**Supporting Information**

# *pyQCM-BraTaDio: A tool for visualization, data mining, and modelling of Quartz crystal microbalance with dissipation data*

*Brandon M. Pardi[1‡], Syeda Tajin Ahmed[1‡], Silvia Jonguitud Flores[2], Warren Flores[1], Jean-Michel Friedt[5], Laura L.E. Mears[2], Bernardo Yáñez Soto[3], Roberto C. Andresen Eguiluz[1,4]**

[1]Department of Materials Science and Engineering, University of California Merced, Merced, California 95344, United States of America

[2]Institute of Applied Physics, Technische Universitaet Wien, Vienna 1030, Austria

[3]Instituto de Física, Universidad Autónoma de San Luis Potosí, San Luis Potosí 78000, Mexico

[4]Health Sciences Research Institute, University of California Merced, Merced, California 95344, United States of America

[5]FEMTO-ST Time & Frequency department, 26 Rue de l'Épitaphe, 25000 Besançon, France

Brandon M. Pardi ORCID: 0000-0001-6483-9858

Syeda Tajin Ahmed ORCID: 0000-0002-2719-9641

Warren Flores ORCID: 0009-0001-1462-4688

Jean-Michel Friedt ORCID: 0000-0003-4888-7133

Silvia Jonguitud Flores ORCID: 0009-0005-4841-4901

Laura L.E. Mears ORCID: 0000-0001-7558-9399

Bernardo Yáñez Soto ORCID: 0000-0002-4273-6513

Roberto C. Andresen Eguiluz ORCID: 0000-0002-5209-4112

**Corresponding Author**

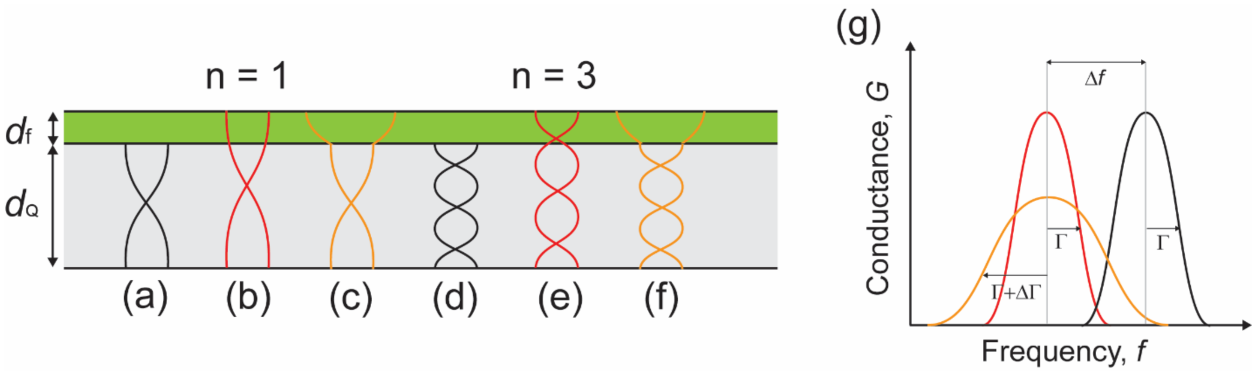
*Roberto C. Andresen Eguiluz

Science and Engineering 2, Room 292
Department of Materials Science and Engineering
University of California Merced
5200 N. Lake Rd., Merced CA 95340, USA
randreseneguiluz@ucmerced.edu

**Keywords:** quartz crystal microbalance with dissipation; analysis tool; open software.

## 1 Quartz crystal microbalance with dissipation (QCM-D) background

QCM-D is an acoustic-based, surface sensitive technique that measures small mass changes and energy losses, in real time, at the surface of a sensor. The sensors are commonly fabricated from a piezoelectric material, such as quartz, which allows thickness-shear of the sensors at its resonance frequencies by applying an alternating electric field. The most common cut of quartz crystals for QCM-D applications is the AT-cut, as it has excellent frequency-temperature characteristics.[1] For these sensors, there are $n$ number of acoustic modes, referred to here as overtones or overtone order, that can be approximated as standing waves perpendicular to the crystal surface with negligible longitudinal wave propagation.



**SI Figure 1.** (a) and (d) Standing wave of overtone orders $n$ = 1 and $n$ = 3 across the thickness of the sensor, $d_Q$. (b) and (e) Increased sensor thickness due to analyte film decreases resonance frequency. (c) and (f) When the analyte film is softer than the sensor, the interface generates a distortion to the standing wave, and the frequency shift becomes proportional to the mass, rather than the thickness. (g) Frequency and bandwidth shifts, $\Delta f$ and $\Delta\Gamma$, respectively, are used to measure changes in film thickness $d_f$ and energy losses.

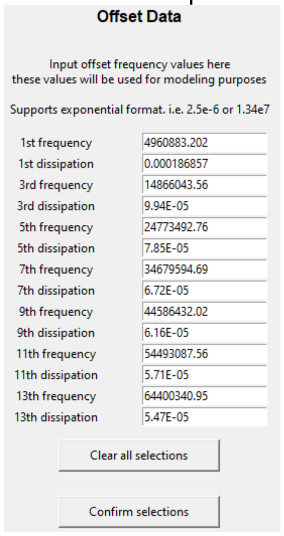
The penetration depth of a 5 MHz shear wave in water is approximately $\delta \approx 250$ nm for the fundamental overtone and $\delta \approx 70$ nm for the 13th overtone,[2,3] making QCM-D a surface-sensitive instrument. For small films formed by analytes at the surface of the sensor, the resonance frequency $f$ is inversely proportional to the total thickness of the plate. That is, the effective thickness $d_{\mathrm{eff}} = d_{\mathrm{Q}} + d_{\mathrm{f}}$ of the sensor increases with increasing amount of analyte coupled to the sensor surface, decreasing its resonance frequency $f$. This relationship was first identified by Sauerbrey,[4] and is schematized in SI Figure 1. When the forming film is rigid or very thin (and homogeneous and continuous), the film-sensor interface does not change the bandwidth ($\Delta\Gamma = 0$). However, if the forming films are viscoelastic (or heterogeneous or discrete), the shift or change in frequency $\Delta f$ is coupled with a change in bandwidth as well ($\Delta\Gamma \neq 0$), and the ideal inverse linear relationship between changes in thickness and changes in frequency no longer apply.[5]

## 2 Using the pyQCM-BraTaDio

The code repository for pyQCM-BraTaDio includes a 'README.md' file, however this next section contains further details on utilizing the software providing more context to use cases.

### 2.1 Entering data file information

In region (1) from the main GUI shown in Figure 3, the user loads the relevant file in any of the supported formats: *.txt, *.csv, *.xls, or *.xlsx, and indicates the data structure by selecting the multi-harmonic instrument that generated the data. Next, the '*relative baseline time'* (*'absolute baseline time'* for openQCM Next) refers to the beginning ($t_0$) and end ($t_f$) of what will be considered the experimental baseline. For example, if the file contains an air-to-liquid transition, followed by an equilibration time, the relative baseline time will consist of the last minutes of the equilibration time. pyQCM-BraTaDio calculates the average $\Delta f_n$ and average $\Delta D_n$ of all datapoints within the selected range (between $t_0$ and $t_f$) and uses those averages to set the reference (i.e., $\Delta f_n \approx 0$ Hz and $\Delta D_n \approx 0$).


Offset Data

Input offset frequency values here
these values will be used for modeling purposes

Supports exponential format. i.e. 2.5e-6 or 1.34e7

| | |
|---|---|
| 1st frequency | 4960883.202 |
| 1st dissipation | 0.000186857 |
| 3rd frequency | 14866043.56 |
| 3rd dissipation | 9.94E-05 |
| 5th frequency | 24773492.76 |
| 5th dissipation | 7.85E-05 |
| 7th frequency | 34679594.69 |
| 7th dissipation | 6.72E-05 |
| 9th frequency | 44586432.02 |
| 9th dissipation | 6.16E-05 |
| 11th frequency | 54493087.56 |
| 11th dissipation | 5.71E-05 |
| 13th frequency | 64400340.95 |
| 13th dissipation | 5.47E-05 |

**SI Figure 2.** Input window for calibration obtained frequency and dissipation values (sometimes referred to as offset values).

**SI Table 1.** Theoretical resonance frequency values for 5 MHz, AT-cut quartz crystals.

| **Overtone order, *n*** | **Frequency, *f* (Hz)** |
|---|---|
| 0 or 1 | 4930000.00 |
| 3 | 14800000.00 |
| 5 | 24700000.00 |
| 7 | 34600000.00 |
| 9 | 44500000.00 |
| 11 | 54400000.00 |
| 13 | 64300000.00 |

The user is next asked to indicate usage of either theoretical or experimental offset values for resonant frequency calculations. Selecting theoretical uses resonant frequency values for 5 MHz AT-cut quartz crystals outlined in SI Table 1, selecting offset will lead to two potential workflows. If the user has selected QCM-I or openQCM-Next, the offset values are automatically taken from the user-inputted baseline time frame. If QSense or AWSensors is selected, the user will be prompted to either enter values in the window shown in SI Figure 2 or edit the 'COPY_PASTE_OFFSET_VALUES_HERE.csv' file in the 'offset_data' directory. It should be noted that these offset values are not required for basic visualization purposes, but model application functions will not be viable as they rely on the resonant frequency values for calculation. It is necessary to keep in mind that the typically reported values of dissipation $D_n$ are related to bandwidth $\Gamma_n$ by:[22]

$$D_n = \frac{\Gamma_n}{f_n} \tag{1}$$

where $D_n$ is the dissipation of order $n$, $\Gamma_n$ is the bandwidth of order $n$, and $f_n$ is the resonance frequency of order $n$.

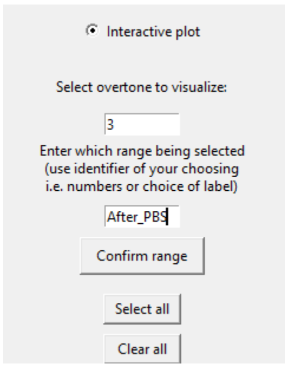
Finally, it is necessary to submit input by clicking the '*Submit file information'*. The user also has the capability of personalizing the plotting aesthetics, such as size and font type for axis titles, $x$ and $y$ plotting ranges, selecting colors for data, among others. These options are described in the Customize Plot Options section in the SI.

## 2.2 Data selection for visualization, mining, and analysis

The data selection region allows the users to choose the frequencies and dissipations from specific overtones for data mining, visualization, and model application purposes. pyQCM-BraTaDio works with overtones $n$ = 1, 3, 5, 7, 9, 11, 13. It is also here where the user selects to work with the raw, full data range (*i.e.*, from the first to last data point acquired) or with the experimental data (*i.e.*, from a predetermined data point to the last data point acquired) and taking the changes in frequencies ($\Delta f_n$) and dissipations ($\Delta D_n$).

## 2.3 Interactive plots

In order to utilize pyQCM-BraTaDio's interactive plot, first, the user is required to select the Interactive plot option under either *Plot raw data (f* and *D)* or *Plot shifted data (*Δ*f* and Δ*D).* Next, it is necessary to designate one of the available overtones for data visualization (*e.g.,* $n$ = 3 in SI Figure 3) and a label to the data range (e.g., After_PBS in SI Figure 3). Note that even though one overtone is selected for visualization, operations done in the interactive plot will apply to all overtones selected in the GUI. Lastly, to enable the interactive plot, the user must check the box labelled 'enable interactive plot'. The interactive plot will then display $f$ and $D$ or Δ$f$ and Δ$D$, depending on the selection. Figure 4 shows the latter case.


Interactive plot

Select overtone to visualize: 3

Enter which range being selected (use identifier of your choosing i.e. numbers or choice of label) After_PBS

**SI Figure 3.** Activating the interactive plot option requires to select an overtone and assign a label or identifier for data mining.

The interactive plot window consists of 5 panels, an input text field to numerically provide a time range of interest in user-predefined units (default is seconds), Figure 4(a). The time range will be highlighted in the frequency, Figure 4(b), and dissipation, Figure 4(d), interactive plots. Alternatively, the user can select a range by clicking and holding the left mouse button in either direction of the frequency or dissipation interactive plots. The input text field will be updated and display the time limits of the selection. Interactive frequency and interactive dissipation plots are coupled, that is, same time ranges will be applied to frequency and dissipation channels. The plots shown in Figures 4(c) and (e) display the selected range only and calculated drift by applying a linear regression. For frequency, it will be in units of Hz over time, and numerical value over time for the dissipation (*i.e.*, seconds, minutes, or hours, depending on the selected unit under the *Customize plot options* menu). pyQCM-BraTaDio will compute the average and standard deviation of the data points contained within the selected range for each overtone selected, every time a selection is made. The range can be moved and adjusted as needed, and the new average and standard deviation computed will be updated each time. Repeating the process overwrites the previously saved average and standard deviation for that specific range identifier. The average and standard deviation will be calculated for all selected overtones (all $f_n$, $\Delta f_n$, $D_n$, and $\Delta D_n$) in a file contained in the *selected_ranges* directory. Details on the file structure can be found in the *Description of directories and output files* section.

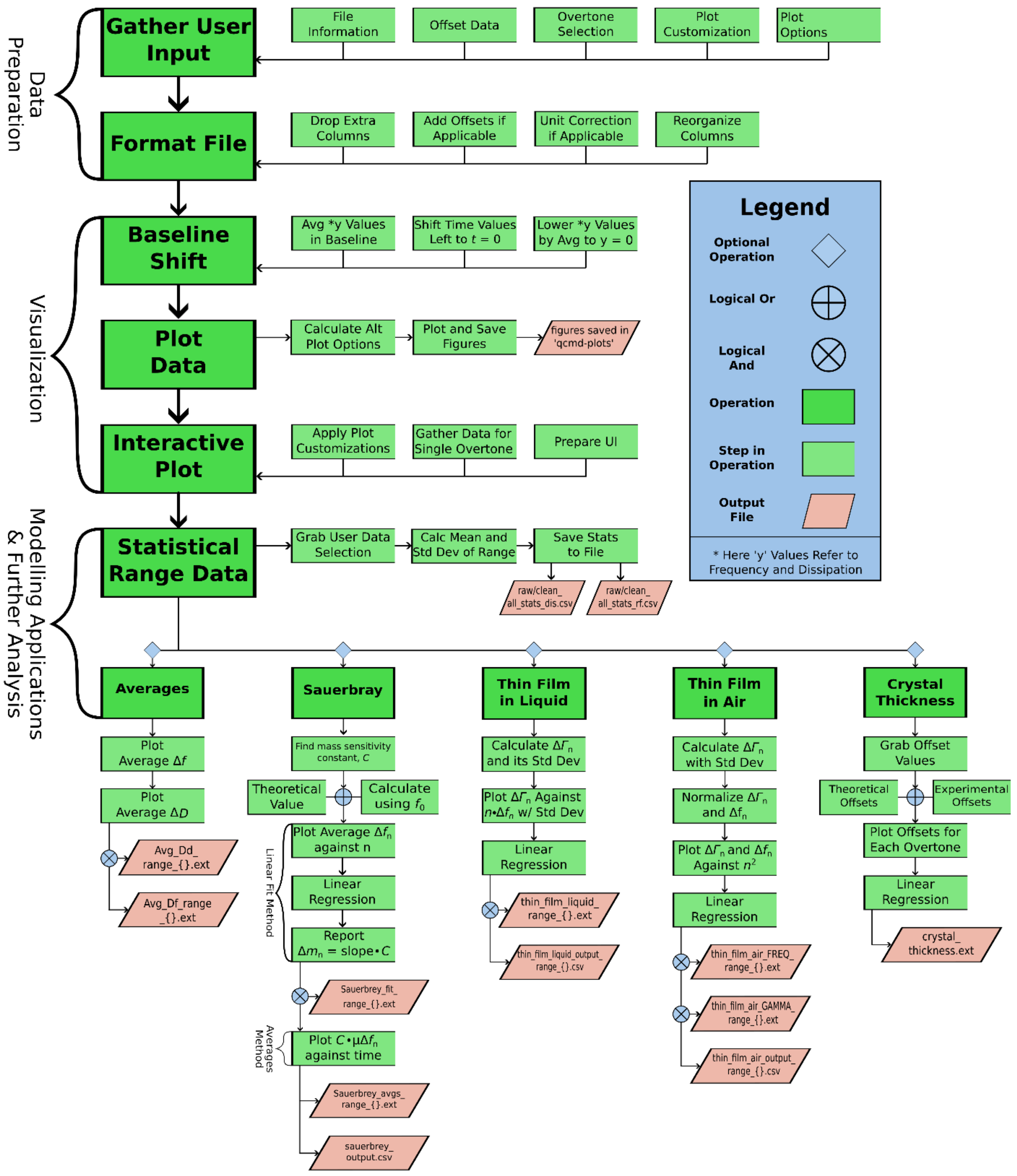
Changing the range identifier signals to pyQCM-BraTaDio that a new experimental step has been selected and averages and standard deviations will be saved in new fields (instead of overwriting the previously stored values). Note that when a new range is selected, data for previous ranges will remain untouched and new selections will only update for the currently identified range. This data will remain untouched until the user clicks the 'Clear saved ranges' button in column 4, which is recommended when switching to new experimental data, or using different models.

### 2.4 A note on execution time

It should be noted that depending on the data file size, combined with computational power, processing time can vary between a fraction of a second to a few seconds. Faster processing times were demonstrated using a Ryzen 9 5900X 12 core (24 threads) processor at 4.5GHz with 32 GB of RAM, while the lower end of software execution speed was using a more conventional computer, consisting of an Intel i7-10510U 4 core (8 thread) processor at 1.8 GHz and 16 GB of RAM.

## 3 Software Workflow

The previous section's workflow is summarized in pyQCM-BraTaDio's architecture, SI Figure 4.



Data Preparation
Gather User Input
File Information
Offset Data
Overtone Selection
Plot Customization
Plot Options
Format File
Drop Extra Columns
Add Offsets if Applicable
Unit Correction if Applicable
Reorganize Columns
Visualization
Baseline Shift
Avg *y Values in Baseline
Shift Time Values Left to t = 0
Lower *y Values by Avg to y = 0
Plot Data
Calculate Alt Plot Options
Plot and Save Figures
figures saved in 'qcmd-plots'
Interactive Plot
Apply Plot Customizations
Gather Data for Single Overtone
Prepare UI
Legend
Optional Operation
Logical Or
Logical And
Operation
Step in Operation
Output File
* Here 'y' Values Refer to Frequency and Dissipation
Modelling Applications & Further Analysis
Statistical Range Data
Grab User Data Selection
Calc Mean and Std Dev of Range
Save Stats to File
raw/clean_all_stats_dis.csv
raw/clean_all_stats_rf.csv
Averages
Plot Average Δf
Plot Average ΔD
Avg_Dd_range_{}.ext
Avg_Df_range_{}.ext
Sauerbray
Find mass sensitivity constant, C
Theoretical Value
Calculate using f0
Linear Fit Method
Plot Average Δfn against n
Linear Regression
Report Δmn = slope•C
Sauerbrey_fit_range_{}.ext
Averages Method
Plot C•μΔfn against time
Sauerbrey_avgs_range_{}.ext
sauerbrey_output.csv
Thin Film in Liquid
Calculate ΔΓn and its Std Dev
Plot ΔΓn Against n•Δfn w/ Std Dev
Linear Regression
thin_film_liquid_range_{}.ext
thin_film_liquid_output_range_{}.csv
Thin Film in Air
Calculate ΔΓn with Std Dev
Normalize ΔΓn and Δfn
Plot ΔΓn and Δfn Against n²
Linear Regression
thin_film_air_FREQ_range_{}.ext
thin_film_air_GAMMA_range_{}.ext
thin_film_air_output_range_{}.csv
Crystal Thickness
Grab Offset Values
Theoretical Offsets
Experimental Offsets
Plot Offsets for Each Overtone
Linear Regression
crystal_thickness.ext

**SI Figure 4** pyQCM-BraTaDio workflow and structure of operations. Beginning with data preparation, pyQCM-BraTaDio takes in the input fields specified by the user in the UI, and formats the file dependent on the experimental apparatus used to record the data. Once data is prepared, it is then visualized. This includes shifting data by the baseline, computing alternate plot options, saving figures, and generating the interactive plot. The optional modelling and further analysis layer can commence after visualization. This starts with the user making selections in the interactive plot and basic statistical calculations being applied on those selections, and ends with the execution of various available models for the data previously selected.

## 4 QCM-D models background

pyQCM-BraTaDio offers several models to apply experimental data to. The offered models are given systematic context in this section.

### 4.1 The Sauerbrey equation for thin, rigid films

The Sauerbrey equation (eq 2) is a linear relationship between the resonance frequency change and the mass change of the acoustic oscillator:[4,10,11]

$$\Delta m_{\mathrm{Sauerbrey}} = -c_{\mathrm{Theo}} \cdot \frac{\Delta f_{\mathrm{n}}}{n} \qquad (2)$$

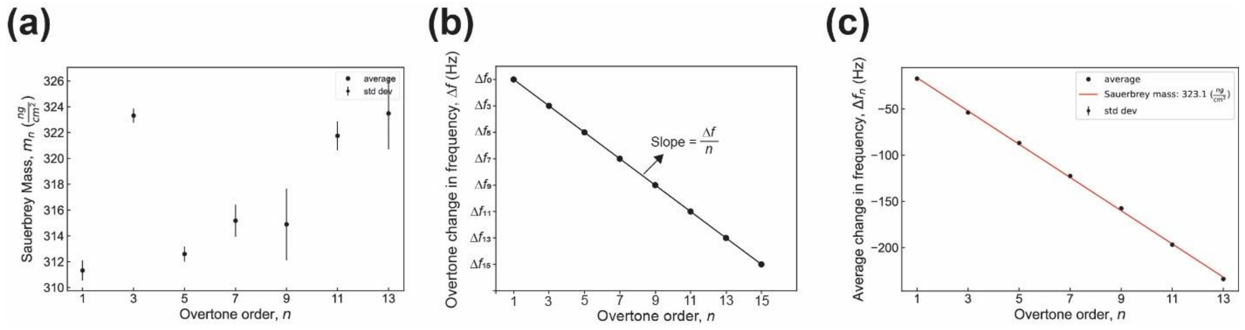
where $\Delta m_{\mathrm{Sauerbrey}}$ is the change in Sauerbrey mass in (ng/cm$^2$) (or mass per unit area), $\Delta f_{\mathrm{n}}$ is the change in frequency of overtone *n* in (Hz), *n* is the overtone order, and $c_{\mathrm{Theo}}$ the mass sensitivity constant in (ng/cm$^2$·Hz).

The Sauerbrey equation is applicable to very thin films, with a change in mass small compared to the quartz crystal, that can be considered rigid (no deformation) and perfectly coupled to the quartz crystal surface (no slip), homogeneous and evenly distributed over the quartz crystal surface. The theoretical value of $c_{\mathrm{Theo}}$ for a 5 (MHz) crystal is 17.7 (ng/(cm$^2$·Hz)). The true $c_{\mathrm{True}}$ can be obtained from eq 3:

$$c_{\mathrm{True}} = \frac{\vartheta_{\mathrm{q}} \cdot \rho_{\mathrm{q}}}{2 f_0^2} \qquad (3)$$

Where $\vartheta_{\mathrm{q}}$ is the is the wave velocity in quartz, $\rho_{\mathrm{q}}$ is the density of quartz, and $f_0$ is the measured, fundamental resonance frequency. It is frequent to report $\Delta m_{\mathrm{Sauerbrey}}$ for one overtone order of interest, SI Figure 5(a). However, more accurate $\Delta m_{\mathrm{Sauerbrey}}$ can be obtained from the slope of the change in frequency of each overtone $\Delta f_{\mathrm{n}}$ as a function of overtone order *n* and multiplying it by the mass sensitivity constant, $c_{\mathrm{Theo}}$. Using the theoretical values is most of the times a good approximation, as $c_{\mathrm{Theo}}$ and $c_{\mathrm{True}}$ are usually in very good agreement. However, when the calibration values are provided, the Sauerbrey mass can be calculated using $c_{\mathrm{True}}$.

The data shown in SI Figure 5 corresponds to the Sauerbrey mass of a BSA film after a PBS wash formed from a 1 mg/mL bulk solution in PBS. Details are described in the *Model Experiments* section in the SI, and plots obtained from these experiments are shown in SI Figures 5, 6, and 7.



**SI Figure 5. Sauerbrey mass as a function of overtone order.** (a) Sauerbrey mass calculated as a funciton of each individual overtone order, (b) linear regression model to obtain the Sauerbrey mass using the changes in frequency from all available overtones, and (c) Sauerbrey mass calculated from performing a fit to the average change in frequency as a function of overtone order for the data range shown in Figrue 6 (BSA after PBS wash).

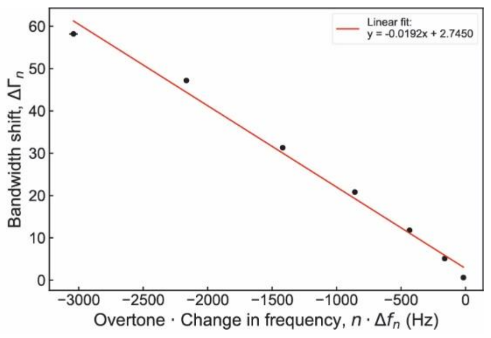
pyQCM-BraTaDio plots the Sauerbrey mass for each selected overtone order, SI Figure 5(a) and saves the Sauerbrey mass values to *sauerbrey_output.csv* file in the *selected_ranges_directory*. It also plots and displays the Sauerbrey mass obtained from calculating the slope of the linear regression and multiplying by the mass sensitivity constant $c_{\mathrm{True}}$, SI Figure 5(c). Similar results for an experiment conducted in a QSense system are shown in SI Figures 8, 11, and 12.

## 4.2 Shear-dependent compliance of a thin viscoelastic film in a Newtonian liquid

For the cases in which the film formed on the surface of the quartz sensor is significantly more rigid than the environment Newtonian liquid, it is possible to obtain the shear-dependent compliance of the film by plotting the change in bandwidth over the negative of the change in frequency, $\frac{\Delta\Gamma}{-\Delta f}$, as a function of the overtone order, *n*, and calculating the slope:[11,12]

$$\frac{\Delta\Gamma}{\Delta f} \approx J_f' \omega \eta_{bulk} = J_f' 2\pi n f_0 \eta_{bulk} \qquad (5)$$

where $J_f'$ is the shear dependent compliance of the film, $\omega$ the angular frequency ($\omega$ = $2\pi n$), and $\eta_{bulk}$ is the viscosity of the Newtonian fluid. This equation assumes that the viscous dependent compliance of the film is much smaller than the shear dependent viscous compliance of the bulk liquid, $J_f'' << J_{bulk}''$. The model was derived by Du and Johannsmann and a detailed derivation can be found elsewhere.[12]



**SI Figure 6.** Shear dependent compliance, $J_f''$ of a BSA film formed from a bulk solution of 1 mg/mL BSA in PBS after a PBS wash.

pyQCM-BraTaDio calculates the shear-dependent compliance of a thin viscoelastic film in a Newtonian liquid by computing the slope of the bandwidth shift for each overtone ($\Delta\Gamma_n$) as a function of the change in frequency times its overtone order value ($n \cdot \Delta f_n$). SI Figure 6 shows the sheard-dependent compliance of a BSA film after a PBS wash formed from a 1 mg/mL bulk solution in PBS. The calculated value $J_f'$ is 0.0192 Pa$^{-1}$. The datapoints displayed in the generated plot are saved in *thin_film_liquid_output.csv* file in the *selected_ranges* directory. Experimental details are described in the *Model Experiments* section, and plots obtained from these experiments are shown in Figures 2, 3, and 4.

### 4.3 Shear-dependent compliance of a thin viscoelastic film in air

For the cases in which the film on the surface of the quartz sensor is exposed to air as the medium instead of a liquid, the inertial effects of the medium become negligible. It is possible to obtain the shear-dependent compliance of the film by plotting the normalized change in frequency $\frac{\Delta f}{n}$, as a function of the square of the overtone order, $n^2$, and calculating the slope:[11,24,25]

$$(\Delta f/n)/\Delta n^2 \approx J_f' \tag{6}$$

If the slope is relatively constant, it suggests that the shear-dependent compliance $J_f'$ is relatively insensitive to the frequency.

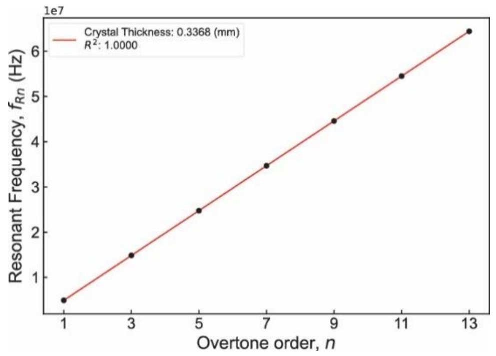
### 4.4 Estimation of the quartz crystal thickness

For the cases in which the true thickness of the crystal is required, and the calibration resonance peaks known, it is possible to estimate the thickness of the crystal $h_q$. The implementation is based on the work by Reviakine *et al.*,[37]

$$f_n = \frac{n}{2h_q}\sqrt{\frac{\mu_{qn}}{\rho_q}} \tag{7}$$

$$\mu_{qn} = \mu_q + \frac{\epsilon}{\kappa} - \frac{8\epsilon^2}{\pi^2 n^2 \kappa} \quad (8)$$

where $n$ is the overtone order, $h_q$ is the crystal thickness, $\rho_q$ the density of quartz (2650 kg/m$^3$), $\mu_{qn}$ is the elastic modulus of quartz considering piezoelectric stiffening, $\mu_q$ is the shear modulus of quartz ($2.93 \cdot 10^{10}$ Pa), $\epsilon$ is the piezoelectric stress coefficient ($-9.24 \cdot 10^{-2}$ C/m$^2$), and $\kappa$ is the dielectric constant ($3.982 \cdot 10^{-11}$ F/m).



**SI Figure 7.** Determination of the quartz crystal thickness.

pyQCM-BraTaDio calculates the thickness of the quartz sensor for the cases in which the calibration resonance frequency values have been provided. The corresponding outputs are a plot displaying the calculated $h_q$ from the user defined number of overtones and saves the computed value in the *crystal_thickness_output.csv* file in the *selected_ranges* directory.

## 5 Related works

In a collection of works,[10–20] Johannsmann and co-workers have developed, refined, and summarized qualitative and quantitative approaches for the analysis and interpretation of QCM-D data, which the reader is encouraged to review. Kanazawa and Gordon[21,22] derived a simple relationship which expresses the change in frequency $\Delta f$ in contact with a fluid only in terms of the material parameters of the fluid (*i.e.*, density $\rho_F$ and viscosity $\mu_F$) and the quartz crystal (*i.e.*, density $\rho_Q$ and shear modulus $\mu_Q$). The relationship is valid for semi-infinite viscoelastic media. Du and Johannsmann[12] derived the elastic compliance $J_f'$ of a viscoelastic film deposited on a quartz crystal surface in a liquid environment from the ratio of bandwidth shift $\Delta\Gamma$ and frequency shift $\Delta f$. Voinova *et al.* derived the general solution describing the dynamics of two-layer viscoelastic materials of arbitrary thickness deposited on a solid (the quartz crystal) surface in a fluid environment.[23] This relationship can be used for QCM-D measurements of layered

## 6 Data interaction

### 6.1 Input file structure

At the time of publication, pyQCM-BraTaDio supports 4 of the major QCM-D devices: openQCM-Next, QCM-I, QSense, and AWSensors. Below is a detailed description of the structure of the files these devices output, and how it is relevant to the software's execution, specifically the file formatting process.

The file structure of openQCM-Next is the file structure that inspired the structure of pyQCM-BraTaDio. Time here is recorded as absolute, meaning every entry has a time stamp in the format of hh:mm:ss, where as other formats record time as relative meaning time starts at 0 counts upwards from there. Columns are listed 'Frequency_n', 'Dissipation_n' where *n* = 0, 1, 2, 3, corresponding to overtones fundamental, 3rd, 5th, 7th, and 9th. The time column is labelled 'Time' in seconds, and temperature column labelled 'Temperature' in degrees Celsius.

QCM-I records time relatively and names its columns as 'Channel A Fundamental Frequency [Hz]', 'Channel A 3. Overtone [Hz]', 'Channel A 5. Overtone [Hz]', …, 'Channel A 13. Overtone [Hz]' and 'Channel A Fundamental Dissipation [ ]', 'Channel A 3. Dissipation [ ]', Channel A 5. Dissipation [ ]', …, 'Channel A 13. Dissipation [ ]'. Its time column is written as 'Channel A QCM Time [sec]' and temperature as 'Channel A Temp [Celsius]. These are the only necessary columns for pyQCM-BraTaDio's workflow. Others are dropped during file formatting. It should be noted that QCM-I also records frequency variation $\Delta f$, and dissipation variation, $\Delta D$, however we opt to use the full values, *f* and *D* respectively. This is due to some of the requirements for modelling downstream in the pipeline that rely on the full values, rather than deltas. Note that we refer to frequency variation and dissipation variation as change in frequency and change in dissipation, respectively.

Both openQCM-Next and QCM-I record data as *f* and *D*, so no offsets need to be considered. There is also no unit conversion required, as frequency and dissipation are recorded in their base units (*i.e.*, frequency values are recorded in Hz not MHz, and dissipation values are reported as $10^0$ and not $10^{-6}$ ).

QSense and AWSensors on the other hand, records $\Delta f$ and $\Delta D$, rather than *f* and *D.* QSense and AWSensors normalize their data by the overtone order ($\Delta f_n/n$), and report dissipation values as $10^{-6}$, meaning un-normalization and scaling is necessary. QSense and AWSensors data files require the most computation to preprocess, due to the three data operations described that other formats do not require. These operations are described in further detail in the next section.

### 6.2 Working with files from QCM-D devices not natively supported by pyQCM-BraTaDio

If you have a device outside of the four supported devices mentioned above, consider the options below and reach out the authors with any questions or concerns. We are also able and willing to meet with readers and discuss adding their file types to be natively supported by pyQCM-BraTaDio.

### 6.2.1 Manually formatting data files

The simplest approach is to manually format non-supported data files. However, it is time-consuming. To do so, follow the below steps:

1. Convert file to *.csv.

If the data file is in *.txt, *.xls, *.xlsx, etc, it must be converted to *.csv by saving the file as *.csv using Excel, or any other spreadsheet software.

2. Remove and rename columns.

   Remove any columns that are not time, temperature, or the frequency/dissipation values for each overtone, as pyQCM-BraTaDio will not need these. pyQCM-BraTaDio references column names using the Pandas Python library. This means that the headers of your sheet must match the column names of the sample *.csv file provided in the 'raw_data' directory.

3. Data formatting.
   a. Magnitude scaling. QSense records dissipation as multiples of $10^{-6}$. pyQCM-BraTaDio reads dissipation as $10^{-0}$. Therefore, the user needs to ensure magnitudes are in terms of $10^{0}$ by multiplying all dissipation values by $10^{-6}$.
   b. Unnormalize data if normalized. pyQCM-BraTaDio relies on non-normalized frequency data (and dissipation if needed). If the user's experimental data is normalized, simply multiply each overtone's data by its corresponding overtone number.
   c. Offset value addition. For proper modeling purposes, change in frequency and change in dissipation columns need to be converted to absolute values. To do this, add offset data for each overtone, to its corresponding overtone's column in the dataset.

This process results in a file that pyQCM-BraTaDio can read. Please note, when saving this type of file following these formatting steps, prepend the word 'Formatted' to the data file (*i.e.*, if the data file was originally named 'qcmi_sample.csv', it should be renamed to 'Formatted-qcmi_sample.csv'). Additionally, if the file records time relatively (*i.e.*, starting at $t$ = 0 seconds) select the QCM-I option in the file options of the GUI. If the file records time absolutely (*i.e.*, using time stamps in the format of HH:MM:SS), then select openQCM-Next.

### 6.2.2 Altering code to automate formatting

This option is more difficult to set and recommended for people with some degree of Python experience, but more efficient long term as it requires the user to add code to the ‘format_file.py’ file to automate the process described above. In that file, the function ‘format_qsense’ is a good example.

1. Define a new function as follows:

```
def_format_<name of experimental device>(fmt_df, calibration_df)
```

Where <name of experimental device> is replaced with an identifier of the device used to generate the data, ‘fmt_df’ is the dataframe to be formatted, and ‘calibration_df’ is the dataframe containing offset data that will be added to ‘fmt_df’. ‘calibration_df’ is only required if user’s data is recorded as changes (*i.e.*, $\Delta f$ and $\Delta D$) rather than absolute values, as is the case with QSense. If the device records absolute values (*i.e.*, $f$ and $D$) then one can follow ‘format_qcmi’ as a more relevant example.

2. Define a dictionary called ‘renamed_cols_dict’ that will contain the key:value pairs, where the key is the original column name, the value is the new column name. For convenience, the new column names to be formatted are globally defined as a list at the top of the script. Using QCM-I formatting as an example, the dictionary will be defined as follows:

```
renamed_cols_dict = {'Channel A QCM Time [sec]':'Time',
        'Channel A Fundamental Frequency [Hz]':freqs[0],'Channel A Fundamental
        Dissipation [ ]':disps[0],
        'Channel A 3. Overtone [Hz]':freqs[1], 'Channel A 3. Dissipation  [ ]':disps[1],
…
'Channel A 13. Overtone [Hz]':freqs[6], 'Channel A 13. Dissipation  [ ]':disps[6],
        'Channel A Temp [Celsius]':'Temp'}
```

With the renaming dictionary defined, next is to call the ‘rename_cols’ function to rename the columns, passing the original dataframe and the dictionary defined. This function returns the formatted dataframe and does not format in place, therefore it is important to set a new variable, ‘fmt_df’ equal to this function. It should also be noted that this function drops columns that are not needed for pyQCM-BraTaDio’s execution.

It is at this point that if data is recorded in the format akin to QCM-I as described earlier, skip step 3.

3. Data formatting (magnitude order conversion, un-normalization, and offsets addition) using provided functions.

Note before proceeding, data may not need all formatting methods. Consult the device manufacturer to understand how the data is recorded, and proceed accordingly applying only needed formatting. That is, data may be recorded as full values, *f* and *D* but it may be in different units and/or be normalized, which as described above, needs correction.

Also note, the order for these operations is crucial, follow the order as described below.

   a. Magnitude order conversion in pyQCM-BraTaDio is done via an applied lambda function. Navigate to the 'format_qsense' function in 'format_file.py' and copy the line:

```
fmt_df.loc[:, disps] = fmt_df.loc[:, disps].apply(lambda x: x*1e-6)
```

and paste this into the function, below the column formatting described in step 2. Note that this converts only magnitudes of dissipation values to multiples of $10^0$. Magnitudes may vary, so adjust accordingly.

   b. Set the 'fmt_df' equal to the 'unnormalize()' function as follows:

```
fmt_df = unnormalize(fmt_df)
```

This will multiply all frequency values by their respective overtone number. Note that if dissipation is also normalized, it is required to adjust the 'unnormalize()' function accordingly.

   c. Set the 'fmt_df' equal to the 'add_offsets()' functions as follows:

```
fmt_df = add_offsets(calibration_df, fmt_df)
```

This is where the calibration file is required. For each overtone it adds the offset value to all recorded values of that respective overtone in the data file.

Now return the fmt_df and proceed.

4. Add to the *if-elif* block in 'format_raw_data.py'.

If the file does not need the offsets, follow the QCM-I formatting:

```
elif src_type == 'QCM-i':
    formatted_df = format_QCMi(data_df)
```

Copy/paste this above the last *else* statement in the *if-elif* block that handles the error case. Then replace the name of the function defined in step 2, and replace the string in quotes that is being evaluated against the 'src_type' variable, with a sensible device identifier. This will be referenced later in the GUI.

5. Adding the data file as an option to choose from in the UI.

Navigate to the 'srcFileFrame' class in 'main.py'. Observe how the other options are added, and insert the file source type in the same fashion.

a. Add the file source type to the list:

```
self.file_src_types = ['QCM-d', 'QCM-i', 'Qsense', 'AWSensors',
'NEW_FILE_SRC_TYPE_HERE']
```

This should match exactly what you put to compare against 'src_type' in the previous step.

b. Add a radio button for the new file source option. You can copy paste the following lines:

```
self.opt3_radio = tk.Radiobutton(self, text="QSense ", variable=self.file_src_var,
value=2, command=self.handle_radios)

self.opt3_radio.grid(row=2, column=0, columnspan=2)
```

Changing the 3 to a 4 in both instances of 'self.opt3', the string in text='Qsense' to the name of the experimental device, the number in 'value=3' to a 4, and number in 'column=0' to a 1

6. Adjust the 'handle_radios' function

In the same class as step 5, scroll down to the 'handle_radios' function. If data is recorded in absolute time, add the new source file 'file_src_type' (the string you appended to the list in step 5a) to the top if line as follows:

```
If self.file_src_type == 'QCM-d' or self.file_src_type == 'NEW_FILE_SRC_TYPE_HERE'':
```

If the data is recorded in relative time, add it to the next if line:

If self.file_src_type == ‘QCM-i’ or self.file_src_type == ‘NEW_FILE_SRC_TYPE_HERE’’:

After following these steps, proceed with data files as with any of the supported formats in pyQCM-BraTaDio.

An OpenQCM-D Next from Novaetech SRL (Pompei, Italy), controlled with openQCM-NEXT-0.1.2. A peristaltic pump (Golander LLC, BQ80S Microflow Variable-Speed) was used to inject buffers and molecule solutions. The native format of the experimental data is already in *.txt format.

File using the “add all” (to include all raw data collected) option before exporting, SI Figure 8(b). OpenQCM-Next generates a file containing all raw data in *.txt format and does not require any export operation.

## 6.3 Description of directories and output files

pyQCM-BraTaDio generates many different output files during execution. Below is the list of directories and files that may be found in them.

- offset_data
    - contains the file ‘COPY-PASTE_OFFSET_VALUES_HERE.csv’.
    - This is the file the user will copy and paste offset values if desired.
    - Alternatively, if user opts to use the offset data window to enter values, this is the file that those values are saved to.
- qcmd-plots
    - Contains ALL plots output by the software.
        - Note that different plotting options are indicated by the file name, but files will be overwritten if the same plots are done again even with different data.
    - modeling
        - Contains all plots generated from functions in the modeling window of the UI.
- raw_data
    - default directory for choosing the data the user would like to work with in the software.
    - Will contain sample data files upon first use, however any other files are placed here by the user.
- selected_ranges

- contains output data from selections made in the interactive plot, as well as any modeling functions that utilize this data. These files include:
    - clean_all_stats_dis.csv and clean_all_stats_rf.csv will contain the statistical data from selections made in the interactive plot of the baseline corrected data, as well as the user-specified name of the range, x-axis (time) bounds, and data file origin.
    - Crystal_thickness_output.csv – overtone, offset values, curve fit of the offset values, and the crystal's thickness corresponding to each overtone.
    - raw_all_stats_dis.csv and raw_all_stats_rf.csv are the same as the clean variant, just for the raw data.
    - Sauerbrey_output.csv – contains overtone, average change in frequency with error, curve fit value of the average change in frequency, average Sauerbrey mass with error, quartz crystal constant calculated by the software, name of the range these values originate from, and the data file the data originates from
    - Thin_film_air_output.csv
    - Thin_film_liquid_output.csv – contains overtone order times the change in frequency for each overtone and its corresponding bandwidth shift, curve fit value of the bandwidth shift, and the range name and data source these values originate from

## 6.4 Obtaining calibration files

Of the natively supported devices, openQCM Next, QCM-I, and QSense, QSense is the only device that generates data that will require an offset value file for more extensive analysis. The offsets are unnecessary if one only wishes to use basic visualization. However, more than just the deltas, $\Delta f$ and $\Delta D$, are required for any functionality beyond. openQCM Next and QCM-I record these values and do not need offset values. For QSense, after the experiment is completed, a *.QSD (or equivalent proprietary) file is generated. This file can only be accessed using QTools (or equivalent proprietary) software. Once opened, the data from the experiment is shown as:

$$Displayed\ frequency\ value = \frac{(real\ value - offset)}{overtone}$$

$$Displayed\ Dissipation\ value = \frac{(real\ value - offset)}{(1 \times 10^{-6})}$$

The file can be exported either as an *.txt or *.xls file. As mentioned earlier, in order to further analyze data using pyQCM-BraTaDio, the real frequency and dissipation value is needed. These values will be later entered either into the offset value window in the UI, or pasted directly into the 'offset_data/ COPY-PASTE_OFFSET_VALUES_HERE.csv' file

## 7 Model experiments

For the data visualization in the main text, we provide here the experimental details for how that data was obtained.

### 7.1 QCM-D experiment

*Materials*

Phosphate Buffer Saline or PBS (Gibco, Catalog No: 10-010-031), Ethanol (ACROS, absolute, 200 Proof, ≥ 99.5%, Product # 61509-5000, Lot # B0542545A), Sodium dodecyl sulfate (MP Biomedicals, ultra-pure, ≥99%, Catalog # 811032, lot # S0709) were purchased. For making base piranha solution: Ammonium Hydroxide (Chemsavers, Product # AMHE500ML, lot # AMHE080521, 28-30% pure), hydrogen peroxide (PERDROGENTM by Honeywell, MDL # MFCD00011333, ≥30% (w/w) stabilized) were used. Gold-coated silica quartz crystals were purchased from Quartz PRO (Product # QCM5140CrAu120-050-Q, resonance frequency- 5 MHz). Ultrapure water was collected from Thermo Fisher Millipore UV water purification system. Bovine Serum Albumin was purchased from Sigma Aldrich (Product # A3294) and prepared at 1 mg/ml in PBS at pH 7 to mimic physiologically relevant conditions.

*Methods*

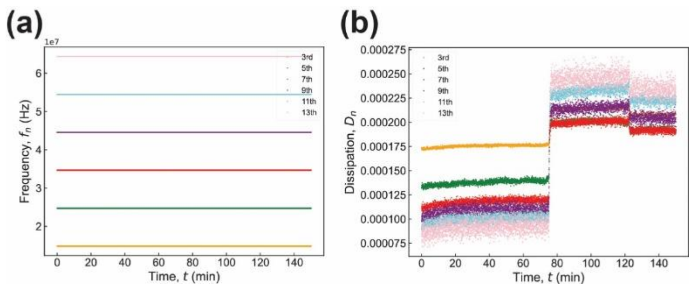
Surface preparation for Quartz Crystal Microbalance with Dissipation (QCM-D)

The gold-coated crystals of 5 MHz base resonance frequency were rinsed copiously with ultrapure water (18.2 MΩ.cm), 2 wt% SDS, and ethanol, respectively, and repeated 3 times. After rinsing, the surfaces were dried with a stream of N2, cleaned with oxygen plasma, and stored in Petri dishes before use.

After using the crystals for protein adsorption experiments in the QCM-D, they were reused after cleaning in base piranha (6:1:1 by volume of water: ammonium hydroxide: hydrogen peroxide) by submerging the surfaces for ~20 seconds at 600 °C. Then the crystals were further rinsed with water and ethanol copiously several times, and then dried with $N_2$.

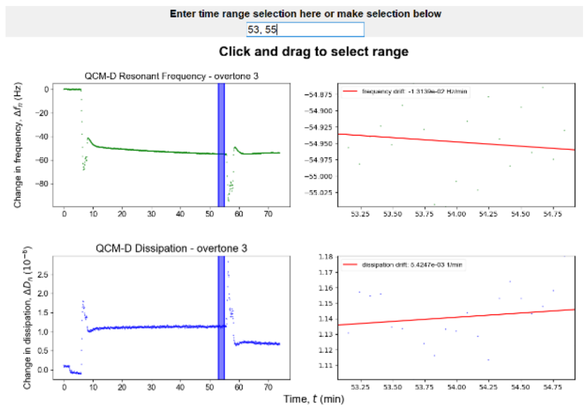
*Quartz Crystal Microbalance with Dissipation (QCM-D) setups*

QCM-D is a non-destructive acoustic shearing technique that uses a piezoelectric quartz sensor to measure changes in frequency and dissipation in real-time. The adsorption of molecules to the gold-coated crystal was assessed in a QCM-I by Gamry MicroVacuum Ltd. (Budapest, Hungary) or QSense (Biolin



**SI Figure 8.** Change in mass observed on the sensor via frequency and dissipation vs time.

Scientific, Sweden). All measurements were at 25 °C. PBS was circulated through the system and allowed approximately 1 hr to equilibrate and establish the baseline. Then the BSA in PBS solution at 1 mg/ml was flowed in the chamber for ~3 mins and then the pump was stopped to

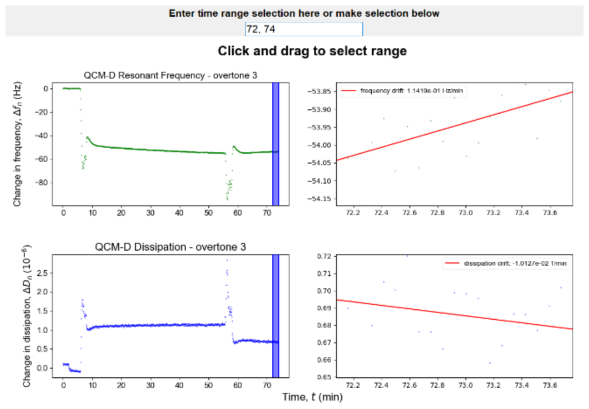

Enter time range selection here or make selection below

53, 55

Click and drag to select range

QCM-D Resonant Frequency - overtone 3
Change in frequency, $\Delta f_n$ (Hz)
frequency drift: -1.3130e-02 Hz/min
QCM-D Dissipation - overtone 3
Change in dissipation, $\Delta D_n$ ($10^{-6}$)
dissipation drift: 5.4247e-03 1/min
Time, $t$ (min)

**SI Figure 9.** Before PBS wash for QCM-I analysis.


Enter time range selection here or make selection below

72, 74

Click and drag to select range

QCM-D Resonant Frequency - overtone 3
Change in frequency, $\Delta f_n$ (Hz)
frequency drift: 1.1419e-01 Hz/min
QCM-D Dissipation - overtone 3
Change in dissipation, $\Delta D_n$ ($10^{-6}$)
dissipation drift: -1.0127e-02 1/min
Time, $t$ (min)

**SI Figure 10.** After PBS wash for QCM-I analysis.

allow the system to reach equilibrium for 30 mins to an hour. The change in mass on the sensor was visualized by plotting the changes in dissipation (ΔD) as a function of changes in frequency (Δf) for different substrates. The higher the negative frequency shift (−Δf), the higher the adsorbed mass; on the other hand, the higher the dissipation or the slope of the lines, the more viscous or hydrated the molecular films were. After the changes in frequency or dissipation signals reached equilibrium, PBS was circulated again to wash away any excess amount of unbound BSA present on the surface. This resulted in a slight reduction in the frequency shift. After waiting for 10-30 mins for the rinsing step to reach equilibrium the experiment and the data collection was stopped.

After every experiment, water and 2% SDS, followed by water, were circulated into the system to clean the inside of the tubing and the connection ports. Then, after purging the chamber of the solutions, the crystal was removed from the chamber. Each measurement was performed at least 3 times independently (N = 3).

The BSA experiment with a QCM-I device described above was replicated in a similar manner using a QSense device in order to test and ensure the consistency of pyQCM-BraTaDio across multiple experimental devices. This experiment is described below.

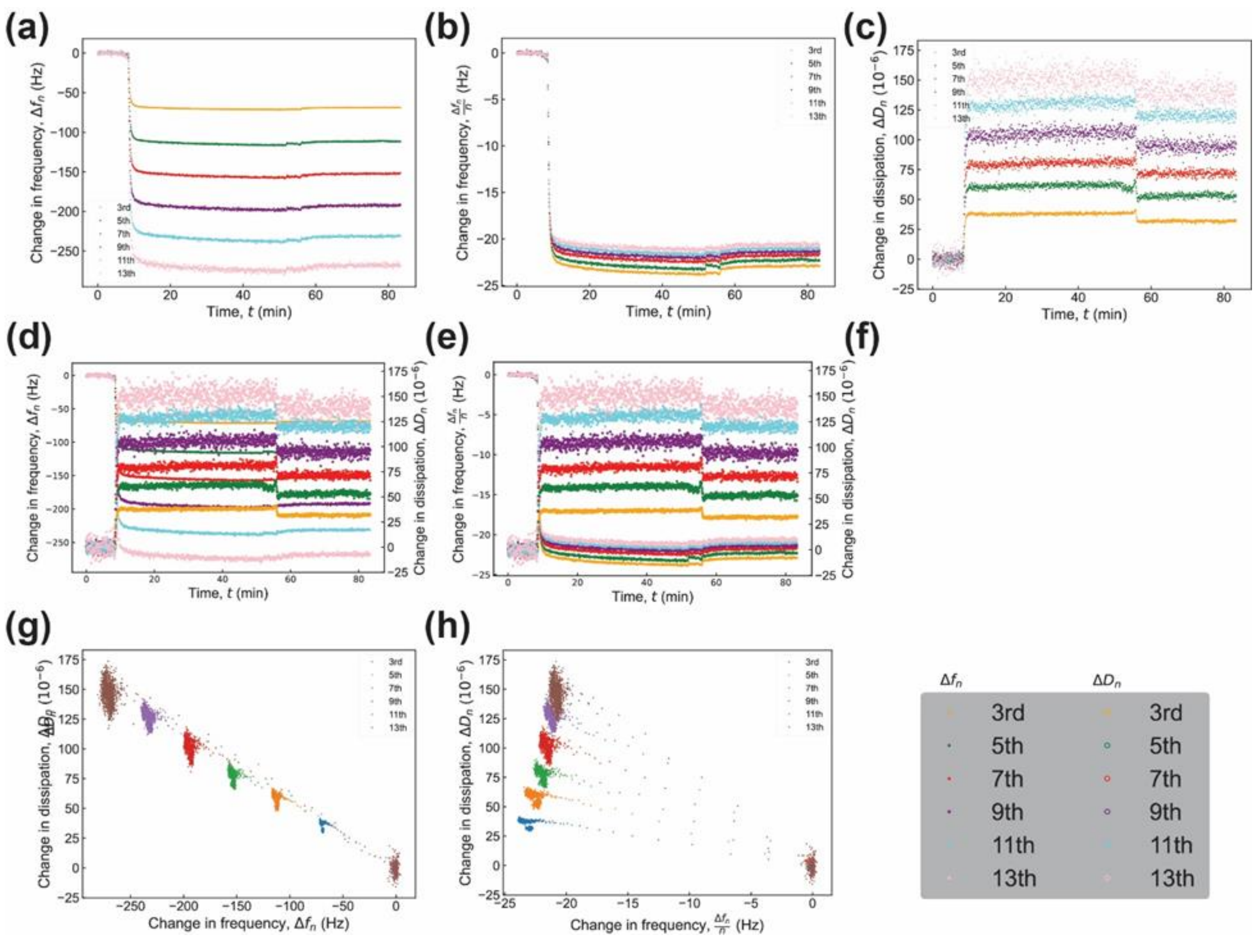
Prior to use, gold-coated quartz sensors were placed in a 5:1:1 $H_2O/H_2O_2/NH_4O$ mixture at 75°C for 10 minutes and later removed to be rinsed thoroughly with deionized water and dried with nitrogen gas.

Quartz Crystal Microbalance with Dissipation (Q-Sense E1, Biolin Scientific) experiments were performed to determine BSA real-time adsorption behavior by measuring frequency and dissipation shifts. A 5 MHz clean quartz crystal sensor was used to record a stable frequency and dissipation baseline in air. Then the chamber was filled with PBS pH 7.4 buffer and allowed to reach a steady baseline, next switched to the BSA solution at a 1 mg/ml concentration. Finally, PBS is introduced again to wash the excess and measure the adsorbed BSA.

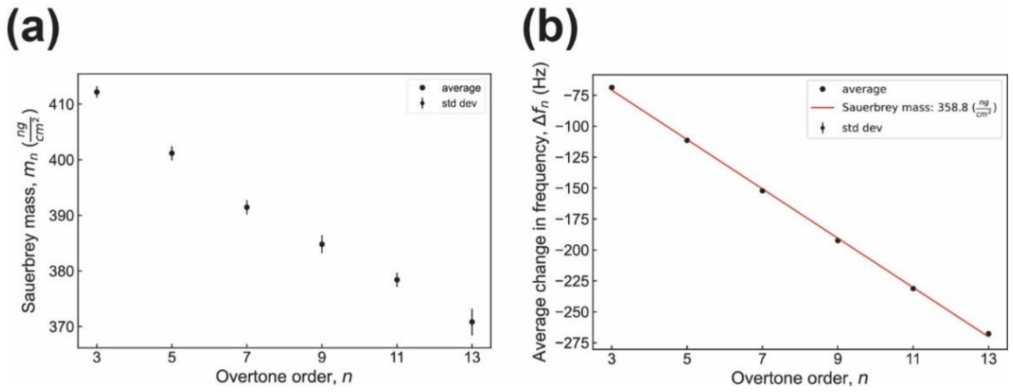
**SI Table 2.** Comparison of results from similar experiments between QCM-I and QSense. Note the fundamental overtone is omitted here due to the inherent noise it possesses. QCM-I selection was made in range t = [3138, 3305] seconds, and QSense in range t = [8725, 8986] seconds.

| | **QCM-I** | **QSense** |
|---|---|---|
| **3rd Overtone Change in Frequency Average, *$\Delta f$* (Hz)** | -55.03 | -72.24 |
| **3rd Overtone Change in Dissipation Average, *$\Delta D$* ($10^{-6}$)** | 1.025 | 1.188 |
| **5th Overtone Change in Frequency Average, *$\Delta f$* (Hz)** | -87.74 | -116.5 |
| **5th Overtone Change in Dissipation Average, *$\Delta D$* ($10^{-6}$)** | 1.206 | 1.173 |
| **7th Overtone Change in Frequency Average, *$\Delta f$* (Hz)** | -123.3 | -159.3 |
| **7th Overtone Change in Dissipation Average, *$\Delta D$* ($10^{-6}$)** | 1.403 | 1.141 |
| **9th Overtone Change in Frequency Average, *$\Delta f$* (Hz)** | -157.8 | -200.7 |

| | | |
|---|---|---|
| **9th Overtone Change in Dissipation Average, $\Delta D$ ($10^{-6}$)** | 1.585 | 1.139 |
| **11th Overtone Change in Frequency Average, $\Delta f$ (Hz)** | -195.8 | -241.5 |
| **11th Overtone Change in Dissipation Average, $\Delta D$ ($10^{-6}$)** | 1.870 | 1.172 |
| **13th Overtone Change in Frequency Average, $\Delta f$ (Hz)** | -235.8 | -279.9 |
| **13th Overtone Change in Dissipation Average, $\Delta D$ ($10^{-6}$)** | 2.081 | 1.126 |
| **Sauerbrey Mass $\left(\frac{ng}{cm^2}\right)$** | 324.4 | 372.5 |
| **Crystal Thickness (mm)** | 0.3368 | 0.3371 |
| **Shear Dependent Compliance $\left(\frac{1}{Pa}\right)$** | 0.0204 | -0.00788 |



**SI Figure 11. Plots generated by BraTaDio for a film formed from a solution of BSA at 1 mg/mL in PBS adsorbed to an Au-coated quartz crystal.** (a) Change in frequency $\Delta f_n$ as a function of time $t$, (b) change in frequency normalized by overtone order, $\Delta f_n$ /$n$ as a function of time $t$, (c) corresponding change in dissipation $\Delta D_n$ as a function of time, (d) change in frequency normalized by overtone order, $\Delta f_n$ /$n$ and corresponding change in dissipation $\Delta D_n$ as a function of time for $n$ = 5 and 7 for clarity, (e) change in dissipation $\Delta D_n$ as a function of change in frequency normalized by overtone order, $\Delta f_n$ /$n$, for $n$ = 3, 5, and 7 for clarity, and (f) temperature T as a function of time. Data collected with a QSense system.

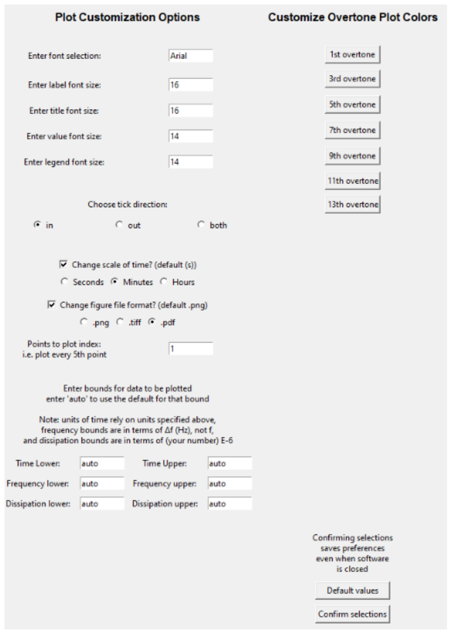


**SI Figure 12. Sauerbrey mass as a function of overtone order.** (a) Sauerbrey mass calculated as a funciton of each individual overtone order and (b) Sauerbrey mass calculated from performing a fit to the average change in frequency as a function of overtone order for the data range shown in SI Figrue 4 (BSA after PBS wash).

## 8

## Customize Plot Options

The aspect of the plots can be customized by accessing the *Customize Plots Option* button. The basic customization parameters are font size, font type, color palette, tick direction, time scale for plots in which the data is plotted as a function of time, figure output formats, and plotting ranges. SI Figure 13 shows the various plot customizations, while SI Figure 14 shows two plots of the same data using different plotting options.



**Plot Customization Options**

Enter font selection: Arial
Enter label font size: 16
Enter title font size: 16
Enter value font size: 14
Enter legend font size: 14

Choose tick direction: [✓ in] out both

[✓ Change scale of time? (default (s))]
Seconds [✓ Minutes] Hours

[✓ Change figure file format? (default .png)]
.png .tiff [✓ .pdf]

Points to plot index:
i.e. plot every 5th point: 1

Enter bounds for data to be plotted
enter 'auto' to use the default for that bound

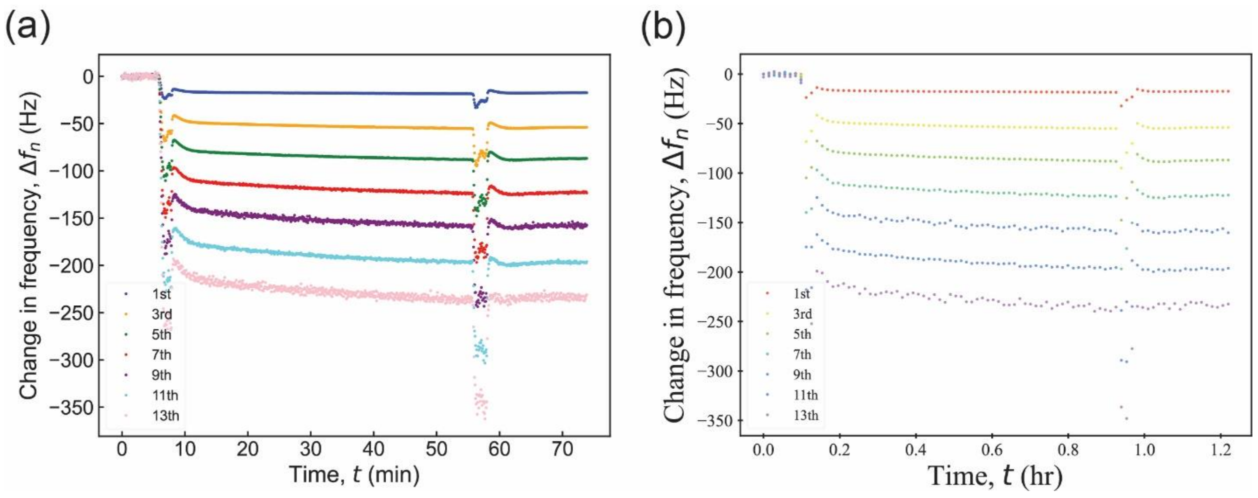
Note: units of time rely on units specified above,
frequency bounds are in terms of Δf (Hz), not f,
and dissipation bounds are in terms of (your number) E-6

Time Lower: auto  Time Upper: auto
Frequency lower: auto  Frequency upper: auto
Dissipation lower: auto  Dissipation upper: auto

**Customize Overtone Plot Colors**

Confirming selections saves preferences even when software is closed

**SI Figure 13.** Plot customization options window



**SI Figure 14.** (a) Example plot using Arial font sizes 16 and 14 for axes titles and values, respectively, plotting every data point with pre-determined color palette. (b) Example plot using Times New Roman font sizes 20 and 12 for axes titles and values, respectively, plotting every 5th data point with a custom color palette.

## 8 Notable package dependencies

pyQCM-BraTaDio utilizes several well-established open-source python packages. Most notably, Pandas,[26] Numpy,[27] for a variety of computational tasks, Tkinter for the interactive plotting engine and application user interface (UI),[28] Matplotlib[29] for the display of scatter plots and models and interactive plot interface, and SciPy[30] for much of the model application capabilities. Pandas is a package providing flexible data structures to make working with relational or labelled data easy. It provides an intuitive way of working with data from spreadsheets, interacting with it with code. It is a high level, fundamental building block data for analysis in Python. NumPy is another core building block, as all the other libraries used rely on it as well. NumPy is used in almost every field of science and engineering in Python, containing multidimensional array and matrix data structures, allowing for easy operations on complexly related data. The number of available operations is enormous, and they are also guaranteed an optimized runtime. Matplotlib is the core visualization tool used for this software. It is a comprehensive library for designing and efficiently plotting static, animated, and interactive plots in Python. SciPy is the final pivotal library for this software. It provides highly optimized fundamental algorithms for model applications and analysis, such as integration, interpolation, statistics, and more. SciPy is the library behind our model application routines. It is a very powerful tool that is relatively simple to use in conjunction with our other libraries.