

TTPL Auswerteprogramm Anleitung

Aaron Flötotto

30. März 2022

Inhaltsverzeichnis

1	Installation	2
1.1	Python	2
1.2	Benötigte Python-Pakete	2
1.3	Programmdateien speichern	2
2	Bedienung	3
2.1	Start-Skript	3
2.2	Dateistruktur	3
2.3	Passende Parameter finden	4
2.4	Dokumentation der Parameter	4

1 Installation

1.1 Python

Hier kann ein Installer der aktuellen Python-Version für Windows heruntergeladen werden. Zusätzlich braucht man für das Auswerteprogramm noch einen (ziemlich beliebigen) Texteditor. Unter Python-Nutzern sind z. B. Microsoft Visual Studio Code oder PyCharm beliebt.

1.2 Benötigte Python-Pakete

Das Auswerteprogramm greift auf Inhalte der folgenden Python-Pakete zu:

- wheel
- numpy
- pandas
- matplotlib
- scipy
- SciencePlots

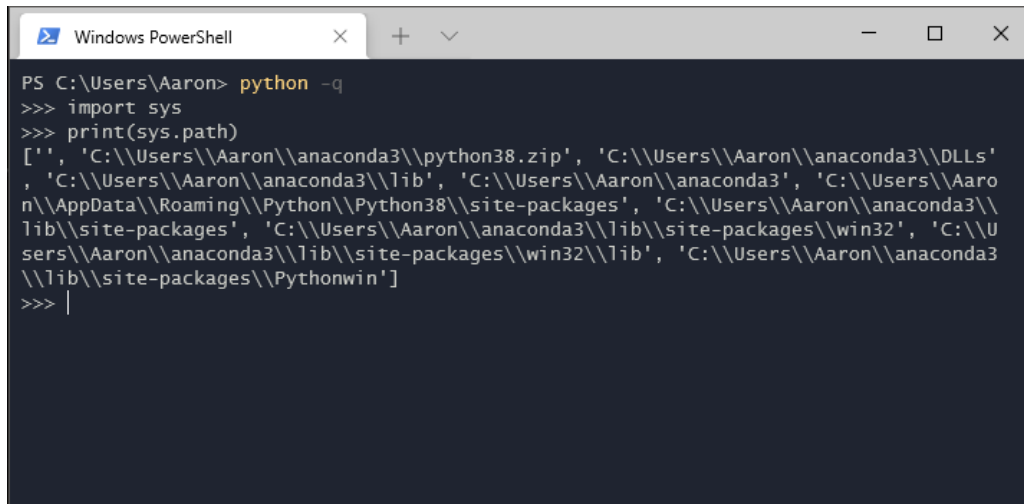
Zur Installation dieser Pakete kann der Paket-Manager pip benutzt werden. pip ist für gewöhnlich in der Python-Installation enthalten.

Um ein Paket zu Installieren öffnet man zunächst die Windows-Eingabeaufforderung (beispielsweise mit “Windows-Taste + r → *cmd* eingeben → Enter”). Hier kann man dann mit dem Befehl “pip install *Paketname*” die oben aufgelisteten Pakete installieren.

Alternativ können PyCharm-Nutzer diese Pakete auch über ein integriertes Tool installieren.

1.3 Programmdateien speichern

Die Programmdateien des Auswerteprogramms sind in diesem Cloud-Ordner. Die Dateien sollten in einem sog. Pythonpath gespeichert werden. Um den zu finden öffnet man wie oben beschrieben die Windows-Eingabeaufforderung. Dort angekommen gibt man die in Abbildung 1 dargestellten Befehle ein. Hierdurch wird eine Liste aller Ordner ausgegeben, in denen Python nach Modulen sucht. Da kann man sich einen Ordner aussuchen und dort die Programmdateien des Auswerteprogramms ablegen.



```
PS C:\Users\Aaron> python -q
>>> import sys
>>> print(sys.path)
['', 'C:\\Users\\Aaron\\anaconda3\\python38.zip', 'C:\\Users\\Aaron\\anaconda3\\DLLs',
 'C:\\Users\\Aaron\\anaconda3\\lib', 'C:\\Users\\Aaron\\anaconda3', 'C:\\Users\\Aaro
n\\AppData\\Roaming\\Python\\Python38\\site-packages', 'C:\\Users\\Aaron\\anaconda3\\
lib\\site-packages', 'C:\\Users\\Aaron\\anaconda3\\lib\\site-packages\\win32', 'C:\\U
sers\\Aaron\\anaconda3\\lib\\site-packages\\win32\\lib', 'C:\\Users\\Aaron\\anaconda3
\\lib\\site-packages\\Pythonwin']
>>> |
```

Abbildung 1: Über die Variable `sys.path` findet man heraus in welchen Ordnern die Programmdateien abgelegt werden können.

2 Bedienung

2.1 Start-Skript

Das Auswerteprogramm wird über ein Python-Skript gestartet. Ein Beispiel dafür kann hier heruntergeladen werden.

Im Skript werden die Parameter des Auswerteprogramms definiert. Diese Parameter sind in den Programmdateien dokumentiert. Eine Kopie dieser Dokumentation befindet sich in Abschnitt 2.4. **Historisch gewachsen sind einige der älteren Parameter etwas redundant; bei Gelegenheit muss ich da aufräumen.** Für ähnliche Spektren (gleiche bzw. ähnliche Wafer und gleiche Monochromator-Einstellungen) müssen die Parameter nicht verändert werden. Zur Auswertung anderer Messungen sollte das Skript kopiert und in der Kopie die Parameter angepasst werden.

Zum Starten des Skripts wird die Windows-Eingabeaufforderung gestartet und der Befehl “python *Pfad zum Skript*” eingegeben. Beispiel: Wenn man sich in der Eingabeaufforderung im Ordner des Skripts befindet lautet der Befehl “python *Skriptname.py*”. In integrierten Entwicklungsumgebungen wie z. B. Microsoft Visual Studio Code oder PyCharm gibt es auch eine Schaltfläche mit der das Skript ausgeführt werden kann.

2.2 Dateistruktur

Ich würde empfehlen alle zusammengehörigen .asc-Messdateien (z. B. alle Messungen eines Defektzyklus oder eines Temperatur-Scans) in einen Ordner zu kopieren und im selben Ordner eine Kopie des Auswertungsskripts zu speichern. Wird nun das Skript gestartet werden alle Messdateien im Ordner ausgewertet.

Bei der Auswertung wird für jede Messdatei eine gleichnamige .pdf-Datei mit einem Plot des Spektrums und der Fitfunktion im gleichen Ordner gespeichert. Abhängig von der Wahl des Parameters *export* wird für jede Messung auch eine Datei mit den Funktionswerten des Spektrums und des Fits sowie der einzelnen Komponenten des Fits gespeichert. Zusätzlich werden in der Datei “_Auswertung.csv” Eigenschaften (Temperatur, $\frac{BTO}{ITO}$ -Peakverhältnis, ...) aller Messungen aufgelistet. Diese Datei kann man zur Visualisierung der Ergebnisse in Origin importieren. **Welche Eigenschaften hier gespeichert werden kann recht leicht verändert werden. Da könnten wir uns mal überlegen was sinnvoll wäre.**

2.3 Passende Parameter finden

Die Parameter des Auswerteprogramms sind in der Dokumentation in den Programmdateien und hier in Abschnitt 2.4 beschrieben. Um passende Werte für die Parameter zu finden ist es hilfreich das Spektrum zu zeichnen und Werte (z. B. Peakpositionen oder Energien für die Baselinekorrektur) abzulesen. Hierfür enthält das Auswerteprogramm die Funktion *TTPL.plot_spektrum(datei)*. Um diese zu Nutzen kann im beispielhaften Start-Skript der Wert der Variable *parameter_finden* zu True geändert werden. Wird nun das Skript wie oben beschrieben ausgeführt, öffnet sich ein Diagramm-Fenster mit Spektrum aus der angegebenen Datei. Wird die Maus in das Diagramm bewegt werden oben rechts im Fenster die Koordinaten der Maus angezeigt. Um anschließend wieder Auswertungen zu starten muss die Variable *parameter_finden* wieder auf False gesetzt werden.

Es ist oft nicht nötig alle Peaks im Spektrum zu fitten. Falls nur die Proben temperatur und das $\frac{BTO}{ITO}$ -Peakverhältnis von Interesse sind, reicht es oft nur $B_{TO}(BE)$, $B_{LO}(BE)$ und die beiden intrinsischen Peaks zu fitten. Hierfür passt man die Parameter *peakarten*, *peakpos*, *peakpos_vergleich* und *T_vergleich* an. Je mehr Peaks man anpasst und je mehr Vergleichswerte für Peakpositionen angegeben sind, desto genauer ist allerdings die lineare Regression zur Kalibrierung der Energie-Achse. Diese hat auch einen Einfluss auf die ausgegebene Proben temperatur.

2.4 Dokumentation der Parameter

peakarten : String

Legt fest welche Funktionen zum Fitten der einzelnen Spektren benutzt werden sollen. Z.B. *peakarten*= 'LGW' für ein Spektrum mit drei Peaks, dessen Peaks mit einem Lorentz-, einem Gauß- und einer Maxwell-Boltzmann-Funktion mit wurzelförmigem Anteil gefitted werden sollen. Sortiert wird aufsteigend nach der Energie. Die Möglichkeiten sind: L (Lorentz), G (Gauß), Q (Maxwell-Boltzmann mit quadratischem Anteil), W (Maxwell-Boltzmann mit wurzelförmigem Anteil), S (Beschreibt ITO und ILO wie im Paper von Pelant: Je eine Faltung von W mit G), B (Bound-Exciton-Peak; Faltung aus Lorentz und Gauß)

ITO_pos : Float

Position des ITO-Peaks in eV.

BTO_pos : Float

Position des BTO-Peaks in eV.

roi : List

Position in eV, die bei der Baselinekorrektur mit Geraden verbunden sollen (Siehe docstring sowie *baseline_ASTM.base()* sowie die ASTM-Vorschrift zu TTPL-Spektroskopie)

P_pos : Float

Position der P-Linie in eV. Falls das Spektrum keine P-Linie enthält kann -1 angegeben werden.

benutze_fit : Bool

Entscheidet, ob die Werte der Fitfunktion zur berechnung der Intensitätsverhältnisse benutzte werden. Falls nicht werden lokale Maxima des gemessenen Spektrums benutzt.

smooth : 2-Tuple

Fensterbreite und Polynomordnung für einen Savitzky-Golay-Filter der Intensitäten. Falls *smooth* = None wird kein Filter angewendet.

x0_diff : Float

Bestimmt die Grenzen beim fit für die Peakpositionen gelten. Bei *x0_diff* = 1 dürfen die Peakpositionen beim Fit in beide Richtungen um bis zu 1 meV verändert werden. Um die Peakpositionen während des Fits effektiv nicht anzupassen, kann hier ein sehr kleiner Wert eingetragen werden.

peakpos : List

Vorgabe für die Positionen der Peaks. Bei *peakpos*=None werden die Peaks mit *scipy.signal.find_peaks()* erkannt. Falls bei *peakpos*!=None in einem Intervall von +- *x0_puffer* um

die vorgegebenen Werte von `scipy.signal.find_peaks()` Peaks erkannt werden, werden deren Positionen benutzt.

`x0_puffer` : Float

Siehe beschreibung von `peakpos`. Angabe ist in meV.

`Delta` : Float

Abstand zwischen ITO und ILO. Wird für die Anpassung des Si-Peaks aus ITO und ILO benötigt.

`sigma_faltung` : Float

Wird für die Anpassung des Si-Peaks aus ITO und ILO benötigt. Halbwertsbreite der Gaußfunktionen, die wie von Pelant beschrieben mit den Maxwell-Boltzmann-Funktionen (Wurzel) gefaltet werden.

`peakpos_vergleich` : List

Vergleichswerte für die Positionen der Peaks für der Kalibrierung der Energie-Achse.

`T_vergleich` : Float

Probentemperaturen bei der Messung der Vergleichswerte. Die Liste muss genau so lang sein wie `peakpos_vergleich`. Achtung: Möglicherweise ist in der Literatur nicht die Probentemperatur sondern die Badtemperatur angegeben.

`export` : Boolean

Bestimmt ob die Daten (E-Achse, Intensität, Fitfunktion, Einzelne Bestandteile der Fitfunktion) als Textdatei importiert werden sollen.

`roi_fenster` : Float

Wird für die Baselinekorrektur benötigt. Beispiel: `roi_fenster = 0.5` -> Es wird in einem Umkreis von 0.5 meV um die Energien in `roi` nach Intensitätsminima gesucht. Die Positionen der Minima werden als 'neues' `roi` genutzt.

`ILO_ratio_grenzen` : 2-Tuple

Grenzen für das ILO/ITO Peakverhältnis beim Pelant-Fit der beiden intrinsischen Peaks. Bei vergleichsweise hohen Temperaturen neigt der Fit dazu eine zu kleine Temperatur zu wählen und das mit einem zu hohen Peakverhältnis auszugleichen. In diesem Fall sollte `ILO_ratio_max` verringert werden. Andernfalls kann getrost ein Wert von 1 benutzt werden.

`sigma_faltung_interval` : 2-Tuple

`sigma_faltung` ist die Standardabweichung der Gaussfunktion, die wie von Pelant beschrieben mit dem Si-Peaks (und den BE-Peaks) gefaltet wird. Dieser Parameter entspricht dem Interval, in dem `sigma_faltung` beim Fit variiert werden kann. Der Mittelwert des Intervals wird als Startwert benutzt.

`E_int` : 2-Tuple

Interval für die Energieachse. Alle Werte außerhalb des Intervals werden abgeschnitten.

`T_start` : Float

Temperatur, die den Startwert für die Breite des intrinsischen Peaks beim Fit bestimmt.