_Kreuzspektrum.py	TSA/20_Linienvergleich/Kreuzspektrum	11
lyapunov_analysis.py	TSA/30_Nichtlineare_Dynamik/Lyapunov	11
Makroturbulenz_Fourier_Methode_batch.py	Mikro_Makroturbulenz	7
Makroturbulenz_Fourier_Methode.py	Mikro_Makroturbulenz	7
MastersErzeugen_CCD_2d_fits.py	Reduce	4
MedianLoessFilter.py	TSA/01_Datenvorbereitung	11
Mergen_Ordnungen.py	Reduce/Echelle	4
Messgroessen_Linienvergleich.py	TSA/20_Linienvergleich/Linienparameter	11
Messwerte_Vergleich.py	TSA/02_Darstellung	11
Messwerteplot.py	TSA/02_Darstellung	11
MittelungVon_fits.py	Serien	6
Multi_File_Cross_RQA.py	TSA/30_Nichtlineare_Dynamik/Rekurrenz/Cross_RQA	11

Skript	Ordner	Kapitel
multifractal_analysis.py	TSA/30_Nichtlineare_Dynamik/Multifraktal	11
newbinning_mittleresSpektrum_v2.py	Rebinning_gemitteltesSpektrum_Differenzspektren	6
NichtlineareDynamik.py	TSA/30_Nichtlineare_Dynamik/Lyapunov	11
Normierung_einzelne_Linien_20260923.py	Normierung	6
Normierung_Ordnungen.py	Reduce/Echelle	4
NRES_auslesen.py	NRES	4
Observatorien.py	Bausteine	13
Orbitelemente_ausJDundGemessenenRV_optimieren.py	binaries_Periodenbestimmung_Orbitalelemente	10
parse_stats_to_csv.py	TSA/01_Datenvorbereitung	11
PDM_Periodenanalyse.py	TSA/10_Periodenanalyse/PDM	11
Periodenanalyse.py	binaries_Periodenbestimmung_Orbitalelemente	10
Phasenberechnung_ausSpektren_mitvorgegebenenToUndPeriode.py	binaries_Periodenbestimmung_Orbitalelemente	10
Phasenberechnung.py	binaries_Periodenbestimmung_Orbitalelemente	10
Phasenfaltung_eine_Linie.py	TSA/10_Periodenanalyse/Phasenfaltung	11
Phasenplot_bekannte_Periode_20260924.py	binaries_Periodenbestimmung_Orbitalelemente	10
Phasenraum_Distanzmatrix_Analyse.py	TSA/30_Nichtlineare_Dynamik/Rekurrenz/Phasenraum	11
PhasenraumRekonstruktion.py	TSA/30_Nichtlineare_Dynamik/Phasenraum	11
PlotAllerOrdnungen.py	NRES	4
PlotAndCrop_1d_ascii.py	ascii	6
PlotsAusSpalten_batch.py	TSA/02_Darstellung	11
PlotsAusSpaltenVon_Tabellen.py	Plots	8
Pollux_Ausschnitt_convol.py	Convolution	9
PSD_plot.py	TSA/12_Rauschen_Modelle	11
RC_und_Distanzmatrix_werteraum.py	TSA/30_Nichtlineare_Dynamik/Rekurrenz/Werteraum	11
Reduce_GUI.py	Reduce/klassischerSpektrograph	4
Regressions_Analyse.py	TSA/20_Linienvergleich/Korrelation_Regression	11
rotationsverbreiterung_fits.py	Convolution	9
rqa_kenngroessen.py	TSA/30_Nichtlineare_Dynamik/Rekurrenz/Werteraum	11
RV_Berechnung_ausOrbitalelementenUndJD.py	binaries_Periodenbestimmung_Orbitalelemente	10
RV_Messung_eineLinieInEinerfitsSerie_Spline.py	RV	7
RV_Werkzeuge.py	Bausteine	13
RVKorrektur_1d_spectra_fits_series.py	RV	6
RVKorrekturPerTemplate_1d_spectra_fits_series.py	RV	6
Simbadabfrage_a.py	Katalogabfragen	3
SinusfitPeriode.py	TSA/10_Periodenanalyse/Sinusfit	11
Sliding_SHO_celerite2.py	TSA/11_Zeit-Frequenz/Sliding_SHO	11
SNR_Bestimmung.py	SNR	5

Skript	Ordner	Kapitel
Spectral_line_asymmetry_interactive.py	TSA/60_Spektrenanalyse/Linienasymmetrie	7
spektren_diagnose.py	Serien	5
spektren_qualitaetskontrolle.py	Serien	5
spektren_stapelplot.py	ascii	8
Spektren_Vergleich.py	Spektrenplots	8
SpektrenDifferenzZumReferenzspektrum_Linie.py	TSA/60_Spektrenanalyse/Linienasymmetrie	7
spektro_ablauf.py	Reduce/klassischerSpektrograph	4
spektro_kern.py	Reduce/klassischerSpektrograph	4
Spektrum_GaussBroadened.py	theoretischeSpektren	9
Statistik_eineVariable_allerLinien.py	TSA/40_Statistik	11
Strukturfunktion.py	TSA/12_Rauschen_Modelle	11
stuetzpunkt_validator_einzelneLinien_20260923.py	Normierung	6
SummenUndDifferenzspektrum_gemessenMinustheoretisches_JePhase_mitIntegralkorrektur.py	binaries_Periodenbestimmung_Orbitalelemente	10
Synchrosqueezing_Analyse.py	TSA/11_Zeit-Frequenz/Synchrosqueezing	11
synthSignal_GUI.py	TSA/03_Testdaten	11
telluric_bc_korrektur.py	BC	6
telluric_trocknung_hitran.py	Trocknung	6
terrLinien_RVKorrektur_RBF_1d_spectra_fits_series.py	RV	6
terrLinien_RVKorrektur_Regression_1d_spectra_fits_series.py	RV	6
ThAr_Aufnahme_Zeigen.py	Reduce/Echelle	4
ThAr_Kalibrierung.py	Reduce/Echelle	4
ThAr_Linienlkatalog_Viewer.py	Reduce/Echelle	4
ThAr_Linienzaehler.py	Reduce/Echelle	4
timeseries_Spektren.py	Spektrenplots	8
TransferEntropie.py	TSA/21_Kausalitaet	11
TSA_Analyse_mehrereMethoden.py	TSA/30_Nichtlineare_Dynamik/Rekurrenz	11
tsa_prewhitener_auto_white.py	TSA/10_Periodenanalyse/Prewhitening	11
UmrechnungSpektrumVonBaryzentrischAufSternBezogen.py	binaries_Periodenbestimmung_Orbitalelemente	10
Universelle_Normierung.py	Normierung	6
VacToAir.py	NRES	4
verrauschen.py	SNR	6
WaveletCrossBispektrum.py	TSA/20_Linienvergleich/Kreuzspektrum	11
Werte_Distanzmatrix_analyse.py	TSA/30_Nichtlineare_Dynamik/Rekurrenz/Werteraum	11
WWZ_Aehnlichkeitsanalyse.py	TSA/20_Linienvergleich/WWZ_Vergleich	11
wwz_kern.py	TSA/11_Zeit-Frequenz/WWZ	11
WWZ_Korrelations_Nachbarn.py	TSA/20_Linienvergleich/WWZ_Vergleich	11

Skript	Ordner	Kapitel
WWZ_Liniengruppierung.py	TSA/20_Linienvergleich/WWZ_Vergleich	11
WWZ_Linienvergleich.py	TSA/20_Linienvergleich/WWZ_Vergleich	11
WWZ_Vergleich_mehrere_Linien.py	TSA/20_Linienvergleich/WWZ_Vergleich	11
wwz.py	TSA/11_Zeit-Frequenz/WWZ	11
ZeitdifferenzenExtremea_zweierLinien.py	TSA/20_Linienvergleich/Extrema_Zeitdifferenzen	11
zeitraum_binning_und_interpolation.py	TSA/01_Datenvorbereitung	11
Zeitserie_EW_Linientiefe_Linienwellenlaenge_FWHM_mehrereModelle.py	LinienparameterBestimmen	7
Zeitserie_EWs_mehrereBereicheInEinerLinie.py	LinienparameterBestimmen	7
Zeitserie_Spektren_smoothen.py	Serien	6
Zeitserie_Uebersicht.py	Serien	5