OES

# Allgemeine Informationen

OES: optical emission spectroscopy

Spektraler Übergang bezeichnet die Wellenlänge λ, welche nach Anregung von Molekülen beim Fall in den stabilen Zustand emittiert wird. Also lässt die Auswertung aller Peaks eines Spektrogramms Rückschlüsse auf das Gasgemisch der verschiedenen Moleküle zu.

Da eine Auswertung der absoluten aufgenommenen Werte nicht sinnvoll/möglich ist, wird ein Referenzpeak (Wasserstoff/Methan) genutzt, um eine Konzentration eines anderen Moleküls zu bestimmen.

Einstieg in die aktuelle Version der Python über die Datei: oes-spa.py

## Schematische Darstellung der Software

Andor Software

* Steuert Spektrometer 🡪 nimmt indirekt Spektren auf (oes)
* Mittels Parameter konfigurierbar bzw. Spektren einstellbar

Spexhex

* In VB programmiert
* Auswertung der Stickstoff Spektren
* Spezial-Software, da nur bestimmte Parameter eingestellt werden können und nur Stickstoff ausgewertet werden kann

Daten/Spektren

Parameter

Oes-spectra-analysis

* Python Programmierung
* Auswertung der Bor Spektren
* Erweiterung der Möglichkeiten von Spexhex um Bor-Auswertung

Andor Messgerät

* Parametereinstellung der Messung
* Durchführung der Spektroskopie

Daten/Spektren

Parameter

Daten

Spektren

Dateisystem

## Bisherige Nutzung

1. Spexhex
   1. Aufnahme der Spektren
      1. Wellenlängenkalibrierung
      2. Speicherung in .spk-Datei
      3. Pixel zu Intensität
      4. Pixel zu Messgeräte-Wellenlänge
      5. Pixel zu korrigierter Wellenlänge
   2. Auswertung der Spektren
   3. Erstellung einer .dat-Datei
      1. Zeitpunkt der Spektrumsaufnahme
      2. Kennwert des Spektrums (z.B. CN-Wert)
      3. Wissenschaftliche Analyse erfolgt anhand der .dat-Datei
2. OES-Analysis (Python-Software)
   1. Untersuchung des Bor-Peaks
   2. Erstellung einer .dat-Datei (s.o.)

## Derzeitige Nutzer

* Peter Knittel
* Philipp Reinke
* Christoph Schreyvogel
* Julia Langer
* Arne Götze
* Volker Cimalla

# Lastenheft (LH)

* Primäre Parameter
  + Wellenlänge
  + Grating
  + Step and Glue
  + Port (zukünftig: 1/2/1 u. 2)
  + Shutter (zukünftig)
* Sekundäre Parameter
  + Belichtungszeit (Exposure time)
  + Anzahl der Akkumulierungen
  + Readout Mode (default: FVB)
  + Vertical/horizontal Pixel Shift (default values)

## Use-Cases

### Paramterierung

* Speicherort?
* Dateiformat?
  + .ini
  + .json

### Messgerätesteuerung

* Unterschiedliche Andor Messgeräte mit verschiedenen Parametersätzen (Anzahl der möglichen Optionen für einen Parameter), z.B. verfügbare Gitter/Ports
* Zugriff nur über Remote-PC möglich, da das Messgerät keine direkte Netzwerkverbindung hat und Datenaustausch nur über USB-Kabel möglich ist

1. Einstellen von Parametern
   1. Auslesen der aktuell eingestellten Parameter
   2. Auslesen des Parametersatzes
   3. Bereitstellen eines Interfaces zum Einstellen möglicher Optionen
   4. Schreiben der Parameter
   5. Standard-Konfiguration(.conf-Datei?)
   6. Primäre und sekundäre Parameter (s.o.)
   7. 🡪 Lese-/Schreibzugriff
2. Ausführen der Spektroskopie
   1. Anzahl/Wiederholrate/Modus (Stick and Glue) /Dauer einstellen
   2. Messung triggern
   3. Messergebnisse speichern/bereitstellen
      1. Dateiformat
      2. Definition der Dateistruktur
      3. Festlegung des Speicherpfades
   4. Machbarkeitsstudie zu Programmierung der Andor Software (Schritt 1)
   5. Übertragung in Analysesoftware (Schritt 2)

### Auswertung/Analyse

1. Auswerten von Peaks
   1. Einstellen des Peaks (N, Bor, P, MH4)
   2. Laden des charakteristischen Spektrums des Peaks
      1. Definierter Speicherort?
   3. Faltung des aufgenommenen und des Referenzspektrums zur Kalibrierung
   4. Berechnung von Kennwerten (z.B. CN-Wert) und Ausgabe
      1. Welche Kennwerte können berechnet werden
         1. Dargestelltes Spektrum hat minimale und maximale Wellenlänge
            1. Peak und Referenzpeaks müssen in dem Bereich liegen, um diese auch analysieren zu können.
            2. Speicherung der Peak/Referenzpeaks und deren Grenzen (in Abhängigkeit eines SNR?)
         2. Abhängig vom Fitting
            1. Boron Fitting hat andere Kennwerte als normales Spektrum, da auch in anderem Wellenlängenbereich analysiert wird
      2. Welche Kennwerte sollen berechnet werden
         1. Verhältnis des zu untersuchenden Peaks zu 1 oder mehreren anderen Referenzpeaks
      3. Wo wird das festgelegt (gleiche Datei/Speicherort wie bei charakteristischen Spektrum
   5. Berechnung des SNR zur Bestimmung der Nutzbarkeit des Spektrums
2. Darstellung von Spektren
   1. Rohes Spektrum (Raw Spectrum)
      1. X-Achse: Pixel (in Abhängigkeit des Gratings)
      2. Y-Achse: Intensität
      3. Berechnen und Einfügen einer Basislinie (Baseline)
   2. Bearbeitetes Spektrum (Processed Spectrum)
      1. X-Achse: Wellenlänge (in Abhängigkeit der zentralen Wellenlänge (Central Wavelength))
      2. Y-Achse: Intensität
   3. Laden von Dateien
      1. Per Browsen
      2. Per Drag&Drop
      3. Anzeige des Speicherpfades
      4. Dateiformate
         1. Spk
         2. Csv (gespeichertes Spektrum; csv zu Unterschied zu Raw Spektrum)
         3. Binary file von der Andor SW (sekundär)
   4. Interaktives Design
      1. Zoom
      2. Einstellen der Achsen
      3. Speichern
         1. Raw\_/processed\_spektrum
            1. Als .csv-Datei, damit der Unterschied zu aufgenommenem Spektrum (.spk) ersichtlich ist
            2. Raw\_spectrum ggf. ein \_raw dem Dateinamen anhängen
            3. Speicherung über das Menü
            4. Ggf. auch im Menü der Batch-Analysis
            5. Gespeichert wird:

X-Daten

Y-Daten

Aufnahmedatum+Zeit

Processed außerdem:

Central Wavelength

Grating

Auswertmethode

Peak height

Ratio

Area

* + - 1. Innerhalb der BatchAnalysis
         1. Angabe oder Auswahl einer .csv-Datei
         2. Auswählen von .spk/.csv-Dateien
         3. Auswahl der zu speichernden Kennwerte
         4. Berechnen und Speichern der Kennwerte über „Calculate“
         5. Clear 🡪 Löschung der Auswahl

# Programmcode von oes-spa.py

(Unstrukturierte Dokumentation)

* openFile
  + AnalysisWindow
    - \_\_init\_\_
      * leer initialisiert W
    - dropEvent
      * Dropped File W
      * getValues R
      * getHead R
    - file\_open
      * opened File W
      * getValues R
      * getHead R
  + BatchAnalysis
    - multi\_calc
      * selected File W
      * getValues R
      * getHead R

## Name Convention

### Klassen

* Jede Klasse ist in einer eigenen Datei definiert
* Datei und Klasse haben den gleichen Namen
* Methoden-Bezeichnungen
  + camelCase, wenn die Methode vom Qt-Backend erkennt werden muss, z.B. Events wie dragEnterEvent
  + snakeCase, wenn die Methode unabhängig com Qt-Backend ist, z.B. file\_open

## Programmpfade

### Laden von Dateien

Zugriff auf Systemdateien über QFileDialog.getOpenFile bzw. über .getOpenFiles zum Laden einer oder mehrerer Dateien (Doku: <https://doc.qt.io/qt-5/qfiledialog.html>).

Mögliche Dateiformate

* .csv
* .spk
* Bindary Dateien von Andor (sekundär, nicht implementiert)

Menüleiste: File 🡪 Open

* keine Datei ausgewählt /Abbruch 🡪 keine Aktion
* eine Datei ausgewählt 🡪 Anzeige der Daten
* mehrere Dateien ausgewählt 🡪 BatchAnalysis & Anzeige der Daten aus erster Datei

Drag&Drop:

* Drag
  + eine Datei ausgewählt 🡪 event.accept()
  + mehrere Dateien ausgewählt 🡪 BatchAnalysis
* Drop
  + eine Datei ausgewählt 🡪 Anzeige der Daten
  + mehrere Dateien ausgewählt 🡪 Anzeige der Daten aus erster Datei

BatchAnalysis: Browse Files

* keine Datei ausgewählt /Abbruch 🡪 keine Aktion
* eine oder mehrere Dateien ausgewählt 🡪 Anzeige der Daten aus erster Datei

## Testen der Anwendung

Schwierigkeiten:

* Testen von Events?!

### PICT

Pairwise Independent Combinatorial Tool 🡪 Reduziert die ausgeführten Tests und behält trotzdem eine gute Testabdeckung bei.

### Test der GUI mit QtTest

Bessere Testbarkeit von GUI-Elementen anstatt mit python.unittest

### Test mit python.unittest

* Verwendung der Bibliothek unittest.
* How to: <https://www.youtube.com/watch?v=1Lfv5tUGsn8>

Ausführen von Tests:

* Öffnen der Anaconda Promt
* Navigation zu dem Ordner mit Modulen und Tests
* Eingabe: python -m unittest [name]
  + -m 🡪 instruct the python module to run as a script
  + [name] (optional) 🡪 run a specific test, if left blank it will run all tests in that directory

# Weitere Beschreibungen und Abhängigkeiten

## Hilfreiche Befehle

* Herausfinden der aktuell verwendeten Python Version
  + Öffnen der Anaconda Prompt
  + Eingeben: python –V
* Ändern der GUI
  + Öffnen der Anaconda Prompt
  + Eingeben: designer
  + Öffnen einer GUI
  + Änderungen vornehmen und Speichern
  + Zurück in der Anaconda Prompt
  + pyuic5 batch\_dialog.ui > ui\_batch\_dialog.py
    - pyuic5 macht aus einem .ui-file ein entsprechendes .py-file
    - das .py-file kann dann eingebunden und verwendet werden
    - Achtung: Überschreibt alle Änderungen, die in der Zwischenzeit in dem .py-file gemacht wurden (hier: ui\_batch\_dialog.py)
    - Abhilfe: Erstellen einer Klasse in UIBatch.py, um Änderungen zu akzeptieren und eigene Änderungen beizubehalten.

## Central Wavelength

* Parametereinstellung in der Andor SW bei der Aufnahme eines Spektrums
* Unabhängig vom Fitting/Grating
* Soll als Parameter mit in die Spk Dateien aufgenommen werden
* Derzeit: Triggered ein Redraw bei Änderung+Enter
* Weitere Verwendung:
  + Auslesen der central Wavelength
    - Berechnen der minimalen und maximalen Wellenlänge in Abhängigkeit des Gratings
      * Mögliche Fittings auswählen, wenn die gesuchten und die zu referenzierenden Peaks innerhalb der berechneten Wellenlängen vorhanden sind