Aufzählung und und Beschreibung der vorhandenen Pythonskripte

von Dr. Lothar Schanne

June 15, 2023

Um die Skripte verwenden zu können müssen die Python-Module in der verwendeten Pythondistribution installiert sein, die am Anfang der Skripte mit den import-Befehlen importiert werden.

Man kann die Skripte auf verschiedenen Wegen starten:

Beim einfachsten geht man in eine Konsole, wechselt in das Verzeichnis, in dem die Spektrumdateien (fits) enthalten sind, die man bearbeiten will und startet das jeweilige Skript mit dem Befehl:

python3 Pfad/zu/dem/Skript/Skriptname.py

Eine andere Methode ist der Start eines Skripts innerhalb einer Entwicklungsumgebung. Ich bevorzuge dafür Spyder, das den Vorteil bietet, dass neben einer Konsole, in der das Skript abgearbeitet wird auch ein Fenster mit dem Pythoncode des Skripts existiert (so dass man schnell z.B. Parameter anpassen oder einzelne Zeilen auskommentieren oder ändern kann). Außerdem werden im Programmablauf erzeugte Variabeln im Variablenmanagerfenster angezeigt. Das kann hilfreich sein, wenn etwas schiefgeht und man z.B. sehen kann, bei welcher Spektrumdatei das Programm unterbrochen wurde, weil dort ein header-Eintrag fehlt. Und man sieht im Dateienfenster die Dateien, welche sich im Arbeitsverzeichnis aktuell befinden (inklusive derjenigen, die zwischenzeitlich vom laufenden Skript erzeugt wurden). Außerdem kann man in Spyder das Arbeitsverzeichnis einstellen und in die Umgebungsvariable PYTHONPATH die Pfade zu den Skripten aufnehmen, so dass die Skripte automatisch in diesen Pfadverzeichnissen gesucht werden.

Bei den meisten Skripten, welche Grafikfenster (z.B. Abbildungen von Spektren) erzeugen, wird am Anfang des Skripts auf ein interaktives backend (QT5Agg) umgeschaltet. D.h. die geöffneten Grafikfenster können aktiv verändert werden, wenn der Fokus auf ein Fenster gelegt wurde. Also kann das Fenster vergrößert oder verkleinert werden, oder von dem dargestellten Inhalt (das Spektrum) kann ein Ausschnitt ausgewählt werden, der Ausschnitt kann verschoben werden, die Grafik kann gespeichert werden etc. So kann man sich Details eines Spektrums genauer anschauen.

Ordner ascii

1d_dat_read_plot_csv_speichern.py

Das Skript liest eine ASCII-Datei und plottet das Spektrum. Die ASCII-Datei hat 2 Spalten mit den Spaltenüberschriften 'WAVE' und 'FLUX'. Die Erweiterung der Datei ist .dat. Das Trennzeichen wird automatisch mit guess=True ermittelt. Plotten und Speichern als csv ohne Spaltenüberschriften (Skriptzeilen 40 und 41).

1d_txt_plotten.py

Das Skript liest ein Spektrum, das in einer ASCII-Datei gespeichert ist. Die Daten sind in 2 Spalten mit Float-Zahlen (Wellenlänge und Fluss) ohne Überschrift. Pfad/Name werden abgefragt. Das Spektrum wird geplottet.

comparison_plot_2spectra_ascii.py

Das Skript ist gedacht um ein gemessenes Spektrum und ein Template (z.B. ein theoretisches Spektrum) übereinander zu plotten und so zu vergleichen.

Es liest das Template und das gemessene Spektrum ein (Eingabe von Pfad/Name). Die 1d-Spektren müssen als tab-separierte ASCII-Tabellen vorliegen mit den Spaltenüberschriften 'WAVE' und 'FLUX'. Der Plot wird als PDF abgespeichert (Zeile 38).

PlotAndCrop_1d_ascii.py

Das Skript hat die Aufgabe, ein im ASCII-Format vorliegendes 1d-Spektrum mit den beiden Spalten 'WAVE' und 'FLUX' zu plotten. Danach kann eine Wellenlängenbereich eingegeben werden, auf den das Spektrum reduziert wird (croppen). Dieser Ausschnitt wird dann geplottet und als ASCII-Datei gespeichert, wobei im Namen der Datei der Wellenlängenbereich genannt wird.

Ordner astroplanner

astroplanner.py

Berechnet die Höhe des Objektes über dem Horizont für einen Zeitpunkt an einem Beobachtungsort und plottet sie über einen Zeitraum von 24 Stunden. Eingabe des Sterns, des Beobachters und des Datums nötig.

astroplanner_fuer_Objektliste.py

Berechnet die Höhen der Objekte einer Objektliste (ascii-Datei (csv) mit den Objektnamen in einer Spalte) über dem Horizont für einen geolanten Zeitpunkt an einem Beobachtungsort und plottet sie über einen Zeitraum von 24 Stunden. Die Grafiken werden als pdf im Arbeitsverzeichnis gespeichert. Eingabe der Liste der Sterne, des Beobachters und des Datums/Zeitpunkts nötig.

Ordner BC

BC_1d_spectra_fits_series.py

Das Skript berechnet die baryzentrische Korrektur BC und das korrigierte JD als HJD einer Serie von 1d-Spektren im Fits-Format eines Objekts. Schreibt eine ascii-Datei (Format-Tab) mit den BC's und HJD's in das Arbeitsverzeichnis. Korrigiert auf Wunsch die eingelesenen Spektren um die berechnete baryzentrische Korrektur BC und speichert sie als fits ab (und nach auskommentieren der Zeilen 206 bis 213 auch als asci-Tabelle).

Vorab müssen im Skript die Koordinaten des Beobachters und des Objekts eingegeben oder auskommentiert/einkommentiert werden. Standardmäßig werden die Koordinaten des Objekts aus dem Internet eingelesen (Internetanschluß nötig), können aber auch manuell im Skript eingetragen werden (Zeilenbereich 71 bis 107).

Ordner Convolution

convol_dat.py

Das Skript wird verwendet, um ein hoch-aufgelöstes (meist ein theoretisches) 1d-Spektrum mit einer Standardabweichung (bestimmt aus dem Apparateprofil des Spektrographen) so zu verbreitern, dass die Linienbreiten dem gemessen Spektren entsprechen.

Einlesen eines 1d-Spektrums in Form einer ASCII-Tabelle mit der Extension .dat, 2-spaltig mit den Spaltenüberschriften 'WAVE' und 'FLUX'. Das Spektrum wird mit einer Standardabweichung convolviert (stddev in der Einheit des step). Der convolvierte Flux wird zusammen mit der Wellenlänge in einer zweispaltigen ASCII-Tabelle abgespeichert (spacer = tab). Mit den Spaltenüberschriften 'WAVE' und 'FLUX'. Außerdem werden die beiden Spektren geplottet und die Plots als PDF und PNG gespeichert.

convol_fits.py

Einlesen eines 1d-Spektrums in Form einer fits-Datei.

Das Spektrum kann mit einer Standardabweichung, ausgedrückt durch die Auflösung R, convolviert (gefaltet) werden. Die gefaltete fits-Datei wird abgespeichert, wobei das R im Namen der Datei genannt wird.

Pollux_Ausschnitt_convol.py

Einlesen eines synthetischen Spektrums in Form einer Tabelle (wie als .spec in der Pollux-Datenbank erhältlich, `http://pollux.graal.univ-montp2.fr/`). In dem begleitenden Pollux-file Spektrum.txt sind Startwellenlänge und step angegeben. Berechnung eines wählbaren Ausschnitts (Pandas Dataframe mit 'bereich' bezeichnet). Geplottet wird die Spalte 'NFLUX' = normierter Flux für das gesamte Spektrum und für bereich. Der Bereich kann mit einer Standardabweichung gefaltet werden (stddev in der Einheit des step). Der convolvierte Flux wird zusammen mit der Wellenlänge in einer zweispaltigen ASCII-Tabelle abgespeichert (spacer = tab, Spaltennamen WAVE und FLUX).

rotationsverbreiterung_fits.py

Einlesen einer Serie von 1d-Spektren im fits-Format (gedacht für synthetische Spektren zum Einrechnen der Rotationsverbreiterung und des limb-darkening-Effekts)).

Durch Eingabe des limb-darkening-Koeffizients und der Rotationsgeschwindigkeit werden die Spektren rotationsverbreitert. Die convolvierten fits-Dateien werden abgespeichert, wobei die Dateinamen mit dem Begriff 'rotverbreitert' ergänzt werden.

Ordner DarstellungGeschwindigkeitsraum

fitsSerie_Linie_imGeschwindigkeitsraum

Einlesen einer Serie von heliozentrisch korrigierten Spektren im fits-Format, Darstellung einer wählbaren Linie im Geschwindigkeitsraum in zwei Plots: Erstens alle Spektren übereinander geplottet. Zweitens alle Spektren mit einem wählbaren offset übereinander geplottet. Die plots können als pdf abgespeichert werden.

fitsSerie_Linie_imGeschwindigkeitsraum_JDBereich.py

Einlesen einer Serie von heliozentrisch korrigierten Spektren im fits-Format, Eingabe eines Zeitraums als JD Anfang und JD Ende. Darstellung einer wählbaren Linie im Geschwindigkeitsraum in zwei Plots:

Erstens alle Spektren übereinander geplottet.

Zweitens alle Spektren mit einem wählbaren offset übereinander geplottet.

Die plots können als pdf und png abgespeichert werden.

fitsSerie_Linien_imGeschwindigkeitsraum_JDBereich.py

Einlesen einer Serie von heliozentrisch korrigierten Spektren im fits-Format, Eingabe eines Zeitraums als JD Anfang und JD Ende. Darstellung mehrerer wählbarer Linien im Geschwindigkeitsraum in einem Plot, mit einem wählbaren offset übereinander geplottet. Die plots können als pdf und png abgespeichert werden.

Order EW

Zeitserie_EW.py

Berechnet für eine Zeitserie AUF DAS KONTINUUM NORMIERTER SPEKTREN im fit-Format die Äquivalentweite einer Linie. Eingabe der Integrationsgrenzen grafisch-interaktiv oder manuell. Die EW-Berechnung setzt voraus, dass für alle Spektren der Serie das gleiche Wellenlängenintervall für die Berechnung des Integrals verwendet werden kann, also keine wesentlichen RV-Änderungen stattfinden, oder wenn doch, dass die

Linie isoliert ist (also der Flux = 1 im Umfeld ist)[1]. Am besten heliozentrisch korrigierte Spektren verwenden. Die EW's werden in einer ASCII-Tabelle mit den Spalten 'Spektrum' und 'EW' abgespeichert.

Zeitserie_EW_Linientiefe_Linienwellenlaenge.py

Berechnet für eine Zeitserie AUF DAS KONTINUUM NORMIERTER SPEKTREN im fits-Format die Äquivalentweite einer Linie, sowie die Linientiefe und die Wellenlänge des Linienminimums. Eingabe der Integrationsgrenzen manuell oder grafisch.

Normierungsfehler werden durch eine Renormierungroutine im Integrationsbereich kompensiert.

Die EW-Berechnung setzt voraus, dass für alle Spektren der Serie das gleiche Wellenlängenintervall für die Berechnung des Integrals verwendet werden kann, also keine wesentlichen RV-Änderungen stattfinden, oder wenn doch, dass die Linie isoliert ist (also der Flux = 1 im Umfeld ist). Am besten baryzentrisch korrigierte Spektren benutzen.

Das erste Spektrum wird geplottet und für 20 Sekunden angezeigt. In diesem Zeitraum ist das Grafikfenster interaktiv geschaltet, so dass man das Spektrum vergrößern kann und den Integrations-Wellenlängenbereich für die EW-Berechnung optisch aussuchen kann. Dieses Reaktionszeitfenster von 205 Sekunden kann in Zeile 78 (plt.pause(ZeitInSekunden) geändert werden.

Zeitserie_EW_BereicheInEinerLinie.py

Berechnet für eine auf das Kontinuum normierter Zeitserie im fits-Format die Äquivalentweite einer in Bereiche aufgeteilten Linie. Eingabe der Integrationsgrenzen (in Angström) über eine Liste, die anzupassen ist. Die EW-Berechnung setzt voraus, dass für alle Spektren der Serie das gleiche Wellenlängenintervall für die Berechnung des Integrals verwendet werden kann, also keine wesentlichen RV-Änderungen stattfinden. Deshalb am besten baryzentrisch korrigierte Spektren benutzen. Die ermittelten EW's und die Flux-Minima und -Maxima der Linien-Bereiche werden in einer ascii-Datei gespeichert.

Ordner fits

1d_fitSpektrum_ansehen.py

Auslesen und Anzeige der Headerdaten eines 1d-Spektrums im fit-Format. Plotten des Spektrums.

1d_fitSpektrum_ansehen_mitLinienidentifikation.py

Einlesen eines 1d-Spektrums im fits-Format, plotten des Spektrums mit der Möglichkeit, eine oder mehrere Laborwellenlängen von Spektrallinien in Form senkrechter Striche im plot zu markieren.

[1] Falls das nicht vorausgesetzt werden kann müssen die Spektren zuerst bzgl. ihrer RV's korrigiert werden, was mit dem Skript RVKorrekturPerTemplate_1d_spectra_fits_series.py im Ordner RV_Korrektur unter Verwendung eines Templates erfolgen kann.

1d_fitSpektrum_ansehen_Mit_Subplots.py

Ansehen eines wellenlängenkalibrierten 1d-Spektrums, Auslesen und Anzeige der Headerdaten. Plotten des gesamten Spektrums und einer Aufteilung in eine wählbare Zahl von Ausschnitten, die dann in entsprechenden Unterplots dargestellt werden. Zusätzlich kann ein beliebiger Wellenlängenbereich ausgewählt und geplottet werden.

badPixFilter.py

Das Skript dient dazu einpixelige Spikes (heisse Pixel) in 1d-Spektren zu entfernen.

Fileliste erstellen für wellenlängenkalibrierte 1d_Spektren einer Serie im fit-Format. Ersatz von einzelnen Pixelwerten, die oberhalb eines *Grenzwertes* (badpixel) sind, durch den Mittelwert der Nachbarpixel und abspeichern aller korrigierten Spektren als sonst unverändertes fit. File-Name um '_badPixRemoved' ergänzt.

badPixFilter_VergleichNachbarpixel.py

Fileliste erstellen für wellenlängenkalibrierte 1d_Spektren einer Serie im fit-Format. Ersatz von einzelnen Pixelwerten, die die Nachbarpixel um einen *Faktor* überschreiten,durch den Mittelwert der Nachbarpixel und abspeichern aller korrigierten Spektren als sonst unverändertes fit. Name um _ badPixRemoved ergänzt.

fit_Serie_in_csv_und_dat.py

Umwandeln einer Serie von wellenlängenkalibrierten 1d-Spektren im fit-Format in Textformat .csv (Komma-separiert) und .dat-Format (tab-separiert). Mit Spaltenüberschriften 'WAVE' und 'FLUX'.

HeaderanzeigeUndKorrektur_Serie.py

Erzeugung einer Spektrenliste der 1d-Spektren im fit-Format in einem Ordner, Korrektur/Ergänzung des Headers aller Spektren.

Headerprint_Serie_csv.py

Liest für eine 1d-Spektrenserie im fits-Format die Headerdaten ein und schreibt sie in getrennte ascii-Dateien (.csv).

JD_EintragInHeader.py

Das Skript liest eine Serie von fits-1d-Spektren ein und berechnet aus unterschiedlichen Headereinträgen für das Beobachtungsdatum das jeweilige JD und trägt es als "JD" in den header des jeweiligen Spektrums ein.

Wellenlaengenbereich_Zeitserie_fits

Fileliste erstellen für wellenlängenkalibrierte 1d_Spektren einer Zeitreihe im fit-Format. Abspeichern der filelist mit Angaben zum Beginn und Ende der Wellenlängenskala. Ausdrucken des gemeinsamen Wellenlängenbereichs.

Ordner Formatumwandlungen

Formatumwandlungen.py

Umwandlung von

- Julianisches Datum in Kalenderdaten
- Kalenderdatum in Julianisches Datum
- RA und DEC in hexagesimaler Form in float
- RA und DEC von float in hexagesimale Form

Ordner KK (Kreuzkorrelationen)

CrossCorrelation_dat_Serie.py

Es wird eine Kreuzkorrelation einer Serie von target-Spektren bzgl. eines template-Spektrums durchgeführt. Alle liegen als Ascii-Datei (.dat) vor. Die berechneten RV's werden gedruckt und in einer Datei gespeichert.

CrossCorrelation_fits_Serie.py

Es wird eine Kreuzkorrelation einer Serie von target-Spektren bzgl. eines template-Spektrums durchgeführt. Beide liegen als fits vor. Die RV's und wahlweise baryzentrisch korrigierten RV's werden ausgedruckt und in einer Datei abgespeichert. Falls die baryzentrische korrigierten RV's berechnet werden sollen, müssen die Beobachter- und die Objektkoordinaten im Skript angepasst werden. Standardmäßig werden die Sternkoordinaten aber durch Eingabe des Objektnamens (z.B. alp Ori) aus dem Internet übernommen. Dazu ist allerdings eine aktive Internetverbindung nötig.

CrossCorrelation_fits_Serie_einzelneLinien.py

Es wird eine Kreuzkorrelation einer Serie von target-Spektren bzgl. eines template-Spektrums durchgeführt. Alle liegen als fits vor. Es wird ein Spektrumausschnitt abgefragt (Umgebung einer Linie oder einer Zentralwellenlänge), der zur KK verwendet wird.

Es wird die RV und wahlweise die baryz. korrigierte RV berechnet. Die Daten werden in eine ascii-Datei geschrieben. Falls die baryzentrische korrigierten RV's berechnet

werden sollen, müssen die Beobachter- und die Objektkoordinaten im Skript angepasst werden. Standardmäßig werden die Sternkoordinaten aber durch Eingabe des Objektnamens (z.B. alp Ori) aus dem Internet übernommen.

Ordner Lichtkurven

AAVSO.py

Übernimmt die Lichtkurvendaten aus einer csv-Datei (welche z.B. von der AAVSO-Datenbank importiert wurde). Man gibt den auszuwertenden Zeitraum ein (JD Anfang und Ende) und das Programm berechnet die Tagesmittelwerte der mag-Werte. Diese werden in eine csv geschrieben und grafisch dargestellt.

Ordner Normierung

automaticNormalization_timeseries_20230615.py

Funktioniert für ein einzelnes oder eine Zeitreihe von 1d-Spektren im Fits-Format. Beim ersten Spektrum werden die Parameter 'inter' und 'sm' optimiert. Nimmt von Pixel zu Pixel ein Intervall von 'inter' Pixel und vergleicht es mit den benachbarten Intervallen. Findet so lokale Maxima = Stützpunkte. Man kann auch mehrere Wellenlängenbereiche von der Bildung von Normalisierungspunkten ausschließen, d.h. löschen (breite Linien). Der Glättungsparameter 'sm' (Dezimalzahl) stellt die Empfindlichkeit des Splines ein der am Ende zur Normalisierung verwendet wird. sm = 0.0 bedeutet: Der Spline geht durch alle Gitterpunkte. Je größer sm (10.0, 100, 1000....), desto steifer ist der Spline. Optimal ist ein sm, wenn der Spline die Kurven des Kontinuums gut nachzeichnet, aber nicht in die Linien hineinragt. Danach können Sie noch Punkte löschen, die um einen bestimmten Prozentsatz größer oder kleiner als der Spline sind. Schließlich werden alle Spektren der Zeitreihen mit den optimierten Parametern normalisiert. Die normalisierten Spektren (sowie die Plots und die Normalisierungsfunktionen) werden auf Wunsch gespeichert. Außerdem werden alle gewählten Parameter in einer ascii-Datei namens Parameterliste.txt gespeichert.

Ordner rebinning_differenzspektren_Periodenanalyse

newbinning_mittleresSpektrum.py

Liest einen Spektrenkatalog ein (fits), rebinnt die Spektren und erzeugt tab-Spektren (tab-Tabelle mit den Spaltenbenennungen WAVE und FLUX) sowie fits-Dateien, alle mit gleicher wählbarer Schrittweite und gleichem wählbarem Wellenlängenbereich. Die Art der Interpolation (linear oder per kubischem Spline) ist durch auskommentieren wählbar (Zeilen 100 bis 106). Außerdem wird ein gemitteltes Spektrum aus dem gemeinsamen Wellenlängenbereich berechnet und als tab-Tabelle abgespeichert mit den Spaltennamen WAVE und FLUX. Das gemittelte Spektrum wird auch geplottet.

Differenzspektren.py

Liest 1d-Spektren einer Serie (im Ascii-Format 'tab', gebildet mit dem Skript newbinning_mittleresSpektrum ein und bildet Differenzspektren zum angegebenen mittleren Spektrum, die dann als tab-Datei gespeichert werden. Alle Differenzspektren werden in einem Plot grafisch dargestellt, der auch abgespeichert wird.

Periodenanalyse.py

Liest eine Ascii-Datei ein mit den beiden Spalten JD und zugehöriger Wert (z.B. RV's), ohne Spaltenüberschrift und plottet die Werte. Macht dann mit beiden Spalten eine Periodenanalyse nach der Lomb-Scargle-Methode, zeigt grafisch das Periodogramm und die mit der Periode gefalteten Daten (Phasendiagramm) und gibt die Periode aus.

Ordner RV_Korrektur

RVKorrektur_1d_spectra_fits_series.py

Das Skript korrigiert eine Serie von 1d-Spektren im Fit-Format eines Objekts um eine manuell eingegebene RV [km/s]. Schreibt die Fit-Dateien und Ascii-Dateien der RV-korrigierten Spektren in das Arbeitsverzeichnis. Die jeweilige RV wird im Header des erzeugten Fits vermerkt.

RVKorrekturPerTemplate_1d_spectra_fits_series.py

Das Skript korrigiert eine Serie von 1d-Spektren im Fit-Format eines Objekts um eine per KK im Vergleich zu einem Template berechnete RV [km/s]. Schreibt die Fit-Dateien der RV-korrigierten Spektren in das Arbeitsverzeichnis. Die jeweilige RV wird im Header des erzeugten Fits vermerkt. Eine ASCII-Tabelle mit den RV's wird ebenfalls gespeichert.

terrLinien_RVKorrektur_Regression_1d_spectra_fits_series

Input: Einlesen einer Spektrenserie im fits-Format. Diese dürfen nicht heliozentrisch korrigiert sein.

Das Skript korrigiert die Kalibrierung durch Bestimmung des Linienminimums von bekannten terrestrischen Linien per Regression und rechnet die Spektren mit der ermittelten mittleren RV um, soweit sie betragmäßig 3 km/s überschreitet. Abspeichern der korrigierten Spektren als fits. Zeigen der modellierten terrestrischen Linien als Grafik. Ausdrucken der ermittelten RV's. Schreiben der RV's, des Mittelwerts und der Standardabweichung in ein ascii-file.

terrLinien_RVKorrektur_RBF_1d_spectra_fits_series.py

Input: Einlesen einer Spektrenserie im fits-Format. Diese dürfen nicht heliozentrisch korrigiert sein.

Das Skript korrigiert die Kalibrierung durch Bestimmung des Linienminimums von bekannten terrestrischen Linien per RBF und Umrechnung der Spektren mit der ermittelten mittleren RV. Abspeichern der korrigierten Spektren als fits. Zeigen der modellierten terrestrischen Linien als Grafik. Ausdrucken der ermittelten RV's.

Die terrestrischen Linien müssen natürlich in den Spektren vorhanden sein. Derzeit sind 3 normalerweise wenig gestörte Wasserlinien zwischen 6543 und 6574 Angström berücksichtigt.

Ordner RV_Messung

Im Ordner RV_Messung sind Skripte vereinigt, deren Aufgabe es ist, an einzelnen Spektren oder an Spektrenserien Radialgeschwindigkeitsmessungen durchzuführen. Dazu dienen verschiedene Methoden:

- KK (Kreuzkorrelation) von Spektrumausschnitten
- Minimumbestimmung an einzelnen Linien.

Für die Minimumbestimmung an definierten Linien existiert ein Modul namens *Linienlisten.py* in diesem Verzeichnis, der gemeinsam von vielen der RV-Bestimmungs-Skripten verwendet wird und deshalb von den Skripten importiert werden muß. Deshalb muss das Modul im gleichen Verzeichnis stehen wie das jeweilige Skript oder in einem Verzeichnis, der in der Umgebungsvariablen PYTHONPATH enthalten ist, damit Python es findet. In dieses Modul können natürlich weitere Linien aufgenommen werden. Dazu muß das Modul nur entsprechend editiert werden.

RV_MessungAnLinie_fits_interaktiv.py

Liest einen Spektrenkatalog ein (Zeitserie). Berechnet aus dem Beobachtungszeitpunkt und den (anzupassenden) Koordinaten des Beobachters und Objekts die heliozentrische Korrektur, zeigt die gewählte Linie in einem Plot, das Minimum der gewählten Linie ist anzuklicken, berechnet aus dem Minimum die heliozentrisch korrigierte Radialgeschwindigkeit RV_bc. Gibt die ermittelten Daten als ASCII-Dateien (tab-separiert) aus.

RV_MessungAnLinie_Zeitserie_dat_perInteraktion.py

Das Skript dient zur Ermittlung der RV einer Linie in Doppelsternsystemen (SB2), die in 2 Linien aufgespalten sein kann.

Liest den Spektrenkatalog ein (Zeitserie von normierten 1d-Spektren im tab-Format, 2 Spalten WAVE und FLUX). Festlegung des Linienminimums per Interaktion und Bestimmung der Radialgeschwindigkeit aus dem Minimum. Plottet alle Spektren zum markieren von bis zu 2 Linien-Minima und gibt ermittelte Daten (RV und Apex) als ascii-Dateien (tab-separiert, als .dat) aus. Die RV's sind nicht baryzentrisch korrigiert.

RV_MessungAnLinie_Zeitserie_dat_perInteraktion_regression_2Linien.py

Wie das vorige Skript, nur dass das Minimum (die Minima der beiden Linien) rechnerisch per Regression (Grad 2, 4 oder 6) bestimmt wird.

Liest einen Spektrenkatalog ein (Zeitserie von normierten 1d-Spektren im tab-Format). Festlegung des Linienminimums per Interaktion und bestimmt aus dem per Regression errechneten Minimum die Radialgeschwindigkeit RV. Plottet zur Kontrolle alle fittings und gibt ermittelte Daten als ascii-Dateien (tab-separiert, als .dat) aus.

RV_MessungAnLinie_Zeitserie_dat_perRegression.py

Liest Spektrenkatalog ein (Zeitserie von normierten 1d-Spektren im tab-Format). Fitted die gewählte Linie per Regression nten Grades und bestimmt aus dem Minimum die Radialgeschwindigkeit RV. Plottet alle fittings und gibt ermittelte Daten als ascii-Dateien (tab-separiert, als .dat) aus.

RV_MessungAnLinie_Zeitserie_fits_perGaussfit.py

Liest einen Spektrenkatalog ein (Zeitserie von normierten (!!!) 1d-Spektren im fits-Format). Berechnet aus dem Beobachtungszeitpunkt und den (anzupassenden) Koordinaten des Beobachters und Objekts die baryzentrische Korrektur, fittet die gewählte Linie per Gaussfit dreimal (gesamt und innere Bereiche) und bestimmt aus dem heliozentrisch korrigierten Minimum die heliozentrisch korrigierte Radialgeschwindigkeit RV. Plottet alle fittings und gibt ermittelte Daten als ascii-Dateien (tab-separiert, als .dat) aus.

RV_MessungAnLinie_Zeitserie_fits_perRBF.py

Liest einen Spektrenkatalog ein (Zeitserie von normierten 1d-Spektren im fit-Format). Berechnet aus dem Beobachtungszeitpunkt und den (anzupassenden) Koordinaten des Beobachters und Objekts die heliozentrische Korrektur, fittet die gewählte Linie per Radial basis function (RBF) und bestimmt aus dem heliozentrisch korrigierten Minimum die heliozentrisch korrigierte Radialgeschwindigkeit RV. Plottet zur Kontrolle alle fittings und gibt ermittelte Daten als ascii-Dateien (tab-separiert, als .dat) aus.

RV_MessungAnLinie_Zeitserie_fits_perRegression.py

Liest einen Spektrenkatalog ein (Zeitserie von normierten 1d-Spektren im fits-Format). Berechnet aus dem Beobachtungszeitpunkt und den (anzupassenden) Koordinaten des Beobachters und Objekts die heliozentrische Korrektur, fitted die gewählte Linie im Minimumbereich per Regression und bestimmt aus dem (heliozentrisch korrigierten) Minimum die (heliozentrisch korrigierte) Radialgeschwindigkeiten RV und RV_bc. Plottet alle fittings und gibt ermittelte Daten als ascii-Dateien (tab-separiert, als .dat) aus.

RV_MessungAnLinie_Zeitserie_fits_perRegression_AbsUndEmi.py

Liest einen Spektrenkatalog ein (Zeitserie von normierten 1d-Spektren im fits-Format). Berechnet aus dem Beobachtungszeitpunkt und den (anzupassenden) Koordinaten des Beobachters und Objekts die heliozentrische Korrektur, fitted die gewählte Linie im Minimum(Absorption)- oder Maximum(Emission)bereich per Regression und bestimmt aus dem heliozentrisch korrigierten Minimum/Maximum die heliozentrisch korrigierte Radialgeschwindigkeit RV. Plottet und speichert alle fittings und gibt die ermittelten Daten als ascii-Datei (Komma-separiert, als .csv) aus.

RV_MessungAnLinie_Zeitserie_fits_perSpline.py

Liest einen Spektrenkatalog ein (Zeitserie von normierten 1d-Spektren im fits-Format). Berechnet aus dem Beobachtungszeitpunkt und den (anzupassenden) Koordinaten des Beobachters und Objekts die heliozentrische Korrektur, fitted die gewählte Linie per Spline und bestimmt aus dem (heliozentrisch korrigierten) Minimum die (heliozentrisch korrigierte) Radialgeschwindigkeiten RV und RV_bc. Plottet alle fittings und gibt ermittelte Daten als ascii-Dateien (tab-separiert, als .dat) aus.

Ordner SNR

SNR_betasigma.py

Berechnet für eine Serie von 1d-fits-Spektren mittels des beta*sigma-Verfahrens das SNR jedes Spektrums. Falls sinnvoll die Parameter N und j im Skript anpassen (Zeilen 40 und 43).

SNR_fits_mehrereBereiche.py

Plottet das gewählte 1d-fits-Spektrum. Wähle die Bereiche für die SNR-Berechnung (Kontinuum) aus, in dem du mit linken Mausklicks die Grenzen setzt und mit doppeltem Drücken der Escape-Taste die jeweilige Eingabe des Bereichs beendest. Es können beliebig viele Bereiche eingegeben werden.

verrauschen.py

Mit dem Skript kann man zusätzliches Rauschen in ein Spektrum einbringen.

Ordner theoreticalSpectra

KunstspektrumErzeugen.py

Das Skript erzeugt ein künstliches, per gaußbroadening auf eine bestimmte FWHM, ausgedrückt durch eine Auflösung R, verbreitertes Linien-Spektrum. In den Zeilen 19 bis 24 werden die Linien, Linienstärken, Wellenlängenbereich und R definiert.

Kurucz_Modell_GaussBroadened.py

Laden eines Kuruzc-Sternatmosphärenmodells (1d-Spektrum) nach Auswahl von T_{eff} und log g. Convolution mit einer Gaußfunktion auf Spektrographenauflösung, Auswahl des Wellenlängenbereiches und Anpassung auf eine gewünschte Schrittbreite. Speicherung als ASCII-tab-Tabelle und als .fit mit dem einzugebenden Filename. Speicherung des Plot als PDF.

Um das Skript verwenden zu können muss im User-Verzeichnis ein Ordner PyAData existieren mit den Modellspektren. Der Ordner wird beim Aufrufen des Skripts 'Get access to the models' in https://pyastronomy.readthedocs.io/en/latest/pyaslDoc/resBasedDoc/kuruczModel angelegt.

Pollux_Ausschnitt_rebinned_convol_spec.py

In der Pollux-Datenbank (http://pollux.graal.univ-montp2.fr/) lassen sich theoretische Spektren für die verschiedensten physikalischen Sterneigenschaften herunterladen. Wir verwenden das Format .spec.

Einlesen eines synthetischen Spektrums in Form einer Tabelle (wie als .spec in der Pollux-Datenbank erhältlich). Verwendet wird der normierte Flux. Berechnung eines wählbaren Wellenlängenausschnitts (Pandas Dataframe mit 'newtable' bezeichnet). Dieser Bereich wird dann mit einer wählbaren Schrittweite rebinned und anschließend noch zusätzlich mit einer wählbaren FWHM (Apparateprofil) gefaltet. Geplottet wird der gewählte rebinnte Wellenlängenbereich und zusätzlich das gefaltete Spektrum. Der rebinnte Flux-Ausschnitt des ursprünglichen .spec und der rebinnte und convolvierte Flux wird zusammen mit der Wellenlänge in je einer zweispaltigen ascii-Tabelle (spacer = tab) und in je einer fits-Datei abgespeichert.

Ordner Zeitreihe

Crop_Auschnitt_Zeitserie_fits.py

Fileliste erstellen für wellenlängenkalibrierte 1d_Spektren einer Zeitreihe im fit-Format. Plotten aller Spektren mit Wahl, ob die Grafik gespeichert werden soll.. Ausdrucken des gemeinsam abgedeckten Wellenlängenbereichs in der Konsole. Beschneiden aller Spektren auf einen wählbaren Wellenlängenbereich und abspeichern aller beschnittenen als fit mit Wellenlängenbereich im Dateinamen.

MittelungVon_fits.py

1d-Spektren im fits-Format. Sie müssen alle gleiches header['CRVAL1'], header['CDELT1'], header['NAXIS1'], header['CRPIX1']) haben. Diese Variablen werden für jedes Spektrum ausgedruckt (Kontrolle der Gleichheit). Mit der Eingabe von 'y' werden alle Fluxe der Spektrenserie gemittelt und das mittlere Spektrum als fit abgespeichert.

timeseries_einPlot_mitOffset.py

Liest alle 1d-fits-Spektren einer Serie ein und plottet sie mit einem wählbaren Offset übereinander. Speichert den Graph als PNG und als PDF.

timeseries_plot_JeEineGrafik.py

Liest alle 1d-fits-Spektren einer Serie ein und plottet sie in getrennten Grafiken. Die Plots werden als PDF gespeichert.

Zeitserie_Beobachtungszeitpunkte.py

Liest die 1d-fits-Spektren einer Spektrenserie ein und druckt die Beobachtungszeitpunkte aus. Außerdem werden der Beginn und das Ende des Wellenlängenbereichs jedes Spektrums ausgegeben. Die Beobachtungzeitpunkte werden in einer ASCII-Datei abgespeichert.

Zeitserie_Spektren_smoothen.py

Einlesen einer Serie von 1d-fits-Dateien und smoothen des fluxes mit einer bestimmten Fensterbreite. Abspeichern der geglätteten fits.