The ALC_EQCM manual
version 1.3

Ivan Scivetti (main author)
Gilberto Teobaldi (edition support)

Ada Lovelace Centre, The Science and Technology Facilities Council, United Kingdom

July 25, 2024

About the code

ALC_EQCM is a software for post-processing Electrochemical Quartz Crystal Microbalance (EQCM) data. The implemented functionalities allow quantitative characterization of electrochemical processes and interpretation of stoichiometric changes. This information is used to automatically generate atomistic models compatible with EQCM data, together with appropriate settings for atomistic level simulations of the derived models. These software capabilities constitute an alternative tool that aims to bridge experimental and computational research of complex electrochemical reactions.

The project started in April 2020 at the Ada Lovelace Centre (ALC) of the Science and Technology Facilities Council (STFC). ALC_EQCM is a serial code written in modern Fortran according to the 2008 standards. Its structure for development and maintenance follows the Continuous Integration (CI) practice and it is integrated within the STFC-GitLab DevOps. The code is open-access, available in STFC-GitHub under the BSD 3-Clause License

We hope the user finds this manual useful, particularly during the initial challenges to familiarise with the capabilities of the code. We remind the user that this manual is a living document, which is subject to modifications and updates. For this reason, we strongly advise users to consult the latest version.

Disclaimer

The ALC does not fully guarantee the code is free of errors and assumes no legal responsibility for any incorrect outcome or loss of data.

Citing ALC_EQCM

Please acknowledge our effort by citing the following papers when generating results using ALC_EQCM:

• Quantitative Resolution of Complex Stoichiometric Changes During Electrochemical Cycling by Density Functional Theory Assisted, Electrochemical Quartz Crystal Microbalance. T-H. Wu; I. Scivetti; J-C. Chen; J-A. Wang; G. Teobaldi, C-C Hu; L.J. Hardwick. ACS Appl. Energy Mater. 3, 4, 33473357 (2020), https://doi.org/10.1021/acsaem.9b02386.

• ALC_EQCM: Automated stoichiometric resolution in electrochemistry through Density Functional Theory aided, Electrochemical Quartz Crystal Microbalance. I. Scivetti and G. Teobaldi. Computational Materials Science 218, 111968, (2023), https://doi.org/10.1016/j.commatsci.2022.111968.

Contents

1 Introduction

Interpretation of experiments to characterize structure and properties of materials and surfaces rests on quantitative knowledge of the atomic composition (stoichiometry) of the measured samples. The capability to simultaneously measure changes of mass and charge of electro-active materials makes Electrochemical Quartz Crystal Microbalance (EQCM) a powerful technique to monitor stoichiometric changes during electrochemical processes. Quantitative stoichiometry resolution during electrochemical cycling is routinely derived by solving the system of two equations for EQCM mass and charge conservation. These equations are coupled through the assumed stoichiometry of the redox process [1, 2].

For electrodeposition processes, knowledge of mass and charge variations is often enough to withdraw quantitative resolution [3, 4]. For ion-intercalation, in contrast, quantitative resolution is invariably lost if the process takes place via co-intercalation of electrolyte solvent molecules, as well as displacement of structural solvent from the host electrode. In other words, whenever more than two chemically inequivalent species are involved in the intercalation process, the system of equations of mass and charge conservation is mathematically underdetermined, with more independent variables (the stoichiometric coefficients of the redox process) than equations [1]. This leads to a set of infinite possible EQCM-compatible stoichiometric compositions and, thence, quantitative uncertainty for the stoichiometry of the material following the reaction. This situation hinders atomistic interpretation of experiments of electrode materials in liquid electrolytes, thus preventing the development of improved solutions for existing battery technologies.

For layered and nanoporous ion-intercalation host materials, widespread practical, yet uncontrolled, approximations for analysis of EQCM data have been the use of arbitrarily defined solvation numbers for the intercalating ions, as well as the neglect of ion-induced displacement of structural solvent inside the host [2]. To investigate further on these ad-hoc approximations, in March 2020 we proposed an approach to building atomistic models that are compatible with EQCM-measurements [1]. These models are subsequently optimised using atomistic simulations. Analysis of the computed results allows identifying energetically and thermodynamically favoured stoichiometries and structures and, consequently, extracting quantitative resolution of stoichiometric changes during electrochemical cycling. This approach, developed at the Scientific Computer Department (SCD) of the Science and Technology Facilities Council (STFC), is applicable to any electrode material that can be electrochemically grown or deposited on an EQCM-electrode.

Despite the novelty of this EQCM-based-computational approach, the complexity of building models manually from EQCM data is substantial, resulting in several tedious and time-consuming steps. To overcome such barriers, here we present ALC_EQCM, a new software suite capable to automate the approach with minimum user's intervention. Through specification of relevant directives, ALC_EQCM can also provide input files for atomistic simulations of the generated structures, as well as scripts to execute simulations in High-Performance-Computing (HPC) facilities. The implemented capabilities allow building models of electro-deposited surfaces as well as host crystals with intercalated species, thus offering the possibility to compute EQCM-compatible structures automatically for the first time, and investigate the intricacies of fundamental electro-induced reactions at the nano-scale level.

We aimed to prepare this manual as user-friendly as possible, in such a way it is accessible to experimental electrochemists with basic knowledge in modelling as well as computational researches with expertise in atomistic simulations with no electrochemistry background. In Sec. 2 we provide a brief and general description of EQCM principles and applications. Users should have a basic knowledge of Calculus and Linear Algebra to understand the theoretical grounds behind the implemented functionalities (Sec. 3). For users interested in EQCM data analysis, section 3.1 to 3.8 describe the implemented functionalities for data processing, quantification of EQCM related variables and stoichiometric analysis. For those users interested in the generation of atomistic structures, the implemented algorithms are described in sections 3.10 and 3.11. Functionalities to generate input files for atomistic simulations and HPC scripts are explained in sections 3.12 and 3.13, respectively. Input-Output file structure is presented in Sec. 4, together with a detailed explanation of available input directives. Instructions for code use and development are given in

Sec. 5. Tutorial exercises of Sec. 6 are intended to help the user to familiarize with the different capabilities of the code through explanatory examples.

We hope the user finds this manual useful and stimulating to investigate EQCM experiments with ALC_EQCM, adding extra value to the already available tool-kit for fundamental research in electrochemistry.

2 Electrochemical Quartz Crystal Microbalance (EQCM): description

In this section we provide a brief general description of the principles and challenges of Electrochemical Quartz Crystal Microbalance (EQCM). For further details we refer the interested user to specialised bibliography.

2.1 Principles of Quartz Crystal Microbalance (QCM)

Charge-transfer reactions for electrodes in solution are commonly affected by the characteristics of the electrode-electrolyte interface. To investigate the intricate underlying mechanisms at these interfaces, surface probe techniques such as Quartz Crystal Microbalance (QCM) have played a crucial role, particularly in the research of heterogeneous electron transfer [5, 6, 7, 8, 9, 10]. Basically, a QCM device is a mass detector composed of a quartz disk in between a top and a bottom electrode, as schematically shown in Fig. 1 (Left). The quartz is exposed to an oscillating external field created in between the electrodes. Due to the inherent piezoelectric properties of the quartz crystal, the oscillating electric field induces mechanical vibrations. For AT-cut quartz [11], shear oscillations are predominant and can be induced efficiently to be parallel to the surface of the crystal. The mechanical response of the quartz is subject to the frequency and the magnitude of the external oscillations and depends on the physical properties of the quartz as well as the surrounding media.

The success of QCM relies on the designed circuit to accurately detect changes in the quartz resonant frequency [12], which are correlated to mass changes at the surface of the electrode. What makes QMC unique is the ability to sensitively monitor mass changes (e.g., 17.7 ng/cm^2/Hz for a 5 MHz quartz crystal) over small time scales, which allows access to mass transport phenomena on a molecular scale [13].

Nowadays, commercial QCM devices offer the possibility to measure frequency changes Δf of vibrating quartz crystals immersed in liquid, while simultaneously monitoring the energy dissipation change ΔD (or the half-bandwidth variation $\Delta \Gamma$) for multiple overtones [14]. This technique is known as QMC-D and allows extracting detailed information of the physical properties of solid-liquid interfaces, including dynamic adsorption and desorption process [9], viscoelasticity and density of liquids [15], electric double layer structure [16, 17] interfacial slip [18] friction [19], etc. QCM has also been combined with ac-electrogravimetry and Electrochemical Impedance Spectroscopy (EIS) to detect separate contributions from anions, cations and free solvent [20, 21].

2.2 Inducing electrochemical reactions with QCM

To trigger electrochemical reactions, QCM is combined with a device to control the applied external voltage and measure the associated current. This setup is known as Electrochemical QCM (EQCM), as schematically shown in Fig. 1 (Right). In an EQCM device, the top electrode of Fig. 1 (Left) is set to be in contact with the electrolyte within the electrochemical cell. The top electrode becomes the working electrode (WE) where the chemical reaction takes place. A voltage difference is applied between the WE and the counter electrode (CE) via a potentiostat, whereas the reference electrode (RE) is used to measure the potential against the other electrodes. The chemical reaction is triggered by changing the applied voltage, and the resulting current is measured at the WE. Mass changes at the WE are obtained from the changes in the QCM frequency (sec. 2.3.1). For further technical details of EQCM and its applications, the user is referred to Refs. [7, 22, 23, 24, 25].

EQCM is designed to induce electrodeposition of chemical species present in the electrolyte, depicted by the green thin slab. This green slab could also represent a host material (deposited ex-situ) and the EQCM is used to trigger (de)intercalation of chemical species. Both electrodeposition and intercalation reactions have been considered in the development of ALC_EQCM.
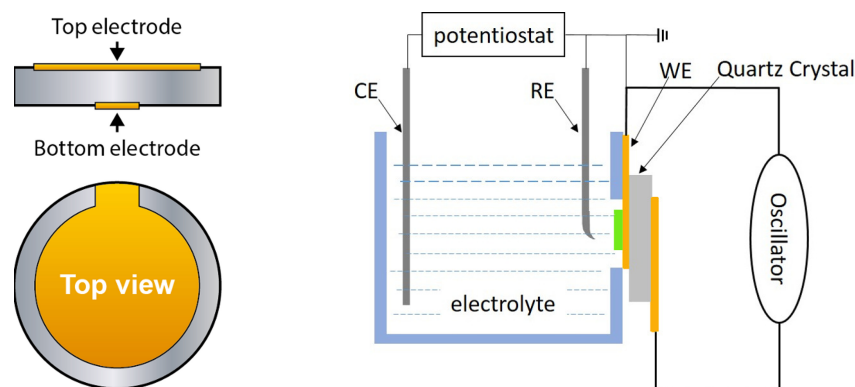
Figure 1: Left: Schematic side and top views of a QCM device with the quartz disk (grey) in between the top and bottom electrodes (orange). Right: Schematic illustration of the EQCM experimental setup. The quartz is exposed to an oscillating external field in between the electrodes. The top electrode is set to be the working electrode (WE) in contact with the electrolyte of the electrochemical chamber, connected to the counter electrode (CE) and the reference electrode (RE). A potentiostat controls the voltage difference between the WE and the CE, and triggers the electrochemical reaction at the WE/electrolyte interface (green slab). Mass changes at the electrode are determined from changes of the quartz crystal frequency. See text for details. Figures adapted from Ref. [26] and [27].

As a non-destructive technique, EQCM also allows in operando study of mechanical property modifications at the electrode/electrolyte interface. Monitoring of frequency changes and energy dissipation for multiple overtones enables probing of the electrode on different length scales [28]. This allows tracking the formation/modification of SEI layers at Li batteries electrodes [25, 29, 30] and investigating the intercalation of lithium in various materials [31, 32]. In addition, multiharmonic response allows analyzing the influence of binders on the mechanical properties of composite electrode materials [33]. EQCM can also be combined with ac-electrogravimetry to experimentally assess the chemical structure and dynamics of the electrical double layer [34]. Other studies have combined EQCM with differential mass spectroscopy [35], impedance and electro-optical transfer functions [36], as well as EIS, Atomic Force Microscopy and Fourier Transform infrared spectroscopy [37].

Regarding SEI, the evolution of mass and the mechanical properties are directly correlated to the electrode impedance. In this context, combined in situ EIS-EQCM-D has demonstrated to provide unique prospects for further studies of the highly dynamic structure-function relationships of electrode interphases in Li ion batteries [38].

Typical responses of current and frequency changes Δf for ion intercalation are shown in the left and right panels of Fig. 2, respectively. This data correspond to the first EQCM cycle of a $Ni(OH)_2$ sample immersed in a 1M solution of LiOH [1]. The voltage is first increased from 0 to 0.45 V (dV>0, black). At approximately 0.37 V, we observe a peak in the measured current, which is correlated to a decrease in the frequency. As we shall see in sec. 2.3.1, a decrease in frequency corresponds to an increase of mass at the WE. This segment of the electrochemical cycle is related to the oxidative intercalation of Li^+ into the host $Ni(OH)_2$ [1]. When the voltage is reversed from 0.45 to 0 V (dV<0, red), a negative peak in the current is observed at approximately 0.27 V, which is correlated to the increase in Δf and a corresponding decrease in mass. This segment is associated to the reductive de-intercalation of Li^+ from the host $Ni(OH)_2$.

With exemption of very specific cases (see Refs [2], [7] and [20] for example), EQCM experiments have been used to investigate electrochemical processes qualitatively. ALC_EQCM aims to provide a software tool to allow quantitative resolution of electrochemical reactions using EQCM charge and mass variations to automatically calculate stoichiometric changes and generate atomistic models of structures compatible
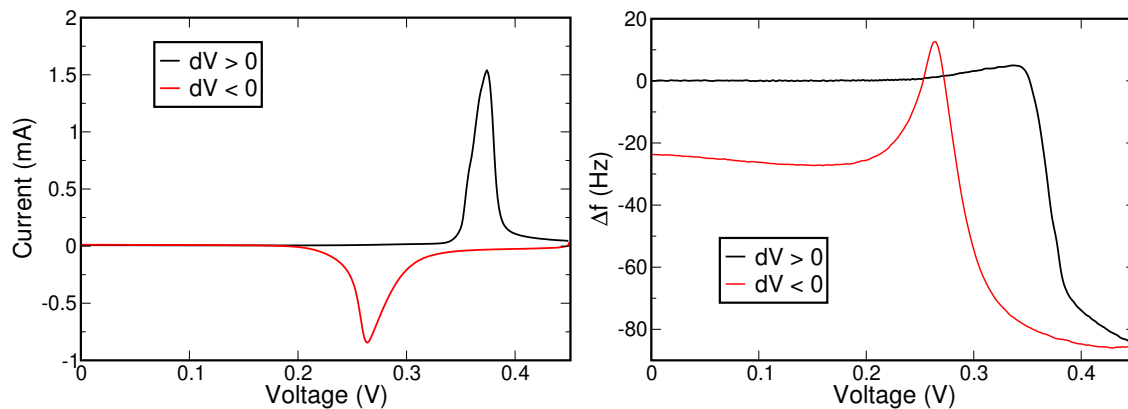
Figure 2: Example of experimental EQCM data. Current (Left) and Δf (Right) as a function of the applied voltage for a host Ni(OH)$_2$ immersed in a 1M solution of LiOH [1]. Responses for positive (dV>0) and negative (dV<0) voltage sweep are shown in black and red, respectively.

with EQCM data.

2.3 Correlating mass with frequency changes

Although EQCM(-D) constitutes an advanced and highly sensitive technique, only data analysis of rigid films in the gas/liquid phase is well established. In fact, EQCM(-D) generally needs complex correlations to extract information about the local deformation, electrolyte viscosity, among other parameters [9]. Thus, quantitative analysis of QCM results depends on the model assumed for the vibrating system to establish the relationship between Δf (frequency changes), ΔD (energy dissipation changes), and $\Delta\Gamma$ (half-width variations) and "macroscopic" physical properties of the surface loads, such as density, viscosity, elasticity, and film thickness. In this context, the influence of surface wetting [23] and interfacial slip at the solid-liquid interface [18, 39, 40], surface roughness [39], microstructure [41] and nanoscale confinement [42] must be carefully incorporated [9].

The challenge to relate the frequency response with the properties of loads is particularly relevant to LIBs, for example, where a complex interplay between Li+ (de)intercalation, SEI formation and changes of the electrolyte viscosity affect the frequency response during battery cycling. These effects significantly complicate the task to obtain the mass variation associated with the redox process.

To the purpose of the present work, we classify available models for EQCM analysis into the Sauerbrey type and those beyond the Sauerbrey limit, which comprise viscoelastic and hydrodynamic models. We remark ALC⌐ must not be considered as a software tool to correlate mass and frequency, but a software to derive stoichiometric resolution from EQCM mass and charge accumulation. ALC⌐EQCM only allows computing the mass load in the limit of the Sauerbrey approximation, for which the user must corroborate its validity for the case under consideration and provide a correct conversion factor. For all other cases where the Sauerbrey conditions are not fulfilled, the user must pre-process the EQCM and provide mass variations as input data for ALC_EQCM execution.

2.3.1 The Sauerbrey model

Independently of the elctrochemical process under consideration, the basic assumption for the Sauerbrey model is the surface load forms a thin, rigid film that can be considered an extension of the supporting quartz crystal. Thus, the acoustic impedance of the unloaded QCM device must remain unaltered during the accumulation of the new material. For such a situation, the shifts in bandwidth are much smaller than

the shifts in frequency. Further, the overtone-normalized shifts in frequency agree between overtones. Only under these conditions, viscous effects can be ignored, and the conversion from frequency changes Δf to mass variation Δm relies on the Sauerbrey equation [7, 43]:

$$\Delta m = -\frac{A_{disk}\Delta f}{C_{f0}} \equiv \Delta \tilde{m} = -\frac{\Delta f}{C_{f0}} \tag{1}$$

where A_{disk} is the electrode disk area and $\Delta \tilde{m} = \Delta m / A_{disk}$ is the mass-density. The theoretical characteristic constant, C_{f0}, is given by the following expression:

$$C_{f0} = \frac{2f_0^2}{\sqrt{\mu_q \rho_q}} \tag{2}$$

where f_0 is the resonant frequency of the fundamental mode, whereas μ_q and ρ_0 are the density and the shear modulus of the AT-cut quartz crystal, respectively. Consequently, C_{f0} a fundamental property of the quartz crystal and independent of the properties of the foreign film.

ALC_EQCM allows the conversion of Δf to Δm by defining the Sauerbrey factor C_{f0}. It is users' responsibility to make sure that the conditions of validity for the Sauerbrey equation are fulfilled. To prevent errors in the input values, the code will abort the execution if the maximum value for Δf is larger than the 2% the value of the quartz crystal fundamental frequency.

For intercalation processes, this correlation is often a very good approximation only if the pristine host electrode material is sufficiently thin in comparison to the quartz thickness. Intercalated species become part of the electrode material, and the gain in mass can confidently be assumed to take place uniformly within the material.

For electrodeposition processes, in contrast, species will accommodate to the morphology of the electrode surface of effective area A_{eff} ($A_{eff} \neq A_{disk}$). Analogous to Eq. (1), the frequency shift Δf can be correlated to the effective mass variation, Δm_{eff}, through the following expression:

$$\Delta f = -\frac{C_f \Delta m_{eff}}{A_{eff}} \tag{3}$$

where C_f is the experimental characteristic constant. As real surfaces exhibit roughness, A_{eff} and C_f are often unknown [44]. Despite this uncertainty, we will show in section 3.8 that the Eqs. (1) and (3) can be used together with the charge conservation to derive an expression for the surface rugosity.

2.3.2 Beyond the Sauerbrey limit: viscoelastic and hydrodynamic models

When viscoelastic and/or hydrodynamic effects become important, the Sauerbrey equation loses validity and complex data modelling is required to extract information for the frequency response. In contrast to Eq. eq:sauerbrey, the sensor mass change is no longer linearly correlated with the sensor resonance frequency change and a shift in resonance bandwidth/dissipation is observed. The values for the normalized resonance frequency changes can also start to deviate from each other as a function of harmonic overtone.

For viscoelastic models, multiple overtones should be utilized to achieve good fits. Polymers swollen with solvent and polymers above the glass transition temperature are example of systems that need viscoelastic corrections. On the other hand, when different overtones will exhibit different shifts in frequency, data analysis require hydrodynamic models, which are currently not included in most commercial fitting software and must be conducted by the user. This is normally the case of porous or particulate films in contact with a liquid. Details of viscoelastic and hydrodynamic models are beyond the scope of the present paper and the reader is referred to Ref. [9] and [13] and references therein.

ALC_EQCM can also be used for systems that exhibit a frequency response beyond the Sauerbrey limit. Nevertheless, mass changes must be provided as input data. It is the responsibility of the user to apply

suitable correlations to obtain correct values for mass changes. The mass-frequency can also be reported, but the code will use the values of mass. If no mass is recorded, the code will apply the Sauerbrey correction to compute the mass-density changes.

3 Software functionalities

In this section we describe each of the functionalities implemented in ALC_EQCM. Hereinafter, and for the sake of clarity, we will use specific fonts in the main text to differentiate between a ***directive*** (with a given *option* or `specification`) and a **&Block** containing specific commands and instructions to ALC_EQCM.

3.1 Spectra analysis of EQCM data

ALC_EQCM offers the possibility to perform spectra analysis of EQCM data. This feature is particularly useful to understand sources of noise in the EQCM signals, if present. Noise can be removed by filtering the data (sec. 3.2). Spectra analysis is executed by setting the option *spectra* for directive ***Analysis***. The schematic illustration of Fig. 3 helps to identify and define the relevant variables for spectra analysis. Let us assume a set of noisy data (in brown) for a given experimental variable Y in the voltage range between V_{low} and V_{top}, measured either during positive or negative voltage sweep. We refer to such a set as $\{Y_i^{exp}\}$ composed of N_{CV} points. The transformation to the reciprocal space is performed via a Fast Fourier Transform (FFT) that requires a total number of N_{FFT} points, equal to a power of 2^N (N is a natural number). ALC_EQCM finds the appropriate value of N for which $2^N = N_{FFT}$ is the closest number to N_{CV} and $N_{FFT} > N_{CV}$. For example, if $N_{CV} = 400$, the code will determine $N = 9$ ($N_{FFT} = 2^9 = 512 > 400$). We have set a sensible maximum limit of $N \leq 15$. Thus, the number of CV points for a single voltage sweep must be lower than $2^{15} = 32768$. N_{CV} points corresponds to the actual experimental data Y_i^{exp}. The rest of the $N_{FFT} - N_{CV}$ points correspond to the padding region ($V < V_{low}$ and $V > V_{top}$). The total number of points for the padding is $N_{pad}^{top} + N_{pad}^{low}$, defined as:

$$\begin{cases} N_{pad}^{top} = N_{pad}^{low} = (N_{FFT} - N_{CV})/2 & \text{if} \quad N_{CV} \text{ is even} \\ \\ \{ N_{pad}^{low} = (N_{FFT} - N_{CV} - 1)/2 & \text{if} \quad N_{CV} \text{ is odd} N_{pad}^{top} = N_{pad}^{low} + 1 \end{cases} \tag{4}$$

in such a way the extent of the padding is evenly distributed outside the voltage range of the experiment. The discretization of the voltage domain, ΔV, is given by:

$$\Delta V = (V_{top} - V_{low})/(N_{CV} - 1). \tag{5}$$

With this construction, the voltage domain for the N_{FFT} points is now defined between V_{pad}^{low} and V_{pad}^{top} and

$$\begin{cases} V_{pad}^{top} = V_{top} + N_{pad}^{top} \Delta V V_{pad}^{low} = V_{low} - N_{pad}^{low} \Delta V \end{cases} \tag{6}$$

The variable Y in the padding region is defined to be constant. To this end, we use N_{end} points from each extreme of the set of experimental data, schematically indicated by the set of experimental data enclosed within the two green squares (Fig. 3), and set

$$Y_{pad}^{top} = \frac{1}{N_{end}} \sum_{k=(N_{CV}-N_{end})}^{N_{CV}} Y_k^{exp} \tag{7}$$

and

$$Y_{pad}^{low} = \frac{1}{N_{end}} \sum_{k=1}^{N_{end}} Y_k^{exp}. \tag{8}$$

The value for N_{end} depends on the particular set of data at hand and must be defined by the user. For example, if the data is not noisy nor flat towards the extremes of the $[V_{low}, V_{top}]$ domain, one could well
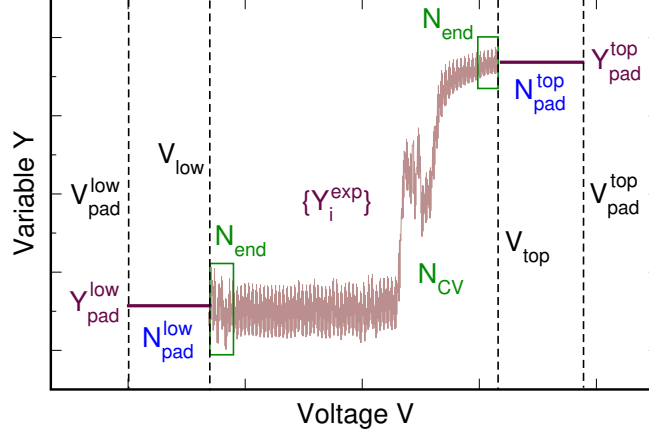
Figure 3: Schematic illustration for a hypothetical set of N_{CV} data (in brown) with noise in the voltage domain between V_{low} and V_{top}. The padding region is composed of N_{pad}^{top} and N_{pad}^{low} points outside this domain, where the value of the variable Y is set to a constant (Y_{pad}^{top} and Y_{pad}^{low}), determined by the average of N_{end} points at both extremes of the set of data (data enclosed by the green squares). See the text for further description of the variables.

assume $N_{end} = 1$, which leads to $Y_{pad}^{low} = Y_1^{exp}$ and $Y_{pad}^{top} = Y_{N_{CV}}^{exp}$. In contrast, if $\{Y_i^{exp}\}$ is flat at the extremes of the voltage domain and exhibits high-frequency noise, it is convenient to take a larger number of points for N_{end}. It is responsibility of the user to decide on the appropriate number for N_{end}. For current and mass-density (or mass, depending on the input data), the number of N_{end} points can be specified with the directive **endpoints_current** and **endpoints_mass**, respectively. Both directives have a default value of 1. We define the set of real data for the for the FFT, $\{Y_i^{FFT}\}$, using N_{FFT} points in the domain $[V_{pad}^{low}, V_{pad}^{top}]$ as follows

$$
\{Y^{FFT}\} = \begin{cases} Y_{pad}^{low} & j = 1, N_{pad}^{low} \\ \\ Y_{j-N_{pad}^{low}}^{exp} & j = (N_{pad}^{low} + 1), (N_{CV} + N_{pad}^{low}) \\ \\ Y_{pad}^{top} & j = (N_{CV} + N_{pad}^{low} + 1), N_{pad}^{top} \end{cases}
\tag{9}
$$

This construction is convenient since the FFT will interpret the set of data $\{Y_i^{FFT}\}$ to be periodic in the domain $[V_{pad}^{low}, V_{pad}^{top}]$. If the number of N_{pad}^{low} and N_{pad}^{top} points is sufficiently large, the boundary condition from the padding region will hardly affect the reciprocal space components that correspond to the experimental values.

If the elapsed time is not recorded in the experiment, the reciprocal space $\tilde{V}$ is given in the 1/V domain, where

$$
-\frac{1}{2\Delta V_{FFT}} < \tilde{V} \le \frac{1}{2\Delta V_{FFT}}
\tag{10}
$$

and

$$
\Delta V_{FFT} = \frac{V_{pad}^{top} - V_{pad}^{low}}{(N_{FFT} - 1)}.
\tag{11}
$$

In contrast, if the elapsed time is recorded, the construction of the padding is performed in time domain and the reciprocal space will be in the frequency [Hz] domain.

The set of N_{FFT} real values are assigned to the real part of a complex array of dimension N_{FFT} while the imaginary part is set to zero. The adopted implementation for the FFT follows the recursive Cooley-Tukey

algorithm [45] and has been adapted to the ALC_EQCM code based on the logic of Ref. [46]. The spectrum is given by the magnitude of complex components that correspond to the $N_{\text{FFT}}/2 + 1$ positive points of the reciprocal space, taking advantage of the FFT symmetry. Spectra are printed to the SPEC_CURRENT and SPEC_MASS files for current and mass-density/mass, respectively, within the **ANALYSIS_EQCM** folder (sec. 4.3.5).

3.2 Noise filtering

Spectra analysis helps to identify the reciprocal space components of EQCM data that are responsible for noise. ALC_EQCM offers the possibility to apply a low-pass Gaussian filter via data convolution in reciprocal space. Assuming the reciprocal space in the 1/V domain (with coordinate $\tilde{v}$), the implemented low pass filter has the following functional form:

$$H(\tilde{v}) = exp\left[-\left(\frac{\tilde{v}}{2\sigma_{cut}}\right)^2\right] \tag{12}$$

where

$$\sigma_{cut} = \frac{f_{cut}}{\sqrt{2\ ln(2)}} \tag{13}$$

and f_{cut} is the filter cutoff. From this definition, $H(0) = 1$ and $H(f_{cut}) = 0.5$, which means that at the chosen filter cutoff, the filter reduces its weight to the half. The value for f_{cut} must be specified with directive *filter_cutoff*. For a set of EQCM data $\{Y^{\text{FFT}}\}$, as defined in Eq. (9), the filtering is executed as follows:

1. **Apply FFT**

    Obtain the complex representation $\{\mathcal{Y}^{\text{FFT}}\} = \mathcal{T}[Y^{\text{FFT}}]$ in reciprocal space, where $\mathcal{T}$ is the FFT.

2. **Modify the transformed set via convolution**

    From the filter defined in Eq. (12), complex data $\{\mathcal{Y}^{\text{FFT}}\}$ is transformed to its filtered representation $\mathcal{Y}^{\text{FFT}}_{\text{filter}}(\tilde{v}) = \mathcal{Y}^{\text{FFT}}(\tilde{v})H(\tilde{v})$

3. **Apply the inverse FFT to the filtered set**

    The real space representation of the filtered data is $\{Y^{\text{FFT}}_{\text{filter}}\} = \mathcal{T}^{-1}[\mathcal{Y}^{\text{FFT}}_{\text{filter}}]$. Only the set of N_{CV} values corresponding to the experimental $[V_{low}, V_{top}]$ voltage domain are meaningful. These values will be printed to the FILTERED files inside folder **ANALYSIS_EQCM** (sec. 4.3.2).

In case the elapsed time is recorded during the EQCM experiment, the reciprocal space is be given in units of frequency (Hz). We refer the reader to sec. 6.3 for a worked example of noise removal.

3.3 Printing raw and filtered EQCM data

ALC_EQCM allows printing current and mass-density/mass data from the set of EQCM data. Raw and filtered values can be printed by choosing the option *print_EQCM_raw* and *print_EQCM_filter* for directive **Analysis**. The term "raw" in this context refers to the data as collected from the EQCM device. Printing raw data is always recommended to check the correctness of the input EQCM values as well as to identify any source of noise. Printing EQCM data is executed for all the EQCM cycles by default. To print only a subset of CV cycles, the user must define the **cycles** directive.

If the *print_EQCM_filter* option is selected, the user will be asked to define the **filter_cutoff** directive, for which it is first recommended to perform a spectra analysis to set an appropriate value.

Raw and filtered values can be printed for the whole voltage domain (given by the set of EQCM data) or for a selected range of voltages, which must be specified by directive **voltage_range**. RAW and FILTERED files for current and mass-density/mass will be generated.

3.4 Mass calibration

Deposition and removal of relatively small masses over rigid deposits can be confidently determined via the Sauerbrey's relation, as long as the offset from the viscous loading has a negligible effect [12]. In such cases, assuming a 100% Faradic efficiency for the reaction, both mass-frequency, Δf, and charge, ΔQ, are related as follows:

$$\Delta f = - \left[\frac{\mathcal{M}_X}{n_e \mathcal{F}} \frac{C_f}{A_{\text{eff}}} \right] \Delta Q \tag{14}$$

where n_e is the number of electrons transferred to induce the electro deposition (or removal) of the atomic species X with molar mass $\mathcal{M}_X$; $\mathcal{F}$ is the Faraday constant. The option *mass_calibration* for directive **Analysis** prints Δf as a function of ΔQ to file MASS_CALIBRATION inside folder **ANALYSIS_EQCM**. Only electrochemical reactions which give a linear correlation between Δf and ΔQ (e.g Ag on Pt) must be considered for mass calibration. By fitting Δf vs ΔQ to a linear function, the calculated slope can be used to determine the factor C_f/A_{eff} of the EQCM device (see sec. 6.8 for a worked example of this functionality).

3.5 Massograms

A massogram corresponds to the rate of mass-density ($d\Delta\tilde{m}/dt$) or or mass ($d\Delta m/dt$) against the applied voltage. This quantity is analogous to the voltammogram and provides a sensitive test to identify the presence of side reactions. Charge transfer processes that correlate to mass variation can be easily observed from direct comparison between massograms and voltammograms [1]. In ALC_EQCM, massograms are obtained by setting the *massogram* option for the **Analysis** directive. Computed mass rates are reported as a function of the applied voltage in file MASSOGRAM, printed inside folder **ANALYSIS_EQCM**. The user is referred to sec. 6.9 for a worked example of this functionality.

3.6 Redox characterization

By redox characterization, we refer to the calculation of mass variation and charge accumulation during oxidation and reduction. Charge and mass variations are useful not only to characterize the response of the system under consideration, but also to compute the stoichiometric changes associated with the electrochemical process. It is first important to differentiate between a segment of a CV cycle and the redox process. As example, we consider the first CV cycle for the electrodeposition of antimony (Sb), as shown in the left panel of Fig. 4. In this experiment, the potential is reduced from the starting value of 0.5 V to -1.0 V (black) and increased again to 0.6 V (red). We define both CV fragments as negative (dV<0) and positive (dV>0) voltage sweeps. It is common to associate the reduction to the process that takes place while the applied bias voltage is being reduced, i.e., for the dV<0 part of the CV cycle. However, when the voltage sweep is reversed at -1.0 V, the value of the current is still negative until $\approx$ -0.46 V where it becomes positive. Thus, even when the voltage sweep changes sign, the sample is still reducing by accumulating negative charge. This becomes evident when plotting the current as a function of the elapsed time (Fig. 4, right panel): the reduction continues until the current becomes positive. The total amount of charge associated with reduction is, therefore, the area denoted with the black stripes. Based on this observation, we have adopted the following convention for the ALC_EQCM code:

• A CV cycle is composed of two consecutive positive and negative voltage sweeps, or viceversa, within a given range of applied voltage. In the example of Fig. 4, the CV cycle consists of a negative voltage sweep from 0.5 V to -1.0 V followed by positive sweep from -1.0 V to 0.6 V. The value for the initial and final voltages do not need to be the same.

• Reduction occurs while the measured current is negative, independently of the applied voltage. Consequently, oxidation occurs while the measured current is positive. In Fig. 4 (left), the sample reduces from 0.5 V to -1.0 V and continues reducing until the potential is reversed back to -0.46 V. Oxidation follows until
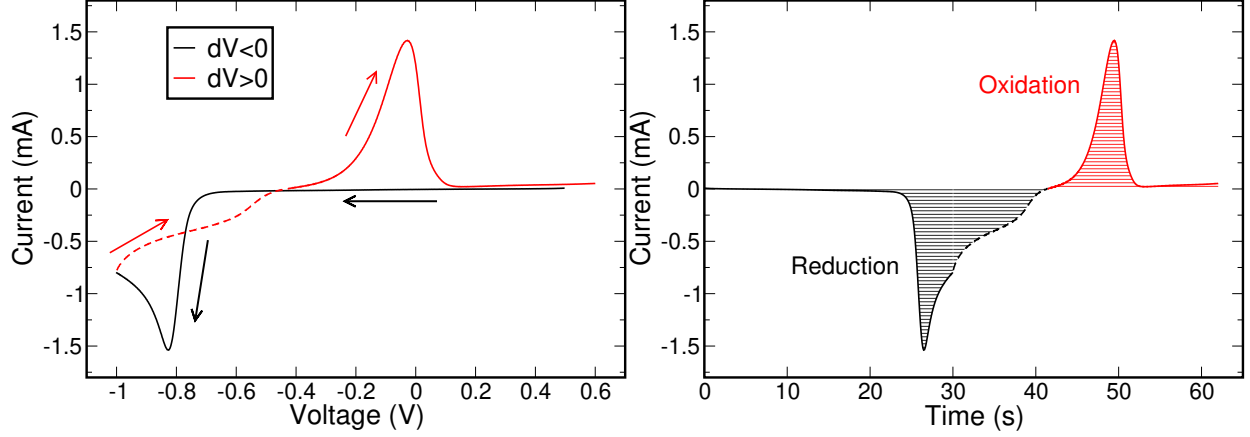
Figure 4: (Left) First CV cycle for the electrodeposition of Sb during negative (dV<0, black) and positive (dV>0, red) voltage sweeps. Voltage values are referred with respect to a Pt wire electrode [4]. At the start of the positive voltage sweep, the current is still negative and changes sign at ≈ -0.46 V. The segment for positive voltage sweep and negative current is indicated with dashed line. (Right) When plotting the current versus the elapsed time, it is evident that even though the voltage sweep has changed sign, the sample is still reducing until the current becomes positive (start of the oxidation).

0.6 V. If the CV cycling would continue, the sample would oxidize until the current becomes negative at the beginning of the second CV cycle.

• A redox cycle is composed of a reduction followed by an oxidation, or viceversa.

Mathematically, we define the total charge associated with reduction

$$\Delta Q_{red} = \int_{\tau[i_{(-)} \to i_{(+)}]}^{\tau[i_{(+)} \to i_{(-)}]} i(t) dt \tag{15}$$

and oxidation

$$\Delta Q_{ox} = \int_{\tau[i_{(+)} \to i_{(-)}]}^{\tau[i_{(-)} \to i_{(+)}]} i(t) dt \tag{16}$$

where $i(t)$ is the current depending on the elapsed time. As defined, $\tau[i_{(+)} \to i_{(-)}]$ and $\tau[i_{(-)} \to i_{(+)}]$ correspond to the instants when the current changes from positive to negative and from negative to positive, respectively. ΔQ_{red} and ΔQ_{ox} are used also to compute the Coulombic Efficiency, CE, as follows:

$$CE = \left\{ \left| \frac{\Delta Q_{red}}{\Delta Q_{ox}} \right| \quad \text{if the sample is reduced first} \quad \left| \frac{\Delta Q_{ox}}{\Delta Q_{red}} \right| \quad \text{if the sample is oxidized first} \right. \tag{17}$$

and the difference between the magnitude of the accumulated charges, δ_Q

$$\delta_Q = |\Delta Q_{ox} + \Delta Q_{red}|. \tag{18}$$

Both CE and δ_Q aim to quantify the irreversibility of the reaction: the larger δ_Q and the more the CE departs from 1, the more irreversible the electrochemical process. Similarly to Eqs. (15) and (16) we define the mass variation associated with reduction

$$\Delta m_{red} = m(\tau[i_{(+)} \to i_{(-)}]) - m(\tau[i_{(-)} \to i_{(+)}]) \tag{19}$$

and oxidation

$$\Delta m_{ox} = m(\tau[i_{(-)} \to i_{(+)}]) - m(\tau[i_{(+)} \to i_{(-)}]) \tag{20}$$

15

where M is mass obtained from the frequency shift of the EQCM device. The amount of mass at half of each redox cycle, $m_{1/2}$, is given by the following expression:

$$m_{1/2} = \begin{cases} m(\tau[i_{(-)} \rightarrow i_{(+)}]) & \text{if the sample is reduced first} \\[2ex] m(\tau[i_{(+)} \rightarrow i_{(-)}]) & \text{if the sample is oxidized first.} \end{cases} \tag{21}$$

Similarly, we define residual mass as the total mass accumulated at the end of the redox cycle. Finally, ALC_EQCM also computes $|\Delta m_{red}/\Delta Q_{red}|$ and $|\Delta m_{ox}/\Delta Q_{ox}|$ which not only provides further insight of the electrochemical efficiency of the process, but can also be used as a rather stringent descriptor when comparing experiments on expectedly the same electroactive material [1].

By setting the option *characterization* for directive **Analysis**, all the above quantities are computed for each redox cycle and printed in file CHARACTERIZATION, located in folder **ANALYSIS_EQCM** (see sec. 4.3.6). If mass-frequency is not recorded, only charge related quantities are computed.

Important: If the elapsed time is not recorded, the user will be asked to specify the voltage scan rate for which the EQCM experiments were performed. Nevertheless, the user must corroborate that cycles were measured at constant scan rate, otherwise, the transformation from voltage to time domain will be incorrect, leading to an inaccurate estimation of the integrated charges.

3.6.1 Characterization restricted to a voltage range

All the above quantities are computed over the range of voltages of the EQCM experiment, and the results are summarised in the CHARACTERIZATION file. Nevertheless, the user might also be interested to compute Δm and ΔQ over a sub-range of voltages. This can be done using the **voltage_range** directive. If such a voltage range is within the voltage range of experiments, a new table with the computed quantities is printed to the OUT_EQCM file just after the table with the computed Δm and ΔQ for the whole voltage range. To the present stage of development, no new file is generated, and the user is responsible to compute manually the characterization related quantities.

3.7 Stoichiometry of intercalation processes

To extract information of stoichiometric changes following the intercalation of species, it is necessary to relate the EQCM data for mass and charge variations with a reaction formula that describes the process. In ALC_EQCM, we have considered a general reaction formula for a set chemical species $\{Z, \tilde{Z}\}$ as follows:

$$\left[\bigcup_j^{N_{fix}} \tilde{Z}_{s_{0j}}^{(j)} \bigcup_k^{N_{var}} Z_{s_{0k}}^{(k)} \right] + \sum_k^{N_{var}} s_k Z^{(k)} \longrightarrow \left[\bigcup_j^{N_{fix}} \tilde{Z}_{s_{0j}}^{(j)} \bigcup_k^{N_{var}} Z_{(s+s_0)_k}^{(k)} \right] - \left[\sum_k^{N_{var}} n_{ox}^{Z^{(k)}} s_k \right] e^-. \tag{22}$$
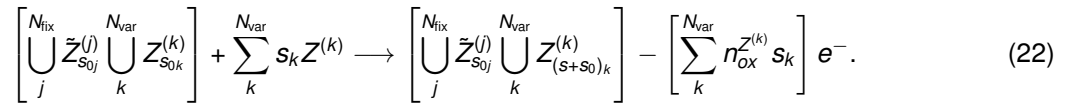
N_{var} is the number of variable chemical species $\{Z\}$ that participate in the reaction. N_{fix} is the number of fixed species $\{\tilde{Z}\}$, whose content remain constant during the reaction. Letters j and k are indexes for the chemical species. From left to right, the terms in the equation account for i) the system before the reaction where each species has stoichiometry $s_{0(j,k)}$, ii) the amount s_k of species $Z^{(k)}$ that participate in the reaction, iii) the system following the reaction and iv) the total accumulated charge. The quantity $n_{ox}^{Z^{(k)}}$ is the oxidation number of species $Z^{(k)}$. The general expression (22) can either represent reduction or oxidation with the arrow indicating the irreversibility of the process, as it is often the case. While stoichiometric coefficients s_k can be either positive or negative, the condition $(s + s_0)_k \geq 0$ must hold.

To activate stoichiometric analysis for intercalation compatible with EQCM data, the user must set options *stoichiometry* and *intercalation* for directives **Analysis** and **Process**, respectively. Implementation details for this functionality are described in the following sections.

16

3.7.1 Defining the species

The **&Block_species** block must define all species that participate in the reaction. The structure of this block is described in Table 3. For each species, the user must set the tag, atomic or molecular mass, oxidation number, pristine stoichiometry and the type of variable (dependent, independent or fixed). Intercalation reactions that involve only one variable species (e.g. lithium intercalation in electrodes) are not implemented to date. For reactions with two and more variable species involved, two species must be defined as dependent. We refer the user to the tutorial of sections 6.10.1 and 6.10.2 for examples of the definition of **&Block_species**.

3.7.2 Defining the set of equations

Based on the general form of Eq. (22), as well as the experimental information for i) mass variation (Δm), ii) charge accumulation (ΔQ) and iii) number of moles of the pristine system (n_{mol}), ALC_EQCM finds the set of variable stoichiometric coefficients s_k consistent with the EQCM data. The mass variation Δm depends on the stoichiometric coefficients as follows:

$$\frac{\Delta m}{n_{mol}} = \sum_{k}^{N_{var}} \mathcal{M}_{Z^{(k)}} s_k \tag{23}$$

where $\mathcal{M}_{Z^{(k)}}$ is the atomic/molecular weight of participating species. The charge accumulated depends on the stoichiometric coefficients in the following way:

$$\frac{\Delta Q}{n_{mol}\mathcal{F}} = -\sum_{k}^{N_{var}} n_{ox}^{Z^{(k)}} s_k, \tag{24}$$

where $n_{ox}^{Z^{(k)}}$ is the oxidation number of species $Z^{(k)}$ and $\mathcal{F}$ is the Faraday constant. Equations (23) and (24) can be expressed in matrix form $\mathbf{D} = \mathbf{BS}$ with

$$\mathbf{D} = \begin{bmatrix} \frac{\Delta m}{n_{mol}} \\ \\ \frac{\Delta Q}{n_{mol}F} \end{bmatrix} , \quad \mathbf{S} = \begin{bmatrix} s_1 \\ s_2 \\ \vdots \\ s_{N_{var-1}} \\ s_{N_{var}} \end{bmatrix} \tag{25}$$

and

$$\mathbf{B} = \begin{bmatrix} \mathcal{M}_{Z^{(1)}} & \mathcal{M}_{Z^{(2)}} & \cdots & \mathcal{M}_{Z^{(N_{var}-1)}} & \mathcal{M}_{Z^{(N_{var})}} \\ -n_{ox}^{Z^{(1)}} & -n_{ox}^{Z^{(2)}} & \cdots & -n_{ox}^{Z^{(N_{var}-1)}} & -n_{ox}^{Z^{(N_{var})}} \end{bmatrix} . \tag{26}$$

The stoichiometric coefficients of matrix **S** are unequivocally defined only if $N_{var} = 2$. If $N_{var} > 2$, the set of solutions is underdetermined: the coefficients defined as "dependent" will depend on the "independent" stoichiometric coefficients, thus leading to an infinite number of possible solutions.

3.7.3 Computing the set of equations

We first describe the procedure for the computation of matrix **S** for the general case of more than two species that participate in the reaction ($N_{var} > 2$). To this end, we select two of the N_{var} coefficients as dependent (dep1 and dep2) and set the rest $N_{var} - 2$ as independent species (indep). The user must decide

which variables are dependent and independent when defining the species in **&Block_species** (see Table 3 and tutorial examples). By arranging the matrix equation $\mathbf{D} = \mathbf{BS}$, we obtain the following expression

$$\mathbf{D} - \mathbf{BS_{indep}} = \begin{bmatrix} \frac{\Delta m}{n_{mol}} - \sum_j^{[N_{var}-2]} \mathcal{M}_{Z^{(j)}} s_j^{indep} \\[6pt] \frac{\Delta Q}{n_{mol}F} + \sum_j^{[N_{var}-2]} n_{ox}^{Z^{(j)}} s_j^{indep} \end{bmatrix} = \begin{bmatrix} \mathcal{M}_{Z^{(dep1)}} & \mathcal{M}_{Z^{(dep2)}} \\[6pt] -n_{ox}^{Z^{(dep1)}} & -n_{ox}^{Z^{(dep2)}} \end{bmatrix} \begin{bmatrix} s_{dep1} \\[6pt] s_{dep2} \end{bmatrix} = \mathbf{B_{dep}S_{dep}}. \tag{27}$$

By inverting the 2x2 matrix $\mathbf{B_{dep}}$ and multiplying to the left by $\mathbf{B_{dep}^{-1}}$, one obtains $\mathbf{S_{dep}}$, which clearly depends on the independent coefficients s_j^{indep}, and form a set of infinite solutions compatible with the experimental input data ΔQ and Δm. The user must select an appropriate set of dependent such that matrix $\mathbf{B_{dep}}$ is not singular, otherwise the code will complain and abort the execution. Physical valid solutions for all variable coefficients (dependent and independent) are recorded by screening the domain of independent variables using a discretization for the stoichiometric domain (set to 0.01 by default). Such discretization, however, can be modified by the user via directive **delta_stoich**. Computed solutions are evaluated against the criteria $s_{dep1} + s_{0,dep1} \geq 0$ and $s_{dep2} + s_{0,dep2} \geq 0$ to be considered as physical valid solutions.

When the system is underdetermined, the set of solutions is computed only for initial redox process (i.e. oxidation or reduction). In fact, the computed solutions depend on the set of initial stoichiometry coefficients $s_{0,j}$ that characterize the pristine system. The multiple set of solutions when $N_{var} > 2$ defines multiple possible configurations and, consequently, multiple possible initial configurations for the next fragment of the redox process. Thus, when $N_{var} > 2$, ALC_EQCM only computes the first fragment of the first redox cycle. Only one file is generated, either INTERCALATION_OX or INTERCALATION_RED, depending on which process occurs first. These files will be printed within folder **ANALYSIS_EQCM**.

Moving to the case of only two variable stoichiometric species ($N_{var} = 2$), there is no independent stoichiometric coefficient and the computed solution is unequivocally determined. Again, the solution is evaluated against the criteria $s_{dep1} + s_{0,dep1} \geq 0$ and $s_{dep2} + s_{0,dep2} \geq 0$ that, in case of not being fulfilled, ALC_EQCM will print an error informing the user and abort the execution. If this is the case, the user should review the input EQCM data and the definition of the participating species, as it could well be that the intercalation process involves more than two species. In contrast, if the criteria is fulfilled, the solution ($Z_{[s_{dep1}+s_{0,dep1}]}, Z_{[s_{dep2}+s_{0,dep2}]}$) is used to define the initial configuration for the next fragment, i.e. $[s_{dep1} + s_{0,dep1}] \rightarrow s_{0,dep1}$ and $[s_{dep2} + s_{0,dep2}] \rightarrow s_{0,dep2}$, and the problem is solved again with the new corresponding values of ΔQ ad Δm. The logic is repeated for all the EQCM cycles, and the computed results are printed to INTERCALATION_OX and INTERCALATION_RED files.

3.7.4 Use of constraints for chemical species

For those process where the number of variable species is larger than two, ALC_EQCM offers the possibility to introduce constraints between the stoichiometric coefficients to reduce the dimensionality of the computed solutions. Constraints must be specified by **&block_constraints_species** in the SET_EQCM file (see Table 3). Specification of constraints can be, for example, solvation numbers for intercalating ions, which are often based on chemical intuition [2] or derived from atomistic simulations [1]. Various types of constraints are available and described as follows:

• target_value: fixes the value either for dependent or independent variables species. Example: to compute solutions with stoichiometry 0.1 for H+ during oxidation, the syntax must be

```
target_value    oxidation    H+    0.1
```

• target_min: finds the set of solutions that gives the minimum value for the species under consideration. Example: to compute solutions with the minimum stoichiometry for H+ during oxidation, the syntax must be

```
target_min    oxidation    H+
```

- `target_max`: finds the set of solutions that gives the maximum value for the species under consideration. Example: to compute solutions with maximum stoichiometry for `Li+` during reduction, the syntax must be

```
target_max    reduction   Li+
```

- `target_range`: for the species under consideration, this constraint finds the set of solutions between the specified range. Example: to compute the set of solution for `H+` with stoichiometry between 0.1 and 0.5 during oxidation, the syntax must be

```
target_range   oxidation   H+   0.1   0.5
```

- `ratio_fixed`: finds solutions for a fixed ratio between two selected species. Example: to compute the set of solutions while keeping the ratio between `H2O` and `Li+` equal to -0.98 during reduction (extraction of Li+ and insertion of H2O, or viceversa), the syntax must be

```
ratio_fixed    reduction   d[H2O]|d[Li+]   -0.98
```

- `keep_ratio`: option to keep the ratio between chemical species equal to the previous process. If oxidation occurs first, specification for this option is only applicable to reduction, and viceversa. Example: to keep the ratio between species `H2O` and `Li+` during reduction equal to oxidation, the syntax must be

```
keep_ratio    reduction   d[H2O]|d[Li+]
```

The format to relate species for options `keep_ratio` and `ratio_fixed` must comply with the specification above. Capital and lower case letter "d"/"D" can be used, as well as any bracket or parenthesis "([{}])". The vertical bar $|$ (indicating division) is compulsory.

ALC_EQCM will check the syntax for constraints and inform the user if there is a problem before aborting the execution. If the number of constraints is less than the number of independent variables, a set of multiple solutions will be generated. The number of generated solutions will be determined by the selected discretization of the stoichiometric space, defined by directive **delta_stoich** (set to 0.01 by default).

3.8 Stoichiometry of electrodeposition

This functionality is activated by selecting options *stoichiometry* and *electrodeposition* for directives **Analysis** and **Process**, respectively. In contrast to intercalation, the number of moles that participate in the electrodeposition process, n_{mol}, cannot be estimated beforehand. In fact, it is fully determined by the amount of species that are deposited and removed from the electrode surface.

It is important to remark that we have not considered electrodeposition processes where multiple species are deposited and stripped during the reaction [47, 48]. This is because such processes cannot be described by a single reaction formula.

3.8.1 Defining the involved species

Similarly to sec. 3.7.1, the user must specify the species that participate in the reaction in **&Block_species**. In contrast to intercalation, all the variable species must be defined as independent, even if they correspond to subspecies (see next section). The value for the stoichiometry s_0 must be the same for all subspecies.

3.8.2 Defining the set of equations (Sauerbrey limit)

For electrodeposition processes that involve only one species, the mass and charge balance equations read:

$$\Delta m_{eff} = -\frac{A_{eff}\Delta f}{C_f} = s n_{mol} \mathcal{M} \tag{28}$$

and

$$\Delta Q = -s n_{\mathrm{mol}} \mathcal{F} n_{ox} \qquad (29)$$

where Δm_{eff} and ΔQ are the effective mass deposited/removed from the supporting electrode and the charge associated with the process, respectively. Note that Δm_{eff} in Eq. (3) is different from Δm in Eq. (1), since in general $A_{\mathrm{eff}} \neq A_{\mathrm{disk}}$. $\mathcal{M}$ is the atomic or molecular mass of the adsorbed species and n_{ox} its oxidation number. $\mathcal{F}$ is the Faraday constant. Dividing (28) by (29) and using Eq. (3) one gets:

$$\frac{1}{\tilde{C}_f} = \left(\frac{A_{\mathrm{eff}}}{A_{\mathrm{disk}}} \right) \frac{1}{C_f} = R \frac{1}{C_f} = \left(\frac{\mathcal{M}}{\mathcal{F} n_{ox} A_{\mathrm{disk}}} \right) \left(\frac{\Delta Q}{\Delta f} \right). \qquad (30)$$

where all terms are divided by the geometric area of the electrode disk, A_{disk}. Different information can be extracted from the above equation:

• If the amount of deposited material is considerably large and/or the surface is not flat, the validity of the Sauerbrey equation can be compromised, and $\tilde{C}_f$ can be interpreted as an effective characteristic constant.

• Under the assumption that the characteristic constant C_f is not affected by the process ($C_f = C_{f0}$) and the disk area is fully covered during the process, then R can be taken as a measure of the surface rugosity using Eq. (1)

$$R = \left(\frac{\mathcal{M}}{\mathcal{F} n_{ox}} \right) \left(\frac{\Delta Q C_{f0}}{A_{\mathrm{disk}} \Delta f} \right) = -\left(\frac{\mathcal{M}}{\mathcal{F} n_{ox}} \right) \left(\frac{\Delta Q}{\Delta m} \right). \qquad (31)$$

It is user's responsibility to extract the relevant information from the file ELECTRO_DEPOSITION, which is written inside folder **ANALYSIS_EQCM**. The amount of moles for the deposited material is calculated from the following expression:

$$s \, n_{\mathrm{mol}} = n_{\mathrm{mol,s}} = \left(\frac{A_{\mathrm{eff}}}{A_{\mathrm{disk}}} \right) \frac{\Delta m}{\mathcal{M}}. \qquad (32)$$

3.8.3 Defining the set of equations (beyond the Sauerbrey limit)

For EQCM processes where non-gravimetric effects are relevant, obtaining the change of deposited mass from changes in the quartz frequency via the Sauerbrey relation is not correct. In those cases, the user is responsible to provide the correct values by reporting the mass Δm_{eff} as a function of the voltage during the EQCM scan. Following oxidation or reduction, Eq. (28) must be written without the Sauerbrey's factor

$$\Delta m_{\mathrm{eff}} = s n_{\mathrm{mol}} \mathcal{M} \qquad (33)$$

whereas (29) remains the same. From these two equations, one can define the theoretical change of mass $\Delta m_{\mathrm{eff}}^{th}$ from the total accumulated charge ΔQ as follows

$$\Delta m_{\mathrm{eff}}^{th} = -\frac{\Delta Q \mathcal{M}}{\mathcal{F} n_{ox}}. \qquad (34)$$

The total number of moles that participate in the reaction is calculated from

$$s \, n_{\mathrm{mol}} = n_{\mathrm{mol,s}} = -\frac{\Delta Q}{\mathcal{F} n_{ox}} \qquad (35)$$

whereas the ratio $\Delta m_{\mathrm{eff}}^{th}/\Delta m_{\mathrm{eff}}$ provides a combined quantification of the rugosity of the electrode surface and the efficiency of the reaction [4].

3.8.4 Subspecies

The presented formalism is also valid when the single species is represented as a set of N_s subspecies $\{s_1, \ldots s_N\}$. Thus $\mathcal{M} = \sum_i \mathcal{M}_i$, where $\mathcal{M}_i$ is the atomic/molecular mass for the subspecies i; $n_{ox} = \sum_i n_{ox}^{(i)}$, where $n_{ox}^{(i)}$ is the oxidation number for the subspecies i; and the number of moles for the subspecies is $n_{mol,s}^{(i)} = n_{mol,s}/N_s$.

3.9 Mole changes, mole fluxes and ionic currents along the CV scan

The computation of stoichiometric coefficients of section 3.7 is conducted using the accumulated charge ΔQ and mass variation Δm for each oxidation/reduction, that is, using values for the entire potential window for oxidation/reduction. ALC_EQCM also offers the capability to compute the amount of moles vs. the along the CV scan, as well as mole fluxes and ionic currents. Such quantities are more relevant when comparing with EQCM profiles and interpreting the potential dependent charge storage response [20]. It is important to emphasize that this capability is only available for intercalaton processes involving two or more species.

The implemented algorithm uses the values for mass-frequency (or mass) $\Delta m(V)$ and the current $i(V)$ reported in the DATA_EQCM. From the $i(V)$ and the scan rate (or elapsed time if measured), the accumulated charge is computed $\Delta Q(V)$. The equations for mass and charge balance are then solved for each value of the voltage V as described in the previous sections. The stoichiometric solution s_k for each variable chemical species $Z^{(k)}$ (see Eq. 22) is computed for each measured voltage. Consequently, the stoichiometric solutions $s_k^j(V)$ depend on the applied potential, V, and the redox cycle number j. The dependence on the cycle number j is due to the irreversibility of the electrochemically induced intercalations processes.

The variations of the number of moles for the involved species is defined as follows,

$$\mathcal{N}_k^j(V) = n_{mol} s_k^j(V) \tag{36}$$

where n_{mol} is number of moles of the pristine system before intercalation. To compute the mole flux corresponding to the species k, $\mathcal{J}_k^j(V)$, the following formula is used:

$$\mathcal{J}_k^j(V) = n_{mol} \frac{ds_k^j(V)}{dt} \tag{37}$$

where the connection between V and t is through the scan rate used for the experiment. In constrat, the ionic currents, $\mathcal{I}_k^j(V)$, are only meaningful for those involved species $Z^{(k)}$ with non-zero oxidation number $n_{ox}^{Z^{(k)}}$ and it is computed as follows

$$\mathcal{I}_k^j(V) = -F n_{ox}^{Z^{(k)}} \mathcal{J}_k^j(V), \tag{38}$$

where F is the Faraday constant. The computation of mole variations, mole fluxes and ionic currents is activated using the option `fluxes` for the **analysis** directive. Values are reported in, nmol, nmol/s (nanomole/sec) and mA for moles, mole fluxes and ionic currents and printed to files CV_MOLES (sec. 4.3.9), MOLE_FLUX (sec. 4.3.10) and IONIC_CURRENT (sec. 4.3.11), respectively. In sec. 6.15 we provide an example exercise of these functionalities.

3.10 Atomistic models of cycled samples

Generating atomistic models compatible with EQCM data is the key functionality of ALC_EQCM and the main reason for which the code was developed. Atomistic models are built using the stoichiometric solutions computed from the mass and charge balance equations as described in the previous sections. Building atomistic models allows screening the space of solutions via computational simulations and identifying energetically/thermodynamically favorable stoichiometric compositions. Insight into computed solutions is crucial to withdraw fundamental atomistic insight of electrochemically induced reactions [1].

For intercalation processes, if the number of participating chemical species is larger than two, the two mass and charge balance equations (Eqs. 23 and 24, respectively) are not sufficient to unequivocally determine the resulting stoichiometry, and the system of equations becomes mathematically underdetermined. If the number of variable species is three, for example, one of the variables is taken as independent (while the other two are dependent through the mass and charge balance equations), and the set of infinite solutions depends on the independent variable. We can interpret such a set as one-dimensional. Analogously, if the number of variable species is four, two variables will be independent and two dependent, and the set of solutions will be in a two-dimensional domain. Consequently, for a process involving N_{var} species, the space of solutions will reside in a ($N_{var} - 2$)-dimensional domain.

For electrodepositon, the implemented algorithm within ALC_EQCM can only generate atomistic models for single species. We have shown that a single species can also be represented as a set of subspecies, where each subspecies contributes with the same number of moles. Still, the mass and charge balance equations are sufficient to determine the number of moles of the participating species.

To generate models for cycled samples one must set the option *model_cycled_sample* for directive **Analysis**. Regardless of the electrochemical process (electrodeposition or intercalation), building atomistic models requires the mass and charge variations obtained from the EQCM data (DATA_EQCM file), which are computed and printed to file CHARACTERIZATION (see sec. 4.3.6) for each reduction/oxidation fragment of the cycle. The user must also provide the INPUT_STRUCTURE file within folder **INPUT_GEOM** (see section 4.2.3) containing a model for the host structure (intercalation) or the slab surface (electrodeposition).

If there are molecular species involved, the user must also provide the corresponding xyz file for the geometry. The name of the file must be species-tag.xyz, where "species-tag" refers to the tag defined in **&Block_species** for that particular species. The file must contain all the atoms for that particular species, and must comply with the format and syntax of section 4.2.4.

Together with the specification in **&Block_species**, species can well be composed by a set of atoms, as it happens with molecular species. Atomistic detail for each species must be provided in

&block_species_components. The syntax and format of this block is shown in Table 3. All the species defined in **&Block_species** must be specified in this block. We refer the user to the tutorial section 6.12 for practical examples. The logic of the implemented algorithm to generate atomistic models can be summarized with the following steps:

1) reading a reference **input model** from file INPUT_STRUCTURE and checking that the simulation cell is consistent with the geometry. Such structure must contain, at least, those atoms of the species that are defined as "fixed" in **&Block_species**, and must represent a reasonable model of a host bulk system (for intercalation) or slab substrate (for electrodeposition),

2) identifying the number of species (those defined in the **&block_species** block) present in the input model and determine the resulting input stoichiometry,
3) grouping atoms that belong to a given species according to the definition of the **&block_species_components** block,

4) defining the final size for the generated atomistic model, which is determined by repetitions of the input reference structure along each of the three cell vectors (see *repeat_input_model* directive),

5) setting the target stoichiometry for the atomistic model. This step depends on the system and the process under consideration (see sections 3.10.1 and 3.10.2),

6) comparing the target stoichiometry with the stoichiometry of the input model. This step determines how many units of a given species must be removed/added from/to the **input** model to approximate the target stoichiometry,

7) removing species from the **input model** to approximate to the target stoichiometry,

8) inserting species to the **input model** to approximate the target stoichiometry,

9) building a repetition of the modified **input model** according to directive ***repeat_input_model***. This multiple repetition of the **input model** is referred to as **sample model**, in contrast to the **input model**. The purpose of building a larger **sample model** is to minimize stoichiometric differences with respect to the targeted values. The size of the generated **sample model** must be set by the user.

10) comparing the target stoichiometry with the stoichiometry of the **sample model** only for those species that change their content during the reaction. This step determines how many units of a given species must be removed from (added to) the **sample model** to match the target stoichiometry.

11) removing species from the **sample model** to match the target stoichiometry,

12) inserting species to the **sample model** to match the target stoichiometry,

13) evaluating the differences between the stoichiometry for the **sample model** and the target stoichiometry. The user will be informed if differences are larger than the value of directive ***stoichiometry_error*** (set to 0.01 by default),

14) printing the generated **sample model** to file. A summary of each generated model is printed as a table in the OUT_EQCM file. The same table is printed to file MODEL_SUMMARY in each subfolder where the atomistic model is printed. Although this construction could be seen as a duplication of the information printed to OUT_EQCM, having a summary of the model in each subfolder becomes convenient to the purpose of identifying the stoichiometry of the generated model when running the simulations.

To insert species, the region within the simulation cell is represented in a 3D-grid set of spatial points, where each cell vector is discretised using the value of directive ***delta_space*** (set to 0.1 Angstrom by default, see Table 3). The centroid for the species is set to coincide with one of these grid points. The species will be inserted when the distance between each of its constituents atoms and the rest of the atoms in the model is larger than a distance criterion specified via directive ***distance_cutoff*** (set to 1.7 Å by default). For molecular species, the code also allows random rotations with respect to the geometry provided in the xyz file (see sec. 4.2.4). This is convenient to building systems with intercalated species as well as amorphous coating. If the species cannot be inserted, the algorithm chooses another grid point. The process is repeated until all the species are inserted.

The format for the generated atomistic models depends on the chosen option for the directive ***output_model_format***. The name of the generated model is SAMPLE.format, where the allowed formats are: vasp, xyz, onetep, castep, siesta, cif and cp2k.

Once all the models are generated, ALC_EQCM creates file OUT_EQCM_BACKUP, which is a copy of OUT_EQCM to keep record of the output information. In addition, ALC_EQCM prints file RECORD_MODELS, which contains specific information of the atomistic models (see sec. 4.3.16). This file allows the user to only generate the input files for atomistic level simulations (sec. 3.12) and scripts with HPC instructions (sec. 3.13), without the need to recompute the atomistic models. Both OUT_EQCM_BACKUP and RECORD_MODELS are printed in folder **RESTART**.

In the following, we present further details for the generation of atomistic models for intercalation and electrodeposition.

3.10.1 Intercalation

The generated models depend on the number of involved species,

• **one variable species**: this case is not implemented yet

• **two variable species**: the stoichiometric solution is unequivocally determined (see tutorial 6.10.1) for each oxidation and reduction. This allows computing changes for the stoichiometric coefficients of the

involved species with the CV cycles. Solutions are built to approximate the target stoichiometries: steps 5 to 14 (above) are repeated for all the oxidation and reduction fragments of the CV cycles. Generated models are printed within folders **ATOMISTIC_MODELS/Xcycle-oxidation** and **ATOMISTIC_MODELS/Xcycle-reduction**, where X is the cycle number. This also applies to the case of N_s variable species with $N_s - 2$ constraints (see sections 3.7.4 and 6.10.3).

• **three or more variable species**: the system of equations is underdetermined and one obtains a set of multiple solutions within the stoichiometric range of those species that are defined as "independent" (see sec. 3.7.3). The number of computed solutions depends on the choice for the discretization of the stoichiometric domain and it is generally large. The user must specify the number of solutions to be modelled using **targeted_number_models**. With this directive, the algorithm selects the most convenient set of solutions (selected solutions) such that the stoichiometric domain is sampled with a reduced number of points, which are are generally different but close to the value set for **targeted_number_models** (hence the word "targeted"). Selected solutions are printed to file **ANALYSIS_EQCM**/SELECTED_SOLUTIONS, and represent a subset of those computed solutions reported in the INTERCALATION files. For each selected solution with a target stoichiometry, steps 5 to 14 (above) are executed. Models are printed in folder **ATOMISTIC_MODELS/1cycle-oxidation/modelX** or **ATOMISTIC_MODELS/1cycle-reduction/modelX**, where X is the model number. The same procedure applies to the case of N_s variable species where the number of constraints is lower than $N_s - 2$. We refer the user to the tutorial example of sec. 6.12.1.

For the process to build atomistic models, ALC_EQCM makes use of random number generators to select one of the grid 3D-grid points inside the simulation cell to be centroid of a given species. If the species is molecular, it is also rotated using random Euler angles. If the species to insert is large, the host structure does not contain sufficient voids and/or the value of **distance_cutoff** is relatively large, the number of iterations to insert the species may reach a pre-defined maximum limit. In such cases, the code will abort and inform the user. If the input structure and species are reasonable, the user is advised to rerun the code. In case of a new failure, the value of **distance_cutoff**, if defined, must be reduced and always be larger than 1.7 Å. Otherwise, the user should re-evaluate the definition of the input structure or consider re-scaling the size of the host model using directive **scale_cell** (sec. 3.14.5).

Finally, it is important to mention that the generation of models is not available for selected values of the potential along the CV scan. So, *fluxes* option for **analysis** does not generate atomic models but only computes mole fluxes and ionic currents (section 3.9).

3.10.2 Electrodeposition

For electrodeposition, the implemented algorithm (see section 3.8) only allows single species to be deposited and removed from a substrate. Such species can be composed of N_{sub} subspecies, where each subspecies contributes with the same number of moles.

In reality, electrode surfaces are far from perfect, containing multiple defects and steps. In addition, species are not uniformly deposited throughout the electrode surface during the reaction. As a result, adsorbed species form rough surfaces, which represent a challenge even for EQCM experiments and the sensitivity for mass accumulation [44]. To further complicate the scenario, the atomic structure at the interface between the electrode material and the adsorbed species is generally very complex and depends on the experimental conditions and the degree of commensurability. Obviously, considering all these factors is practically unfeasible from the modeling perspective, and further approximations are required.

To build electrodeposition models with ALC_EQCM, the user is advised to consider the lowest energy facets of the substrate, locally free of steps. Models are generated by assuming that species are adsorbed uniformly across the electrode. This might be a rather crude approximation, but allows to determine the amount of adsorbed species η_i from the computed number of moles $n_{mol,s}^{(i)}$ [see Eq. 32 or 35] and the area

A_{slab} for the modelled slab, as follows

$$\eta_i = NINT\left(n_{mol,s}^{(i)} N_A \frac{A_{\text{slab}}}{A_{\text{disk}}}\right), \tag{39}$$

where $NINT$ is the nearest integer and N_A the Avogadro's number. Generally, the dimensions of A_{disk} and A_{slab} are cm^2 and Å^2, respectively.

The implemented algorithm scans the substrate model and finds automatically the outermost part from where to start depositing species. Alternatively, the user can set directive **deposition_level**, which is convenient to model systems with intercalated species at the subsurface. To add species, the surface plane is scanned uniformly. In the case of multiple subspecies, each subspecies is added to the surface one after the another in a loop. For example, for 3 subspecies, the sequence of deposition is 1, 2, 3, 1, 2, 3, ..., etc . Species are deposited when all atoms comply with the distance cutoff criterion, while accounting for PBCs imposed by the simulation cell. Depending on the amount of species, the role of the distance cutoff can play a crucial role. In fact, if many species are added and the selected distance cutoff is small, the resulting model for the interface will be in an unrealistically high-energy configuration. If the distance cutoff is large, the process of depositing the species could leave large voids at the interface. It is user's responsibility to select a sensible value for the cutoff distance This can be conveniently optimised by means of a small set of auxiliary DFT simulations aimed at determining the the cutoff distance leading to the lowest energy for the as prepared systems. In addition, the vacuum region must be sufficiently large to prevent the simulation cell to be filled with species.

For the reversed process, Eq. (39) gives a negative number, thus indicating the amount of species to be removed from the model. The removal of species is conducted from the outermost part of the surface. Generated models are printed within folders **ATOMISTIC_MODELS/Xcycle-oxidation** and **ATOMISTIC_MODELS/Xcycle-reduction**, where X is the cycle number. We refer the user to the tutorial example of sec. 6.12.3.

IMPORTANT: By default, species are deposited over **only one side** of the slab. This is not appropriate for implicit solvent models (sec. 3.14.7), as the solvation will be applied to both surfaces of the slab, which will be chemically inequivalent. To build models with species deposited at both sides of the slab, the user must set the **both_surfaces** directive to `.True.`.

3.11   Atomistic models of pristine samples

ALC_EQCM also offers the possibility to generate atomistic structures of pristine samples. This functionality can be set with the option *Model_pristine_sample* for directive **Analysis**. In contrast to the option *Model_cycled_sample*, building pristine models does not require processing EQCM data nor solving of the stoichiometric problem from the charge and mass balance equations. Thus, definition of dependent/independent species is not relevant in this case, and they will all be treated as variable species allowed to adjust their content to match the input stoichiometry values as defined in **&block_species**. The logic of the implemented algorithm is as described by steps 1 to 14 in the previous section, but in step 5 the target stoichiometry is obtained from the information in **&block_species**. The resulting model is printed in folder **ATOMISTIC_MODELS/pristine** with the name SAMPLE.format, where the allowed formats are vasp, xyz, onetep, castep, siesta, cif and cp2k, as specified with directive **output_model_format**. Tutorial of sec. 6.12.2 exemplifies this functionality.

For intercalation processes, this functionality is convenient to screen the stoichiometry of the system before the electrochemical cycling. An example is the generation of atomistic models with different content of intercalated water [1].

For electrodeposition, ALC_EQCM determines the number of species to be adsorbed or removed based on the stoichiometry set in **&block_species** and the number of units for the fixed species (those belonging

to the electrode model) identified in the input model. The user can modify the amount of atoms to deposit by simply changing the stoichiometry value of the independent (sub)species in **&block_species**. We refer the user to the tutorial examples of section 6.12.4. The user can also set the ***both_surfaces*** directive to `.True.`, which will instruct the code to deposit surfaces at both sides of the slab.

3.12   Input files for atomistic level simulations

ALC_EQCM allows generating atomistic models compatible with EQCM data (Sec. 3.10), as well as for pristine structures with a defined stoichiometry (Sec. 3.11). The purpose is to use the generated atomic structures as the starting point for atomistic level simulations. However, such structures are not sufficient by their own, as further details are needed to define the interaction between the participating atomic species. This section describes how ALC_EQCM can generate additional input files with appropriate directives to perform simulations of the atomistic structures.

The development of a general infrastructure to automatically generate input files for simulations is challenging due to the following reasons (to name a few):

• Materials in nature exhibit very different electronic properties, which in turn require of appropriate settings for meaningful simulations.

• Bulk and surfaces are conceptually different systems that need different treatments.

• Simulation settings depend on the particular property the user aims to investigate.

• Available codes for simulations have their own particular characteristics and intricacies, not only in the way to compute the interatomic interaction but also in the large amount of different possible options for the execution of simulations.

• Not all computational codes contain the same functionalities.

Relevant to the electrochemical response of materials, we have only considered the generation of input files for **geometry relaxation** and **molecular dynamics (MD)** simulations. The generation of input files is activated with the **&block_simulation_settings** block. The generated files and information depend on the choice of the ***output_model_format*** directive. At the present stage of development, it is only possible to generate files for DFT simulations using the codes VASP [49], CASTEP [50], CP2K [51] and ONETEP [52]. The plan is to progressively account for other computational codes. In case there is a missing specification or an incorrect setting, the code will abort and ask the user to fix the change the settings. In the following, we will only explain those directives that are essential to the purpose of generating input files for simulations. The full set of directives and options for **&block_simulation_settings** are listed and explained in Table 3 of section 4.

Inside **&block_simulation_settings**, ALC_EQCM will first ask for directive ***simulation_type***. This should be set to either *relax_geometry* or *MD*. The next compulsory directive is the level of theory to describe the interaction between the atoms, ***theory_level***. To date, the only possible option for this directive is *DFT*. In the future, we aim to implement directives to generate input files using classical force fields, as well as machine learning potentials [53, 54, 55, 56] and tight-binding DFT parameterizations [57].

By default, it is assumed that the system is neutral, but the charge state can be changed with the ***net_charge*** directive.

3.12.1   Settings for DFT input files

If option *DFT* is selected for ***theory_level***, ALC_EQCM will ask the user to specify the **&DFT_settings** block, which must be closed with directive **&end_DFT_settings**. The user is referred to the tutorial sec. 6.13 for examples of DFT settings. Inside this sub-block, the user must consider the following points:

- The level for the exchange correlation (XC) approach via directive **XC_level** (LDA or GGA) and the XC version using **XC_version** (See Table 4 for options). Dispersion corrections can be added with directive **vdW** (See Table 4). Hybrid functionals are currently not accounted for in the released ALC_EQCM, although work is planned to include them in the near future.

- To compute the electronic problem, optimised methods for diagonalization are set by default. For all these methods, specification of the smearing method (**smearing**, see Table 4) is compulsory, while the smearing width (**with_smear**) is optional but subject to the selected options.

- For CP2K, the user can also select the Orbital Transformation method with the **OT** directive to efficiently compute large systems with band gap. Directive **smearing** and **with_smear** are not compatible with this setting. In addition, setting the **gapw** directive to .True. defines the required instructions for simulations using the Gaussian Augmented Plane Waves (GAPW) method.

- For CASTEP, the user can choose the option *fix_occupancy* for **smearing** to fix the occupancy for the bands. This is very convenient to compute insulators. For metals or semiconductors, the user can only select options *gaussian* or *fermi*, which set the Density Mixing (DM) scheme by default. The user can select the option *edft* (Ensemble DFT) instead of DM, which instructs CASTEP to perform a self-consistent all-bands wavefunction search.

- For ONETEP, ALC_EQCM assumes by default that the system is an insulator or a wide-gap semiconductor. For metals, the user must select the option *edft* (Ensemble DFT) only with option *fermi* for the **smearing** directive. In addition, the user must specify sub-block **&ngwf** to specify the number and radius (in Å) of the NGWFs to describe the orbitals.

- Directives **energy_cutoff**, **scf_energy_tolerance** and **scf_steps** are compulsory.

- By default, only the Γ point is considered to sample the Brillouin zone. The user can define a mesh for multiple kpoints using directive **kpoints**.

- Directives **precision** is compulsory only for VASP simulations.

- The pseudopotential for each atomic tag must be specified via the **&pseudo_potentials** sub-block. All pseudo potential files shall be located inside the **DFT/PPs** folder. Only for those codes that use atomic-like basis sets, the user must also specify the name of the basis (see Table 4 for the implemented options of basis sets, via the definition of **&basis_set**). We refer to sec. 4.2.5 and 4.2.6 for the specification of the pseudo potential input files and basis set, respectively. If **&pseudo_potentials** is not defined, ALC_EQCM will not set input files with pseudopotentials, and the generated information will not be sufficient to execute a DFT simulation.

- Directive **spin_polarised** must be set for spin polarised simulations. For VASP, specification of directive **max_l_orbital** is also required.

- Initial magnetic moments for the atomic sites can be specified with the sub-block **&magnetization**.

- The total magnetization is allowed to change by default. To fix the total magnetization, the user must set the target value with directive **total_magnetization**. The assigned value must be equal to the initial total magnetic moment, which is computed from the initial magnetic moments and the total number of atoms in the sample model. If there is an inconsistency, ALC_EQCM will complain and abort the execution.

- Inclusion of Hubbard corrections is possible via the specification of the **&hubbard** sub-block, which must specify the corrected l-orbital for each participating atomic species, as well as the value of the Hubbard U and exchange J terms. If there is any value of J different from zero, the anisotropic DFT+(U-J) approach of Anisimov [58] and O'Reagan [59, 60] will be set for VASP and ONETEP, respectively. In contrast, if all values of J are zero, the isotropic DFT+U method of Dudarev [61] will be considered for all codes. For CP2K and CASTEP, in contrast, the anisotropic approach is not implemented, and the value of (U-J) will only be treated as isotropic. The **&hubbard** sub-block requires the specification of sub-block **&magnetization**.

Option **bands** can be set in case the user needs to improve convergence of the electronic minimization. In VASP, the specified value will correspond to the total number of bands for the system, whereas for CASTEP and CP2K it corresponds to the inclusion of extra bands. Finally, and only for VASP simulations, the user must specify how the number of bands that are treated in parallel using directive **npar** [49]. If the Brillouin zone needs to be sampled using multiple k-points, the user can further optimise the parallelization using directive **kpar** [49].

3.12.2   Ionic degrees of freedom

Block **&motion_settings** is needed to specify how ionic degrees of freedom will be described in the simulation. Directives within this block must be compatible with the option selected for **simulation_type**. It is compulsory to set the directive **ion_steps** to specify the number of ionic steps for the simulation.

If **simulation_type** is set to *relax_geometry*, the user must specify directive **relax_method**. The convergence value for the ionic forces can be set using the **force_tolerance** directive. By default, the volume and shape of the simulation cell are kept fixed. Thus, if the user wants to relax the volume and the shape of the cell, directives **change_cell_volume** and **change_cell_shape** must be set to .True., respectively. ONETEP does not allow simulations with variable cell. Cell relaxation can be carried out under an external pressure defined by **pressure**. IMPORTANT: simulation settings are not set to preserve the crystal symmetry. This is because the implemented algorithm to generate atomistic models uses a random number generator to insert and remove species, which inevitably breaks any space group symmetry.

To set a MD simulation, the adopted ensemble and the system temperature must be specified using **ensemble** and **temperature**, respectively. The time step for the simulation is set by default to 1.0 fs, but it can be modified via directive **timestep**. This information is sufficient for the NVE ensemble. Specification of extra directives depends on the choice for the ensemble:

• NVT: **thermostat** and **relax_time_thermostat** are needed. ALC_EQCM only sets thermostats that act over the whole system.

• NPT: the **thermostat** and **pressure** directives must be defined, as well as **relax_time_thermostat** and **relax_time_barostat**. Only for the CASTEP code, the user must define **relax_time_barostat**. For VASP and CP2K, the barostat properties are determined from the choice of the thermostat. Both **change_cell_volume** and **change_cell_shape** must be activated. To date, NPT simulations are not implemented in ONETEP.

• NPH: **pressure** and **relax_time_barostat** are needed. As for NPT, both **change_cell_volume** and **change_cell_shape** must be .True.. Specification of directive **barostat** is also needed for CASTEP settings. To date, ONETEP does not allow NPH simulations.

In case of species containing isotopes, the user can use the sub-block **&masses** to specify the masses of the involved atomic species. See sec. 3.14.4 for more details.

Note: ALC_EQCM only allows generating input files for geometry optimization and MD simulations. Any other type of calculations such as band structure, EELS, Time-dependent DFT, Phonons, NMR, non-colinear magnetization, etc, are excluded from ALC_EQCM. Files for single point simulations, even though of little use within the context of ALC_EQCM, can also be set by choosing the option *relax_geometry* for directive **simulation_type** and setting the **ion_steps** directive to 1.

3.12.3   Specification of extra directives

Although the implemented capabilities aim to offer a general framework to building input files for simulations, they are not yet sufficient to include all the possible available options and features of the codes. To try to amend for this limitation, ALC_EQCM offers the sub-block **&extra_directives** where the user can specify additional directives that are not considered in the implementation.

Even though this block aims to increase the range of applications and functionalities of the generated input files, the user is fully responsible for the correct definition of these extra directives. The code will only check that there is no duplication in the definition of directives. Definition of blocks within **&extra_directives** are not allowed.

This block is very convenient to define specific parallelization and optimization directives, as well as instructions to improve the convergence of the calculations and control input/output information. At present this functionality is implemented only for the VASP, CASTEP and ONETEP codes.

3.12.4   Arbitrary settings for selected CP2K directives

Settings for **&block_simulation** are aimed to capture most of the functionalities for geometry relaxation and MD simulations with CP2K. Still, we had to arbitrarily define additional (unavoidable) directives based on our previous experience with the code. Such directives are listed in Appendix A. If any of such directives needs to be changed, the user has to do it manually, as the use of **&extra_directives** (sec. 3.12.3) is not allowed for the CP2K format.

3.12.5   Summary description of simulation settings

Following the generation of input files for simulations, ALC_EQCM prints a summary of the simulation settings in the OUT_EQCM file. Depending on the settings, there might be simulations directives that need to be adjusted by the user, either to improve the convergence of the electronic problem, optimise the ionic relaxation or correct the settings for MD simulations. To guide the user with this task, ALC_EQCM prints a message with the following header:

```
************************************************************

Aspects to take into consideration for simulations with XYZ

************************************************************
```

indicating which directives need to be monitored for a better performance and correctness of the simulation. XYZ will be the name of code selected to build the input files. There could be directives that ALC_EQCM fails to define correctly. It is user's responsibility to check that the generated parameters are suitable to the problem under consideration.

3.13   Script files for HPC execution

Simulations for the generated atomistic models (secs. 3.10 and 3.11) necessarily require of parallel computing infrastructures to divide the large computational task in smaller tasks that are executed simultaneously using several CPUs. Such a multi-CPUs infrastructure is known as High-Performance-Computing (HPC) facility. To execute parallel simulations in HPC facilities, however, it is necessary to generate script files that instruct the HPC platform (e.g. SLURM) how to make use of the available CPUs and nodes to distribute the computational task. A node is defined as a set of CPUs.

Generation of HPC script is possible via the specification of **&block_HPC_settings** (see Table 3). In case there is a wrong or missing specification, ALC_EQCM will abort and ask the user to address the problem. Even though ALC_EQCM has been developed and tested using the STFC facility SCARF [62], the definition of settings can be applied to any other HPC facilities. The order for the directives as shown in Table 3 is not mandatory.

The user must provide the name of the HPC machine used for the simulations via directive *machine_name*. Directive *platform* refers to the job scheduling implemented in the HPC facility, and it is required if the *machine_name* is different from *SCARF*. Various options are possible for the HPC platform (SLURM, LSF, PSUB, etc), but only SLURM has been implemented to date.

Directive **project_name** is optional, while **job_name** is compulsory and used to tag the simulation during the run. The number of processes (or tasks) used for the simulation is defined with **number_mpi_tasks** and must be carefully set according to the problem to be computed. Relevant to the available HPC architecture, the user must set the number of requested nodes (**number_nodes**), as well as the number of CPUs per node (**CPUs_per_node**). The choice for **parallelism_type** (MPI-Only or MPI-OpenMP) depends on how the code was built and compiled. If option MPI-OpenMP is set for **parallelism_type**, the user will be asked to provide the value for **threads_per_CPU**.

To select the HPC partition to job submission, user must specify directive **queue**, together with the maximum allocation time via **time_limit**. To specify **time_limit**, the user must set a list of three integers, which correspond to days, hours and minutes, respectively. To date, the maximum time limit for jobs in SCARF is 7 days. It is users' responsibility to verify the maximum allocation time of the HPC facility.

Directive **executable** is compulsory and can be provided as a path. If the code needs loading modules before job execution, the user must provide them using sub-block **&modules**. Directive **exec_options** must be defined if the user needs to specify an option for mpirun (different from -np). Similarly, if the code was compiled with MLK libraries, directive **mkl** must be set to True.

HPC script files will be generated in each of the sub-folders together with the atomistic models and input files for simulations. These files are named as hpc_script-format.sh, where, to date, *format* can either be vasp, cp2k, castep or onetep, depending on the option for **output_model_format**. A summary of HPC settings is also printed towards the end of the OUT_EQCM for user's reference. The tutorial exercise of sec. 6.14 provides a worked example of this functionality.

IMPORTANT: Relevant to CPU use and HPC optimization, the user can also specify the memory requirements with **memory_per_CPU** (in MegaBytes). Nevertheless, unless the user is fully certain of the memory requirements for the simulation, we strongly recommend to omit the specification of **memory_per_CPU**.

3.14 Extra functionalities and features

3.14.1 Atomistic models as initial input for the EPSR-software

The Empirical Potential Structure Refinement (EPSR) is a computational tool that aims to address the problem of calculating three-dimensional structures based on the information contained in diffraction data experiments [63]. It is based around an atomistic Monte Carlo simulation of a system, correcting the employed pair interaction potentials by comparison with the diffraction data sets.

From the functionalities described in section 3.10 and 3.11, it is possible to generate atomistic models as initial input for EPSR. This can be done by setting the option cif for **output_model_format**, which generates structures with file name SAMPLE.cif, in .cif format. To generate the required .ato file extension for the execution of EPSR, the user must open SAMPLE.cif with the software ATEN [64]. Once opened, the user has the option to remove and build the bonds, and save the structure with .ato extension.

3.14.2 Building models for disordered systems

In section 3.11 we presented the capability of ALC_EQCM to build models for pristine samples. The target stoichiometry for the models is set in **&block_species**. The implemented algorithm reads the input structure, identifies the number of variable species and modifies their amount to match the target stoichiometry.

Although the main scope of ALC_EQCM is to provide models for intercalation and electrodeposition, the code also allows building models for disordered systems, such as melted crystals or liquids. This can be executed by setting the option *model_disordered_system* for **Analysis**, together with the definition of a set of "fixed" species. In contrast to building models for intercalation or electrodeposition, "fixed" species in this context refers to a set of atoms used as starting configuration to generate the disordered

system. The minimum number of "fixed" species is 1. The minimum number atoms/molecules is also 1. A single "fixed" species can be, for example, a solute atom (such as Li) or an organic molecule (e.g. benzene). It is also possible to have many "fixed" species. Fixed species must be defined as "solute" in **&block_species_components**.

Independently of the number of atoms/molecules for the "fixed" species, the user must define the species with the minimum number of constituents, and set to 1 the pristine stoichiometry for such fixed species in **&block_species**. The stoichiometry of any other "fixed" species must be defined with respect to the species with the minimum number of constituents. Likewise, the stoichiometry values for variable species to be inserted within the simulation box will depend on the amount of target constituents. For example, if we have species "1" with only 2 constituents and species "2" with 3 constituents, one must set the pristine stoichiometries equal to 1 and 1.5 for species "1" and "2" in **&block_species**, respectively. If we aim to insert 300 water molecules within the simulation cell, we must define water species as "dependent" and set a value of 300/2=150 for its stoichiometry. We refer the user to the tutorial example of sec. 6.16.4.

3.14.3 Files for simulations and HPC scripts without re-computing atomistic models

Building atomistic models (sec. 3.10 and 3.11) is significantly more time consuming than generating input files for simulation (sec. 3.12) or scripts files for HPC (sec. 3.13). In fact, depending on the system and the involved species, generating atomistic models can take up to several minutes in contrast to less than a second to derive input files for simulation and HPC. In the previous sections, we have shown that all these files can be obtained at once, as long as the user includes the corresponding blocks in the SET_EQCM file.

Let us assume the two following hypothetical scenarios, where the user executes ALC_EQCM to obtain atomistic models

1) **without having defined** the required blocks for generating input files for atomistic simulations and/or files for submission of simulations to a HPC machine,

2) **having defined** the blocks to generate input files for atomistic simulations and/or HPC script, but after further analysis the user realises either that simulation or HPC settings (or both) must be adjusted/changed.

Re-running the code with the new simulation/HPC settings via option *model_cycled_sample*, *model_pristine_sample* or *model_disordered_system* is not convenient, because atomistic models will be rebuilt, and this is exactly what the user wants to avoid. In addition, the new generated models will be different from those generated in the first place, because the insertion and removal of species are conducted using a random number generator (sec. 3.10). To generate or update simulation/HPC files and keep the atomistic models, ALC_EQCM offers the option *hpc_simulation_files* for **Analysis**. With this option, the code reads file RECORD_MODELS, located in folder **RESTART**, which is created when running the code with option *model_cycled_sample*, *model_disordered_system* or *model_pristine_sample*. The RECORD_MODELS file contains the minimum information for each of the atomistic models in a format that allows ALC_EQCM to characterize them. This information, together with the new specification for the simulation and HPC blocks is sufficient to generate the corresponding files. Following this step, the algorithm corroborates that relevant folders and files exist. By comparison between the new generated files with the information found for each model, the code will inform the user which files remain unchanged and which files have been generated or updated to comply with the new specifications.

Finally, the code prints a summary of the new specifications to file OUT_EQCM together with recommendations to execute the atomistic simulations based on the new settings. See section 6.16.1 for an example case of this functionality.

3.14.4 Including isotopes for MD simulations

By default, there is no requirement to specify the atomic masses for MD simulations. Nevertheless, if the user wants to consider isotopes in the simulation, ALC_EQCM offers the sub-block **&masses**, within **&motion_settings**, where the user must define the mass (in atomic units) for all the atomic tags.

The assigned atomic masses must be consistent with the masses of the species defined in **&Block_species**, whose atomic composition is set in **&block_species_components**. For example, let us assume we need to define the species deuterated water D2O. We use the following line in **&block_species_components**:

```
&Block_species_components
  ⋮
  D2O  2  molecule  1.2
  D H 2 || Ow O 1
  ⋮
&End_block_species_components
```

And define the masses for these atomic constituents as follows:

```
&masses
  tags   ...   D   ...   Ow
  values ...   2.0  ...   16.0
&end_masses
```

Consequently, the mass assigned to the species D2O in **&Block_species** must be equal 20.0, otherwise the code will abort the execution.

Note 1: For CASTEP MD simulations with the sub-block **&masses**, all those atomic tags that are not set equal to their chemical element must be defined using the symbol ":". The chemical element must be defined before the ":" and any additional specification after ":". For example, deuterium could be defined as "H:D".
Note 2: This functionality is not available for ONETEP.

3.14.5 Optimising the intercalation of species

During the generation of atomistic models for the intercalation of species, it may happen that the input structure is not large enough to accommodate the required amount of species to match the target stoichiometry. As explained in sections 3.10 and 3.11, the implemented algorithm uses a random number generator to select spatial points within the simulation cell and rotate molecular species. If the value assigned to *distance_cutoff* is rather large and/or the input structure does not contain sufficient empty space, the number of attempts to insert the species might reach the maximum number of interactions. For such cases, the user will get an error message like the following:

```
***ERROR: fail to insert unit 3 (out of 3) for species "Water"
The maximum limit of attempts to insert the species has been reached.  Possible reasons:
1) an incorrect geometry for the input structure.
2) the input structure is not large enough to accommodate the required species.  The
user might consider enlarging the input structure by setting directive "scale_cell" to
larger than 1.
3) a rather large value for the cutoff distance between species, set to 2.10 Angstrom.
The user could try reducing the value of directive "distance_cutoff".
```

```
4) for large molecular species, it is convenient to set "rotate_species" to .False.  and
adapt the orientation of the molecule to the input structure.
```

In this case, the algorithm fails to insert the last water molecule. The error message is meant to guide the user to fix this problem. The first suggestion is to check the geometry of the input structure. Assuming that the user has not made a mistake here, the next two advises instruct the user to consider either enlarging the input structure with directive **scale_cell** (default is 1.0) or reduce the value of **distance_cutoff** (set to 2.10 Å). The value for **scale_cell** should be used with care to avoid increasing the distances between atoms beyond a reasonable physical limit. Depending on the system, the user should decide which option is the most convenient.
Inserting large molecular species could be also problematic. For such cases, we advice to define the structure of the molecular species in the .xyz file in a convenient orientation with respect to the geometry of the input structure and prevent random rotations by setting **rotate_species** to .False. to maximise the chances of success. We refer the user to the example case of sec. 6.16.3.
For electrodeposition, one can always increase the vacuum region. For large molecular species, however, one must consider geometrical orientation of the molecule to being inserted.

3.14.6   Grand-Canonical DFT for electrochemical systems

To simulate interfaces under electrochemical conditions, it is required to fix the electrode potential. Consequently, the system under consideration can freely exchange electrons with the external circuit (electron reservoir). For such systems, one needs to use the Grand-Canonical extension of DFT (GC-DFT), where the relevant quantity to minimize is not the total energy but the grand-potential [65]. To date, generating input files for GC-DFT simulations with ALC_EQCM is only possible for the ONETEP code via the definition of the **&gcdft** sub-block (see Table 3). We refer to section 6.16.5 for a example case that allows generating input files for GC-DFT simulations with ONETEP.

3.14.7   Including implicit solvent

The simulation of solid-liquid interfaces still constitutes a major challenge to computational research. This is not only due to the complexity of the interactions at the interface between two different phases (liquid and solid), but also related to the need of sampling multiple configurations for the liquid constituents to extract meaningful thermodynamic quantities, which requires of lengthy simulations and intensive computational power.

To circumvent this inherent limitation, empirical models for describing bulk liquids [66], suitable for solid-liquid interfaces have been implemented with the DFT framework [67] and helped to make important advances towards reliable, yet efficient, computation of electrochemical systems. Such empirical models allow including the effect of the solvent implicitly. We refer the reader to the paper of A. Roldan [68] and references therein for a more detail description of the challenges and advances in this field.

To date, ALC_EQCM allows setting parameters for simulations with implicit solvent only for the ONETEP code [69]. This is done with the definition of the **&solvation** block, which asks the required input directives for the computation of the Poisson equation.

The implicit solvent is included by defining a smooth dielectric cavity around the explicit atomic system. Within the cavity, the relative permittivity is 1, whereas in the bulk region of the implicit solvent it must be ϵ_{bulk}. Such a transition for the permittivity must be also smooth, so the solution of the electrostatic equation is numerical stable. The value of ϵ_{bulk} must be set with directive **permittivity_bulk**, where the word bulk refers to the region of the solvent far from any other phase or solute.

ALC_EQCM offers the possibility to set instructions for three different models of the permittivity. The selected model must be defined with directive **dielectric_function**. If the option chosen is *Fattebert-Gygi*, the permittivity is set to depend on the charge density $\rho(\mathbf{r})$ as follows:

$$\epsilon[\rho(\mathbf{r})] = 1 + \frac{\epsilon_{bulk} - 1}{2}\left(1 + \frac{1 - (\rho(\mathbf{r})/\rho_0)^{2\beta}}{1 + (\rho(\mathbf{r})/\rho_0)^{2\beta}}\right) \tag{40}$$

where the spatial dependence of ϵ is subject to the spatial dependence of ρ [70]. Parameters for the exponent β and the density threshold ρ_0 must be defined with directives **beta** and **density_threshold**, respectively. If the *Andreussi* option is selected for the **dielectric_function** directive, the functional form for ϵ is given by the following expression [67]:

$$\epsilon[\rho(\mathbf{r})] = \begin{cases} 1 & \rho(\mathbf{r}) > \rho_{max} \\[2ex] exp[t(\ln \rho(\mathbf{r}))] & \rho_{min} < \rho(\mathbf{r}) < \rho_{max} \\[2ex] \epsilon_{bulk} & \rho(\mathbf{r}) < \rho_{min} \end{cases} \tag{41}$$

where the function $t(\ln \rho(\mathbf{r})$ reads

$$t(\ln \rho(\mathbf{r})) = \frac{\ln[\epsilon_{bulk}]}{2\pi}\left[2\pi\frac{(\ln \rho_{max} - \ln \rho(\mathbf{r})}{(\ln \rho_{max} - \ln \rho_{min})} - \sin\left(2\pi\frac{(\ln\rho_{max} - \ln \rho(\mathbf{r})}{(\ln\rho_{max} - \ln\rho_{min})}\right)\right]. \tag{42}$$

The value of ρ_{min} and ρ_{max} must be specified with the **density_min_threshold** and **density_max_threshold** directives, respectively.

The last implemented option for **dielectric_function** is *soft_sphere*. This method is based on defining the cavity using atom-centered overlapping spheres. Details for this method can be found in Ref. [71]. In contrast to the previous two methods, the transition region is controlled using the Δ parameter, which must be set with the **soft_sphere_delta**. By default, the radii for the soft spheres are obtain from the vdW radii and can be scaled using the **soft_sphere_delta**. Alternatively, the soft sphere radii for each species can be defined with the **soft_sphere_radii** block (see Table 6).

From Ref. [72], an important parameter for convergence in ONETEP is the width if the Gaussian functions used to smear the charge distribution of the ions. This must be set with the **smear_ion_width** directive, and the recommended value is 0.8, which must be given in atomic units.

During the simulation, it is expected that the cavity changes its shape, influenced by the changes in geometry and electronic density. By selecting the *self_consistent* for the **cavity_model** directive, the shape of the cavity is allowed to change during the simulation. This is, however, inconvenient from the computational and numerical point of view, as extra terms in the energy gradients are needed. Because these terms tend to vary rapidly over very localised regions of space, their accurate calculation usually requires unreasonably fine grids and becomes prohibitively difficult for large systems [69].

This problem can be circumvented by choosing the *fixed* option for the **cavity_model** directive. This setting keeps fixed the shape of the cavity, which is computed from an initial simulation in vacuum.

Relevant to the computation with implicit solvent is the inclusion of the apolar terms using the **apolar_terms** directive. For a comprehensive review of the standard methods and approximations, the user is referred Refs. [70, 67, 71]. Depending on the approximation set for **apolar_terms**, ALC_EQCM will ask for different directives, including the surface tension of the solvent (**solvent_surface_tension**) and the pressure used to calculate the SAV contribution (**solvent_pressure**). To include the cavitation term only, the *only_cavitation* option must be selected, and ALC_EQCM will ask the user to set **apolar_scaling** equal to 1.0. Apolar corrections can be omitted by selecting the *None* for **apolar_terms**. See Table 6 for details.

Inclusion of electrolyte ions

ALC_EQCM also allows the user to define electrolyte ionic species to be part of the implicit solvent. Such ions are referred to as Boltzmann ions, and the problem is solved using the Poisson-Boltzmann equation

(PBEq), which is coupled with the DFT problem for the explicit atomic species. Details of the implemented algorithms for the solution of the PBEq in ONETEP can be found in Refs. [73, 74].

In ALC_EQCM, the settings for electrolyte ions are defined using the **&electrolyte** block. We refer the user to Table 7 for details. The user must first specify the treatment for the computation of the PBEq with the *solver* directive, and opt between the full non-linear solution (*full*) and the linear approxima- tion (*linearised*). To compensate for the addition of extra charge to the system, the user must also define the *neutral_scheme* directive. When the explicit quantum system is charged, electrolyte ions with opposite charge will be attracted. To prevent the Boltzmann ions from becoming too close to the solute interface and leading to unphysical concentrations, the *steric_potential* directive offers the possibility to introduce a repulsive potential which, depending on the selected option, it requires the definition of the *steric_isodensity* and *steric_smearing* directives. To help the numerical stability for the convergence of the full PBEq, the user can also define a value for the (exponential) *capping* directive.

The temperature for the Boltzmann ions must be set with the *boltzmann_temperature* directive. The solvent radius cutoff for each atomic species must be defined with the **&solvent_radii** block. Finally, the model parameters for the electrolyte Boltzmann ions must be specified in the **&boltzmann_ions**. ALC_EQCM is robust enough to help the user to define the settings depending on the selected options.

Settings for the DL_MG solver

The user can also use the **&extra_directives** block to define settings for the multigrid solver (DL_MG) [75]. Relevant to calculations with implicit solvent, the user should consider to adjust the following keywords (see ONETEP manual for description):

- mg_use_cg

- mg_max_iters_vcycle

- mg_vcyc_smoother_iter_pre

- mg_vcyc_smoother_iter_post

- mg_defco_fd_order

- mg_max_iters_newton

In section 6.16.5 the user will find instructions for a example case that allows generating input files for DFT including implicit solvent and electrolytes for ONETEP.

3.14.8 Additional scripts

ALC_EQCM also offers a set of scripts files located within folder **scripts** in the root directory. At the present stage of development, the following scripts are available:

- **cif-input-geom.py**: this is a Python script that allows reading the INPUT_STRUCTURE file in `.cif` format. This script must be copied to the working directory. Both Python and ASE [76] softwares must be installed in the machine. If the script is not found, ALC_EQCM will abort the execution and instruct the user.

- **cif-output-geom.py**: this is a Python script that allows generating atomistic structures in `.cif` format. This script must be copied to the working directory. Both Python and ASE [76] softwares must be installed in the machine. If the script is not found, ALC_EQCM will abort the execution and instruct the user what to do.

- **clean_data.sh**: in case the user decides to rerun ALC_EQCM, the execution of this script is very con- venient to delete all the generated files and folder generated previously without taking the risk to deleting any relevant input file by mistake. This scripts can be copied to the working directory or invoked from the working directory. The user must use the Linux command **sh** to execute this script. For example, if the file has been copied to the working directory, the user must execute "`sh clean_data.sh`".

In addition, we provide the following template files with directives to the execution of different analysis:

- SET_EQCM-characterization

- SET_EQCM-mass_calibration

- SET_EQCM-massogram

- SET_EQCM-model_cycled_sample

- SET_EQCM-model_cycled_sample

- SET_EQCM-print_EQCM_filter

- SET_EQCM-print_EQCM_raw

- SET_EQCM-spectra

- SET_EQCM-stoichiometry

The user must remove the symbol "#" symbol in front of the directive and complete the specification. To help on this task, the user can find instructions for directives in Table 3 or follow the settings examples of the tutorials.

3.15   Units convention

Table 1 summarizes the units that ALC_EQCM adopts for each input variable and directive, as well as the valid input units. Those input directives not listed below are dimensionless. Adopted units for mass, mass-density and reciprocal space are ng, ng/cm^2 and Hz (or 1/V), respectively.

Table 1: Adopted internal units and possible valid units for all input variables

Variable	Adopted internal units	Accepted input units
Data analysis		
Potential or Voltage	V (Volts)	V
Current	mA (mili-Amperes)	A , mA, uA (microA), nA (nanoA)
Charge	mC (mili-Coulombs)	C, mC , uC (micro) or nC (nano)
Time	s (seconds)	s
Mass	ng	mg (mili), ug (micro), ng (nano)
Frequency	Hz	Hz, V
Resistance	V	V
Sauerbrey	Hz ng-1 cm2	Hz ng-1 cm2
Quartz_freq	Hz	Hz, kHz, MHz
Electrode_mass	ng	g, mg, ug and ng
Electrode_area	cm2	cm2, inch2
voltage_range	V	V, mV
scan_rate	V s-1	V s-1, mV s-1
Filter_cutoff	V-1, Hz	V-1, Hz
Atomistic models		
distance_cutoff	Angstrom	Angstrom, Bohr
delta_space	Angstrom	Angstrom, Bohr
Note: Units for the following directives are transformed internally but adopted as specified. For example, if units for Energy_cutoff are defined as eV, the adopted internal units will be eV. For quantifies defined within blocks, please refer to their description for the valid input units.		
DFT		
Energy_cutoff		eV, Ry
Smear_width		eV
SCF_energy_tolerance		eV, Hartree
temperature		K
pressure	As specified by the input unit	kb (or kbar)
force_tolerance		eV Angstrom-1, Hartree Bohr-1
timestep		fs (or fsec)
relax_time_thermostat		fs (or fsec)
relax_time_barostat		fs (or fsec)
EDFT		
reference_potential	As specified by the input unit	eV
electrode_potential		V
Implicit solvation		
smeared_ion_width		Bohr
solvent_surface_tension		N m-1
solvent_pressure	As specified by the input unit	GPa
Boltzmann_temperature		K
steric_smearing		Bohr

4 Input-Output structure

In this section we describe the content and structure of the input files for the execution of ALC_EQCM, as well as the generated output.

4.1 General description

Table 2 summarises the input/output filing structure for all the implemented types of analysis.

Table 2: Description of the required (input) and the generated (output) files for all the possible types of analysis. Files are listed with bullet points. Folders are reported in bold. Files listed below folders indicates that such files are stored within that particular folder. Input files SET_EQCM and DATA_EQCM as well as output file OUT_EQCM are located in the working directory.

Option for **Analysis**	Required input files	Output files and folders
print_EQCM_raw (section 3.3)	• SET_EQCM (4.2.1) • DATA_EQCM (4.2.2)	• OUT_EQCM (4.3.1) **ANALYSIS_EQCM** • RAW_CURRENT • RAW_MASS (See sec. 4.3.2)
print_EQCM_filter (section 3.3)	• SET_EQCM (4.2.1) • DATA_EQCM (4.2.2)	• OUT_EQCM (4.3.1) **ANALYSIS_EQCM** • FILTERED_CURRENT • FILTERED_MASS (See sec. 4.3.2)
mass_calibration (section 3.4)	• SET_EQCM (4.2.1) • DATA_EQCM (4.2.2)	• OUT_EQCM (4.3.1) **ANALYSIS_EQCM** • MASS_CALIBRATION (See sec. 4.3.3)
massogram (section 3.5)	• SET_EQCM (4.2.1) • DATA_EQCM (4.2.2)	• OUT_EQCM (4.3.1) **ANALYSIS_EQCM** • MASSOGRAM (See sec. 4.3.4)
spectra (section 3.1)	• SET_EQCM (4.2.1) • DATA_EQCM (4.2.2)	• OUT_EQCM (4.3.1) **ANALYSIS_EQCM** • SPEC_CURRENT • SPEC_MASS (See sec. 4.3.5)
characterization (section 3.6)	• SET_EQCM (4.2.1) • DATA_EQCM (4.2.2)	• OUT_EQCM (4.3.1) **ANALYSIS_EQCM** • CHARACTERIZATION (See sec. 4.3.6)

stoichiometry (section 3.7 and 3.8)	• SET_EQCM (4.2.1) • DATA_EQCM (4.2.2)	• OUT_EQCM (4.3.1) **ANALYSIS_EQCM** • CHARACTERIZATION (4.3.6) If **process** *intercalation*: • INTERCALATION_OX • INTERCALATION_RED (See sec. 4.3.7) If **process** *electrodeposition*: • ELECTRO_DEPOSITION (See sec. 4.3.8)
fluxes (section 3.9	• SET_EQCM (4.2.1) • DATA_EQCM (4.2.2)	• OUT_EQCM (4.3.1) **ANALYSIS_EQCM** • CV_MOLES (4.3.9) • MOLE_FLUX (4.3.10) • IONIC_CURRENT (4.3.11)
model_cycled_ sample (section 3.10)	• SET_EQCM (4.2.1) • DATA_EQCM (4.2.2) **INPUT_GEOM** • INPUT_STRUCTURE (4.2.3) • .xyz files (4.2.4) **DFT** • BASIS_SET (4.2.6) • vdW kernel (4.2.7) **DFT/PPs** • pseudo-potentials (4.2.5)	All output files from *stoichiometry* + **ANALYSIS_EQCM** • SELECTED_SOLUTIONS (only for multiple solutions, see sec. 4.3.12) **ATOMISTIC_MODELS/subfolders** • SAMPLE.format (4.3.13) • MODEL_SUMMARY (4.3.14) • Files for atomistic simulations (4.3.17) • HPC scripts for simulations (4.3.18) **RESTART** • OUT_EQCM_RESTART (4.3.15) • RECORD_MODELS (4.3.16)
model_pristine_ sample (section 3.11)	• SET_EQCM (4.2.1) **INPUT_GEOM** • INPUT_STRUCTURE (4.2.3) • .xyz files (4.2.4) **DFT** • BASIS_SET (4.2.6) • vdW kernel (4.2.7) **DFT/PPs** • pseudo-potentials (4.2.5)	• OUT_EQCM (4.3.1) *No EQCM related file is generated* **ATOMISTIC_MODELS/pristine** • SAMPLE.format (4.3.13) • MODEL_SUMMARY (4.3.14) • Files for atomistic simulations (4.3.17) • HPC scripts for simulations (4.3.18) **RESTART** • OUT_EQCM_RESTART (4.3.15) • RECORD_MODELS (4.3.16)
model_disordered_ system (section 3.14.2)	• SET_EQCM (4.2.1) **INPUT_GEOM** • INPUT_STRUCTURE (4.2.3) • .xyz files (4.2.4) **DFT** • BASIS_SET (4.2.6) • vdW kernel (4.2.7) **DFT/PPs** • pseudo-potentials (4.2.5)	• OUT_EQCM (4.3.1) *No EQCM related file is generated* **ATOMISTIC_MODELS/disordered** • SAMPLE.format (4.3.13) • MODEL_SUMMARY (4.3.14) • Files for atomistic simulations (4.3.17) • HPC scripts for simulations (4.3.18) **RESTART** • OUT_EQCM_RESTART (4.3.15) • RECORD_MODELS (4.3.16)

hpc_simulation_files (section 3.14.3)	• SET_EQCM (4.2.1) **DFT** • BASIS_SET (4.2.6) • vdW kernel (4.2.7) **DFT/PPs** • pseudo-potentials (4.2.5) **RESTART** • RECORD_MODELS (4.3.16)	• OUT_EQCM (4.3.1) **ATOMISTIC_MODELS/subfolders** • Files for atomistic simulations (4.3.17) • HPC scripts for simulations (4.3.18)

4.2 Input files

4.2.1 The SET_EQCM file

This file contains the directives for analysis and must be present in the folder where the ALC_EQCM is executed from, otherwise the program will print an error message and abort. Comments can be added using the symbol #. It is recommended to add a descriptive header in the first lines of the file (for revision purposes). We report the description of implemented directives and valid blocks in Table 3. Different files will be printed Depending on the selected options. The code is flexible enough to recognize directives independently of capitalization. For example, directive *Analysis*, *aNalysIs*, *anAlySiS*, etc, will all be interpreted as *analysis*. To complement the description of Table 3, the implemented options for selected DFT directives are shown in Table 4 and 5.

The code checks for the correct syntax and format of the defined directives and blocks. If a problem is found, an error message will be printed in file OUT_EQCM and the execution is aborted. A summary of these directives is printed to file OUT_EQCM. For all options of *Analysis* except *model_pristine_sample*, *model_disordered_system* and *hpc_simulation_files*, the code reads the EQCM data from file DATA_EQCM (see sec. 4.2.2), and the information is compared against the directives of the SET_EQCM to check the feasibility of the requested simulation. In case there is an inconsistency but the calculation can still proceed, ALC_EQCM prints a warning message. If the required settings are not sufficient to perform the calculation, the program will stop and print an error message instructing the user what to fix.

Table 3: Description for directives of file SET_EQCM

Directive/Block	Format	Description and Specification
Analysis	*String*	Type of analysis to be executed. Options are: • *print_EQCM_raw*: prints raw EQCM current and mass-density (or mass) as a function of the applied potential. • *print_EQCM_filter*: prints filtered EQCM current and mass-density/mas after applying a Gaussian low-pass filter. • *mass_calibration*: prints mass-frequency and charge for calibration of the EQCM device. • *massogram*: computes and prints the mass rate as a function of the applied voltage. • *Spectra*: computes and prints current and mass-density/mass spectra in reciprocal space. • *Characterization*: performs quantitative EQCM analysis for each redox cycle. • *Stoichiometry*: computes stoichiometric changes from EQCM data. This option requires the specification of directive **process** (see below). If **process** is set to *intercalation*, ALC_EQCM computes the set of stoichiometric solutions consistent with the charge and mass balance equations for the species specified in **&block_species**. If **process** is set to *electrodeposition*, ALC_EQCM computes the number of moles involved in the process and the ratio R as defined in sec. 3.8. • *fluxes*: computes mole changes, mole fluxes and ionic current along the CV scan for intercalation reactions (sec 3.9). • *Model_pristine_sample*: generates a single atomic model compatible with the stoichiometry defined in **&block_species**. If **&block_dft_settings** and **&block_hpc_settings** are defined in the SET_EQCM file, input files for atomistic level simulation and scripts files for HPC submission are also generated. • *Model_cycled_sample*: generates a set of atomic models compatible with the EQCM data following the stoichiometric solution from the charge and mass mass balance equations. If **&block_dft_settings** and **&block_hpc_settings** are defined in the SET_EQCM file, input files for atomistic level simulation and scripts files for HPC submission are also generated. • *Model_disordered_system*: generates a single atomic model for a disordered system compatible with the stoichiometry defined in **&block_species**, suitable for liquids. If **&block_dft_settings** and **&block_hpc_settings** are defined in the SET_EQCM file, input files for atomistic level simulation and scripts files for HPC submission are also generated. • *hpc_simulation_files*: only generates input files for simulation and scripts for HPC submission without the need to recompute atomistic models.
Software	*String*	Software used for EQCM data collection. This directive is optional. Options are: • *QCM200*: Stanford Research Systems QCM200 QCM • *Metrohm*: Metrohm Autolab • *CH-Inst*: CH Instruments, Inc. Electrochemical Instrumentation.

Cycles	cycle1 cycle2	Range of cycles to be considered. Both cycle1 and cycle2 are positive integers and cycle1 < cycle2. If specified, cycle2 must not be lager than the number of cycles the code identifies from file DATA_EQCM, otherwise the code will print an error message and abort. If not specified, the code considers all the CV cycles that have been identified from DATA_EQCM.
V_to_Hz	real	Conversion factor, only needed if mass-frequency is recorded in Volts.
Filter_cutoff	real units	Cutoff frequency to filter raw EQCM data. Value must be positive. Units must be either V-1 (meaning 1/V) or Hz.
endpoints_current	integer	Number of points at the extremes of the raw current data used to define the values in the padding region (sec. 3.1). Default is 1.
*endpoints_mass_*	integer	Number of points at the extremes of the raw mass/mass-frequency used to define the values in the padding region. Default is 1.
sauerbrey	real units	Sauerbrey factor to convert frequency changes to mass densities. Units must be given in Hz ng-1 cm2 (Hz ng^{-1} cm^2). The user is responsible to provide the right value for these units.
quartz_freq	real units	Resonant quartz frequency for the EQCM device. This directive is to used to compare with the maximum change of the mass-frequency change and corroborate the validity of the Sauerbrey's relation. Valid Units must be in either Hz, kHz or MHz.
scan_rate	real units	Voltage scan rate. This directive is needed to compute charge accumulation when the elapsed time is not recorded. Valid settings for units must be either V s-1 or mV s-1. The user is responsible to verify the scan rate was kept constant during the EQCM experiment.
electrode_area	real units	Quartz disk geometric area. Valid units: cm2 (cm^2) or inch2 (inch^2).
area_scale	real	Multiplicative factor for the area of the electrode disc. Default is 1.0. This is convenient to model effective surface areas, including surface rugosity of electro-deposited films
voltage_range	real1 real2 units	Range of voltages to print EQCM data. Condition real1 < real2 must be satisfied. Units can be given either in V or mV.
current_offset	logical	Offset the current using the value at the beginning of the cycling.
process	*string*	Either *intercalation* or *electrodeposition*.
efficiency	real	Real must be a value between 0 and 1 and indicate the amount of material that participates in the reaction. Only relevant to intercalation. Default is 1.0
delta_stoich	real	Discretization of the stoichiometric domain for each independent chemical species. Default is set to 0.01.
electrode_mass	real units	Mass of the electrode host material before intercalation. Valid units: g, mg, ug and ng.
electrode_moles	real	Moles of the electrode host material before intercalation
input_model_format	*String*	Format of file INPUT_STRUCTURE. Options: *vasp*, *cif*, *xyz*, *onetep* and *castep*.
output_model_format	*String*	Format of the generated atomistic models. Options: *vasp*, *cif*, *xyz*, *onetep*, *siesta* and *castep*
distance_cutoff	Real units	Minimum distance between atoms of different species (default is 1.7 Angstrom). Valid units: Angstrom or Bohr.

deposition_ level	Real units	Sets the level from where species start to be deposited. If not defined, the code identifies the level from the outermost atoms. Valid units: `Angstrom` or `Bohr`.
shift_ structure	Logical	Instruct if atomic positions will be shifted with respect to the configuration of the input structure. If `.True.` (default), the algorithm selects the closest atom to the origin of the frame of reference, and shifts the whole structure such that this atom is set at (0,0,0). The position of all the other atoms are changed to comply with the PBCs.
repeat_ input_model	r1 r2 r3	Defines how many repetitions of the input model, along cell vectors (c_1, c_2, c_3), are set to defined the size of the sample model.
rotate_ species	Logical	Instructs the code to randomly rotate (or not) molecular species for the process of building the atomistic models. This directive is `.True.` by default.
both_ surfaces	Logical	Instructs the code to build models with species deposited at both sides of the slab. This directive is `.False.` by default.
stoichiometry _error	Real	Error tolerance between the targeted stoichiometry and the stoichiometry of generated the models. Default is 0.01.
delta_space	Real units	Discretization criterion along each vector of the simulation cell. Valid units: `Angstrom` or `Bohr`. Default value is 0.1 `Angstrom`
scale_cell	Real	Defines the scaling factor for the input structure. Default is set to 1.
multiple_ input_atoms	Integer	This directive must be defined only if the code asks for it. For guidance, see the generated error message that requests this change. Default value is 10000.
multiple_ output_atoms	Integer	This directive must be defined only if the code asks for it. For guidance, see the generated error message that requests this change. Default value is 2.

&block_species   Block for the specification of the N_s chemical species. Format and syntax:

```
&block_species
  number_species        Ns
  Label1         mass1         nox1         s01         Typevar1
  Label2         mass2         nox2         s02         Typevar2
  :
  LabelNs        massNs        noxNs        s0Ns        TypevarNs
&end_block_species
```

`number_species`: amount of chemical species.
`Label`: species name defined by user (string)
`mass` : Atomic or molecular weight of the species (real > 0)
`nox` : Oxidation number of the species (real)
`s0` : Stoichiometry of species in the pristine sample (real > 0)
`Typevar`: Type of variable for stoichiometry analysis: `fixed`, `dependent` or `independent`. Variables defined as `fixed` must be set for those species that do not participate in the reaction.

&block_constraints_species   Definition of the N_{cstr} constraints between variable chemical species defined in **&block_species**. Format and syntax:

```
&block_constraints_species
  number_constraints        Ncstr
  Type1       cv_fragment1        constrained_species1        set_values1
  Type2       cv_fragment2        constrained_species2        set_values2
```

```
        ⋮
    Type_{N_{cstr}}        cv_fragment_{N_{cstr}}        constrained_species_{N_{cstr}}        set_values_{N_{cstr}}
&end_block_constraints_species
```

Type: type of constraint.
cv_fragment: Oxidation or reduction.
constrained_species: Chemical species to be constrained among dependent and independent variables defined in **&block_species**.
set_values: Constraint values. Two values are required for type target_range. No value is required for type keep_ratio. The rest of the constraints require only one value.

targeted_ number_models	Integer	In case there are multiple stoichiometric solutions, this option instructs the code to select a subset of models among the many possible solutions. Depending on the system, the generated number of models could be different than the value specified for this variable.
normal_vector	*String*	Only for electrodeposition. This option must indicate which lattice vectors is perpendicular to the surface. Options: *c1*, *c2* or *c3*.

&block_species_components Atomistic details for the species defined in **&block_species**. This block is compulsory to build atomistic models. Format and syntax:

```
&block_species_components
 Label_1      Nc_1       class_1       maxbond_1
 Tag_{1,1}   Element_{1,1}  Stoich_{1,1} || ... || Tag_{Nc_1,1}   Element_{Nc_1,1}   Stoich_{Nc_1,1}
 Label_2      N_{c2}      class_2       maxbond_2
 Tag_{1,2}   Element_{1,2}  Stoich_{1,2} || ... || Tag_{Nc_2,2}   Element_{Nc_2,2}   Stoich_{Nc_2,2}
  ⋮
 Label_{N_s}     Nc_{N_s}      class_{N_s}      maxbond_{N_s}
 Tag_{1,N_s} Element_{1,N_s} Stoich_{1,N_s} || ... || Tag_{Nc_{N_s},N_s} Element_{Nc_{N_s},N_s} Stoich_{Nc_{N_s},N_s}
&end_block_species_components
```

Label$_i$: Label of species i, as defined in **&block_species**.
Nc$_i$: Number of atomic components for species i (integer). Example: 2 for Water (H and O).
class$_i$: class for species i. Options: crystal, solute, molecule or atom, depending on the selected option for Typevar in **&block_species**.
maxbond$_i$: Maximum intramolecular bonding distance for species i (only if class = molecule).
||: denotes a separator between component specification. It can be any string (&, $, etc).
Tag$_{i,j}$: Tag for the atomic component i of species j (String). Maximum is four characters.
Element$_{i,j}$: Chemical element for the atomic component i of species j.
Stoich$_{i,j}$: Stoichiometry of the atomic component i in the species j (integer).

Specification for the atomic components of each species does not need to follow the order of **&block_species**, as long as all species are defined. Species labelled with crystal must be defined as fixed. Atomic tags must be unequivocally defined. Note that Tag is not necessarily equal to the atomic element. Example: hydrogen tag could set as H+, but Element must be H.

&block_input_composition For each Tag$_{i,j}$ defined in **&block_species_components**, this block must specify number of atoms present in the INPUT_STRUCTURE file. This block is compulsory. Format and syntax:

```
&block_input_composition
  tags       Tag_{1,1}  Tag_{2,1} ... Tag_{Nc_1,1}   Tag_{1,2} ... ... Tag_{Nc_{N_s},N_s}
  amounts    Nat_{1,1}  Nat_{2,1} ... Nat_{Nc_1,1}   Nat_{1,2} ... ... Nat_{Nc_{N_s},N_s}
```

```
&end_block_input_composition
```

$Nat_{i,j}$: Number of $Tag_{i,j}$ atoms present in file INPUT_STRUCTURE.
Specification must include all the $Tag_{i,j}$ atomic components defined in **&block_species_components**. The order does NOT need to coincide with the specification of **&block_species_components** but must be consistent with order of atoms as listed in file INPUT_STRUCTURE. If atoms with $Tag_{i,j}$ are not part of the INPUT_STRUCTURE file, one must set 0 (zero) for that particular $Tag_{i,j}$.

&block_input_cell   Specification for the cell vectors of the input model, only needed if the format of INPUT_STRUCTURE is XYZ. Format and syntax:

```
&block_input_cell
  c1,1    c1,2    c1,3
  c2,1    c2,2    c2,3
  c3,1    c3,2    c3,3
&end_block_input_cell
```

Each $c_{i,j}$ correspond to the j component of cell vector i. IMPORTANT: values must be given in Å.

&block_simulation_settings   Block with the set of directives to build input files for simulations. Directives do not need to follow a particular order. Comments must start with #.

```
&block_simulation_settings
 simulation_type      String
 theory_level         String
 net_charge           Real
 &DFT_settings (if option DFT is selected for theory_level)
  XC_level             String
  XC_version           String
  vdw                  String
  mixing_scheme        String
  smearing             String
  width_smear          Real      Units
  energy_cutoff        Real      Units
  scf_steps            Integer
  scf_energy_tolerance Real      Units
  bands                Integer
  kpoints              String    Integer₁   Integer₂    Integer₃
  precision            String    (Only for VASP)
  npar                 Integer   (Only for VASP)
  kpar                 Integer   (Only for VASP)
  OT                   Logical   (Only for CP2K)
  GAPW                 Logical    (Only for CP2K)
  EDFT                 Logical    (Only for CASTEP and ONETEP)

  &gcdft (Only available in ONETEP)
    reference_potential   Real      Units
    electrode_potential   Real      Units
    electron_threshold    Real
  &end_gcdft

  &ngwf (Compulsory only for ONETEP)
    tags       Tag1,1   Tag2,1 ... TagNC1,1   Tag1,2 ... ... TagNCNs,Ns
```

```
   number      N_{1,1}    N_{2,1} ...   N_{Nc_1,1}    N_{1,2} ... ...   N_{Nc_{N_s},N_s}
   radius      R_{1,1}    R_{2,1} ...   R_{Nc_1,1}    R_{1,2} ... ...   R_{Nc_{N_s},N_s}
&end_ngwf

spin_polarised         Logical
max_l_orbital          Integer      (Only for VASP)
total_magnetization    Real
&magnetization
   tags       Tag_{1,1}  Tag_{2,1} ... Tag_{Nc_1,1}  Tag_{1,2} ... ... Tag_{Nc_{N_s},N_s}
   values      μ_{1,1}    μ_{2,1} ...   μ_{Nc_1,1}    μ_{1,2} ... ...   μ_{Nc_{N_s},N_s}
&end_magnetization

&hubbard
   tags        Tag_{1,1}  Tag_{2,1} ... Tag_{Nc_1,1}  Tag_{1,2} ... ... Tag_{Nc_{N_s},N_s}
   l_orbital   𝓛_{1,1}    𝓛_{2,1} ...   𝓛_{Nc_1,1}    𝓛_{1,2} ... ...   𝓛_{Nc_{N_s},N_s}
   U           U_{1,1}    U_{2,1} ...   U_{Nc_1,1}    U_{1,2} ... ...   U_{Nc_{N_s},N_s}
   J           J_{1,1}    J_{2,1} ...   J_{Nc_1,1}    J_{1,2} ... ...   J_{Nc_{N_s},N_s}
&end_hubbard

&pseudo_potentials
   Tag_{1,1}        PPfilename_{1,1}
      ⋮                ⋮
   Tag_{Nc_1,1}     PPfilename_{Nc_1,1}
   Tag_{1,2}        PPfilename_{1,2}
      ⋮                ⋮
   Tag_{Nc_{N_s},N_s}  PPfilename_{Nc_{N_s},N_s}
&end_pseudo_potentials

&basis_set     (Only for CP2K)
   Tag_{1,1}        ABS_{1,1}
      ⋮                ⋮
   Tag_{Nc_1,1}     ABS_{Nc_1,1}
   Tag_{1,2}        ABS_{1,2}
      ⋮                ⋮
   Tag_{Nc_{N_s},N_s}  ABS_{Nc_{N_s},N_s}
&end_basis_set

&solvation (See Table 6)
&end_solvation
&electrolyte (See Table 7)
&end_electrolyte
&end_DFT_settings

&motion_settings
  ion_steps            Integer
  #=== geometry relaxation (simulation_type set to relax_geometry)
  relax_method         String
  force_tolerance      Real      Units
  #=== Molecular dynamics (if simulation_type is set to MD)
```

46

```
      timestep               Real      Units
      ensemble               String
      temperature            Real      Units
      pressure               Real      Units
      thermostat             String
      relax_time_thermostat Real       Units
      relax_time_barostat    Real       Units
      change_cell_volume     Logical
      change_cell_shape      Logical
    &masses
       tags       Tag₁,₁   Tag₂,₁ ... Tag_{NC₁,1}   Tag₁,₂ ... ... Tag_{NC_{Ns},Ns}
       values     m₁,₁     m₂,₁ ...   m_{NC₁,1}     m₁,₂ ... ...   m_{NC_{Ns},Ns}
    &end_masses
  &end_motion_settings

  &extra_directives
    directives_1

      ⋮

    directives_N
  &end_extra_directives
&end_block_simulation_settings
```

simulation_type: Type of simulation. Valid options for *String*: *relax_geometry* (geometry relaxation) and *MD* (Molecular dynamics)

theory_level: Level of theory to describe the interactions between atoms. To date, the only valid option is *DFT*.

net_charge: Net charge for the system. If negative, there is an excess of electrons in the system.

XC_level: Level of approximation for the XC. Valid options for *String*: *LDA* and *GGA*.

XC_version: version for the exchange and correlation term. See Table 4 for implemented options.

vdw: Type of van der Waals corrections. See Table 4 for options.

mixing_scheme: Type of mixing for the electronic minimization. See Table 4 for options.

smearing: Type of smearing for electronic convergence (see Table 4).

width_smear: Width for electronic smearing. Real must be positive (default is 0.2 eV). Units: eV.

energy_cutoff: Energy cutoff. Real must be positive. Valid units: eV and Ry.

scf_steps: Maximum number of steps for electronic convergence.

scf_energy_tolerance: Energy tolerance for electronic convergence. Real must be positive (default is 1.0×10^{-4} eV). Valid options for Units: eV and Hartree.

bands: In VASP, this flag must be the total number of bands. In CASTEP and CP2K, this number must be the extra bands added to the set of default bands.

kpoints: Information of the reciprocal space. String must be the method to define the mesh, either MPack or Automatic. The set of three integers specifies the amount of kpoints for each dimension of the Brillouin zone.

precision: Precision for the simulation (only for VASP). Options: *normal*, *single* and *accurate*.

npar: number of bands treated in parallel (only for VASP).

kpar: number of k-points treated in parallel (only for VASP).

OT: activates Orbital Transformation (optional only for CP2K).

GAPW: activates the Gaussian Augmented Plane Waves method (optional only for CP2K. The default method for CP2K is GPW).

EDFT: activates Ensemble-DFT (optional only for ONETEP and CASTEP).

&gcdft: Sub-block to generate instructions for Grand-Canonical DFT simulations.This functionality is only available for ONETEP and requires activation of EDFT. Directives within this sub-block must be *reference_potential* (`Units`: eV, often set to -4.44 eV), *electrode_potential* (`Units`: V) and *electron_threshold* (`Real`> 0).

&ngwf: Sub-block to specify the number and radius of the NGWF for each participating atomic species. The number of NGWF must be larger than zero. If set to -1, ONETEP will assign default values. IMPORTANT: The radii for the NGWFs must be given in Å.

spin_polarised: Flag to activate spin polarised calculations. Default is `.False.`.

max_l_orbital: (only for VASP) Maximum l-orbital among all the participating atomic species. Allowed values: 0, 1, 2 and 3 (for s, p, d and f orbitals, respectively).

total_magnetization: sets the total magnetization. `Real` must be given in `Bohr magneton`.

&magnetization: Sub-block to specify the initial magnetization for the participating atomic species. The user must list all the atomic tags of **&block_input_composition**. Valid units: `Bohr magneton`.

&hubbard: Sub-block to specify the Hubbard corrections applied to specific orbitals of the participating chemical species. The user must list all the atomic tags of **&block_input_composition**. The orbitals to be corrected must be listed for each atomic tag, following the specification of `l_orbital`. Values for the Hubbard `U` and exchange `J` terms must be specified for each atomic tag in units of eV. If there is any value of `J` different from zero, the anisotropic DFT+(U-J) approach of Anisimov [58] and O'Reagan [59, 60] will be set for VASP and ONETEP, respectively. In contrast, if all values of `J` are zero, the isotropic DFT+U method of Dudarev [61] will be considered for all codes. For CP2K and CASTEP, in contrast, the anisotropic approach is not implemented, and the value of (U-J) will only be treated as isotropic. Block **&hubbard** needs the specification of **&magnetization**.

&pseudo_potentials: Sub-block to specify the name of the pseudo potential files (PPfilename) for each of the participating atomic species. The user must list all the the tags of **&block_input_composition**. All PPfilename files must be located within folder **DFT**/**PPs**. If this block is not defined, pseudopotentials files will not be generated.

&basis_set: Only required for CP2K. Sub-block to specify the type of atomic basis set (ABS) adopted for each of the participating chemical species. The user must list all the atomic elements using the tags of **&block_input_composition**. All ABSs must be within file BASIS_SET in folder **DFT**. See Table 4 for the different possible choices of ABS.

ion_steps: Number of ionic steps for the simulation.

relax_method: Method for ionic relaxation. See Table 4 for implemented options. Not all options are available for the implemented codes. ALC_EQCM shall inform the user if the option is not compatible with the choice of the output format.

force_tolerance: Force tolerance for ionic relaxation. `Real` must be positive. Valid options for units: `eV Angstrom-1` (eV/Angstrom) and `Hartree Bohr-1` (Hartree/Bohr).

timestep: Time step for MD simulations. `Real` must be positive (default is 0.1 fs). Units must be in `fs` (or `fsec`).

ensemble: Type of ensemble for the execution of molecular dynamics. See Table 4 for options.

temperature: Temperature for MD simulations. `Real` must be positive. Units must be in `K` (Kelvin).

pressure: Pressure for MD simulations, only needed for `NPT` and `NPH` ensembles. Allowed units: `kb` (kilobars).

thermostat: Thermostat used for MD simulations. See Table 4 for options. Not all options are available in the implemented codes.

relax_time_thermostat: relaxation time for the thermostat.

relax_time_barostat: relaxation time for the barostat (only needed for CASTEP).

change_cell_volume: If set to `.True.`, the volume of the simulation cell is allowed to change.

change_cell_shape: If set to `.True.`, this flag allows for changes in the shape of the simulation cell. Default is `.False.`.

&masses: Sub-block to specify the masses of the participating chemical species. The user must list all the atomic tags of **&block_input_composition**. Values must be given in atomic units.

&extra_directives: Sub-block to include directives corresponding to functionalities that are not implemented. This block is not available for the CP2K format.

&block_HPC_settings   Block with the set of directives to build script files for HPC submission. Directives do not need to follow any particular order. Comments must start with #.

```
&block_HPC_settings
 machine_name            String
 platform                String
 project_name            String
 job_name                String
 number_mpi_tasks        Integer
 number_nodes            Integer
 CPUs_per_node           Integer
 memory_per_CPU          Integer
 parallelism_type        String
 threads_per_process     Integer
 queue                   String
 time_limit              Integer1   Integer2   Integer3
 executable              String
 mkl                     Logical
 exec_options            String
 &modules
   module1

    ⋮

   moduleN
 &end_modules
&end_block_HPC_settings
```

machine_name: Name of HPC facility to run the simulation.

platform: Job scheduling implemented in the HPC machine. To date, only the *SLURM* option is implemented. This directive is not needed if *machine_name* is set to SCARF.

project_name: Name of the project. This is optional

job_name: Name/Tag for HPC simulation.

number_mpi_tasks: Number of processes (tasks) for the simulation.

number_nodes: Number of nodes for the simulation.

CPUs_per_node: Number of CPUs per node, which is determined by the computing architecture where the DFT job will be executed. This number is often equal to 16, 20, 24, 32, etc.

memory_per_CPU: Requested memory per CPU, in MegaBytes (MB). This is optional and should only be used if the user is certain about the memory requirements of the code.

parallelism_type: Parallelization for process distribution (either be *MPI-Only* or *MPI-OpenMP*).

threads_per_process: Number of threads per process. This directive is compulsory if *parallelism_type* is set to *MPI-OpenMP*. If Values for the Integer must be ≥ 1.

queue: Partition of the HPC facility where the job will be submitted to.

time_limit: Time limit for the simulation. Integer1, Integer2 and Integer3 correspond to days, hours and minutes, respectively. The user must check beforehand the time limits of the machine and the queue where the simulation will be submitted.

executable: name of the executable to be launched. String can be also be set as a path to the executable, including the executable.

mkl: it must be set to .True. if MKL libraries were used for the compilation of the code.

exec_options: String to specify options for running, which must be different from -np X.

&modules: Sub-block to include modules that might be required for the execution of the job.

Table 4: Implemented options for **XC_version** (LDA and GGA type), **vdW**, **ensemble**, **smearing**, **thermostat**, **barostat**, **basis_set**, **relax_method** and **mixing_scheme**. Listed options are not available for all codes. ALC_EQCM will informs the user if a selected option is not implemented for the requested code. See Table 5 to identify which options are implemented for the different codes.

XC_version (GGA-type)	**XC_version** (LDA-type)		
AM05 (Armiento-Mattsson [77])	CA (Ceperley-Alder [78])		
PW91 (Perdew-Wang [79])	PZ (Perdew-Zunger [80])		
PBE (Perdew-Burke-Ernzerhof [81])	Wigner (Wigner [82])		
RP (revised PBE with Padé Approximation [83])	VWN (Vosko-Wilk-Nusair [84, 85])		
revPBE [86]	PW92 [87]		
PBEsol [88]	HL (Heding-Lundqvist) [89]		
BLYP [90, 91]	PADE [92]		
XLYP [93]	**Ensemble**		
WC [94]	NVE (microcanonical)		
	NVT (canonical)		
	NPT (isothermal-isobaric)		
vdw	NPH (isoenthalpic-isobaric)		
DFT-D2 (Grimme [95])			
DFT-D3 (Grimme with zero damping [96])			
DFT-D3-BJ (Grimme with Becke-Jonson damping [97])	**Smearing**		
TS (Tkatchenko-Scheffler [98])	Gaussian		
TSH (Tkatchenko-Scheffler with Hirshfeld partitioning [99])	Fermi (Fermi-Dirac distribution)		
MBD (Many-body dispersion [100, 101])	Tetrahedron		
dDsC (Dispersion correction [102])	Window		
g06 (the same as DFT-D2)	MP (Methfessel-Paxton)		
OBS (Ortmann, Bechstedt and Schmidt [103])	Fix_occupancy		
JCHS (Jurečka, Černy, Hobza and Salahub [104])			
The following options generally require kernel files			
vdW-DF [105]			
optPBE [106]	**Basis_set** (only for CP2K)		
optB88 [106]	SZ (Single Zeta)		
optB86B [107]	DZ (Double Zeta)		
vdW-DF2 [108]	SZP (Single Zeta Polarizable)		
vdW-DF2-B86R [109]	DZP (Double Zeta Polarizable)		
SCAN+rVV10 [110]	TZP (Triple Zeta Polarizable)		
VV10 [111, 112]	TZ2P (Triple Zeta 2-Polarizable)		
AVV10S [113]			
	relax_method		
thermostat	CG (Conjugate Gradient)		
Andersen	QN (Quasi-Newton)		
Nose-Hoover	BFGS (Broyden-Fletcher-Goldfarb-Shanno)		
CSVR (Canonical Sampling through Velocity Rescaling)	LBFGS (Linear BFGS)		
Multi-Andersen (Multiple-Andersen)			
GLE (Generalised Langevin Equation)	**barostat** (CASTEP explicit options)		
Langevin	Andersen-Hoover		
AD_Langevin (Adaptative Langevin)	Parrinello-Rahman		
mixing_scheme Linear, Kerker [114], Broyden[115], Broyden-2nd [116], Multisecant [51], Pulay [117], Tchebycheff [118]. ONETEP specific [52]: damp_fixpoint, LNV,dnk_linear, ham_linear,dnk_pulay, ham_pulay,dnk_listi, ham_listi, dnk_listb, ham_listb			

Table 5: Availability for the implemented options of Table 4 for the codes VASP, CASTEP, CP2K and ONETEP. Specifications for **basis_set** and **barostat** are only available for CP2K and CASTEP, respectively, and they are not listed in this table.

Option	VASP	CASTEP	CP2K	ONETEP
XC_version (LDA-type)				
CA [78]	Yes	No	No	No
PZ [80]	Yes	Yes	Yes	Yes
Wigner [82]	Yes	No	Yes	No
VWN [84, 85]	Yes	No	Yes	No
PW92 [87]	No	No	Yes	Yes
HL [89]	Yes	No	Yes	No
PADE [92]	No	No	Yes	No
XC_version (GGA-type)				
AM05 [77]	Yes	No	Yes	No
PW91 [79]	Yes	Yes	No	Yes
PBE [81]	Yes	Yes	Yes	Yes
RP [83]	Yes	No	Yes	Yes
revPBE [86]	Yes	Yes	Yes	Yes
PBEsol [88]	Yes	Yes	Yes	Yes
BLYP [90, 91]	Yes	Yes	Yes	Yes
XLYP [93]	No	No	Yes	Yes
WC [94]	No	Yes	Yes	Yes
vdW (dispersion interactions)				
DFT-D2 [95]	Yes	No	Yes	Yes
DFT-D3 [96]	Yes	No	Yes	No
DFT-D3-BJ [97]	Yes	No	Yes	No
TS [98]	Yes	Yes	No	No
TSH [99]	Yes	No	No	No
MBD [100, 101]	Yes	Yes	No	No
dDsC [102]	Yes	No	No	No
g06 [95]	No	Yes	No	No
OABS [103]	No	Yes	No	No
JCHS [104]	No	Yes	No	No
vdW-DF [105]	Yes	No	Yes	Yes
optPBE [106]	Yes	No	Yes	Yes
optB88 [106]	Yes	No	Yes	Yes
optB86B [107]	Yes	No	Yes	No
vdW-DF2 [108]	Yes	No	Yes	Yes
vdW-DF2-B86R [109]	Yes	No	Yes	No
SCAN+rVV10 [110]	Yes	No	Yes	No
VV10 [111, 112]	No	No	No	Yes
AVV10S [113]	No	No	No	Yes
smearing				
Gaussian	Yes	Yes	No	No
Fermi	Yes	Yes	Yes	Yes
Tetrahedron	Yes	No	No	No
Window	No	No	Yes	No
MP	Yes	No	No	No
Fix_occupancy	No	Yes	No	No

relax_method				
CG	Yes	No	Yes	No
QN	Yes	No	No	No
BFGS	No	Yes	Yes	Yes
LBFGS	No	Yes	Yes	Yes
ensemble				
NVE	Yes	Yes	Yes	Yes
NVT	Yes	Yes	Yes	Yes
NPT	Yes	Yes	Yes	No
NPH	Yes	Yes	Yes	No
thermostat				
Andersen	Yes	No	No	Yes
Multi-Andersen	Yes	No	No	No
Nose-Hoover	Yes	Yes	Yes	Yes
CSVR	No	No	Yes	No
GLE	No	No	Yes	No
Langevin	Yes	Yes	No	Yes
AD_Langevin	No	No	Yes	No
mixig_scheme				
Linear	No	Yes	Yes	No
Kerker	Yes	Yes	Yes	No
Broyden	No	No	Yes	No
Broyden-2nd	Yes	Yes	Yes	No
Multisecant	No	No	Yes	No
Pulay	Yes	Yes	Yes	Yes
Tchebycheff	Yes	No	No	No

See Table 4 for options only specific to ONETEP.

Table 6: Description for directives of the **&solvation** sub-block, which shall be defined within **&block_simulation_settings**. For the sake of clarity, we have opted to specify this sub-block separately. The **&solvation** sub-block allows the specification of simulations including implicit solvent. To date, ALC_EQCM only allows the definition of implicit solvents for the ONETEP code.

```
&solvation
 cavity_model              String
 in_vacuum_first           Logical
 dielectric_function       String
 permittivity_bulk         Real
 smear_ion_width           Real Units
 # Fattebert-Gygi model
 density_threshold         Real
 beta                      Real
 # Andreussi model
 density_min_threshold     Real
 density_max_threshold     Real
 # Soft sphere model
 &soft_sphere_radii
     tags     Tag₁,₁  Tag₂,₁ ... Tag_Nc₁,₁   Tag₁,₂ ... ... Tag_NcNs,Ns
     values    R₁,₁    R₂,₁ ...   R_Nc₁,₁     R₁,₂ ... ...   R_NcNs,Ns
 &end_soft_sphere_radii

 soft_sphere_delta         Real
 soft_sphere_scale         Real
 # Apolar corrections
 apolar_terms              String
 sasa_definition           String
 apolar_scaling            Real
 solvent_surface_tension   Real Units
 solvent_pressure          Real Units
&end_solvation
```

cavity_model: type of cavity for the implicit solvent. String: *fixed* or *self_consistent*.
in_vacuum_first: if set to .True., instructions will set a DFT calculation of the system in vacuum prior to including the implicit solvent.
dielectric_function: function type to describe the dielectric function. String: *Fattebert-Gygi*, *Andreussi* or *soft_sphere*.
permittivity_bulk: relative permittivity of the bulk part of the implicit solvent.
smear_ion_width: the width of the smeared-ion Gaussians, only in units of Bohr. The recommended value is 0.8 Bohr. See Ref. [72].
density_threshold: parameter ρ_0 for the model in atomic units (see Eq. 40).
beta: parameter β for the Fattebert-Gygi model (see Eq. 40).
density_min_threshold: parameter ρ_{min} for the Andreussi model (see Eq. 41).
density_max_threshold: parameter ρ_{max} for the Andreussi model (see Eq. 41).
soft_sphere_radii: This block defines the radii of the involved species for the soft-sphere model [71]. Values must be given in units of Bohr. If this block is undefined, soft-sphere radii will be set to the vdW radii scaled by *soft_sphere_scale*.
soft_sphere_delta: Controls the steepness of the transition from vacuum to the bulk permittivity.
soft_sphere_scale: Scales the default vdW radii, used to defined the size of the soft spheres. The default is 1.33. This is incompatible with if the radii are defined with the **soft_sphere_radii** block.

apolar_terms: selects how the apolar terms will be considered. Implemented options for *String*: *None*, *only_cavitation*, *SASA* and *SAV*.

sasa_definition: defines the method used in the difference method which calculates the solvent accessible surface area (SASA) of the dielectric. Implemented options: *Density* and *Isosurface*.

apolar_scaling: controls the scaling of the apolar term for the solute-solvent dispersion-repulsion corrections.

solvent_surface_tension: surface tension for the computation of the apolar terms. *Units* must be `N m-1`.

solvent_pressure: pressure used to calculate the SAV contribution to the apolar term. *Units* must be `GPa`.

Table 7: Description for directives of the **&electrolyte** sub-block, which must be defined within **&block_simulation_settings**. For the sake of clarity, we have opted to specify this sub-block separately. The **&electrolyte** sub-block allows the specification of Boltzmann electrolytes within the implicit solvent. To date, ALC_EQCM only allows the definition of implicit solvents for the ONETEP code.

```
&electrolyte
 solver                    String
 neutral_scheme            String
 steric_potential          String
 steric_isodensity         Value
 steric_smearing           Value     Units
 boltzmann_temperature     Value     Units
 capping                   Value


 &solvent_radii
     tags      Tag₁    Tag₂ ... Tag_Nc₁,₁   Tag₁,₂ ... ... Tag_NcNs,Ns
     values    R^sol₁,₁  R^sol₂,₁ ... R^sol_Nc₁,₁   R^sol₁,₂ ... ... R^sol_NcNs,Ns
 &end_solvent_radii

  &boltzmann_ions
    tags          Tag₁   Tag₂ ... Tag_NB
    charge         q₁     q₂ ...    q_NB
    conc           C₁     C₂ ...    C_NB
    necs_shift     x₁     x₂ ...    x_NB
  &end_boltzmann_ions


&end_electrolyte
```

solver: treatment of the solution of Poisson-Boltzmann equation. String: *full* or *linearised*.

neutral_scheme: method for charge neutralization. String: *jellium*, *accessible_jellium*, *counterions_auto*, *counterions_auto_linear* or *counterions_fixed*.

steric_potential: type of potential to prevent the unphysical accumulation of Boltzmann ions close to the explicit quantum system. String: *Hard-core* or *smoothed-Hard-core*.

steric_isovalue: sets the density isovalue used to determine the solute radial cutoff. Value must be in atomic units. This is only relevant if ***steric_potential*** is NOT *None*. Default is 0.003

steric_smearing: smearing width for the smoothed hard-core cutoff. Value must be positive in Bohr. This is only relevant if ***steric_potential*** is *smoothed-hard-core*. Default is `0.4 Bohr`.

> ***boltzmann_temperature***: sets the temperature of the Boltzmann ions in Units of K.
> ***capping***: sets the exponential cap to help numerical stability when ***solver*** is *full*.
> **&solvent_radii**: defines the solvent radial cutoff for each atomic species. Values must be given in units of Bohr, and this is the responsibility of the user.
> **&boltzmann_ions**: defines the Boltzmann ions of the electrolyte. Species tag, charge (in atomic units) and concentration (conc, in mol/L) must be provided. If the *counterions_fixed* option is selected for the neutral_scheme directive, the NECS_shift parameters are also needed.

4.2.2   The DATA_EQCM file

This file must contain the EQCM data, and it is needed for all options of directive ***Analysis*** except *Model_pristine_sample* and *model_disordered_system*. Data must be provided in columns, and each column labelled in the header of the file. ALC_EQCM will identify the number of columns using the first line containing numerical data, and will assume that the number of EQCM variables is equal to the number of columns found. The first non-empty line of DATA_EQCM must contain the labelling for the columns. Labelling can be potential or voltage, time, current, charge, frequency (which refers to mass-frequency),mass (needed for those cases where the Sauerbrey equation does not hold) and resistance. Any other variable will not be recognized and the code will abort. If both frequency and mass are recorded, ALC_EQCM will use the values of mass assuming that the user has previously evaluated the applicability of the Sauerbrey correlation.

Data for the CV cycles must be set in a concatenated manner, i.e. data set for cycle 1 must be followed by data set for cycle 2, cycle 3, etc, until all the CV cycles are added. It is recommended to separate each set of data using empty lines to help with the visualization of the data. Data can also be set without any separation: the code is flexible to identify when a CV cycle has ended and a new cycle starts (see definition of a CV cycle sec. 3.6). If no potential is provided the execution will be aborted. For more information of the allowed format and syntax for the header, we refer to the tutorial exercise of sec. 6.5.

4.2.3   The INPUT_STRUCTURE file

This file is needed if option *Model_cycled_sample*, *model_disordered_system* or *model_pristine_sample* is set for the ***Analysis*** directive. This file must contain the input structure used to build atomistic models, and must be located within folder **INPUT_GEOM**. Atomic coordinates in this file must be consistent with the specification of **&block_input_composition**. We refer the reader to sec. 6.12.1 for an example of how to set this file correctly. The format for the INPUT_STRUCTURE file must be specified via directive ***input_model_format***. Allowed options are: *vasp* (POSCAR), *onetep* and *castep* (.geom), *xyz* and *cif*.

4.2.4   Input geometry for molecular species

In case there are molecular species that participate in the reaction, the code will request to include files with the geometry definition for such molecular species. The name of these files must be species-tag.xyz, where species-tag is the tag defined in **&block_species**. These files must be located inside folder **INPUT_GEOM**.

Lets consider the example of intercalated water, which is tagged as H2O. Atomic species are tagged as Hw and Ow in **&block_species_components**. The geometry structure of this molecular species must be defined in file H2O.xyz as follows

```
 3
 geometry
 O   0.000    0.000   0.0   Ow
 H   0.707    0.707   0.0   Hw
 H  -0.707    0.707   0.0   Hw
```

The structure corresponds to the XYZ format, but it requires an additional column (last column) for the specification of the atomic tags. Such tags must be consistent with those defined in **&block_species_components**. The second line can be anything (usually a descriptive comment) but must be present. The order of atomic species is irrelevant.

4.2.5 Pseudopotential files

Pseudopotentials are needed to compute the electronic structure with DFT. Pseudopotential (PP) files for each atomic species must be copied to folder **DFT/PPs**. The name of the PP file for each species must coincide with the specification within sub-block **&pseudo_potentials**. If this sub-block is not defined, PPs are not set.

CASTEP: Valid PP formats are `.recpot` and `.usp` (ultra-soft). If the **&pseudo_potentials** sub-block is not defined, ALC_EQCM will set instructions to execute the on-the-fly generation of PPs.

ONETEP: To date, only the `.recpot` format is allowed.

VASP: PPs must comply with the `POTCAR` format.

CP2K: PPs format must be consistent with those available in the CP2K repository [119].

IMPORTANT: no pseudopotential file is provided with ALC_EQCM. The user is therefore expected to retrieve suitable pseudopotential files as per code-specific licensing conditions.

4.2.6 The BASIS_SET file

Definition for the basis set is only required to build files for simulation with the CP2K code. The basis set for each atomic tag must be defined in sub-block **&basis_set**. All basis sets must be defined in the BASIS_SET file, which must be located within folder **DFT**. It might occur that a particular requested basis sets for a given element has not been included in the BASIS_SET file or it does not exist within the CP2K repository [119]. In such a case, ALC_EQCM will stop and inform the user. We recommend the following steps (in the given order):

i) changing the atomic basis set for the element,

ii) changing "XC_level" and "XC_version",

iii) contacting the CP2K forum for assistance at https://groups.google.com/g/cp2k

iv) generating a new basis set for the element (only for expert users).

4.2.7 Kernel files to set vdW corrections

To set vdW corrections as part of input files for DFT calculations, the user might need to include a kernel file, which must be located inside the **DFT** folder. The name of this file depends on the requested vdW correction and the DFT code. It is important to mention that not all the implemented vdW corrections need of kernel files. ALC_EQCM will inform the user if a kernel is needed. See Table 4 for details.

4.3 Output files

Several files are generated depending on the selected option for directive **Analysis**. See Table 2 for details. To support the development of this code, Gnuplot [120], XmGrace [121] and ASE [76] softwares were used to plot and verify the correctness of the generated data. Nevertheless, the format of the generated data is suitable to be imported using other software, in principle.

4.3.1 The OUT_EQCM file

This file contains a summary of the requested analysis and will be generated in the same folder where ALC_EQCM is executed. The file starts with an introductory banner and a brief description of the requested analysis. For all the possible options of directive **Analysis** except options *model_pristine_sample*, *model_disordered_system* and *hpc_simulation_files*, the relevant EQCM settings are reported according to the units convention adopted by ALC_EQCM (see sec. 3.15). This is followed by a section that reports details of the EQCM data:

• Number of columns detected in the DATA_EQCM file

• A table that indicates what variable is listed in each column

• Number of total CV cycles identified in DATA_EQCM

• A summary table with information of number of points and voltage range for the positive and negative voltage sweep for each CV cycle. If no data has been recorded for either of positive of negative voltage sweep, the fragment of the CV cycle is classified as undefined.

After this stage, reported results depend on the requested option for the **analysis** directive. We recommend the user to investigate the structure of the OUT_EQCM files generated for the different options of directive **analysis** following the tutorial examples of sec. 6.

4.3.2 The RAW and FILTERED files

If option *Print_EQCM_raw* or *Print_EQCM_filter* is set for directive **Analysis**, the code will print the raw or filtered values for current and mass-frequency, respectively, as a function of the applied potential. The generated RAW and FILTERED files are located in folder **ANALYSIS_EQCM**. If option *Print_EQCM_filter* is selected, directive **Filter_cutoff** must be specified, otherwise the execution will be aborted. If no current nor mass-frequency are found in the DATA_EQCM file, the execution will stop and a message will be printed to inform the user. If no reported data is provided for a given segment of a particular cycle, either for positive or negative voltage sweep (or both), there will not be reported data for that particular segment. For option *Print_EQCM_filter* it might well happen (very rarely) that the number of endpoints used to defined the padding is larger than the actual number of points within a cycle segment. For such cases, no filtered value will be printed. Instead, the user will be informed and a warning will be generated in the OUT_EQCM file. We refer the reader to tutorial exercises of sec. 6.1 and 6.3 to check the structure of RAW and FILTERED files.

4.3.3 The MASS_CALIBRATION file

If option *mass_calibration* is set for the **Analysis** directive, the code will generate the file MASS_CALIBRATION inside folder **ANALYSIS_EQCM** with the values of Δf (in Hz) and accumulated charge ΔQ (in mC). Both current and mass-frequency have to be recorded in the DATA_EQCM file, otherwise the code aborts the execution. We refer the reader to the tutorial example of sec. 6.8.

4.3.4 The MASSOGRAM file

If option *massogram* is set for directive **Analysis**, ALC_EQCM generates the file MASSOGRAM with the computed values of mass-density rate (in ng/cm2/s) and applied voltage. If the EQCM is reported, the MASSOGRAM contains mass rates (in ng/s). This file is located in folder **ANALYSIS_EQCM**. If directive **Filter_cutoff** is not specified, massogram values will be computed using the raw EQCM data. If either mass-frequency or current are not recorded in DATA_EQCM, the execution is aborted. We refer the user to the exercise of sec. 6.9 to inspect the structure of the MASSOGRAM file.

4.3.5 The SPEC files

ALC_EQCM performs spectra analysis if option *spectra* is set for directive **Analysis**. The code will print the results for the magnitude of the current and mass in reciprocal space to files SPEC_CURRENT and SPEC_MASS, respectively, inside folder **ANALYSIS_EQCM**. If the elapsed time is recorded, the code will automatically use the frequency domain, otherwise, the reciprocal space will be represented in units of 1/V. If option *cycles* is not specified, the spectra analysis will be conducted for all the CV cycles that have been identified from the DATA_EQCM file. We refer the reader to the exercise of sec. 6.2 to inspect the structure of SPEC_CURRENT and SPEC_MASS files.

4.3.6 The CHARACTERIZATION file

This file is printed inside folder **ANALYSIS_EQCM** only if the directive **Analysis** is set to *characterization*, *stoichiometry* or *model_cycled_sample*. From the current profiles, the code determines the number of redox cycles (in contrast to the CV cycles, see sec. 3.6) and the total charge will be computed from the elapsed time (if provided) or from the voltage values and the **scan_rate**. If no elapsed time is recorded, the user is responsible to make sure that each cycle has been measured at constant scan rate, otherwise the integrated charge will be incorrect. If mass-frequency is not measured, option *characterization* only computes the charge-related variables. Table 8 describes the meaning for each of column of the CHARACTERIZATION file (see sec. 3.6). We refer the user to exercise of sec. 6.4 for an example of EQCM characterization with a detailed analysis of the calculated quantities.

Table 8: Description of the variables printed to file CHARACTERIZATION. In parentheses, the corresponding column for the variable when mass-frequency/mass is not recorded.

Label	Units	Col.	Description
#Cycle	N/A	1	Redox cycle
Dmass_oxidation	ng	2	ΔM for oxidation
Dmass_reduction	ng	3	ΔM for reduction
Dmass_residual	ng	4	Residual mass at the end of the redox nth cycle
Dmass_half_redox	ng	5	Total accumulated mass at half of the redox cycle. If the sample is first oxidised (reduced), this quantity is the total accumulated mass after the *nth* oxidation (reduction).
Q_oxidation	mC	6(2)	Total charge accumulated during oxidation
Q_reduction	mC	7(3)	Total charge accumulated during reduction
$\|Q\_ox + Q\_red\|$	mC	8(4)	Charge difference between oxidation and reduction.
$\|Q\_ox/Q\_red\|$	N/A	9(5)	Magnitude of the ratio between oxidation and reduction charge (if the sample is oxidized first).
$\|Q\_red/Q\_ox\|$	N/A	9(5)	Magnitude of the ratio between reduction and oxidation (if the sample is reduced first).
$\|Dmass/Q\|\_red$	ng/mC	10	Magnitude of the ratio between ΔM and the charge for reduction.
$\|Dmass/Q\|\_ox$	ng/mC	11	Magnitude of the ratio between ΔM and the charge for oxidation.

4.3.7 The INTERCALATION_OX and the INTERCALATION_RED files

ALC_EQCM offers the option to compute the stoichiometric solutions compatible with the EQCM data if directive *Analysis* is set to *stoichiometry* (or *model_cycled_sample*) and *process* is set to *intercalation*. We refer the user to section 3.10.1 for a detailed discussion of the implemented algorithm. Depending on which process occurs first (oxidation or reduction), the number of variable species and the number of constraints, INTERCALATION_OX and/or INTERCALATION_RED will be generated containing single or multiple stoichiometric solutions. These files are printed to folder **ANALYSIS_EQCM**. We refer the user to the explanatory examples of sec. 6.10 to familiarise with the information reported in the files.

4.3.8 The ELECTRO_DEPOSITION file

For electrodeposition experiments involving one chemical species, ALC_EQCM computes the A_{eff}/A_{disk} ratio and total amount of moles from the EQCM data (see sec. 3.8). The user must set directive *Analysis* to *stoichiometry* (or *model_cycled_sample*) and select the option *electrodeposition* for directive *process*. File ELECTRO_DEPOSITION will be generated within folder **ANALYSIS_EQCM**. In contrast to INTERCALATION files, information for oxidation and reduction is reported within the same file. We refer the user to the worked example of sec. 6.11.1.

4.3.9 The CV_MOLES file

This file is generated when selecting the *fluxes* option for the *analysis* directive. Mole changes for each participating species are computed as a function of the applied potential according to Eq. (36). Units for delta moles are in nmol (nanomoles). Please refer to sec. 6.15 for an example exercise to generate examples of this file.

4.3.10 The MOLE_FLUX file

This file is generated when selecting the *fluxes* option for the *analysis* directive. Mole fluxes for each participating species are computed as a function of the applied potential according to Eq. (37). Units for mole fluxes are in nmol/s (nanomole/sec). Please refer to sec. 6.15 for an example exercise to generate examples of this file.

4.3.11 The IONIC_CURRENT file

This file is generated when selecting the *fluxes* option for the *analysis* direcitve. Ionic currents for the variable species with **non-zero oxidation number** are computed as a function of the applied potential according to Eq. (38). Units for ionic currents are in mA. Please refer to sec. 6.15 for an example exercise to generate examples of this file.

4.3.12 The SELECTED_SOLUTIONS file

For intercalation process with N_s variable species subject to N_{cstr} constraints, where $N_{cstr} < (N_s\text{-}2)$, a set of multiple stoichiometric solutions compatible with the charge and mass balance equations is obtained and printed to any of the INTERCALATION files (see sec. 4.3.7). The number of solutions is generally large ($>$ 100) and depends on the discretization parameter defined with directive *delta_stoich* (0.01 by default). Building atomistic models for such a large set is obviously not convenient. The user can reduce the number of stoichiometric solutions to build atomistic model using directive *targeted_num_models*. This option selects a set of solutions that conveniently sample the domain of computed stoichiometries. The number of selected solutions is generally different but close to the number specified by *targeted_num_models*. These solutions are printed to file SELECTED_SOLUTIONS inside folder **ANALYSIS_EQCM**.

4.3.13   Atomistic structure files

These files contain the generated atomistic models. We refer the user to sec. 3.10 and 3.11 for a more detailed explanation of the implemented algorithm.

Generated structures will be printed to file SAMPLE.format. If option *Model_cycled_sample* is set for directive **Analysis**, files are printed in the corresponding subfolders within **ATOMISTIC_MODELS**. If option *Model_pristine_sample* is selected, SAMPLE.format is printed inside folder **ATOMISTIC_MODELS/pristine**. Similarly, SAMPLE.format is printed inside folder **ATOMISTIC_MODELS/disordered** if the user selects option

Model_disordered_system for the **Analysis** directive.

The format for the generated models is set with the **output_model_format** directive. Options are: *vasp*, *XYZ*, *onetep*, *castep*, *siesta*, *cif* and *CP2K*.

4.3.14   The MODEL_SUMMARY file

Together with the SAMPLE.format file, a summary of the generated model is recorded to file MODEL_SUMMARY. This file is a copy of the table printed in the OUT_EQCM file for that particular model. The table contains the amount of units for each species in the model and how the modelled stoichiometry compares with the target stoichiometry.

4.3.15   The OUT_EQCM_BACKUP file

If **Analysis** is set to *model_cycled_sample*, *Model_disordered_system* or *model_cycled_sample*, file OUT_EQCM is copied OUT_EQCM_BACKUP inside folder **RESTART**. This file is generated to keep record of the information for the generation of atomistic models.

4.3.16   The RECORD_MODELS file

This file is generated if **Analysis** is set either to *model_cycled_sample*, *Model_disordered_system* or *model_pristine_sample*. This file contains basic specification for each of the generated atomistic models, and allows the user to obtain input files for atomistic simulations and/or files for HPC submission without the need to recompute the atomistic models (see sec. 3.14.3 for details). This file is stored within folder **RESTART**.

4.3.17   Generated input files for simulations

If the **Analysis** directive is set either to *model_cycled_sample*, *Model_disordered_system* or

model_pristine_sample, various files are generated depending on the format specified by the directive **output_model_format**. These files contain all the requested information set in

&block_simulation_settings and allow the user to perform atomistic-level simulation of the generated models. We consider each format separately:

• VASP: the generated atomistic model in file SAMPLE.vasp is copied to file POSCAR. The user will find the POTCAR file with the pseudopotentials consistent with the atomic species of the POSCAR file. This POTCAR file is generated via the concatenation of the pseudo potential files for each atomic species, previously copied to folder PP by the user. File KPOINTS contains the specification for sampling of the reciprocal space. The INCAR file contains all the directives for the DFT computation of the electronic problem as well as for the dynamics of the ions. IMPORTANT: the user must hold the VASP license to copy POTCAR files.

- CP2K: the generated input.cp2k file contains all the required directives for simulation. File SAMPLE.cp2k with the atomistic structure is read automatically by CP2K from the specification of directive COORD_FILE_NAME. BASIS_SET file and the user defined file with PPs are also copied to each sub-folder.

- CASTEP: the generated model.param and model.cell files contain all the required directives for simulation. If **&pseudo_potentials** is defined, each of the pseudo potential files in **DFT/PPs** will be copied to the subfolders. File CI-castep.cell is only for CI purposes: it contains the same information as model.cell without the cell vectors and the atomic coordinates.

- ONETEP: the generated model.dat file contains all the required directives for simulation. Each of the pseudo potential files defined in sub-block **&pseudo_potentials** is also be copied to the subfolders. File CI-onetep.dat is only for CI purposes: it contains the same information as model.dat without the cell vectors and the atomic coordinates.

4.3.18 The HPC script files

These files contain directives for allocation and execution of HPC jobs, and are generated only if **&block_HPC_settings** is defined in SET_EQCM (see sec. 3.13). HPC files are named as hpc_script-format.sh in each of the subfolders where the atomistic models are generated. Specification for format depends on the option set for *output_model_format*.

5 Instructions for use and development of ALC_EQCM

In this section we detail the steps to download, build, compile, test and execute ALC_EQCM. We also describe the preparation of input files and suggest a protocol for efficient execution. For developers, we provide instructions to contribute and maintain code within the framework of STFC GitLab [122].

5.1 Setting up the code

File ./README.md in the root directory provides a brief introduction of the code, and describes the structure and content of files. Instructions for building and compiling ALC_EQCM are described in detail in file ./cmake_building.md. To this purpose, the user must have access to the CMake platform [123] and Fortran compilers, either GNU (gfortran) or Intel (ifort/ifx).

We assume the user `username` in the machine `wherever` has compiled the code using the GNU-Fortran compiler and the Release option in folder **/home/username/codes/alc_eqcm/build_gnu**. The user has also created folder **example**, from where the code will be executed. In such folder, the executable must be called from the command line as follows:

`username@wherever:example$` /home/username/codes/alc_eqcm/build_gnu/bin/alc_eqcm.x

Alternative, the path to the executable can be set in the file .bashrc.

5.2 Preparation of input files

Independently of the type of calculation requested, the user will always need the SET_EQCM file (sec. 4.2.1). We refer the user to Table 3 with a detailed specification of each directive. In SET_EQCM, all characters after symbol "#" are interpreted as comments. We recommend to add a header at the top of the file with a brief description of the calculation.

Execution of ALC_EQCM requires of different input files depending on the option for directive ***Analysis***. Information of the required files is detailed in Table 2. ALC_EQCM is sufficiently robust to guide the user, via error messages, to complete file SET_EQCM if directives are missing for the requested analysis and inform if any input file is missing. We refer the user to the tutorial exercises of sec. 6 for examples of how to set the SET_EQCM file from scratch.

For all options of ***Analysis*** except *model_pristine_sample* and *Model_disordered_system*, file DATA_EQCM is compulsory (see sec. 4.2.2). The first line of this file must be a header with the specification of the recorded variables. If the header is set in any other line, the execution will be aborted. Experimental values should be given in columns. ALC_EQCM will first find the first line with numerical values, determine the number of columns (variables) and print the following message in the OUT_EQCM:

`The code has detected "X" EQCM variables (columns)`

Once the number of variables is determined, the algorithm returns to the first line and reads the name of the variables and units for each column. The structure for the header must comply with the following requirements:

• allowed labels for the variables are: voltage (or potential), current, charge, frequency, mass, resistance and time.

• the string used to label each variable can be different from the above options, as long as the allowed label is part of the string. Moreover, the use of capital or lower case strings is irrelevant. For example, label `123-Voltage-up` will be interpreted as voltage.

• string to label variables must not contain any forward slash, parentheses or brackets. For example `External(1).voltage` is NOT a valid option.

• labels for the variables must be followed by the specification of the units. Units can be separated from the variable labels by a space, forward slash, parentheses or brackets. Example: `voltage (V)`, `voltage V`, `voltage/V` and `voltage [V]` are valid options.

• data for different cycles must be set one after the other, and it is recommended to avoid any extra headers in between these sets of data. For the sake of tracking, the user it is also recommended separate data of different CV cycles using blank lines.

It may happen that EQCM files generated from the acquisition software contain non-ASCII characters. Similarly, new line characters might be introduced when copying a set of data from a Windows spreadsheet to a text file, for example. We advise the user to check for non-ASCII characters beforehand and remove them. In Linux one can use the following execution command to clean any raw-file.dat and write it to a new clean-file.dat as follows:

```
tr -cd '\11\12\40-\176' < raw-file.dat > clean-file.dat
```

No commas are accepted to represent real values. If present, the user must replace them by the standard dot. We refer the user to sec. 6.5 for a tutorial example of how to set a correct DATA_EQCM file. Similarly, in sec. 6.7 we show an example where the DATA_EQCM contains non-ASCII characters and how one should proceed to remove them.

Files INPUT_STRUCTURE (sec. 4.2.3) and species-tag.xyz (in case the species is a molecule, sec. 4.2.4) are needed for building atomistic models. These files must be located in folder **INPUT_GEOM**. File INPUT_STRUCTURE must contain the atomic coordinates of the reference input model (supporting slab or host bulk) and comply with the format specified in directive ***input_model_format***, otherwise the execution will be terminated. In case the INPUT_STRUCTURE file is in xyz format, information of the simulation cell vectors must be provided via **&block_input_cell**. File species-tag.xyz is only required for those molecular species that participate in the reaction.

Finally, if the user requests to build input files for DFT simulations, folder **DFT** must be created. This folder must contain:

• the BASIS_SET file (only for DFT codes that user localized basis sets, see sec. 4.2.6)

• the kernel file (only for those vdW corrections that need a kernel). See sec. 4.2.7 and Table 4.

• subfolder **PPs** with pseudopotential files (see sec. 4.2.5).

5.3   Suggested protocol for execution and analysis

Based on the experience we have accumulated during the development of the code, we recommend the following steps for a rigorous execution and analysis:

• set the DATA_EQCM file carefully according to the guidelines of sec. 5.2.

• always print the raw data via option *print_EQCM_raw* and check both current and mass-density/mass profiles. In addition, look carefully at the information for the CV cycles printed in table of the OUT_EQCM file.

• in case the data contains strong noise, perform a spectra analysis.

• to filter data, select the appropriate cut off and endpoints to build the padding region. Always compare the raw and filtered data in the same plot.

• for characterization analysis, always check the table with the redox information printed in OUT_EQCM and the content of the CHARACTERIZATION file.

• prior stoichiometry analysis, it is advisable to perform a characterization analysis beforehand to check the correctness of the computed quantities. For stoichiometric analysis, check that the information provided in **&block_species** is correct.

• to build atomistic models with option *Model_cycled_sample*, the user should run a stoichiometric analysis beforehand to check the correctness of the stoichiometry, related blocks and the EQCM data.

• none of the previous steps is needed for option *Model_pristine_sample*, *Model_disordered_system* or *hpc_simulation_files*.

• independently of the selected option for the generation of models, it is crucial to check the input atomic structure for the host structure (INPUT_STRUCTURE file) and the participating molecular species (files species-tag.xyz).

Finally, and most importantly, the user should not panic nor give up if the code gives an error. Instead, read the error message at the end of the printed OUT_EQCM and try to understand the source of the problem. In case the user gets a segmentation fault of finds an unexpected malfunctioning, we urge to contact the developers by email, ALWAYS attaching the input files for a quicker and efficient assessment. We will be pleased to assist you.

5.4 Instructions for developers

5.4.1 Development of ALC_EQCM

The instructions that developers must follow to contribute to ALC_EQCM are documented in file ./CI_instructions.md. The strategy is based on Continuous Integration (CI), which consists in the set of practices to merging developers' working copies to the main version of the code.

To this purpose, developers must use the GitLab [122] platform and have access to the ALC_EQCM project, for which they must be granted permission from the responsible individuals. The responsible developers of ALC_EQCM are Ivan Scivetti and Gilberto Teobaldi from the Theoretical and Computational Physics Group of the STFC.

5.4.2 Adopted protocol

The protocol adopted to develop ALC_EQCM is detailed in file ./coding_protocol.md. This protocol is based on the refactoring work for the DL_POLY code. Developers are urged to follow Fortran2008 standards, for which we recommend to consult specialised bibliography [124].

Complying with these rules is crucial to the purpose of consistency and maintenance. Any written code that does not comply with the protocol **must NOT be implemented** within the main ALC_EQCM version without further revision, which often leads to a tedious and laborious iterative process that developers and the reviewers are always pleased to avoid.

6 Tutorial examples

In this section we describe the different functionalities of ALC_EQCM with application examples. Files can be found in folder **tutorial**. We advice the user to investigate the generated output files to get familiarized with the reported information. Based on the examples and discussion, we also propose additional exercises for user's consideration. Words after the symbol # are interpreted by ALC_EQCM as comments, which we shall use here for description purposes.

6.1 Printing raw EQCM data

We first show how to print raw EQCM data from the information provided in file DATA_EQCM. By "raw" we refer to data as collected from the EQCM equipment. To this purpose, we consider the electrochemical cycling of a $Ni(OH)_2$ sample in a 1M solution of KOH. This electrochemical process leads to the (de)intercalation of K^+ cations [1]. Input files for this exercise are provided in **tutorial/exercise-01**. From the header of DATA_EQCM file, we observe that voltage, cur rent and mass-frequency have been recorded. To generate files with the raw current and mass-density as a function of voltage only for the first CV cycle, we define the following directives in the SET_EQCM file:

```
# Intercalation/de-intercalation of K ions into/from a (NiOH)2 host
# Input directives to print raw EQCM data
Software     CH-Inst              # Acquisition software (optional)
Analysis     print_EQCM_raw       # Print raw EQCM data
Sauerbrey    -0.1464 Hz ng-1 cm2  # Sauerbrey factor
Quartz_freq  8.0 MHz              # Quartz frequency
Cycles       1    1               # Range of cycles to be considered
```

Option *print_EQCM_raw* instructs ALC_EQCM to print the measured EQCM current (in mA) and mass-density (in ng/cm^2) as a function of the applied potential to files RAW_CURRENT and RAW_MASS, respectively. These files are stored within folder **ANALYSIS_EQCM**. Since changes of the electrode mass are recorded as changes in frequency, ALC_EQCM needs specification of the conversion factor from frequency to mass-density (directive **Sauerbrey**) as well as the natural frequency of the quartz (directive **quartz_frequency**), equal to 8.0 MHz in this case. The user can also specify the acquisition software used to collect the data with the **software** directive.

Results for the first CV cycle for positive (dV>0) and negative (dV<0) voltage sweep are shown in Fig. 6.1.1, where the units of mass-density are reported in $\mu g/cm^2$. Voltages are reported with respect to a Ag/AgCl reference electrode. In this experiment, the initial voltage is set to zero and increased to 0.45 V (dV>0, black). A peak is observed at approximately 0.35 V, from which the mass-density starts to increase until the voltage is reversed at 0.45 V. This process corresponds to K^+ intercalation into the host electrode (oxidation). When the voltage is reversed, we find a different peak at 0.26 V, which is correlated to a drop in the mass-density. This corresponds to the de-intercalation of K^+ (reduction). Interestingly, at the end of the cycle the value for the mass-density is larger than zero. This indicates an accumulation of mass in the electrode and demonstrates the irreversibility of the reaction.

Exercise 1: Run ALC_EQCM again to obtain all the cycles. What can we learn from the current and mass-density profiles as the number of cycles increase?

Exercise 2: Modify SET_EQCM to print results in the voltage window between 0.3 and 0.45 V.

6.2 Spectra analysis

To show the capability of ALC_EQCM to extract spectra information from EQCM data, we consider mass-frequency changes for the layer electrodeposition of Ag on Au during the first CV cycle [3]. Following the
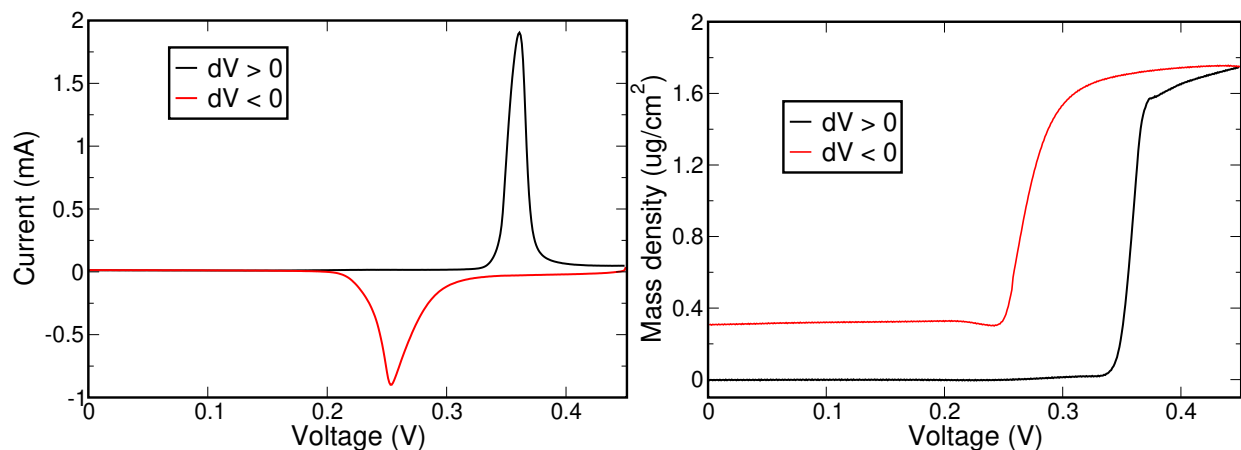
Figure 6.1.1: EQCM current (left) and mass-density (right) as a function of the applied potential during the first CV cycle of Ni(OH)$_2$ in a 1M solution of KOH. Voltages are reported with respect to a Ag/AgCl reference electrode.
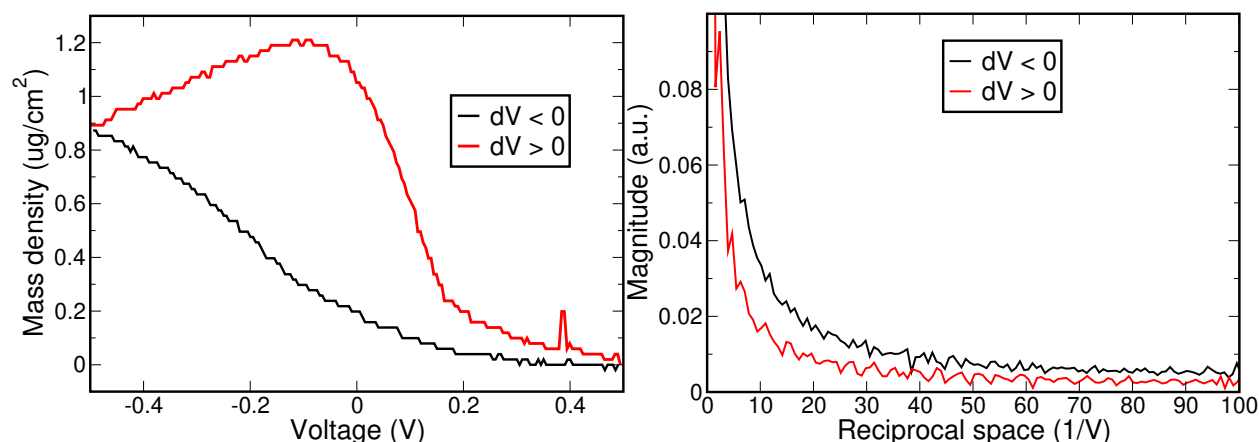

Figure 6.2.1: Left: EQCM mass-density as a function of the applied potential during the first calibration cycle of a EQCM device for negative (black, dV<0) and positive (red, dV>0) voltage scan. Right: spectra analysis for positive and negative voltage scan. Voltages are reported vs. Ag/10 mM AgNO$_3$ + 0.1 M TBAPF6 in acetonitrile reference electrode.

proposed protocol of sec. 5.3, we first plot the raw data. We base on the example of the previous section. Input files for this example can be found in folder **tutorial/exercise-02/print**. We set the following directives in the SET_EQCM file based on the reported specification of the EQCM device [3]:

```
# Input directives for printing raw data
# Electrodeposition of Ag on Au
Software     QCM200                # Acquisition software (optional)
Analysis     print_EQCM_raw        # Print raw data
Sauerbrey    -0.0769231 Hz ng-1 cm2  # Sauerbrey factor
Quartz_freq  5.0 MHz               # Quartz frequency
```

Results for mass-density as a function of the applied voltage is shown in Fig. 6.2.1 (Left) for negative

(black, dV<0) and positive (red, dV>0) voltage scan. Voltages are reported vs. Ag/10 mM AgNO$_3$ + 0.1 M TBAPF6 in acetonitrile reference electrode. Strong noise in the recorded EQCM data is generally not expected if the experimental setup has been properly adjusted. Nevertheless, in this example we observe that the resolution settings for the measured values is not sufficiently fine and, consequently, mass-profiles exhibit a step-like pattern. To further investigate on the nature of the recorded signals, ALC_EQCM offers the possibility to compute the spectrum in reciprocal space. Input files for this exercise are provided in folder **tutorial/exercise-02/spectra**. The user must set the following directives in the SET_EQCM file, where we highlight in blue the differences with respect to the previous directives:

```
# Input directives for computation of signal spectra
# Electrodeposition of Ag on Au
Software        QCM200                  # Acquisition software (optional)
Analysis        spectra                 # Transform data to reciprocal space
Sauerbrey       -0.0769231 Hz ng-1 cm2  # Sauerbrey factor
Quartz_freq     5.0 MHz                 # Quartz frequency
endpoints_mass  2                       # points for FFT padding
```

Option *spectra* for directive **Analysis** instructs the code to perform the spectra analysis for current and mass-density. The last directive corresponds to the number of points of the mass-frequency used to calculate the padding, needed for the calculation of the spectra via FFT (see sec. 3.1). In this case we consider the 2 values above/below the lowest/highest voltage. The user is responsible to define the appropriate value for this directive, which depends on the EQCM profiles and the physical process under consideration.

These settings generate the file SPEC_MASS inside folder **ANALYSIS_EQCM** with the computed FFT magnitude for each component of the reciprocal space. If elapsed time is provided in the DATA_EQCM file, the reciprocal space will correspond to the frequency domain in units of Hz. If the elapsed time is not recorded, as in the present case, the reciprocal space will correspond to the inverse of the voltage, in units of 1/V. Results are shown in Fig. 6.2.1 (Right) both for positive and negative voltage sweeps. For the sake of visualization, y-values (magnitudes of the FFT) have been normalised with respect to the computed magnitude for the zero component of the reciprocal space 1/V. Both spectra exhibit the expected trend with lower magnitudes for larger components of the reciprocal space. Although these magnitudes are relatively small, they are responsible for step-like pattern of the recorded data. The purpose of applying a filter is to eliminate the reciprocal space components above a given cutoff and smooth the data. We refer the reader to the following section.

Whenever the EQCM experiment is conducted at constant voltage scan rate, as in the present case (0.1 V/s), one could multiply the x-values of Fig. 6.2.1 (Right) and represent the spectra in the frequency domain (Hz).

Exercise: Compute the spectra using the EQCM data of section 6.1 only for the first CV cycle. How do results compare with those reported in Fig. 6.2.1?

6.3   Removing noise from signal

Depending on the magnitude of the noise, it might be beneficial to apply a filter not only for a better visualization but also for improved data analysis. ALC_EQCM offers the possibility to apply a low pass Gaussian-type filter (see sec. 3.2) using a cutoff value (in units Hz or 1/V) that must be specified by the user.

Here we apply a filter to the EQCM data of the previous section 6.2. Input files for this example can be found in folder **tutorial/exercise-03**. We set the following directives in the SET_EQCM file:
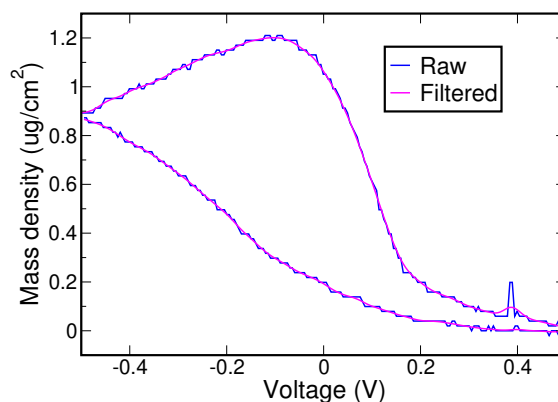
Figure 6.3.1: Raw (black) and filtered (red) mass-density for the first EQCM cycle of electrodeposition of Ag on Au. Cutoff frequency is 2.0 Hz. Raw data from Ref [3].

```
# Input directives for noise filtering and printing
# Electrodeposition of Ag on Au
Software        QCM200                  # Acquisition software (optional)
Analysis        print_EQCM_filter       # Filter and print data
Sauerbrey       0.0769231 Hz ng-1 cm2   # Sauerbrey factor
Quartz_freq     5.0 MHz                 # Quartz frequency
Filter_cutoff   2.0 Hz                  # Specification for the cutoff frequency
scan_rate       0.1 V s-1               # voltage scan rate
```

Option *print_EQCM_filter* instructs ALC_EQCM to filter the raw data and print the values in file

FILTERED_MASS (since only mass-frequency is recorded in the DATA_EQCM file). Directive **Filter_cutoff** sets a filter of 2.0 Hz. Directive **scan_rate** is needed to define the filter, as the elapsed time is not recorded during the experiment. Filtered results are shown in Fig. 6.3.1 (magenta) and compared with the raw data of Fig. 6.2.1 (blue), where we have used the same color for positive and negative scan. We observe the signal has been smoothened.

Exercise: Run the simulation with a different cutoff frequency. How would you define a reasonable value?

6.4   Quantitative EQCM characterization

The previous sections provide examples for visualization, spectra analysis and noise filtering of raw EQCM data. In this section, we focus on the quantitative characterization by computing charge and mass variations associated to electrochemical process. We consider experimental data for the intercalation of Li$^+$ into Ni(OH)$_2$ [1]. Input EQCM files are provided in folder **tutorial/exercise-04**. As per guidance of section 5.3, it is first recommended to print the raw data to check the EQCM profiles. The user must follow the example of sec. 6.1 and set the file SET_EQCM to print raw values.
To perform quantitative characterization from the EQCM data, the user must set the following directives in the SET_EQCM file:

```
# Input directives for quantitative characterization
Software      CH-Inst                   # Acquisition software (optional)
Analysis      Characterization          # Integrated charge and total mass
Sauerbrey     -0.146455224 Hz ng-1 cm2  # Sauerbrey factor
Quartz_freq   8.0 MHz                   # Quartz frequency
Electrode_area  0.030418811 inch2       # Electrode area
scan_rate     5.0 mV s-1                # Scan rate for voltage sweep
current_offset  .True.                  # Correction of zero-voltage current offset
```

Option *characterization* instructs the code to compute the total charge and the total mass for each oxidation/reduction. Here, no cycle range is specified and, consequently, ALC_EQCM considers all cycles found in the DATA_EQCM file. Together with the specification for the Sauerbrey's factor (**sauerbrey** directive), quantitative characterization requires the specification for the electrode area (**electrode_area** directive). Following execution, the OUT_EQCM file contains a table with information of the total charge (in mC) and mass (in ng) computed for each fragment:

```
Cycle   Process   Total charge [mC]      Mass change [ng]
-----------------------------------------------------------
1       oxidation          7.398                89.908
        reduction         -8.183               -73.286
2       oxidation          7.191                98.781
        reduction         -8.035               -85.318
3       oxidation          7.001               111.078
        reduction         -7.834               -99.796
4       oxidation          6.832               121.165
        reduction         -7.631              -109.372
5       oxidation          6.636               123.385
        reduction         -7.455              -112.689
6       oxidation          6.446               120.403
        reduction         -7.262              -111.777
7       oxidation          6.263               115.187
        reduction         -7.072              -109.547
8       oxidation          6.098               108.236
        reduction         -6.858              -103.701
9       oxidation          5.955               102.578
        reduction         -6.505              -100.536
-----------------------------------------------------------
```

The code generates file CHARACTERIZATION within folder **ANALYSIS_EQCM** with information (see Table 8 of sec. 4.3.6 for a detailed description of the computed values) that quantifies the electrochemical response of the host material. These computed quantities are reported in the panels of Fig. 6.4.1 as a function of the cycle number. The observed differences for the magnitudes of the collected charges during oxidation and reduction (Top left), and the fact that $|Q_{ox}+Q_{res}| \neq 0$ and $|Q_{ox}/Q_{res}| \neq 1$ (Top right), suggest that the process is irreversible. This is further demonstrated by the different amounts of ΔM for oxidation and reduction (Bottom left), which leads to an increment of residual mass M_{res} (at the end of the redox cycle) and accumulated mass after each oxidation ($M_{1/2}$, half a cycle). As proposed in Ref. [1], the ratio between mass and charge variation for oxidation and reduction (Bottom right) provides further insight of the electrochemical response for the material.

Exercise 1: Repeat the exercise for a selected range of voltages (between 0.25 and 0.35 V) using the **voltage_range** directive.

Exercise 2: Edit the directives of file SET_EQCM in **tutorial/exercise-04** and evaluate how these computed
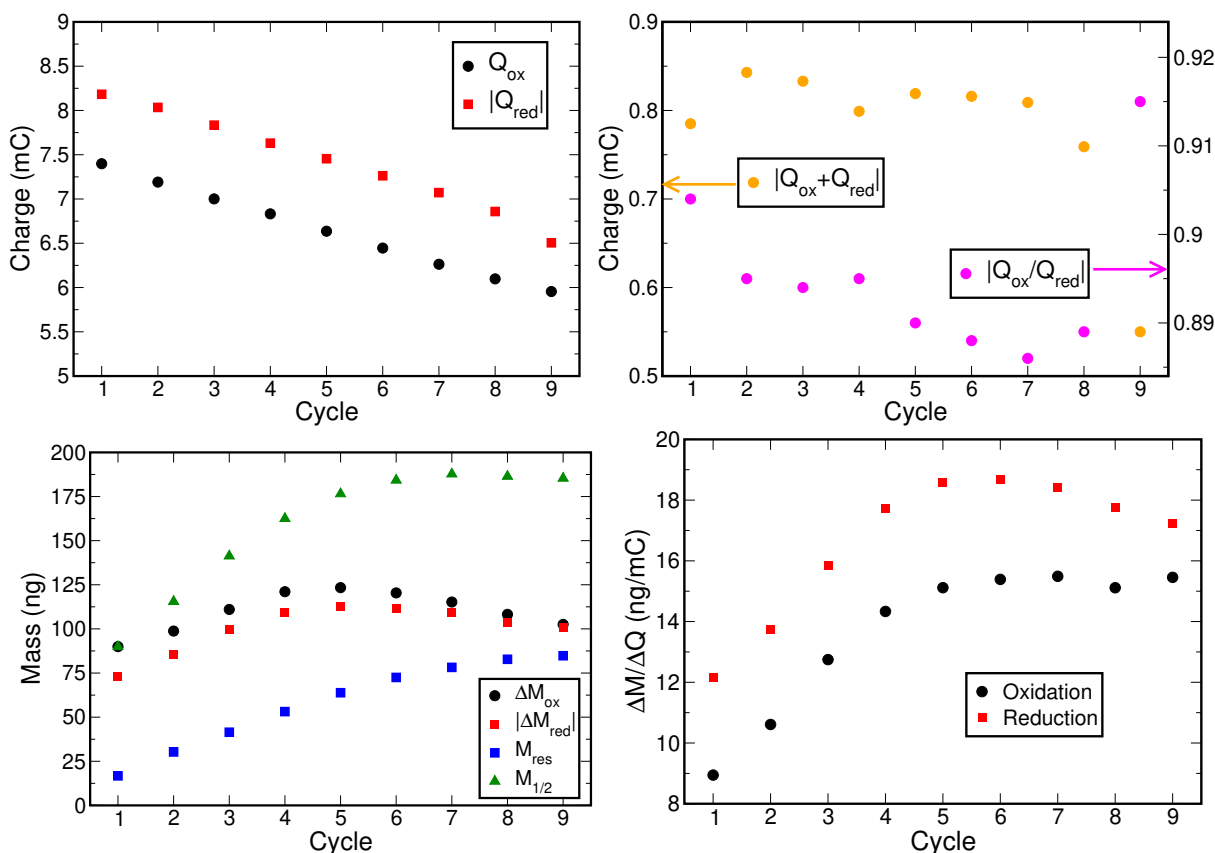
Figure 6.4.1: Quantitative characterization for the intercalation of Li^+ in $Ni(OH)_2$ as a function of the number of cycles. Top left: differences in the accumulated charge during oxidation (black) and reduction (red) demonstrate the irreversibility of the process. Top right: charge difference between oxidation and reduction (orange, left y-axis) and corresponding ratio (magenta, right y-axis). Bottom left: mass variations for oxidation (black) and reduction (red) further demonstrates the irreversibility of the intercalation and leads to the accumulation of mass at the end of each cycle, here called residual mass (blue). Such residual mass leads to a continuous increase of the total accumulated mass at the end of every oxidation ($M_{1/2}$, green). Bottom right: mass to charge ratio for oxidation (black) and reduction (red).

quantities are affected by filtering.

Exercise 3: How the results of Fig. 6.4.1 are affected if we set ***current_offset*** to `.False.`? Why?

6.5   Setting DATA_EQCM from a set of experimental files

When recording the EQCM data, the ideal scenario would be having the whole set of values within a single file (one cycle recorded after the other). Unfortunately, this is not always the case as the cycles might be recorded in separate files, each file corresponding to a different CV cycle. In addition, experimentalists might use labels for the EQCM variables according to their best convenience. For such cases, the correct execution of ALC_EQCM requires of extra (minor) preparation of the DATA_EQCM file to define the appropriate labelling and arrange the experimental data correctly. The following explanatory example is aimed to guide the user with this task. We consider the following files, generated after recording five CV cycles for the electrodeposition of Antimony on Platinum [125]:

3_Sb2mM_PtQCM,newcell_CV_0.6to-1Vat0.05Vs-1_Scan1.txt

3_Sb2mM_PtQCM,newcell_CV_0.6to-1Vat0.05Vs-1_Scan2.txt

3_Sb2mM_PtQCM,newcell_CV_0.6to-1Vat0.05Vs-1_Scan3.txt

3_Sb2mM_PtQCM,newcell_CV_0.6to-1Vat0.05Vs-1_Scan4.txt

3_Sb2mM_PtQCM,newcell_CV_0.6to-1Vat0.05Vs-1_Scan5.txt

Input files for this exercise can be found in folder **tutorial/exercise-05**. The first line of each file correspond to the labels used for the columns:

```
Potential applied (V)   WE(1).Current (A)   WE(1).Charge (C)   Time (s)   External(1).External 1
(V)
External(1).External 2 (V)
```

Here, we observe the last two columns are named as External 1 and External 2. At this point, we ask the researcher who performed the EQCM experiment and find that these labellings correspond to mass-frequency and resistance, respectively, both measured in Volts. We proceed by replacing the label for these variables in the first line (header) of file 3_Sb2mM_PtQCM,newcell_CV_0.6to-1Vat0.05Vs-1_Scan1.txt to read:

```
Potential applied (V)   WE(1).Current (A)   WE(1).Charge (C)   Time (s)   Frequency 1 (V)
Resistance 2 (V)
```

Following the guidelines of sec. 5.2, we remove the headers (first line) from the rest of the files. We also make sure each file is free of non-ASCII characters (see exercise of sec. 6.7). In Linux this is done by executing the following command:

```
tr -cd '\11\12\40-\176'  <  3_Sb2mM_PtQCM,newcell_CV_0.6to-1Vat0.05Vs-1_ScanX.txt  >
scanX.txt
```

where X=1,2,3,4,5. We then concatenate file scan1.tex, followed by scan2.tex, etc. and write the output to DATA_EQCM as follows:

```
cat scan1.txt scan2.txt scan3.txt scan4.txt scan5.txt > DATA_EQCM
```

File SET_EQCM file is set to consider the five CV cycles. Execution of the code will give the following error:

```
***ERROR in file DATA_EQCM - Specified units " appl " for potential/voltage is
different
from "V" (Volts).  Check labelling for header
```

By inspection, we can verify that instead of "V" (Volts) the code interpreted the first four characters of the word "applied" as the units of variable "potential". Guidelines of sec. 5.2 indicate that the name for a variable should be given in one string without separation. Thus, we can either remove "applied" or add any character in between the two strings to make one string. With this fix, we execute the program again and get a different error:

```
***ERROR in file DATA_EQCM - Name of variable for the header of column 2
is NOT recognized:  "we" .  Please check syntax for the label
```

The code interpreted we (the lower case of WE) as a name for the variable, which is obviously wrong. Guidelines of 5.2 instruct that labels for variables should not contain parentheses nor brackets. Consequently, we must remove the parenthesis from the labelling or simply delete the prefix "WE(1).". We observe the same applies to the variable label of the third column. Thus, we modify header to read:

```
Potential (V)   Current (A)   Charge (C)   Time (s)   Frequency 1 (V)   Resistance 2 (V)
```

and execute the code once again, which gives a new error:

```
***ERROR in file DATA_EQCM - Specified units " 1 " for mass-frequency is not valid.
Options:  V or Hz.  Check labelling for header
```

This error arises because we have used two strings to define the last two variables, and "1" was assumed as the units for frequency. By removing the "1" and "2" from the last two labels the program finally runs successfully. Even though this exercise is obvious, it demonstrates ALC_EQCM is capable to guide the user to setting the DATA_EQCM file correctly.

6.6 Defining input directives of the SET_EQCM file from scratch

This exercise aims to help the user to get familiarized with some of the errors of ALC_EQCM when setting directives for the SET_EQCM file from scratch. To this purpose we consider the first EQCM five cycles for the electrodeposition of Antimony, and use the DATA_EQCM file from the previous exercise. Files for this exercise are available in **tutorial/exercise-06**. The aim is to perform a *characterization* analysis.

Let us assume that the experimental data was collected using the Metrohm software. From the experimental setup, we know the quartz crystal disc has a resonant frequency of 5.0 MHz and diameter is 1.3 cm. We start by setting the following two directives in the SET_EQCM file:

```
 # Input directives for characterization analysis
 Software  Metrohm          # Acquisition software (optional)
 Analysis  Characterization  # Characterization of EQCM data
```

Execution AC_EQCM leads to the first error:

```
***ERROR in file SET_EQCM - The selected option for the "analysis" directive needs the

definition of the "current_offset" directive, which must be set to .True.  or .False.

depending on the CV profile and the problem under consideration.  Please amend and
rerun.
```

This error indicates the the *characterization* analysis needs the specification of the **current_offset** directive. We add this line as follows:

```
 # Input directives for characterization analysis
 Software        Metrohm          # Acquisition software (optional)
 Analysis        Characterization  # Characterization of EQCM data
 current_offset  .True.           # Offset the current
```

Re-runing the code we obtain:

```
Units of mass-frequency are given in Volts, and this requires of the V_to_Hz
conversion factor
```

```
***ERROR - No specified value of V_to_Hz in file SET_EQCM
```

As the error message indicates, we need the voltage to Hz conversion factor in the specification. Adding this new directive to the SET_EQCM file, we have:

```
 # Input directives for characterization analysis
 Software        Metrohm          # Acquisition software (optional)
 Analysis        Characterization  # Characterization of EQCM data
 current_offset  .True.           # Offset the current
 V_to_Hz         200.0            # Transformation factor
```

With this extra line we execute again and get a new error:

***ERROR - Computation of the Sauerbrey factor from electrode properties is not
implemented. The user must define directive "sauerbrey" in file SET_EQCM, in units
of [Hz ng-1 cm2].

Calculating this factor from Eq. 1 from the disc area and quartz crystal properties, we include the ***sauerbrey***
directive, bearing in mind the sign:

```
 # Input directives for characterization analysis
 Software        Metrohm               # Acquisition software (optional)
 Analysis        Characterization      # Characterization of EQCM data
 current_offset  .True.                # Offset the current
 V_to_Hz         200.0                 # Transformation factor
 Sauerbrey       0.0566 Hz ng-1 cm2    # Sauerbrey factor
```

Upon execution, ALC_EQCM gives the following error:

To corroborate the validation of the Sauerbrey approximation, the change in
mass-frequency
must be compared with the quartz frequency of the EQCM device.
***ERROR - No specification for "quartz_freq" in file SET_EQCM

which clearly indicates we have missed the specification for the resonant frequency of the quartz crystal.
We add this directive

```
 # Input directives for characterization analysis
 Software        Metrohm               # Acquisition software (optional)
 Analysis        Characterization      # Characterization of EQCM data
 current_offset  .True.                # Offset the current
 V_to_Hz         200.0                 # Transformation factor
 Sauerbrey       0.0566 Hz ng-1 cm2    # Sauerbrey factor
 Quartz_freq     5.0 MHz               # Quartz frequency
```

Execution give a new error:

***ERROR - No specification for "electrode_area" in file SET_EQCM

which leads us to the correct final completed version of the file

```
 # Input directives for characterization analysis
 Software        Metrohm               # Acquisition software (optional)
 Analysis        Characterization      # Characterization of EQCM data
 current_offset  .True.                # Offset the current
 V_to_Hz         200.0                 # Transformation factor
 Sauerbrey       0.0566 Hz ng-1 cm2    # Sauerbrey factor
 Quartz_freq     5.0 MHz               # Quartz frequency
 electrode_area  1.3273 cm2            # Electrode area
```

and a successful execution of the job.

6.7 Removing non-ASCII characters

It might happen that the acquisition software generates files containing non-ASCII characters. Such characters could also be introduced when editing the data in spreadsheets and copying it to text files. In folder **tutorial/exercise-07**, we provide an example of EQCM data (only for EQCM current) that contain non-ascii characters. Upon execution, we get the following error message:

```
***ERROR in file DATA_EQCM - Heading MUST be in the first line (no previous empty
lines)
```

Indeed, the header must be the first line of the file. We edit the file and run the program again to find another error:

```
***ERROR in file DATA_EQCM - It seems there are wrong settings with the file....
1) Remove all non-ASCII characters from file before running the code.
In Linux, this can be done by copying DATA_EQCM to file input_file.dat, for example,
and then execute:
tr -cd '\11\12\40-\176' < input_file.dat > DATA_EQCM
```

The message indicates the presence of non-ASCII characters. Indeed, when we open the DATA_EQCM with the program Vim in Linux, we notice the presence of the character M̂, which is a carriage-return character between the set of data. This is the typical example of a file generated with the DOS/Windows where an end-of-line is marked by a carriage return/newline pair. In the Linux environment, in contrast, the end-of-line is marked by a single newline. By applying the command of the error message above these characters are removed. In contrast to Vim, we would have not observed the M̂ characters if we had opened DATA_EQCM with the Nano editor, for example, but an extra empty line. By saving the file with Nano, the problem is eliminated (as empty lines can be handled by ALC_EQCM).

6.8 Mass calibration of the EQCM device

This exercise shows an example of mass calibration using the available data of Ref. [3] for the electrodeposition of Ag on Au, obtained with the QCM200 software. Input files for this exercise are provided in folder **tutorial/exercise-08**. By setting the option *mass_calibration* for directive **Analysis**, ALC_EQCM prints charge and frequency changes either for all cycles or a selected range of cycles via directive **cycles**. We set the following directives in the SET_EQCM file:

```
# Input directives for mass calibration
# Electrodeposition of Ag on Au
Software        QCM200          # Acquisition software (optional)
Analysis        mass_calibration # Print mass-frequency and charge
scan_rate       100.0  mV s-1   # Voltage scan rate
current_offset  .False.         # Do not offset EQCM current
```

In this example, only voltage, current and mass-frequency were recorded in the EQCM experiment. Thus, integration of current requires of the voltage scan rate, as specified above. The implicit assumption is that the scan rate must be kept constant during the whole experiment. It is user's responsibility to fulfill this condition, otherwise the computed charges will be wrong. Figure 6.8.1 demonstrates the linear dependence between the measured mass-frequency Δf and the computed charge for the first five cycles. Data can be fitted and the calculated slope used together with Eq. (14) to determine the ratio C_f/A_{eff}, under the assumption of $n_e = 1$ for the electrodeposition of Ag. The calibration coefficient can be used to determine mass variation for other electrodeposition reactions.

Exercise: Rerun the simulation, this time setting **current_offset** to .True.? How would you interpret the data for calibration?
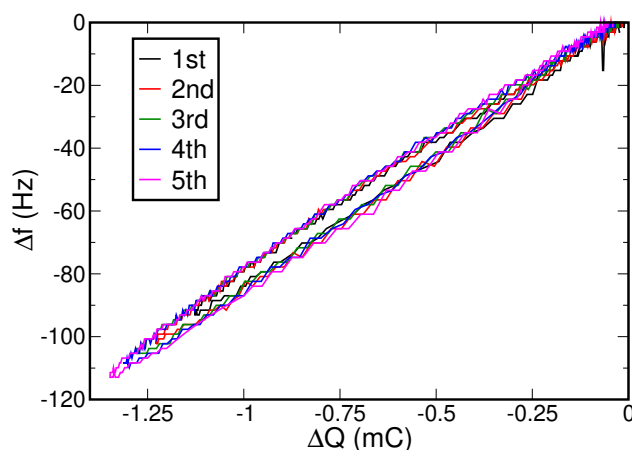
Figure 6.8.1: Relation between Δf and ΔQ for the first five electrodeposition cycles of Ag in Au. We observe a clear linear correlation between both variables.

6.9 Massogram profiles

To exemplify the computation of massograms, we consider EQCM data for the electrodeposition of Antimony on a Platinum electrode. Particularly, we focus on the first and the fifth cycle. Input files SET_EQCM and DATA_EQCM are provided in folder **tutorial/exercise-09**. We start by extracting the mass-density from the raw DATA_EQCM file for cycle 1 and cycle 5 separately. This can be done with the *cycle* directive. We leave this task to the user (Hint: user *print_EQCM_raw* option for directive *Analysis*). Results for mass-density are shown in the left panel of Fig. 6.9.1. We observe that mass is first accumulated during reduction (dV<0) due to the electrodeposition of Sb. During oxidation (dV>0), Sb atoms are stripped from the surface and the amount of mass reduces. The process is irreversible with a net accumulation of mass at the end of the cycle. This accumulation explains the larger values for the mass densities during cycle 5 in comparison with cycle 1.

To obtain massograms from EQCM mass-density changes, the user must select the option *Massogram* for directive *Analysis*. File SET_EQCM is set as follows:

```
# Input directives for massogram analysis
# Case:  Electrodeposition of Sb on Pt
Software        Metrohm               # Acquisition software (optional)
Analysis        Massogram             # Compute massogram from EQCM data
V_to_Hz         200.0                 # Transformation factor
Sauerbrey       0.0566 Hz ng-1 cm2    # Sauerbrey factor
Quartz_freq     5.0 MHz               # Quartz frequency
Cycles          1 1                   # Only first cycle
Current_offset  .True.                # Offset current
```

where, as per setting of directive *Cycles*, only the first cycle is considered. Computation of the massogram only for cycle 5 can be executed by setting `Cycles 5 5`. Alternatively, all cycles 1 to 5 can be computed at once. In this example, frequency values are recorded is given in units of Volts (see DATA_EQCM file), hence the specification of directive *V_to_Hz* to transform the values to Hz. The resulting mass-density rates for both cycles are reported in central panel of Fig. 6.9.1 and exhibit a strong noise. This is surprising given the apparent smooth mass-densities profiles of the left panel. Nevertheless, raw mass-densities are only smooth in the scale of the left panel: they actually exhibit a saw-like pattern, which is determined by the resolution settings for the acquisition of the EQCM data. We invite the user to investigate the mass-density
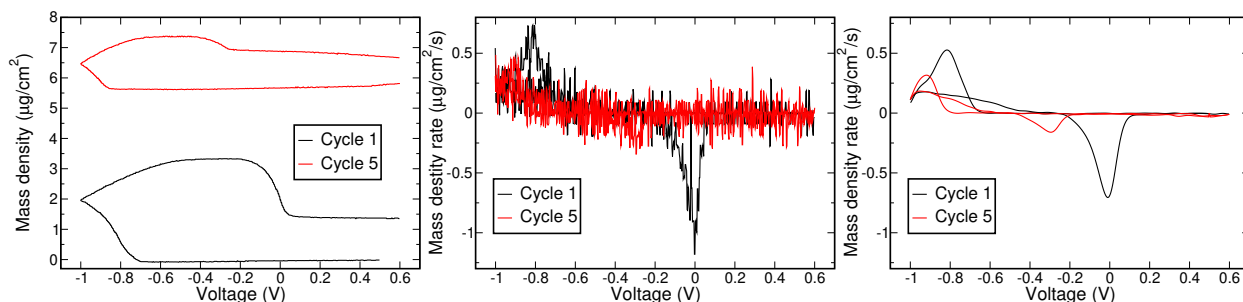
Figure 6.9.1: Massogram analysis for the electrodeposition of Sb on a Pt electrode. Left: Voltammograms for the first (black) and fifth (red) cycle. Centre: computed massograms using the raw data of the left panel. Right: computed massograms using a filter of 0.5 Hz for the raw data.

values using a different scale. Such a behavior explains the strong noise component of the massograms, which severely complicates the analysis. To remove this noise, we apply a filter (see sec. 6.3) with a cutoff frequency of 0.5 Hz, by adding the directive *Filter_cutoff 0.5 Hz* to the SET_EQCM file. This filter removes the noise from the massograms, as shown in the right panel of Fig. 6.9.1.

Exercise 1: perform a spectra analysis for all the 5 cycles. Can you identify the common source of noise?

Exercise 2: compute the massograms for cycle 2, 3 and 4 using a frequency cutoff of 0.5 Hz. Can you elaborate a qualitative argument to explain the change in the massograms with cycling?

6.10   Stoichiometry analysis for the intercalation of species

This section provides examples to compute stoichiometric changes from EQCM data. Setting subscripts and superscripts in standard text editors (Vim, Emacs, etc) can be cumbersome and their use must be avoided when defining chemical species. Tags for species must be defined using alphanumeric strings. For example, we shall refer to species Ni_2O, H_2O, K^+ and H^+ using the tags `Ni2O`, `H2O`, `K+` and `H+`, respectively. Hereinafter, we will refer to species using either notation.

6.10.1   Two variable species

To exemplify the stoichiometric analysis for ion intercalation data involving two chemical species, we consider the electrochemical cycling of $Ni(OH)_2$ immersed in a 1M solution of LiOH. Input files can be found in folder **tutorial/exercise-10/2-species**. Note that the experimental data for this example differs from exercise 6.4 (it corresponds to a different experiment for the same process). To conduct the stoichiometric analysis, we append the following directives to the file SET_EQCM of sec. 6.4 (and set *Stoichiometry* for **Analysis**):

```
# Directive for stoichiometric analysis
######################################
process         intercalation  # Type of process
efficiency      1.00           # Efficiency of the reaction is set to 100%
electrode_mass  6.73138E-06 g  # Mass of the pristine electrode
```

```
# Stoichiometric species
&Block_Species
   number_species  4       # Number of species
   # Name           Weight  Oxidation  s0   Type of variable
   NiO2            90.692  0.0        1.0  fixed
   H2O             18.015  0.0        1.0  fixed
   Li+             6.940   1.0        0.0  dependent
   H+              1.008   1.0        2.0  dependent
&End_Block_Species
```

To instruct ALC_EQCM that the process under study corresponds to the intercalation of species, the user must set directive **process** to *intercalation*. Crucial to EQCM data analysis is to offset the value of the measured current at the beginning of the cycle. This is done by setting **current_offset** to .True.. Another important variable to take into account is the efficiency of the intercalation process, which specifies the ratio of the pristine host electrode material that participates in the reaction. This is specified by directive **efficiency**, here set to 1.00 (100%), which is also the default. The electrode mass is specified by directive **electrode_mass** and it is used together with the molecular/atomic weights of the species defined in **&Block_Species** to compute the total amount of moles of the pristine host electrode material. Alternatively, ALC_EQCM offers the **electrode_moles** directive to set the amount of moles.

Relevant to the stoichiometric analysis is the specification of the participating species during intercalation. Such information must be set in block **&Block_Species**. We refer the user to Table 3 for the general format and syntax of this block. Previous experiments for this material [1] revealed a layered turbostratic structure with intercalated water with stoichiometry of 1. Thus, the formula unit for the pristine sample is found to be $Ni(OH)_2(H_2O)$. For this example, we assume the intercalation of Li+ ions from the surrounding solution occurs at expenses of removing only H+ from the host structure, while the amount of H_2O and NiO_2 remains fixed. For the specification of **&block_species**, the first line must be the directive **number_species**, in this case set to 4. Comment lines (starting with #) are allowed within this block. Here, we use a comment (shown in magenta for clarity) to instruct the user with the correct syntax/format for the specification of the species. Each species must be defined with a convenient label (preferably associated with its chemical composition), followed by its atomic/molecular weight, oxidation number, stoichiometry within the pristine sample (s0) and type of variable. In this case, we have two variables species (Li+ and H+) that must be defined as dependent and two fixed species (NiO2 and H2O) that must be defined as fixed. The block must be closed with the directive **&End_Block_Species**.

For the present case, the initial stoichiometry $Ni(OH)_2(H_2O)$ is specified by setting values 1.0, 1.0 and 2.0 for species NiO2, H2O and H+ (see block). The oxidation number for Li+ and H+ is 1.0, whereas it is zero for NiO2 and H2O.

Execution of this example first performs the EQCM characterization analysis, and prints the computed values to CHARACTERIZATION file. Computed quantities for accumulated mass and charge are used to solve the charge and mass balance equations, and obtain the stoichiometric coefficients for Li+ and H+ within the sample for each oxidation and reduction. Results are printed in files INTERCALATION_OX and INTERCALATION_RED. Stoichiometric coefficients are reported in Fig. 6.10.1. Information of the total amount of electrons per unit cell is also computed. These files are located within folder **ANALYSIS_EQCM**. In file OUT_EQCM, the users will find a summary of the calculation, as well as a table with the description of each chemical species, as defined in **&Block_Species**.

To interpret the information from the computed stoichiometric solutions, we consider the first oxidation from the INTERCALATION_OX file. Using the composition for the pristine sample from **&Block_Species** we can write the reaction as follows:

$$Ni(OH)_2(H_2O) + 0.547Li^+ - (2 - 0.089)H^+ \rightarrow NiO_2H_{0.089}^+Li_{0.547}^+(H_2O) + 1.364e^- \qquad (43)$$
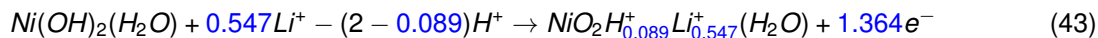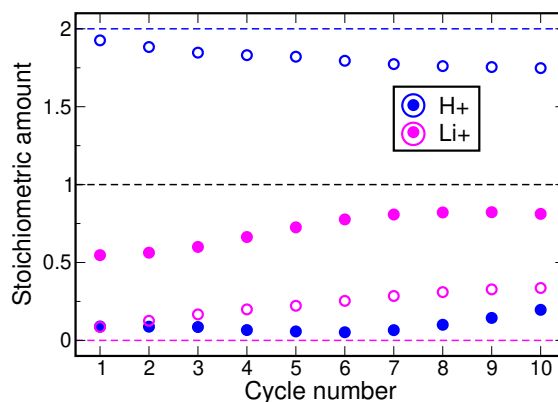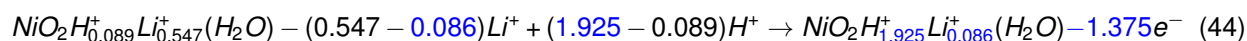
Figure 6.10.1: Stoichiometry values for Li+ and H+ following oxidation (filled symbols) and reduction (empty symbols) as a function of the cycle number.

while the first reduction reaction (file INTERCALATION_RED):

$$NiO_2H^+_{0.089}Li^+_{0.547}(H_2O) - (0.547 - 0.086)Li^+ + (1.925 - 0.089)H^+ \rightarrow NiO_2H^+_{1.925}Li^+_{0.086}(H_2O) - 1.375e^- \quad (44)$$

In the last two equations, we have highlighted in blue the stoichiometric coefficients as they are reported for the OX/RED files for the first cycle. Such values indicate the content of the involved species within the sample after the reaction, while the number of electrons per formula unit denotes the charge associated with the process.

Exercise 1: What can we learn from the computed results of Fig. 6.10.1?

Exercise 2: What happens to the amount of accumulated electrons (holes) as the cycling progresses?

Exercise 3: Assuming the standard oxidation number of -2 for Oxygen, use the compute values for Li+ and H+ to calculate the effective oxidation number for Ni. Plot the values as a function of the cycles.

Exercise 4: How do results change if the value for directive *efficiency* is decreased?

6.10.2 Three variable species

For ion intercalation processes with three variable chemical species, we consider the electrochemical cycling of Ni(OH)$_2$ immersed in a 1M solution of KOH. In contrast to sec. 6.10.1, in this exercise we also allow intercalated water within the host electrode to change its content. Input files for the first six CV cycles can be found in the **tutorial/exercise-10/3-species** folder. Inspection of the SET_EQCM file shows that the amount of mass for the pristine electrode is different (6.80988E-06 g) with respect to the previous section. Specification of the chemical species follows:

```
# Stoichiometric species
&Block_Species
number_species   4        # Number of species
# Name          Weight    Oxidation  s0    Type of variable
NiO2            90.692    0.0        1.0   fixed
H2O             18.015    0.0        1.0   dependent
K+              39.0983   1.0        0.0   dependent
H+              1.008     1.0        2.0   independent
&End_Block_Species
```

where both H2O and K+ are set as dependent variables, whereas H+ is the independent one. For a description of the settings we refer the user to the example of sec. 6.10.1. With 3 variables chemical species, the
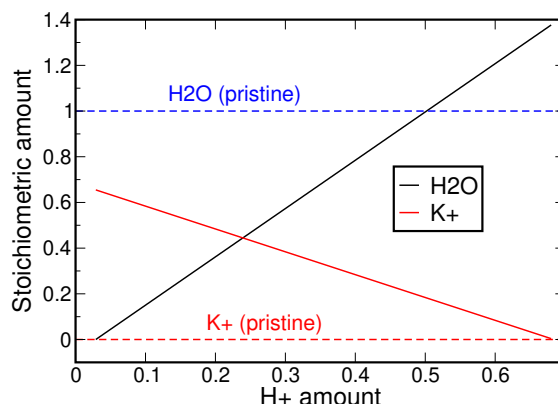
Figure 6.10.2: Stoichiometric solutions for the sample compatible with the EQCM data following the first oxidation. Both K+ (solid red) and H2O (solid black) coefficients are reported as a function of the H+ content. The blue horizontal dashed line corresponds to the content of H_2O within the pristine sample.

system of two equations for mass and charge balance is underdetermined, and a multiple set of solutions is obtained for the first fragment of the CV cycle (in this case oxidation). In ALC_EQCM, the stoichiometric domain for each `independent` species is discretised using the value of directive **delta_stoich**, in this case set to 0.001 (0.01 by default). This setting for the discretization leads to 652 possible solutions, with stoichiometries printed to the INTERCALATION_OX file (inside folder **ANALYSIS_EQCM**). In principle, each of these solutions is a valid starting stoichiometry for the next reduction. This uncertainty prevents the analysis for the subsequent reduction and the execution is terminated. Results of INTERCALATION_OX file for the content of K+ and H2O are plotted as a function of the independent H+ species in Figure 6.10.2. We find that the computed solutions contain a considerable reduction of the H+ content (equal to 2 for the pristine sample).

Further analysis for the stoichiometric solutions at both extremes of the domain reveals interesting information about the process. For the solution with [H+]=0.68 (rounded to the second decimal), we find [K+]=0.004 and [H2O]=1.38. This solution indicates that a minimum amount of K+ is intercalated within the sample together with H2O. The computed ratio d[H2O]/d[K+]=0.38/0.004=95 shows that each K+ would intercalate together with 95 H2O molecules. From chemical intuition, this ratio appears to be exceedingly large. In contrast, for the water free solution with [H+]=0.03, [K+]=0.65 (and [H2O]=0.00), the computed ratio d[H2O]/d[K+]=-1.0/0.65=-1.53, thus indicating the insertion of each K+ entails the removal of ≈ 1.5 H2O molecules from the host electrode. Even though the two solutions represent a qualitative different process for the intercalation of K+, they are both possible mathematical solutions to the stoichiometric problem compatible with the EQCM data. To elucidate which solution within the domain reported in Fig. 6.10.2 is the energetically favored, it is required to assess the problem using atomistic simulations. The strategy is to use a selected number of solutions, build atomistic models with the corresponding stoichiometries and compute each of these models separately. The configuration with lowest formation energy will correspond to the stoichiometric composition of the sample following oxidation [1].

Exercise 1: What is the consequence in the computed results if directive **current_offset** is set to `.False.`?

Exercise 2: Repeat the exercise this time by fixing the amount of intercalated water, as for the settings of tutorial of sec. 6.10.2. What do you obtain?

6.10.3 Three species with constraints

Results of the previous section show that when the number of involved chemical species (allowed to change content) during an intercalation process is larger than two, one obtains a set of multiple solutions compatible with the EQCM data. To reduce the dimensionality of the computed solutions, it is necessary to impose constraints between the chemical species. Such constraints can be derived from chemical intuition or atomistic level simulations. In ALC_EQCM, constraints are specified via **&Block_constraints_species**. To obtain only one solution from the set of multiple solutions of Fig. 6.10.2, we must specify only one constraint for the involved variables. Input files for this example can be found in folder **tutorial/exercise-10/3-species-constraints**. Previous research demonstrated that following oxidation, the lowest formation energy structure is obtained for configurations with minimum H+ content [1]. In ALC_EQCM this constraint can be imposed by adding the following lines to the SET_EQCM file of the previous section:

```
# Constraints
&Block_constraints_Species
number_constraints  1          # Number of constraints
target_min          oxidation  H+
&End_Block_constraints_Species
```

where the type of constraint `target_min` instructs ALC_EQCM to only consider the solution with the minimum value for H+ after the oxidation. The computed solution ([H+]=0.03, [K+]=0.65, [H2O]=0.00) is printed to file INTERCALATION_OX and used as the starting point for the subsequent reduction process. Similarly to the example of previous section, the lack of constraints for the reduction leads to the set of multiple possible solutions reported in Fig. 6.10.3. The solution with the lowest value of [H+]=1.56, [K+]=0.50 and no H2O content indicates that the amount of K+ reduces (in 0.15) only at expenses of the reinsertion of H+ without any H2O involved. In contrast, the solution with [H+]=2.05, [H2O]=1.04 and [K+]=0 has a composition that is close to the pristine sample ([H+]=2.00, [H2O]=1.00 and [K+]=0). Each of the multiple solutions computed following reduction will represent a possible starting point for the second oxidation and, consequently, the stoichiometric analysis is stopped. To obtain a single reductive solution, it is required to add a constraint for reduction. From the computed solution after oxidation, we found that the intercalation of each K+ leads to the removal of 1.53 H2O molecules. Based on Ref. [1], we hypothesize that the same ratio is kept during reduction (e.g. the removal of each K+ ion leads to the re-insertion of 1.53 H2O molecules) and set this condition by adding an extra constraints as follows:

```
# Constraints
&Block_constraints_Species
number_constraints  2          # Number of constraints
target_min          oxidation  H+
keep_ratio          reduction  d[H2O]|d[K+]
&End_Block_constraints_Species
```

With this constraint, the stoichiometric composition after reduction is [H+]=1.96, [H2O]=0.865 and [K+]=0.09, which demonstrates a residual amount of K+ after the first redox cycle.

The constraints for oxidation and reduction lead to a single solution for each process. This allows computing stoichiometry changes with cycling. Results following oxidation(reduction) for each cycle are printed to files INTERCALATION_OX(RED). Computed values for oxidation (filled symbols) and reduction (empty symbols) are shown in Fig. 6.10.4 for each participating species. After each oxidation, the structure contains no water, the amount of K+ remains practically constant and the content of H+ monotonically increases with the cycles. After reduction, in contrast, the amount of H+ is practically recovered with respect to the pristine composition, while the amount of H2O and K+ decreases and increases monotonically, respectively. The rather large variation of water content between oxidation and reduction was put forward to explain the observed instability for this system, responsible of the phase transformations and electrochemical degradation of this material [1].
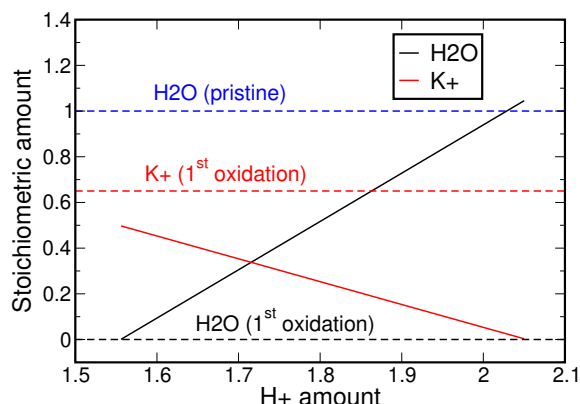
80

Figure 6.10.3: Stoichiometric solutions compatible with the EQCM data following the first reduction. Both K+ (solid red) and H2O (solid black) coefficients are reported as a function of the H+ content. Black and red dashed lines correspond to content of H_2O and K+ following oxidation. The blue dashed line corresponds to the H2O content within the pristine sample.

Exercise 1: Follow the procedure of section 6.10.1, write the chemical reactions corresponding to the first and oxidation and reduction.

Exercise 2: The user is asked to try other types of constraints for the reduction process and check the effect on the computed stoichiometric solutions.

6.11   Electrodeposition analysis

6.11.1   Antimony on Pt

To exemplify the capabilities of ALC_EQCM to compute electrodeposition quantities, we consider Sb deposited over an electrode of Pt. Input files for this example can be found in folder **tutorial/exercise-11**. In the SET_EQCM file, the user must set the *electrodeposition* option for the ***process*** directive. Moreover, species Pt and Sb must be defined as fixed and independent in **&Block_constraints_Species**, respectively. In contrast to intercalation, the total amount of moles that participate in the reaction is computed for
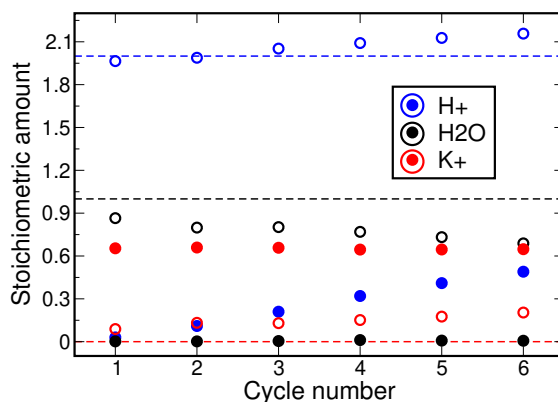

Figure 6.10.4: Stoichiometric solutions compatible with the EQCM data for the first 6 cycles. Results for H+ (blue), H2O (black) and K+ (red) are shown following oxidation (filled symbols) and reduction (empty symbols). Dashed lines correspond to the stoichiometry of the pristine sample.

81

each oxidation/reduction fragment, and results are printed to ELECTRO_DEPOSITION file within folder **ANAL-YSIS_EQCM**. Positive (negative) values for moles indicate deposition (removal) of the chemical species. The disk area, A_{disk}, charge accumulation and mass variations (reported in file CHARACTERIZATION) are used to compute the ratio $R = A_{eff}/A_{disk}$, as detailed in sec. 3.8. If the electrode area is fully covered with the deposited species, and the amount of material is such that the Sauerbrey equation is still valid, R can be taken as an indicative measure of the surface rugosity [44]. In contrast, R can be taken as a correction to the Sauerbrey factor, C_f, computed via Eq. (2), as long as the non-gravimetric effects can be neglected. Figure 6.11.1 shows the computed ratio R for the first five electrodeposition cycles. We find that the amount of deposited Sb is always larger (more than 1.9 times) than the amount of removed Sb, thus leading to an accumulation of Sb over the Pt electrode disk with cycling. Interestingly, we find R is practically the same for oxidation and reduction from the 4th cycle, which suggest Sb is deposited (stripped off) on (from) a surface that is already fully covered by Sb. Comparison with two experimental values for the surface roughness of Sb(111) (blue dashed lines) [126] indicate that the computed values for R correlates to a fully covered Sb-electrode.
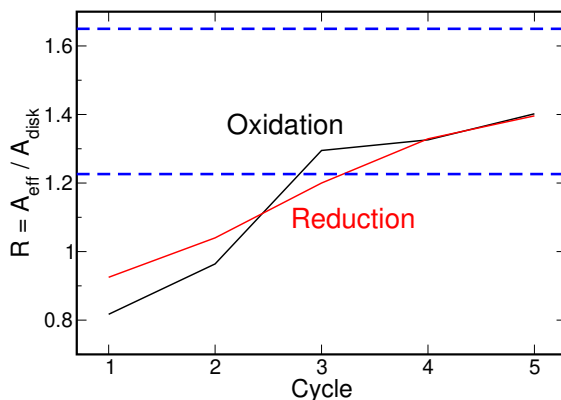

Figure 6.11.1: Computed $R = A_{eff}/A_{disk}$ following reduction (deposition) and oxidation (removal) for the electrodeposition of Sb atoms over Pt for the first five EQCM cycles. The blue dashed lines correspond to the roughness measured for Sb(111) using two different preparation methods [126].

Exercise: Repeat the exercise by removing the species Pt in **&Block_species**. What do you obtain?

6.12   Building atomistic models

6.12.1   Intercalation of K+ cations (multiple solutions)

Here we base on the data of section 6.10.2 for the electrochemical cycling of Ni(OH)$_2$ immersed in a 1M solution of KOH. Input files for this example can be found in folder **tutorial/exercise-12/K+**. From the settings in **&block_species** we have three variable species that participate in the reaction (H2O, K+ and H+), whereas NiO2 is set as fixed. This leads to an underdetermined system of equation with an infinite number of possible solutions, numerically represented in a grid.

We start by setting option *Model_cycled_sample* for directive **Analysis** to request building atomistic models compatible with the EQCM data. To generate atomistic structures, we need to specify the atomistic details of the species defined in **&Block_species**. We do this with **&Block_species_components**:

```
&Block_species_components
  #NiO2
  NiO2  2  crystal
  Ni Ni 1 || Oh O 2
  #H2O
  H2O   2  molecule  1.2
  Hw H 2 || Ow O 1
  #K+
  K+    1  atom
  K+ K 1
  #H+
  H+    1  atom
  H+ H 1
&End_block_species_components
```

Comments within this block (shown in magenta) must start with the symbol #. We refer the user to Table 3 for a general description of this block. For this particular example, species tagged as NiO2 was defined as fixed in **&Block_species** (its content does not change during the reaction). This species is composed of 2 atomic components (Ni and O), and its class must be defined as crystal (fixed set of species that are part of the host structure). The next line defines the information of each individual atomic component. Nickel is tagged as Ni (it could be any other tag), which corresponds the element Ni, and contributes with 1 atom to the stoichiometry of the NiO2 unit. Specification for oxygen must follow after the separator ||. The separator can be any string and means to help the user to track the settings, especially when the species is composed of many atomic constituents. We tag the oxygen atoms of NiO2 as Oh, which obviously corresponds to the chemical element O and contributes to the NiO2 unit with 2 atoms.

For the intercalated species H2O, we set the number of atomic components equal to 2 (H and O), followed by the class of the species, molecule. This is in contrast to the unit NiO2 which is fixed and part of the crystal structure. The user must also specify the maximum intramolecular bonding length for water. This parameter depends on the geometry of the molecular species under consideration. Here, we set a sensible value of 1.2 Å. In the next line, we first declare the tag Hw for hydrogen (to distinguish water hydrogen from other hydrogen present in the sample), followed by the atomic element (H) and number of hydrogen atoms within H2O, equal to 2. Similarly, for the oxygen component we set the tag Ow (different from the NiO2 oxygen), followed by the atomic element O and the number of oxygens per water molecule, equal to 1.

For the atomic species K+, we specify the tag K+, followed by 1 (only one constituent) and atom, which is the class for this component. In contrast to the case of water, there is obviously no need to specify any bonding distance criteria for atomic species. In the next line we specify the tag for this constituent, set to K+ (but it could be anything else), followed by the chemical element K and the number of atoms for this species, equal to 1. Settings for the definition of H+ are analogous to K+. In this case we keep the tag H+ to differentiate these hydrogen atoms from those of the water molecule. Even though H+ bonds to Oh to form Ni(OH)$_2$, its content is not fixed but allowed to change, and we define its class as atom.

It is important to remark that the order for the specification of the atomic component for both crystal and molecule species is irrelevant. In fact, we could first define Oh O 2 and then Ni Ni 1 for NiO2, for example. Moreover, no bonding distance is needed for the specification of the crystal type of species. Finally, the order for the definition of the atomic components of species does not need to follow the order of **&Block_species**, as long as all species are defined.

Building atomistic models requires of an input set of coordinates that serves as a structural starting point. Such input model must be given in file INPUT_STRUCTURE (located within folder **INPUT_GEOM**), and must contain at least all atoms for the species defined as fixed. In this example, the input structure file is consistent with the vasp format (POSCAR). The first 8 lines read:

```
NiOH-hydrated
1.0000000000000000
 5.3752349157118466 0.0449292506095459 -0.0647497768438475
-0.0072728085682880 18.8094842744397610 -0.7187960385029701
-1.9019035222542005 -0.4409798504095773 7.9492302078989248
Ni O H O H
12 24 24 10 20
Cartesian
```

The format of the this file must be specified with directive **input_model_format** in SET_EQCM, set to vasp in this case. We recommend the user to follow the guidelines of Ref. [127] for a detailed explanation of the POSCAR structure. For the moment we focus on the sixth line, which lists the order for the atomic elements of the model. The number of atoms for each of these elements is provided in line 7. According to these settings, the first 12 atomic coordinates (below Cartesian, no shown) correspond to Ni atoms, and the next 24 lines correspond to oxygen. Inspection of the file shows that this set of coordinates corresponds to 12 NiO2 units of the model. The 24 following lines are hydrogen atoms, which are part of the $Ni(OH)_2$ structure. Finally, the remaining set of 10 O and 20 H atoms correspond to 10 intercalated water molecules. To relate the tags (defined in **&Block_species_components**) for each of the elements above, the user must define the following block in the SET_EQCM file:

```
&Block_input_composition
tags    Ni Oh H+ Ow Hw K+
amounts 12 24 24 10 20 0
&End_block_input_composition
```

In this way, ALC_EQCM can identify each atom listed in the INPUT_STRUCTURE and relate it to the atomic tag it belongs to. It is important that the user provides the atomic coordinates of the INPUT_STRUCTURE file consistent with the information of **&Block_input_composition**, otherwise the code will complain and abort. Moreover, the tags in **&Block_input_composition** must be defined only once, even those tags that do not have atoms in the INPUT_STRUCTURE file, as it occurs with K+. This condition in the definition of input models is, maybe, the only non-general aspect of the input settings. The format of the generated atomistic models is set via directive **output_model_format**. Note that the stoichiometry of the input model is $(NiO2)_1 H_2 (H2O)_{0.833}$, which does not need to coincide with the pristine stoichiometry defined in **&Block_species**, namely $(NiO2)_1 H_2 (H2O)_1$.

Returning to the stoichiometry for this problem, the set of computed solutions is represented in a grid, as shown in Fig. 6.10.2. The number of solutions (652, see INTERCALATION_OX) depends on the setting for directive **delta_stoich** (0.001). Building 652 atomistic models is obviously not practical. Directive **targeted_num_models** offers the option to reduce the number of atomistic models to be generated with stoichiometric compositions that adequately screen the domain of possible solutions. In this case, we ask the code to generate 8 models. We return to this point later.

The user also needs to specify the distance cutoff between atoms of different species via **distance_cutoff**, set to 1.80 Angstrom in this case. The size of the generated models is controlled with directive **repeat_input_model**. In this case, the size of the generated models will be 3, 2 and 2 times the repetition of the input model along the c1, c2, and c3 cell vectors, respectively.

Since H2O is the only molecular species, we also need the H2O.xyz file, which specifies the geometry of the species. This file must be located in folder **INPUT_GEOM**. If this file is not present or its format is incorrect, the code will abort.

Execution of this example first computes the mass and charge variations for all the cycles identified in DATA_EQCM and solves the stoichiometry problem. Results are printed to the CHARACTERIZATION file. Since the problem is underdetermined, a set of multiple 652 solutions is computed and printed to IN-TERCALATION_OX. The code continues with the generation of 8 models that are printed to folders **ATOM-**
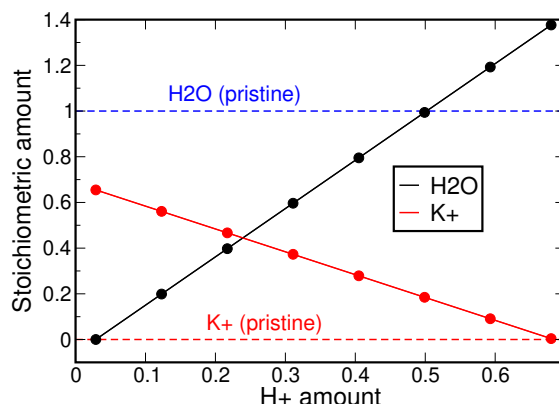
Figure 6.12.1: Stoichiometric solutions for the sample compatible with the EQCM data following the first oxidation. Both K+ (red) and H2O (black) coefficients are reported as a function of the H+ content. The solid lines represent the 652 solutions of the mass and charge balance equations. Symbols corresponds to the set of 8 selected solutions used to build atomistic models. The blue horizontal dashed line corresponds to the content of H_2O within the pristine sample as defined in **&Block_species**.

ISTIC_MODELS/1cycle-oxidation/modelX, where X is the model number. Each model has a target stoichiometry as reported in the SELECTED_SOLUTIONS file (inside **ANALYSIS_EQCM**). These solutions are shown in Fig. 6.12.1 for K+ and H2O as a function of H+ (filled circles) and compared with the 652 solutions results (solid line) of Fig. 6.10.2.

Details for each of the generated models are printed to OUT_EQCM file as tables. For example, let us consider model 1:

```
-------------------------------------------------
        | Total |      Stoichiometry
Species | number | Model  Target  Difference
-------------------------------------------------
  NiO2      144   1.000   1.000    0.000
  H2O         0   0.000   0.000   -0.000
   K+        94   0.653   0.655   -0.002
   H+         4   0.028   0.029   -0.001
-------------------------------------------------
***SUCCESS: target and modelled stoichiometries agree within the error tolerance.
```

For this stoichiometric solution, we observe that the intercalated water is fully removed from the sample, and very little H+ is left. In contrast, the amount of K+ is significantly large. In comparison to the pristine composition of $Ni(OH)_2(H_2O)$, intercalation of K+ takes place via the removal of H2O and H+ from the host sample. The table also shows the amount of species units for each kind, as well as the model and target stoichiometries together with the computed differences. If such differences are larger than the tolerance (see directive *stoichiometry_error*, here not defined but set to a default value of 0.01), the code will inform the user accordingly. For referencing purposes, this table is also printed to file **ATOMISTIC_MODELS/1cycle-oxidation/model1**/MODEL_SUMMARY.

In addition, the **RESTART** folder is created, which contains the file RECORD_MODELS with relevant information of the generated models as well as a copy of the OUT_EQCM, named as OUT_EQCM_BACKUP.

To determine which of the generated structures is the energetically/thermodynamically favorable composition following the first oxidation, one must compute each of the generated models via atomistic simulations. As the generated models contain different atoms of different kind, the user must compare the formation

(free-) energy between the samples. Since the structures are generated by removing and inserting species randomly within the system, it is recommended to generate several models for the same stoichiometric solution by running the code several times (and changing the name of directory

ATOMISTIC_MODELS/1cycle-oxidation to avoid being rewritten). Finding the energetically favorable structure allows to withdraw important information of the process of ion-intercalation [1].

Exercise: Following the guidelines of the presented exercise, build atomistic models for the intercalation of Li+ using the files and settings of sec. 6.10.1. Repeat the exercise by allowing the content of water to change. How do results compare with the intercalation of K+?

6.12.2   Changing the content of intercalated water within Ni(OH)$_2$

Computation of the mass and charge balance equations relies on the knowledge of the stoichiometric composition of the pristine sample, which we have set to Ni(OH)$_2$(H$_2$O), i. e. intercalated water with stoichiometry of 1, based on previous research for this system [1]. However, there is not always certainty about the composition of the pristine sample previous to CV cycling. Clearly, no EQCM data can be used in this case to guide the search for the favorable composition of the host material. Thus, one must explore different configurations and generate models with different stoichiometries for the pristine material.

In the present case we know that the host material is Ni(OH)$_2$ and we aim to build models with different content of intercalated water. Input files for this example can be found in folder **tutorial/exercise-12/H2O**. In this case, INPUT_STRUCTURE file (inside folder **INPUT_GEOM**) contains a model for the Ni(OH)$_2$ only. Since we will change the content of water, we must also provide the H2O.xyz file, which is borrowed from the previous exercise.

In file SET_EQCM we choose the options *Model_pristine_sample* and *intercalation* for directives

Analysis and **process**, respectively. Settings for **&Block_species** and **&Block_species_components** involve three species, where H+ is set to be fixed and part of crystal. The structure of **&Block_input_composition** only contains non-zero number of atoms (amounts) for Ni, Oh and H+. The target stoichiometry is specified with the s0 values in **&Block_species**. We set an initial stoichiometry of 0.5 for H2O. Upon execution, the details for the input model are first compared with the target stoichiometry in the OUT_EQCM:

```
Atomistic details for the INPUT_STRUCTURE model
------------------------------------------------
         | Total |      Stoichiometry
 Species | number |  Model  Target  Difference
------------------------------------------------
   NiO2        12  1.000  1.000   0.000
    H+         24  2.000  2.000   0.000
   H2O         12  0.000  0.500  -0.500      **** EXCEEDS ERROR TOLERANCE
------------------------------------------------
```

which clearly shows that the input model is free of water, and the difference with respect to the target stoichiometry s0 is -0.5. Since the difference in H2O content is larger than the stoichiometry error tolerance (not defined in the SET_EQCM file; 0.01 by default), we obtain a warning message **** EXCEEDS ERROR TOLERANCE. To match the target stoichiometry for water, the algorithm will insert water molecules. The generated model is also reported as a table:

```
-----------------------------------------------
        | Total |       Stoichiometry
Species | number |  Model  Target  Difference
-----------------------------------------------
  NiO2          12  1.000  1.000    0.000
    H+          24  2.000  2.000    0.000
   H2O          12  0.500  0.500    0.000
-----------------------------------------------
 ***SUCCESS: target and modelled stoichiometries agree within the error tolerance.
```

The generated structure is saved to SAMPLE.vasp file within folder **ATOMISTIC_MODELS/pristine**. For reference, the previous table is also printed to file MODEL_SUMMARY.

Exercise: Build pristine models by changing the content of intercalated water from 0 to 1.2 using an increment of 0.1. Are all the generated models accurate enough within the error tolerance? By how much the size of the models has to be increased to obtain accurate models?

6.12.3 Electrodeposition of Antimony on Pt(111)

In this section we build models for the electrodeposition of Antimony (Sb) on Platinum (Pt) compatible with the EQCM data of sec. 6.11.1. Input files for this example can be found in folder **tutorial/exercise-12/Sb**. To model the bare Pt electrode surface we use a "perfect" flat slab exhibiting the (111) facet with 5 atomic layers, each layer containing 9 atoms. This slab is provided in file **INPUT_GEOM**/INPUT_STRUCTURE in xyz format. For the generated models we select the cp2k format. Since the input format is xyz, we also need to include the information of the three lattice vectors of the simulation cell via **&Block_input_cell**. If this block is omitted, the code aborts the execution.

Settings for **&Block_species_components** show that the class of species Pt is defined as crystal. For guidance on how to define this block we refer the user to the detailed explanation of sec. 6.12.1. Since there are only Pt atoms in the **INPUT_GEOM**/INPUT_STRUCTURE file, we define **&Block_input_composition** with 45 atoms for Pt and zero atoms for Sb.

Building models for electrodeposition also require the specification of the cell vector that is perpendicular to the slab surface. This is instructed via directive **normal_vector**, which must be set to c3 in this case. Finally we select a distance cutoff of 2.2 Å.

Execution of this example leads to 10 models, which corresponds to the oxidation and reduction fragments for the first 5 CV cycles. Similar to intercalation, each model is summarized as a table in the OUT_EQCM file. For example, the first reduction gives

```
-----------------------------------------------
        | Total |       Stoichiometry
Species | number |  Model  Target  Difference
-----------------------------------------------
    Pt          45  1.000  1.000    0.000
    Sb          93  2.067  2.074   -0.008
-----------------------------------------------
 ***SUCCESS: target and modelled stoichiometries agree within the error tolerance.
```

Specification of the stoichiometry for species Pt in **&Block_species** is set to 1, which represent the 45 atoms atoms defined in INPUT_STRUCTURE. Thence, the number of atoms of Sb added to the systems divided by 45 gives the stoichiometry of the sample, which is 2.067 for this case. We observe that the amount of Sb atoms deposited during reduction is always larger than the atoms stripped during oxidation, thus leading to an accumulation of atoms with CV cycling. As in the previous examples, this table is printed to file MODEL_SUMMARY together with the model (SAMPLE.cp2k).

Exercise 1: The (332) facet is also observed for Pt surfaces. Using the cif file 332-Pt.cif, compute electrode-position models only for the first oxidation and reduction. Compare the results with the models generated for the (111) facet. Note: the user must copy the cif-input-geom.py from the folder `scripts` to the working directory (see sec. 3.14.8). Python and ASE [76] softwares must be available.

Exercise 2: Repeat exercise 1, this time by generating the atomistic models in .cif format.

6.12.4 Al$_2$O$_3$ coating of LiMn$_2$O$_4$

In section 6.12.2 we provided an example for the generation of pristine sample models for bulk Ni(OH)$_2$. Here, we build models of coating materials over a supporting substrate. To exemplify this capability we consider the (001) facet of LiMn$_2$O$_4$ (LMO), where the content of Li at the surface is reduced, leading to a net stoichiometry of 0.9167 for Li. Input files for this example can be found in folder **tutorial/exercise-12/Al2O3**. The input model in the **INPUT_GEOM**/INPUT_STRUCTURE file is in vasp format.

We consider amorphous Al$_2$O$_3$ as the coating material, which is represented by two molecular subspecies, namely AlO and AlO$_2$ as specified in **&Block_species**. Unless directive *rotate_species* is set to `.False.` (default is `.True.`) the implemented algorithm rotates randomly each subspecies before they are deposited, subject to distance cutoff condition. This randomness in the orientation of the deposited subspecies is convenient to model amorphous coating, such as in the present case.

In **&Block_species**, AlO and AlO$_2$ have the same value for the initial stoichiometry. This is a compulsory requirement as both subspecies are part of the same species Al$_2$O$_3$. From **&Block_Species_Components**, oxygen atoms are tagged differently depending on the species. The same is set for Al atoms. We refer the user to example of sec. 6.12.1 for a detailed explanation of how to set **&Block_Species_Components**. The amount of subspecies to be added to the input model will depend on not only the assigned stoichiometry but also the amount of atoms of the input model. In this case, we notice that the stoichiometry of 1 corresponds to 24 atoms. Thus, a stoichiometry of 0.25 mean that 6 units of AlO and AlO$_2$ will be added to the input model. Since we request the size of the generated model to be 2x2 times in the plane of the surface (see *repeat_input_model*) a total of 24 units of AlO and AlO$_2$ are added to the model, as stated in the last table of file OUT_EQCM. The generated model is printed to the **ATOMISTIC_MODELS/pristine**/SAMPLE.vasp file. Due to the randomness in building the models, the user must generate and compute several of such structures for a rigorous assessment.

Exercise: using the generated structure, set a new SET_EQCM file to deposit Li on top of the Al$_2$O$_3$ coating with stoichiometry of 1, while keeping the size of the model unchanged.

6.13 Building input files for atomistic level simulations

Examples of sec. 6.12 show how to build atomistic models, either for pristine samples or for configurations with stoichiometries that comply with the EQCM data. In this section we provide examples of how to build input files for the atomistic simulation of the generated models. Generated files depend on the format/code selected by the user via directive *output_model_format* and the level of theory to describe the interactions between atoms. At the DFT level, the information required by ALC_EQCM to build simulation files is generally independent of the code, apart from very few specific directives. To generate input files for simulations, we need to define **&block_simulation_settings**, which must contain two blocks: the block related to the level of theory and the block related to the motion of the constituents atoms (**&motion_settings**). Files for simulations can be obtained together with the atomistic models via *model_cycled_sample* (or *model_pristine_sample*) or after having generated atomistic models using the option *hpc_simulation_files* (see sec. 3.14.3). We present and discuss examples for each of the implemented codes.

6.13.1 Files for DFT simulations (VASP)

As a working example to generate input files for VASP simulations, we base on the exercise of sec. 6.12.1 for the intercalation of K+ into Ni(OH)$_2$. The user will find the relevant files in folder **tutorial/exercise-13/vasp**. Here we have two options: i) we can copy the whole **ATOMISTIC_MODELS** and **RESTART** folders generated for the exercise of sec. 6.12.1 or ii) we use the given files DATA_EQCM, SET_EQCM as well as folders **INPUT_GEOM** and **DFT**. If we choose i) the models have been already generated and we need to set option *hpc_simulation_files* for **Analysis** to only generate the input files for DFT simulation. In contrast, if we choose ii) we leave directive **Analysis** as it is, and the atomistic models will be generated again. For this example, we will choose option ii). We refer the reader to sec. 6.16.1 if option i) is selected. In either case, we must add **&block_simulation_settings** in the SET_EQCM file (see next page).

The first two directives within this block show that we aim to build file for geometry relaxation using DFT, for which we must define the block **&DFT_settings** accordingly. This block contains several instructions needed to define input settings for DFT simulations. As developers, we fully empathize if the user feels intimidated when exposed to these directives for the first time. Unfortunately, there is an inherent complexity to set DFT files for simulations that cannot be circumvented.

In this example we ask the GGA level (version PBEsol) for the XC term, as well as spin polarised treatment, since we include magnetization and Hubbard corrections (see sub-block below).

Regarding the convergence of the electronic problem, we set a Gaussian smearing with a width of 0.1 eV, a maximum of 100 scf steps and a criterion of 0.0001 eV for the energy convergence. The energy cutoff is set to 550 eV and the Brillouin zone is sampled with only one k-point (Γ-point). These last settings are pretty much standard for all DFT codes. For this particular example, and because option *output_model_format* had been set to vasp, we need to specify the VASP related directives *precision*, *npar*, *kpar* and *max_l_orbital* (see Table 3 for description). Although *precision*, *npar* and *kpar* are always compulsory for the vasp format, *max_l_orbital* is only compulsory for spin polarised calculations [49].

Because the system under consideration is a transition metal oxide, and the requested XC term is at the GGA level, electronic occupancies must be corrected with the inclusion of Hubbard terms. This is done with sub-block **&hubbard**, for which the user must specify the l-orbital to be corrected, as well as the U and J terms for each atomic tag defined in **&block_species_components**. We only correct the d-orbitals (l_orbital 2) of nickel. The settings for the Hubbard corrections required of the initial magnetization via sub-block **&magnetization**.

Finally, the information for the pseudo-potentials of each tag must be defined in sub-block **&pseudo_potentials**. Each atomic tag has to be assigned a pseudopotential file, which must be located in folder **DFT/PPs**. Given IP restrictions, we cannot distribute these files. Only if the user holds the license for the VASP code and has access to the pseudopotential repository, PBE pseudo-potentials for each of the elements defined in **&block_species_components** must be copied and named as shown in the sub-block **&pseudo_potentials**. It is important to mention that if any other XC type for the pseudo potential files is provided (such as PW91 or LDA), the code will complain and abort the execution. The same applies if the user provide the pseudopotential of a wrong element.

```
&block_simulation_settings
   simulation_type relax_geometry
   theory_level DFT

  ### DFT directives
  &DFT_settings
    XC_level GGA
    XC_version PBEsol
    spin_polarised .True.
    smearing Gaussian
    width_smear 0.1 eV
    scf_steps 100
    energy_cutoff 550.0 eV
    scf_energy_tolerance 0.0001 eV
    kpoints MPack 1 1 1
    precision normal
    npar 4
    kpar 1
    max_l_orbital 2
    &magnetization
     tags    Ni Oh  H+  Ow  Hw  K+
     values 3.0 0.0 0.0 0.0 0.0 0.0
    &end_magnetization
    &hubbard
    tags       Ni  Oh  H+  Ow  Hw  K+
     l_orbital 2   1   0   1   0   0
     U         6.0 0.0 0.0 0.0 0.0 0.0
     J         1.1 0.0 0.0 0.0 0.0 0.0
    &end_hubbard
    &pseudo_potentials
     Ni POTCAR_Ni
     K+ POTCAR_K
     Oh POTCAR_O
     H+ POTCAR_H
     Ow POTCAR_O
     Hw POTCAR_H
    &end_pseudo_potentials
  &end_DFT_settings

  ### Motion of ions
  &motion_settings
   force_tolerance 0.01 eV Angstrom-1
   ion_steps 100
   relax_method CG
   change_cell_volume .True.
   change_cell_shape .True.
  &end_motion_settings

&end_block_simulation_settings
```

Regarding the settings for geometry relaxation (**&motion_settings**), the chosen algorithm is conjugate gradients (GC), while we set the force tolerance of 0.01 eV/Å and the maximum of 100 displacements. Both volume and shape of the supercell are allowed to change during the relaxation.

After execution, files INCAR, KPOINTS, POTCAR and POSCAR are generated at each subfolder, together with the atomistic model SAMPLE.vasp and the MODEL_SUMMARY file. File POSCAR is a copy of SAMPLE.vasp file. A summary of the specifications of **&block_simulation_settings** is described towards the end of the OUT_EQCM file. This is important for reference. We read:

```
=========================================================
Summary of the generated input settings for simulations
=========================================================
=== General settings:
- type of simulation requested:  geometry relaxation
- level of theory for interatomic interactions:  DFT
- neutral supercell
Input files have been generated for execution with the VASP code.
Files INCAR, KPOINTS, POTCAR and POSCAR are located at each sub-folder.
POSCAR is a copy of the generated atomic structure (SAMPLE.vasp).
=== DFT settings:
- spin-polarised calculation
- XC level:  gga
- XC version:  PBE for solids (PBEsol)
[G. I. Csonka et al.  Phys.  Rev.  B, 79:155107, Apr 2009]
- vdW corrections are NOT included
- maximum SCF steps for electronic convergence:  100
- energy cutoff:  550.00 ev
- energy tolerance:  0.1000E-03 ev
- Pulay mixing scheme is set by default
- smearing method:  gaussian
- smearing width:  0.100 ev
- precision:  normal
- only the Gamma point is used for the reciprocal space
- an initial magnetization is assigned to each atomic site
- anisotropic (U-J) Hubbard corrections are imposed to correct for the occupancy
of selected atomic sites
=== Ion-related settings:
- number of ionic steps:  100
- relaxation method:  cg
- force tolerance:  0.01000 ev angstrom-1
```

We observe that the default mixing scheme for the electronic minimization is set to Pulay. However, the mixing can be changed using the *mixing_scheme* directive (see Tables 4 and 5). In addition, ALC_EQCM provides a guidance for the execution of the VASP simulations. We read:

```
********************************************************************
Aspects to take into consideration for simulations with VASP
********************************************************************
Specification of directives for the generated files might need to be adjusted depending on the
system under study.
Implementation of directives has been tested and validated using version 5.4.1 of the code.
The user is responsible to check if the defined settings are compatible with the version of the
VASP code used for the simulations.
```

```
For more information visit https://www.vasp.at/, and read the corresponding section of the
ALC_EQCM manual.
The efficiency in the parallelization can be optimised by:
- adjusting NPAR via ALC_EQCM directive "npar".
- setting directives LREAL and/or LPLANE using the &extra_directives block
In case of problems in the electronic convergence, the user should try:
- changing the mixing parameters (AMIX and BMIX) using the &extra_directives block
- increasing the value of NBANDS via ALC_EQCM directive "bands"
- changing the settings of MAXMIX, NELMIN and NELMDL using the &extra_directives block
- changing default values for MIXPRE and INIMIX using the &extra_directives block (see manual)
- increasing the value of SIGMA via ALC_EQCM directive "width_smear"
- selecting a different option for "mixing_scheme"
If problems persist, try setting "IALGO = 48" using the &extra_directives block
I/O can be controlled using the &extra_directives block with the following
directives (see VASP manual):
- LWAVE, LCHARG, LVTOT and LVHAR (depending on the system and memory requirements)
- LORBIT (for DOS analysis and printing of magnetic moments)
***IMPORTANT*** From the requested settings of &block_simulation_settings, it is RECOMMENDED
to consider:
- changing the settings for AMIX_MAG and/or BMIX_MAG directives using the &extra_directives
block (if electronic convergence fails from the inclusion of Hubbard corrections)
```

We observe that the code recommends using the block **&extra_directives** (see sec. 3.12.3) to add additional information that is not accounted for the implemented ALC_EQCM directives. Definitions inside this block is user's responsibility. To avoid recomputing the atomistic models, the user can add this block to file SET_EQCM and rerun with option *hpc_simulation_files* for *analysis*, as explained in sec. 6.16.1.

Exercise 1: Add the block &extra_directives to specify LORBIT=11 and LWAVE=F. What happens if we add SIGMA=0.2 to the block?

Exercise 2: We ask to set the correct directives to generate input files for MD simulations using the NPT ensemble (300 K and 1 kb).

6.13.2 Files for DFT simulations (CP2K)

The structure of **&block_simulation_settings** is rather general and independent of the choice of the DFT code, apart from few specific directives. For example, VASP uses plane waves as a basis set, and no further specification is needed apart from the information of the **&pseudo_potentials** sub-block. In contrast, CP2K not only needs the sub-block for the pseudopotentials, but also the sub-block **&basis_set**, which must define the type of localised atomic basis set (ABS) adopted for each of the participating atoms (see Table 3). In this section we discuss the settings to build input files for molecular dynamics (MD) simulations of exercise 6.12.4. The user will find the required files in folder **tutorial/exercise-13/cp2k**. Directives for **&block_simulation_settings** can be found in the next page.

Directive *simulation_type* is set to MD. Here, we will not explain all directives as in the previous section. We refer the user to Table 3 for details. Instead, we will focus on directive **vdw** and the information provided in sub-blocks **&pseudo_potentials**, **&basis_set** and **&motion_settings**. In contrast to the VASP, CP2K files for PPs and basis sets are not subjected to license restrictions, for which we have attached the required files in the folder. Alternatively, the user can download the files from the **data** folder of the open-source repository [119].

```
&block_simulation_settings
   simulation_type MD
   theory_level DFT
   ### DFT
   &DFT_settings
    XC_level GGA
    XC_version PBE
    vdw vdW-DF2-B86R
    energy_cutoff 400.0 Ry
    spin_polarised .True.
    smearing Fermi
    SCF_energy_tolerance 0.001 eV
    SCF_steps 100
    bands 200

   &pseudo_potentials
      Li GTH_POTENTIALS
      Mn GTH_POTENTIALS
      O GTH_POTENTIALS
      Al1 GTH_POTENTIALS
      O1 GTH_POTENTIALS
      Al2 GTH_POTENTIALS
      O2 GTH_POTENTIALS
   &end_pseudo_potentials

   &basis_set
      Li DZP
      Mn TZP
      O DZP
      Al1 TZP
      O1 DZP
      Al2 TZP
      O2 DZP
    &end_basis_set

   &end_DFT_settings

   ### Motion of ions
   &motion_settings
      timestep 1.0 fs
      ensemble NVT
      ion_steps 2000
      temperature 300.00 K
      thermostat nose-hoover
      relax_time_thermostat 200.0 fs
   &end_motion_settings

 &end_block_simulation_settings
```

In this example we set the vdW-DF2-B86R option for vdW corrections (see Table 4), which requires of kernel file vdW_kernel_table.dat (located in the **DFT** folder). Sub-block **&pseudo_potentials** must contain

the pseudo potential files for each atomic species. In this case we use the same file GTH_POTENTIALS that contains the specification for all the PPs. This file must be located within folder **DFT/PPs**. In sub-block **&basis_set**, we must specify the type of localised basis set for each atomic species: here we choose DZP for Li, O, O1 and O2 and TZP for Mn, Al1 and Al2. Such basis sets must be defined in file BASIS_SET, which must be located in the **DFT**.

Regarding sub-block **&motion_settings**, we set a time step of 1.0 fs and 2000 ionic steps, while the temperature is 300 K for the NVT ensemble. Finally, we set a Nose-Hoover type of thermostat with a relaxation time of 200 fs.

Successful execution of this example generates several files in folder **ATOMISTIC_MODELS/pristine**. In particular, file input.cp2k contains all the directives for the CP2K simulation of the generated atomistic model (file SAMPLE.cp2k) using files vdW_kernel_table.dat, GTH_POTENTIALS and BASIS_SET. As in the example of the previous section, ALC_EQCM prints a summary of the simulation settings as well as recommendations for the simulation of the generated files with CP2K.

Exercise 1: Repeat the exercise of sec. 6.13.1 with the option cp2k for directive ***output_model_format***. What do you get? Is there any irrelevant or missing specification? Get the relevant input information from the cp2k repository [119] to execute the code successfully. Analyse the information printed in OUT_EQCM.

6.13.3   Files for DFT simulations (CASTEP)

In this section, the user is asked to base on the tutorial exercises of sections 6.13.1 and 6.13.2 and adapt the input directives to generate simulation files for CASTEP simulations. In addition, we suggest the user to play with option EDFT (Ensemble DFT) and ***bands***, and corroborate the settings of file model.cell when module **&pseudo_potentials** is removed from file SET_EQCM. In case the user decides to define **&pseudo_potentials**, PP files compatible with CASTEP must be provided in folder **DFT/PPs**. To obtain tabulated PPs for CASTEP, either norm-conserving (.recpot) or ultra-soft (.usp), the user should look at the **pseudopotentials** folder of the CASTEP repository. Alternatively, the user is referred to the OPIUM website [128] or the FAQ of Ref. [129] for .recpot PPs.

6.13.4   Files for DFT simulations (ONETEP)

The user is now asked to base on the tutorial exercises of sections 6.13.1 and 6.13.2 and adapt the input directives to generate simulation files for ONETEP simulations. In particular, the user is asked to pay attention to the definition of sub-block **&ngwf**. The user must obtain pseudopotentials compatible with ONETEP and copy them to folder **DFT/PPs**. To date, ONETEP users were advised only to use norm-conserving PPs of extension .recpot, which can be obtained via the OPIUM software [128]. See FAQ of Ref. [129] for more details.

6.14   Script files for execution of HPC simulations

Examples of section 6.13 show how to instruct ALC_EQCM to generate atomistic models and input directives for simulations. The generated files are, in principle, sufficient to start the simulation with the selected computational code. Nevertheless, given the size of the modelled systems and/or the amount of CPU power required to compute the DFT problem, simulations must generally be executed in HPC facilities (sec. 3.13). To this purpose one needs of script files that instruct the HPC platform how to make use of the available computational resources.

ALC_EQCM also offers the possibility to generate HPC scripts automatically, based on directives provided by the user. To exemplify this capability, we consider the example of section 6.13.2. The user can find the files for this example in **tutorial/exercise-14**. We assume that we will run our simulations in SCARF, using the partition (queue) "scarf", which is the default. We investigate who was the individual that compiled

CP2K in SCARF, corroborate the version of the code (cp2k files generated by ALC_EQCM will only work for version 8.1.1) and ask which modules were loaded for the compilation. Let us assume 2 modules were loaded: module1 and module2. Let us also assume the executable cp2k.popt is located in the path **/home/vol08/scarf628/codes/**. We also find that the code was compiled using MPI-OpenMP. To build script files for HPC simulations, we add the following block to the SET_EQCM file:

```
&block_hpc_settings
   machine_name   SCARF
   platform       SLURM
   project_name   ALC_EQCM
   job_name       Al2O3
   number_mpi_tasks   24
   number_nodes    2
   CPUs_per_node 24
   parallelism_type   MPI-OpenMP
   threads_per_process    2
   queue          scarf
   executable     /home/vol08/scarf628/codes/cp2k.popt
   mkl     .True.
   &modules
     module1
     module2
   &end_modules
   time_limit    1 12 0
&end_block_hpc_settings
```

SCARF uses the SLURM platform, thence the specification above. Project name is not strictly needed but useful for reference. The jobname for this example is set to Al2O3, which here refers to the coating material but it can be anything. The number of MPI tasks is set to 24. Since the parallelism type for CP2K is MPI-OpenMP, we must specify ***threads_per_process***, which we set to 2. Among all the nodes that belong to the partition (queue) scarf, we set to use only those that contain 24 CPUs. With these settings, a total of 48 MPI tasks will be initiated, with 12 tasks per node (each task uses 2 CPUs from directive ***threads_per_process***). The path to the executable is specified via directive ***executable***. Since this version of CP2K has been compiled with MKL libraries, ***mkl*** is set to .True.. The required modules used for compilation of the code must be listed within sub-block **&modules**. Finally, the user must specify the time limit for the simulation, here set to 1 day and 12 hours.

Execution of ALC_EQCM with the settings generates the script file hpc_script-cp2k.sh. The content of this file are shown below. This file is copied to **ATOMISTIC_MODELS/pristine**, where the atomistic model and the input files for CP2K simulations are also stored. The user must then change to the **ATOMISTIC_MODELS/pristine** folder and execute:

```
sbatch hpc_script-cp2k.sh
```

and the job will hopefully be submitted to the queue. Together with the generation of the hpc_script-cp2k.sh file, a summary of the HPC directives is also printed to file OUT_EQCM. In this particular example, since we have left the option *model_pristine_sample* for ***Analysis***, the atomistic model has also been generated together with the input files for simulation. However, it is not necessarily required to recompute the atomistic model to generate input files for simulation and/or HPC script for job submission. We refer the user to sec. 6.16.1.

```
#!/bin/bash
## Submission script for scarf
#SBATCH -comment=ALC_EQCM # Project name
#SBATCH -job-name=Al2O3 # job name
#SBATCH -o %J.out
#SBATCH -e %J.err
#SBATCH -time=1-12:00:00 # days-hh:mm:ss
#
#SBATCH -partition=scarf # queue (partition)
#SBATCH -ntasks=24
#SBATCH -nodes=2
#SBATCH -ntasks-per-node=12
#SBATCH -cpus-per-task=2

export OMP_NUM_THREADS=2
export MKL_NUM_THREADS=2

## Load required modules
module load module1
module load module2

## Define executable
exec="/home/vol08/scarf628/codes/cp2k.popt"

## Execute job
mpirun -n 24 $exec -i input.cp2k -o output.cp2k
```

Exercise: For the example of sec. 6.13.1 generate a **&block_hpc_settings** from scratch, by setting reasonable specifications for the requested directives. Hint: VASP is compiled in SCARF with MPI-only. Bear in mind the time limit for jobs in SCARF is 7 days.

6.15   Mole changes, mole fluxes and ionic currents along the CV scan

In section 3.9 we presented the capability of ALC_EQCM to compute mole-related quantities along the CV scan. To exemplify such capabilities, in this section we shall consider the EQCM measurements for the intercalation of H^+ and H_2O in $Ti_3C_2T_x$, as reported in Ref. [20]. The EQCM data of the DATA_EQCM file has been kindly provided by Prof. Zifeng Lin. Files for execution are available in the **tutorial/exercise-15** folder. The user should create a separate folder and follow the instructions of sec. 6.1 to print the Δm and current profiles and corroborate the correctness of the EQCM data.

To compute mole changes, mole fluxes and ionic currents, the user must set the *fluxes* option for the **analysis** directive (SET_EQCM file). For the moment, the user is asked to leave the symbol # to comment the line with the **Filter_cutoff** directive. This will compute the relevant quantities from the raw EQCM data without any filtering. Together with the specification for stoichiometric analysis, the user will find the **&Block_species** block withe the definition of the involved chemical species. For a detailed description of this block, the user is referred to sec. 6.10. The host material has a tag Ti3C2Tx, and a stoichiometric composition of $Ti_3C_2O_{0.84}(OH)_{0.06}F_{0.25}$ with a total of 186.83 gr/mol. This species is labelled as fixed. Species H+ and H2O are the dependent variable species. For this example, we have hypothetically set a content of 1.0 for the H2O species. From this molar masses and the setting for the **electrode_mass** directive, ALC_EQCM computes the total moles of the pristine sample before intercalation.

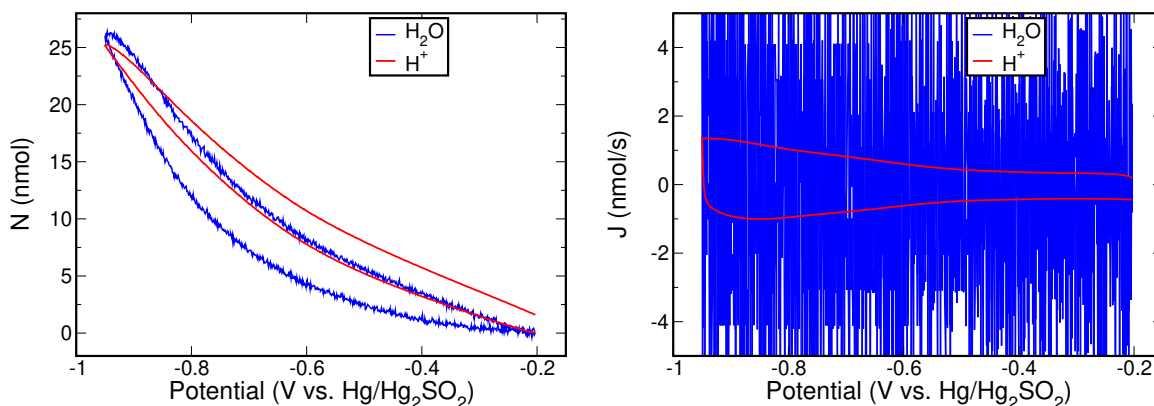Execution of this example generates three files within the **ANALYSIS_EQCM** folder:

Figure 6.15.1: Intercalation of H^+ and H_2O in $Ti_3C_2T_x$ as a function of the potential for the raw EQCM data. Left: Changes in the amount of H_2O moles (blue) exhibit more noise than H^+ (red). Right: such a noise component for the computed amount of H_2O moles leads to an even noiser, practically undiscernible, mole flux (blue) in contrast to computed values for H^+ (red).
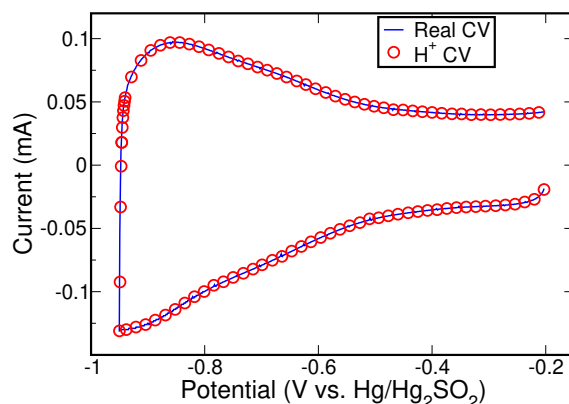

Figure 6.15.2: Comparison between the measured CV (solid blue) and the computed ionic current for H^+ (red circles). Only selected values for the ionic current are displayed for the sake of visualization.

1) CV_MOLES: contains the changes in the number of moles (in nmols) for the changing species H^+ and H_2O as a function of the potential.

2) ION_CURRENT: reports the ionic current (in mA) of those species with non-zero oxidation number (H^+ in this case) as a function of the potential.

3) MOLE_FLUX: contains the computed values for the mole fluxes (in nmols/s) of the species H^+ and H_2O as a function of the potential.

Mole changes are reported in the Fig. 6.15.1(Left). Values show that the content of both species increase/reduce for reduction/oxidation. At the end of the CV cycle (at -0.2 V), we observe a minor residual amount H^+ and practically no H_2O content, which indicates that the reaction is highly reversible. We also notice that the changes in the amount of moles for H_2O (blue) exhibit more noise in contrast to the computed values for H^+ (red). Such a noise arises from the QCM part of the experiment and leads to an undiscernible computed mole flux profile for the H_2O species, as shown by the solid blue lines of Fig. 6.15.1 (Right). In contrast, the mole flux for the H^+ (red) is smooth, and this is related to the smoother signal for the CV current. In fact, H^+ is the only species that contributes to the measured CV current, as demonstrated in the
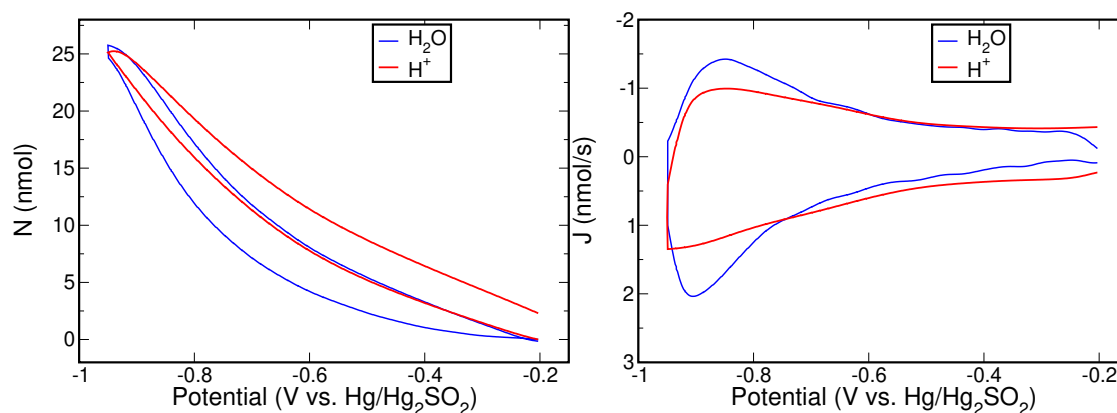
97

Figure 6.15.3: Intercalation of H^+ and H_2O in $Ti_3C_2T_x$ as a function of the potential. In contrast to the results of Fig. 6.15.1, the computed values result from using a low-pass filter of 0.25 Hz to reduce the noise of the EQCM data. Left: Changes in the amount of moles for H_2O (blue) and H^+ (red). Right: mole fluxes for H_2O (blue) and H^+ (red). Note the significant improvement for the mole flux of H_2O in comparison with the values of the Fig. 6.15.1(Right).

reported results of Fig. 6.15.2.

To improve the results for the mole fluxes of Fig. 6.15.1(Right), we instruct the user to remove the symbol # to uncomment the line with the *Filter_cutoff* directive, and execute the code again. By applying a low-pass filter of 0.25 Hz, the noise for the computed amount of H_2O moles is almost elliminated, as shown in Fig. 6.15.3 (Left). Consequently, the computed mole flux is now identifiable (see the right pannel of Fig. Fig. 6.15.3) and follows a similar trend as for the H^+, in agreement with the data processing via fitting of EQCM data [20]. Here, in contrast, the filter does the smoothing automatically. Although highly convenient, we advise using the filtering of data only for visualization purposes. Integrated quantities must be computed using the raw data (i.e. with no filter).

The presented example involves only two variable species. ALC_EQCM also offers the possibility to compute mole-related quantities and ionic currents for reactions involving three or more variable species, as long as the number of constraints is such that stoichiometric solutions are single defined along the CV scan.

Exercise 1: Perform a characterization analysis to evaluate the magnitude of the redox quantities (with and without filtering).

Exercise 2: Change the value of the cutoff frequency for the filter and analyse the change of the computed fluxes.

Exercise 3: Using the files of sec. 6.10.1, change the *stoichiometry* option to *fluxes* to compute mole fluxes and ionic currents for the species Li+ and H+.

Exercise 4: Using the example files of sec 6.10.3, change the *stoichiometry* option to *fluxes* to compute mole fluxes and ionic currents for the species H_2O, K+ and H+. Use the results of the example to set valid constraints.

6.16 Additional capabilities

6.16.1 Simulation files and HPC scripts without recomputing atomistic models

In sections 6.13 and 6.14 we have shown that input files for simulation and script files for HPC execution can be generated together with the atomistic models, as long as the user defines **&block_simulations_settings** and **&block_hpc_settings** in the SET_EQCM file. Nevertheless, it may happen the user decides to generate atomistic models only, omitting the inclusion of these blocks, as we showed in the examples of sec. 6.12. Other scenarios may be that the user requests to generate atomistic models and input files for simulation only (without HPC scripts), or generate atomistic models and HPC scripts only (without input files for simulations). It might also happen that the user performs the atomistic simulation for one of the generated models with the selected computational code, and decides it is convenient add/change/remove a simulation/HPC directive. For all these hypothetical cases, the user needs to amend the corresponding changes and rerun ALC_EQCM. However, if the option of directive *Analysis* is set to either *model_pristine_sample* or *model_cycled_sample*, atomistic models will be generated again.

ALC_EQCM offers the possibility to change simulations and/or HPC settings without the need to rebuild the atomistic models. This is possible by setting the option *hpc_simulation_files* for *Analysis*, which instructs the ALC_EQCM to read file **RESTART**/RECORD_MODELS and generates/updates simulations and HPC files according to the new specification of the blocks. By comparison with the files generated previously (if generated), ALC_EQCM informs the user about the new modifications.

To exemplify this feature, we consider the example of section 6.14 and all the generated files. The user will find the files for this example in folder **tutorial/exercise-16**.

Let us assume that, after inspection, the user realises that vdW corrections are not needed for the production run, the DPZ basis for Al1 and Al2 is sufficient to the purposes of the calculation and the MD simulation must be at 400 K. In addition, the user increases the amount of MPI tasks to 48, changes the number of nodes to 4 accordingly, and sets the time limit to 3 days. These changes are recorded in file SET_EQCM_changed. The user must copy this file to SET_EQCM and rerun. The new OUT_EQCM file contains a summary of which files have changed together with the new specifications for simulation and HPC settings. We see that both hpc_script-cp2k.sh and input-cp2k.sh files changed. In addition, a summary with the new simulation and HPC specifications is printed. This new OUT_EQCM should be interpreted as a complement of the OUT_EQCM_BACKUP file inside the **RESTART** folder, which was generated when the atomistic models were built.

Exercise: For the example of sec. 6.13.1 generate a new set of simulation files by changing the *kpoints* to 2 2 1. In addition, change the energy cutoff to 450 eV and choose the Quasi-Newton method for the geometry relaxation.

6.16.2 Defining the atomic masses for MD simulations

For this section, the user is asked to base on the tutorial exercise of section 6.13.1 and correct the directives to generate input files for MD simulations using:

1) the NPT ensemble (300 K and 1 kb) and

2) intercalated D_2O (deuterated water) instead of normal H_2O

3) sub-block **&masses** to specify the masses f the involved atomic species

The hydrogen named as H+ should be left as normal hydrogen. The user can choose the option for output format of the generated files (except ONETEP). Please refer to sec. 3.14.4 for further guidance.

6.16.3 Enhancing the intercalation of species

Depending on the input structure and the target stoichiometry, there might be situations where the algorithms fails to intercalate species. We refer the user to sec. 3.14.5 for details. The example of folder **tutorial/exercise-17** is meant to model a pristine configuration with formula unit $Ni(OH)_2(H_2O)_{1.2}$ and a cutoff distance of 2.1 Å. Executing this example will give an error (hopefully), as the code cannot accommodate the required amount of extra water molecules within the input structure. We refer the reader to investigate the error message.

If the user aims to keep the size of the simulation cell fixed, directive *distance_cutoff* must be reduced. If, in contrast, the user still aims to keep the separation criteria between atoms to 2.1 Å, the size of the simulation cell must be increased via directive *scale_cell*. We ask the user to try either way or a combination of both, to model the required stoichiometric composition.

6.16.4 Building disordered systems

The main purpose of ALC_EQCM is to generate atomistic models for intercalation and electrodeposition using EQCM experimental data. Nevertheless, the implemented algorithms have been extended to allow building models of disordered systems. We refer the reader to sec. 3.14.2 for details.

We first consider an example of a single Li atom in water solution. Input files can be found in folder **tutorial/exercise-18/Li-solution**. Option *model_disordered_system* for **Analysis** instructs the code to generate a disordered model. As for the option *model_pristine_sample*, the generation of a disordered model does not need of EQCM input data. Specification of the stoichiometry for the model is set in **&block_species**. In this example, we have only one Li atom, which is the "fixed" species, and we set the pristine stoichiometry to 1. We aim to insert 32 water molecules within the box. Species "H2O" is defined as "dependent" and the stoichiometry is set to 32. For the specification of **&block_species_components**, the species Li must be defined as solute, whereas species H2O is set as in sec. 6.12.1. Consistent with file INPUT_STRUCTURE, **&block_input_composition** defines only 1 Li atom, and zero Ow and Hw. Execution of this example generates folder **disordered** that contains the atomistic model.

As a second example of a disordered system, we consider NaCl liquid, which is an example of molten salt. Input files for this example are given in **tutorial/exercise-18/NaCl-liquid**. We choose to build a model with 108 atoms of Na and Cl. As per description in section 3.14.2, we need to set at least one atom as a solute. We choose a Na atom, which is declared as Na1 species, with pristine stoichiometry of 1. Consequently, we define a second species of Na atom, Na2, with stoichiometry 107 (which adds to a total of 108 Na atoms). Likewise, we define species Cl with a stoichiometry of 108. Execution of this example generates the model within folder **disordered**.

Exercise: Generate input files for simulation by specification of **&block_simulation_settings** following the examples of section 6.13.

6.16.5 Including implicit solvent and electrolyte ions within GC-DFT

To exemplify this functionality, we provide the input files of the **tutorial/exercise-19** folder. The user will find the SET_EQCM file with the relevant blocks **&gcdft**, **&solvation** and **&electrolyte** to define settings for CG-DFT, implicit solvent and electrolyte ions, respectively. The user is advised to identify the keywords, make changes according to the description of Secs. 3.14.6 and 3.14.7 and become familiarised with the generated information and input files files. For further details, please refer to Refs. [73] and [74]. As instructed in sec. 3.10.2 and 3.11, the user must set the *both_surfaces* directive to `.True.` to generate correct solvation models.

A Defined settings for CP2K directives

The following table reports only for those CP2K directives (in blue) that have been arbitrarily defined for building input files for the simulation of the generated atomistic models. See Ref. [51] for further details.

Directive	Setting	Notes
PRINT_LEVEL	LOW	Determines a "low" level of generated output information during the run
WALLTIME	300000	Maximum execution time for the run is set to 300000 seconds
PLUS_U_METHOD	MULLIKEN	Mulliken population analysis
&OT		
PRECONDITIONER	FULL_SINGLE_INVERSE	Based on H-eS cholesky inversion.
MINIMIZER	DIIS	Direct inversion in the iterative subspace
N_DIIS	7	Number of history vectors
&END OT		
&OUTER_SCF		
MAX_SCF	10	Maximum number of outer loops
&END OUTER_SCF		
SCF_GUESS	RESTART	Initial guess for the wavefunction
&DIAGONALIZATION		
ALGORITHM	STANDARD	Algorithm to be used for diagonalization
&END DIAGONALIZATION		
&KPOINTS		
SYMMETRY	TRUE	Use symmetry when possible
WAVEFUNCTIONS	REAL	Use real wavefunctions
FULL_GRID	TRUE	Use full non-reduced kpoint grid
&END KPOINTS		
&MGRID		
NGRIDS	4	Number of multi-grids
&END MGRID		
&POISSON		
POISSON_SOLVER	ANALYTIC/PERIODIC	ANALYTIC for slabs/PERIODIC for bulk
&END POISSON		
PRESSURE_TOLERANCE	1.00E+02	Pressure tolerance for cell optimization
&THERMOSTAT		
REGION	MASSIVE	region attached to the thermostat
&NOSE		
LENGTH	3	Length of the Nose-Hoover chain
YOSHIDA	3	Order of the yoshida integrator
MTS	2	multiple timesteps for the NH chain
&END NOSE		
&END THERMOSTAT		
&QS		
METHOD	GPW	This is the default
If *gapw* is set to `.True.`:		
METHOD	GAPW	Gaussian Augmented Plane Waves
ALPHA0_H	10	Exponent for hard compensation charge
EPS_PGF_ORB	1.0E-8	Precision of the overlap matrix elements
LMAXN1	6	max L number
&END QS		

B References

[1] Tzu-Ho Wu, Ivan Scivetti, Jia-Cing Chen, Jeng-An Wang, Gilberto Teobaldi, Chi-Chang Hu, and Lawrence J. Hardwick. Quantitative Resolution of Complex Stoichiometric Changes during Electrochemical Cycling by Density Functional Theory-Assisted Electrochemical Quartz Crystal Microbalance. *ACS Applied Energy Materials*, 3(4):3347–3357, 2020.

[2] S. I. Cordoba-Torresi, C. Gabrielli, A. Hugot-Le Goff, and R. Torresi. Electrochromic Behavior of Nickel Oxide Electrodes: I . Identification of the Colored State Using Quartz Crystal Microbalance. *Journal of The Electrochemical Society*, 138(6):1548–1553, jun 1991.

[3] O.I. Istakova, D.V. Konev, T.O. Medvedeva, O.A. Goncharova, and M.A. Vorotyntsev. Datasets of EQCM-controlled deposition and cycling of thin polypyrrole films in acetonitrile electrolyte solution. *Data in Brief*, 29:105360, 2020.

[4] Simon J. Reeves, Yasir J. Noori, Wenjian Zhang, Gillian Reid, and Philip N. Bartlett. Chloroantimonate electrochemistry in dichloromethane. *Electrochimica Acta*, 354:136692, 2020.

[5] H.K. Pulker. Factors influencing the accuracy of a quartz-crystal-oscillator as a thickness monitor for thin film deposition. *Thin Solid Films*, 1(5):400–402, 1968.

[6] Mark R. Deakin and Owen Melroy. Underpotential metal deposition on gold, monitored in situ with a quartz microbalance. *Journal of Electroanalytical Chemistry and Interfacial Electrochemistry*, 239(1):321–331, 1988.

[7] Mark R. Deakin and Daniel A. Buttry. Electrochemical applications of the quartz crystal microbalance. *Analytical Chemistry*, 61(20):1147A–1154A, 1989.

[8] Stephen J. Martin, James J. Spates, Kurt O. Wessendorf, Thomas W. Schneider, and Robert J. Huber. Resonator/Oscillator Response to Liquid Loading. *Analytical Chemistry*, 69(11):2050–2054, 1997. PMID: 21639245.

[9] Xiaoxi Qiao, Xiangjun Zhang, Yu Tian, and Yonggang Meng. Progresses on the theory and application of quartz crystal microbalance. *Applied Physics Reviews*, 3(3):031106, 2016.

[10] Abdulrahman Alassi, Mohieddine Benammar, and Dan Brett. Quartz Crystal Microbalance Electronic Interfacing Systems: A Review. *Sensors*, 17(12), 2017.

[11] F. R. Lack, G. W. Willard, and I. E. Fair. Some improvements in quartz crystal circuit elements. *Bell System Technical Journal*, 13(3):453–463, 1934.

[12] SRS Standfor Research Systems. *QCM200 Operation and Service Manual*, 6 2018. Rev. 2.5.

[13] Alexandra D. Easley, Ting Ma, Chikaodinaka I. Eneh, Junyeong Yun, Ratul M. Thakur, and Jodie L. Lutkenhaus. A practical guide to quartz crystal microbalance with dissipation monitoring of thin polymer films. *Journal of Polymer Science*, 60(7):1090–1107, 2022.

[14] Michael Rodahl, Fredrik Höök, Claes Fredriksson, Craig A. Keller, Anatol Krozer, Peter Brzezinski, Marina Voinova, and Bengt Kasemo. Simultaneous frequency and dissipation factor qcm measurements of biomolecular adsorption and cell adhesion. *Faraday Discuss.*, 107:229–246, 1997.

[15] Xiaoxi Qiao, Xiangjun Zhang, Yanbao Guo, Shikuan Yang, Yu Tian, and Yonggang Meng. Boundary layer viscosity of cnt-doped liquid crystals: effects of phase behavior. *Rheologica Acta*, 52(10):939–947, Nov 2013.

[16] Minoru Yoshimoto, Shin Tokimura, and Shigeru Kurosawa. Characteristics of the series resonant-frequency shift of a quartz crystal microbalance in electrolyte solutions. *Analyst*, 131:1175–1182, 2006.

[17] Roberto Etchenique and Thomas Buhse. Viscoelasticity in the diffuse electric double layer. *Analyst*, 127:1347–1352, 2002.

[18] Han Zhuang, Pin Lu, Siak Piang Lim, and Heow Pueh Lee. Effects of Interface Slip and Viscoelasticity on the Dynamic Response of Droplet Quartz Crystal Microbalances. *Analytical Chemistry*, 80(19):7347–7353, 2008. PMID: 18767868.

[19] J. Krim, D. H. Solina, and R. Chiarello. Nanotribology of a Kr monolayer: A quartz-crystal microbalance study of atomic-scale friction. *Phys. Rev. Lett.*, 66:181–184, Jan 1991.

[20] Kai Zheng, Yongqiu Xian, and Zifeng Lin. A method for deconvoluting and quantifying the real-time species fluxes and ionic currents using in situ electrochemical quartz crystal microbalance. *Advanced Materials Interfaces*, 9(16):2200112, 2022.

[21] Carlota Ridruejo Arias, Catherine Debiemme-Chouvy, Claude Gabrielli, Christel Laberty-Robert, Alain Pailleret, Hubert Perrot, and Ozlem Sel. New insights into pseudocapacitive charge-storage mechanisms in li-birnessite type mno2 monitored by fast quartz crystal microbalance methods. *The Journal of Physical Chemistry C*, 118(46):26551–26559, 2014.

[22] Daniel A. Buttry and Michael D. Ward. Measurement of interfacial processes at electrode surfaces with the electrochemical quartz crystal microbalance. *Chemical Reviews*, 92(6):1355–1379, 1992.

[23] Gregory Jerkiewicz, Gholamreza Vatankhah, Shin-ichi Tanaka, and Jean Lessard. Discovery of the Potential of Minimum Mass for Platinum Electrodes. *Langmuir*, 27(7):4220–4226, 2011. PMID: 21401076.

[24] Yao Yang, Yin Xiong, Rui Zeng, Xinyao Lu, Mihail Krumov, Xin Huang, Weixuan Xu, Hongsen Wang, Francis J. DiSalvo, Joel. D. Brock, David A. Muller, and Héctor D. Abruña. Operando Methods in Electrocatalysis. *ACS Catalysis*, 11(3):1136–1178, 2021.

[25] Netanel Shpigel, Mikhael D. Levi, and Doron Aurbach. Eqcm-d technique for complex mechanical characterization of energy storage electrodes: Background and practical guide. *Energy Storage Materials*, 21:399–413, 2019.

[26] Schematic Electrochemical Quartz Crystal Microbalance, by Ruoha Zhao. https://en.wikipedia.org/wiki/Electrochemical_quartz_crystal_microbalance#/media/File:EQCM1.jpg.

[27] Schematic side and top views of a QCM sensor, from the Bio Scientific webpage. https://www.biolinscientific.com/blog/how-does-qcm-technology-work.

[28] Netanel Shpigel, Mikhael D. Levi, Sergey Sigalov, Olga Girshevitz, Doron Aurbach, Leonid Daikhin, Piret Pikma, Margus Marandi, Alar Jänes, Enn Lust, Nicolas Jäckel, and Volker Presser. In situ hydrodynamic spectroscopy for structure characterization of porous energy storage electrodes. *Nature Materials*, 15(5):570–575, May 2016.

[29] Mikhael D. Levi, Leonid Daikhin, Doron Aurbach, and Volker Presser. Quartz crystal microbalance with dissipation monitoring (eqcm-d) for in-situ studies of electrodes for supercapacitors and batteries: A mini-review. *Electrochemistry Communications*, 67:16–21, 2016.

[30] Paul G. Kitz, Matthew J. Lacey, Petr Novák, and Erik J. Berg. Operando investigation of the solid electrolyte interphase mechanical and transport properties formed from vinylene carbonate and fluoroethylene carbonate. *Journal of Power Sources*, 477:228567, 2020.

[31] Pierre Lemaire, Thomas Dargon, Daniel Alves Dalla Corte, Ozlem Sel, Hubert Perrot, and Jean-Marie Tarascon. Making advanced electrogravimetry as an affordable analytical tool for battery interface characterization. *Analytical Chemistry*, 92(20):13803–13812, 2020. PMID: 32945170.

[32] Netanel Shpigel, Maria R. Lukatskaya, Sergey Sigalov, Chang E. Ren, Prasant Nayak, Mikhael D. Levi, Leonid Daikhin, Doron Aurbach, and Yury Gogotsi. In situ monitoring of gravimetric and viscoelastic changes in 2d intercalation electrodes. *ACS Energy Letters*, 2(6):1407–1415, 2017.

[33] Pierre Lemaire, Ozlem Sel, Daniel Alves Dalla Corte, Antonella Iadecola, Hubert Perrot, and Jean-Marie Tarascon. Elucidating the Origin of the Electrochemical Capacity in a Proton-Based Battery HxIrO4 via Advanced Electrogravimetry. *ACS Applied Materials & Interfaces*, 12(4):4510–4519, 2020. PMID: 31850732.

[34] Ezzoubair Bendadesse, Anatolii V. Morozov, Artem M. Abakumov, Hubert Perrot, Jean-Marie Tarascon, and Ozlem Sel. Deciphering the double-layer structure and dynamics on a model lixmoo3 interface by advanced electrogravimetric analysis. *ACS Nano Article ASAP*, 2022. PMID: 35984450.

[35] Z. Jusys, H. Massong, and H. Baltruschat. A new approach for simultaneous DEMS and EQCM: Electro-oxidation of adsorbed CO on pt and pt-ru. *Journal of The Electrochemical Society*, 146(3):1093–1098, mar 1999.

[36] C. Gabrielli, M. Keddam, H. Perrot, and R. Torresi. Lithium insertion in wo3 studied by simultaneous measurements of impedance, electrogravimetric and electro-optical transfer functions. *Journal of Electroanalytical Chemistry*, 378(1):85–92, 1994.

[37] D. Aurbach, M. Moshkovich, Y. Cohen, and A. Schechter. The study of surface film formation on noble-metal electrodes in alkyl carbonates/li salt solutions, using simultaneous in situ afm, eqcm, ftir, and eis. *Langmuir*, 15(8):2947–2960, 1999.

[38] Paul G. Kitz, Matthew J. Lacey, Petr Novák, and Erik J. Berg. Operando EQCM-D with Simultaneous in Situ EIS: New Insights into Interphase Formation in Li Ion Batteries. *Analytical Chemistry*, 91(3):2296–2303, 2019.

[39] G. McHale and M. I. Newton. Surface roughness and interfacial slip boundary condition for quartz crystal microbalances. *Journal of Applied Physics*, 95(1):373–380, 2004.

[40] Binyang Du, Ilchat Goubaidoulline, and Diethelm Johannsmann. Effects of laterally heterogeneous slip on the resonance properties of quartz crystals immersed in liquids. *Langmuir*, 20(24):10617–10624, 2004. PMID: 15544393.

[41] Leonid Daikhin and Michael Urbakh. Effect of surface film structure on the quartz crystal microbalance response in liquids. *Langmuir*, 12(26):6354–6360, 1996.

[42] Xiaoxi Qiao, Xiangjun Zhang, Yu Tian, and Yonggang Meng. Modeling the response of a quartz crystal microbalance under nanoscale confinement and slip boundary conditions. *Phys. Chem. Chem. Phys.*, 17:7224–7231, 2015.

[43] Günter Sauerbrey. Verwendung von Schwingquarzen zur Wägung dünner Schichten und zur Mikrowägung. *Zeitschrift für Physik*, 155(2):206–222, Apr 1959.

[44] Jutae Kim and Gregory Jerkiewicz. Influence of the Surface Roughness of Platinum Electrodes on the Calibration of the Electrochemical Quartz-Crystal Nanobalance. *Analytical Chemistry*, 89(14):7462–7469, Jul 2017.

[45] James W. Cooley and John W. Tukey. An algorithm for the machine calculation of complex Fourier series. *Mathematics of Computation*, 9:297–301, 1965.

[46] Rosetta code: FFT. https://rosettacode.org/wiki/Fast_Fourier_transform#Fortran.

[47] M. O. Solaliendres, A. Manzoli, G. R. Salazar-Banda, K. I. B. Eguiluz, S. T. Tanimoto, and S. A. S. Machado. The processes involved in the Se electrodeposition and dissolution on Au electrode: the H2Se formation. *Journal of Solid State Electrochemistry*, 12(6):679–686, Jun 2008.

[48] M. F. Cabral, M. L. Calegaro, and S. A. S. Machado. Nanogravimetric study of lead underpotential deposition on selenium thin films as a semiconductor alloy formation procedure. *RSC Adv.*, 2:2498–2503, 2012.

[49] The VASP Manual. https://www.vasp.at/wiki/index.php/The_VASP_Manual.

[50] CASTEP. http://www.castep.org/.

[51] CP2K Open Source Molecular Dynamics. https://www.cp2k.org.

[52] ONETEP. https://www.onetep.org/.

[53] Albert P. Bartók, Risi Kondor, and Gábor Csányi. On representing chemical environments. *Phys. Rev. B*, 87:184115, May 2013.

[54] Ryosuke Jinnouchi, Jonathan Lahnsteiner, Ferenc Karsai, Georg Kresse, and Menno Bokdam. Phase Transitions of Hybrid Perovskites Simulated by Machine-Learning Force Fields Trained on the Fly with Bayesian Inference. *Phys. Rev. Lett.*, 122:225701, Jun 2019.

[55] Ryosuke Jinnouchi, Ferenc Karsai, and Georg Kresse. On-the-fly machine learning force field generation: Application to melting points. *Phys. Rev. B*, 100:014105, Jul 2019.

[56] Volker L. Deringer, Albert P. Bartók, Noam Bernstein, David M. Wilkins, Michele Ceriotti, and Gábor Csányi. Gaussian Process Regression for Materials and Molecules. *Chemical Reviews*, 121(16):10073–10141, 2021. PMID: 34398616.

[57] B. Hourahine, B. Aradi, V. Blum, F. Bonafé, A. Buccheri, C. Camacho, C. Cevallos, M. Y. Deshaye, T. Dumitrică, A. Dominguez, S. Ehlert, M. Elstner, T. van der Heide, J. Hermann, S. Irle, J. J. Kranz, C. Köhler, T. Kowalczyk, T. Kubař, I. S. Lee, V. Lutsker, R. J. Maurer, S. K. Min, I. Mitchell, C. Negre, T. A. Niehaus, A. M. N. Niklasson, A. J. Page, A. Pecchia, G. Penazzi, M. P. Persson, J. Řezáč, C. G. Sánchez, M. Sternberg, M. Stöhr, F. Stuckenberg, A. Tkatchenko, V. W.-z. Yu, and T. Frauenheim. DFTB+, a software package for efficient approximate density functional theory based atomistic simulations. *The Journal of Chemical Physics*, 152(12):124101, 2020.

[58] V. I. Anisimov, I. V. Solovyev, M. A. Korotin, M. T. Czyżyk, and G. A. Sawatzky. Density-functional theory and NiO photoemission spectra. *Phys. Rev. B*, 48:16929–16934, Dec 1993.

[59] Edward B. Linscott, Daniel J. Cole, Michael C. Payne, and David D. O'Regan. Role of spin in the calculation of Hubbard U and Hund's J parameters from first principles. *Phys. Rev. B*, 98:235157, Dec 2018.

[60] Okan K. Orhan and David D. O'Regan. First-principles Hubbard U and Hund's J corrected approximate density functional theory predicts an accurate fundamental gap in rutile and anatase TiO_2. *Phys. Rev. B*, 101:245137, Jun 2020.

[61] S. L. Dudarev, G. A. Botton, S. Y. Savrasov, C. J. Humphreys, and A. P. Sutton. Electron-energy-loss spectra and the structural stability of nickel oxide: An LSDA+U study. *Phys. Rev. B*, 57:1505–1509, Jan 1998.

[62] SCARF: Scientific Computing Application Resource for Facilities. https://www.scarf.rl.ac.uk/.

[63] https://www.isis.stfc.ac.uk/Pages/Empirical-Potential-Structure-Refinement.aspx.

[64] The ATEN project. https://www.projectaten.com/.

[65] Ravishankar Sundararaman, William A. Goddard, and Tomas A. Arias. Grand canonical electronic density-functional theory: Algorithms and applications to electrochemistry. *The Journal of Chemical Physics*, 146(11):114104, 2017.

[66] Aleksandr V. Marenich, Christopher J. Cramer, and Donald G. Truhlar. Universal Solvation Model Based on Solute Electron Density and on a Continuum Model of the Solvent Defined by the Bulk Dielectric Constant and Atomic Surface Tensions. *The Journal of Physical Chemistry B*, 113(18):6378–6396, 2009. PMID: 19366259.

[67] Oliviero Andreussi, Ismaila Dabo, and Nicola Marzari. Revised self-consistent continuum solvation in electronic-structure calculations. *The Journal of Chemical Physics*, 136(6):064102, 2012.

[68] Alberto Roldan. Frontiers in first principles modelling of electrochemical simulations. *Current Opinion in Electrochemistry*, 10:1–6, 2018.

[69] ONETEP solvent and electrolyte model. https://www.onetep.org/pmwiki/uploads/Main/Documentation/implicit_solvation_v3.pdf.

[70] Damián A. Scherlis, Jean-Luc Fattebert, François Gygi, Matteo Cococcioni, and Nicola Marzari. A unified electrostatic and cavitation model for first-principles molecular dynamics in solution. *The Journal of Chemical Physics*, 124(7):074103, 2006.

[71] Giuseppe Fisicaro, Luigi Genovese, Oliviero Andreussi, Sagarmoy Mandal, Nisanth N. Nair, Nicola Marzari, and Stefan Goedecker. Soft-Sphere Continuum Solvation in Electronic-Structure Calculations. *Journal of Chemical Theory and Computation*, 13(8):3829–3845, 2017. PMID: 28628316.

[72] Dziedzic, Jacek and Fox, Stephen J. and Fox, Thomas and Tautermann, Christofer S. and Skylaris, Chris-Kriton. Large-scale DFT calculations in implicit solvent: A case study on the T4 lysozyme L99A/M102Q protein. *International Journal of Quantum Chemistry*, 113(6):771–785, 2013.

[73] Jacek Dziedzic, Arihant Bhandari, Lucian Anton, Chao Peng, James C. Womack, Marjan Famili, Denis Kramer, and Chris-Kriton Skylaris. Practical Approach to Large-Scale Electronic Structure Calculations in Electrolyte Solutions via Continuum-Embedded Linear-Scaling Density Functional Theory. *The Journal of Physical Chemistry C*, 124(14):7860–7872, 2020.

[74] Arihant Bhandari, Lucian Anton, Jacek Dziedzic, Chao Peng, Denis Kramer, and Chris-Kriton Skylaris. Electronic structure calculations in electrolyte solutions: Methods for neutralization of extended charged interfaces. *The Journal of Chemical Physics*, 153(12):124101, 2020.

[75] James C. Womack, Lucian Anton, Jacek Dziedzic, Phil J. Hasnip, Matt I. J. Probert, and Chris-Kriton Skylaris. DL_MG: A Parallel Multigrid Poisson and Poisson Boltzmann Solver for Electronic Structure Calculations in Vacuum and Solution. *Journal of Chemical Theory and Computation*, 14(3):1412–1432, 2018. PMID: 29447447.

[76] ASE. https://wiki.fysik.dtu.dk/ase/#.

[77] R. Armiento and A. E. Mattsson. Functional designed to include surface effects in self-consistent density functional theory. *Phys. Rev. B*, 72:085108, Aug 2005.

[78] D. M. Ceperley and B. J. Alder. Ground State of the Electron Gas by a Stochastic Method. *Phys. Rev. Lett.*, 45:566–569, Aug 1980.

[79] John P. Perdew, J. A. Chevary, S. H. Vosko, Koblar A. Jackson, Mark R. Pederson, D. J. Singh, and Carlos Fiolhais. Atoms, molecules, solids, and surfaces: Applications of the generalized gradient approximation for exchange and correlation. *Phys. Rev. B*, 46:6671–6687, Sep 1992.

[80] J. P. Perdew and Alex Zunger. Self-interaction correction to density-functional approximations for many-electron systems. *Phys. Rev. B*, 23:5048–5079, May 1981.

[81] John P. Perdew, Kieron Burke, and Matthias Ernzerhof. Generalized Gradient Approximation Made Simple. *Phys. Rev. Lett.*, 77:3865–3868, Oct 1996.

[82] E. Wigner. On the Interaction of Electrons in Metals. *Phys. Rev.*, 46:1002–1011, Dec 1934.

[83] B. Hammer, L. B. Hansen, and J. K. Nørskov. Improved adsorption energetics within density-functional theory using revised Perdew-Burke-Ernzerhof functionals. *Phys. Rev. B*, 59:7413–7421, Mar 1999.

[84] S. H. Vosko, L. Wilk, and M. Nusair. Accurate spin-dependent electron liquid correlation energies for local spin density calculations: a critical analysis. *Canadian Journal of Physics*, 58(8):1200–1211, 1980.

[85] S. H. Vosko and L. Wilk. Influence of an improved local-spin-density correlation-energy functional on the cohesive energy of alkali metals. *Phys. Rev. B*, 22:3812–3815, Oct 1980.

[86] Yingkai Zhang and Weitao Yang. Comment on "Generalized Gradient Approximation Made Simple". *Phys. Rev. Lett.*, 80:890–890, Jan 1998.

[87] John P. Perdew and Yue Wang. Accurate and simple analytic representation of the electron-gas correlation energy. *Phys. Rev. B*, 45:13244–13249, Jun 1992.

[88] Gábor I. Csonka, John P. Perdew, Adrienn Ruzsinszky, Pier H. T. Philipsen, Sébastien Lebègue, Joachim Paier, Oleg A. Vydrov, and János G. Ángyán. Assessing the performance of recent density functionals for bulk solids. *Phys. Rev. B*, 79:155107, Apr 2009.

[89] L Hedin and B I Lundqvist. Explicit local exchange-correlation potentials. *Journal of Physics C: Solid State Physics*, 4(14):2064–2083, oct 1971.

[90] A. D. Becke. Density-functional exchange-energy approximation with correct asymptotic behavior. *Phys. Rev. A*, 38:3098–3100, Sep 1988.

[91] Chengteh Lee, Weitao Yang, and Robert G. Parr. Development of the Colle-Salvetti correlation-energy formula into a functional of the electron density. *Phys. Rev. B*, 37:785–789, Jan 1988.

[92] S. Goedecker, M. Teter, and J. Hutter. Separable dual-space Gaussian pseudopotentials. *Phys. Rev. B*, 54:1703–1710, Jul 1996.

[93] Xin Xu and William A. Goddard. The X3LYP extended density functional for accurate descriptions of nonbond interactions, spin states, and thermochemical properties. *Proceedings of the National Academy of Sciences*, 101(9):2673–2677, 2004.

[94] Zhigang Wu and R. E. Cohen. More accurate generalized gradient approximation for solids. *Phys. Rev. B*, 73:235116, Jun 2006.

[95] Stefan Grimme. Semiempirical GGA-type density functional constructed with a long-range dispersion correction. *Journal of Computational Chemistry*, 27(15):1787–1799, 2006.

[96] Stefan Grimme, Jens Antony, Stephan Ehrlich, and Helge Krieg. A consistent and accurate ab initio parametrization of density functional dispersion correction (DFT-D) for the 94 elements H-Pu. *The Journal of Chemical Physics*, 132(15):154104, 2010.

[97] Stefan Grimme, Stephan Ehrlich, and Lars Goerigk. Effect of the damping function in dispersion corrected density functional theory. *Journal of Computational Chemistry*, 32(7):1456–1465, 2011.

[98] Alexandre Tkatchenko and Matthias Scheffler. Accurate Molecular Van Der Waals Interactions from Ground-State Electron Density and Free-Atom Reference Data. *Phys. Rev. Lett.*, 102:073005, Feb 2009.

[99] Tomáš Bučko, Sébastien Lebègue, Já nos G. Ángyán, and Jürgen Hafner. Extending the applicability of the Tkatchenko-Scheffler dispersion correction via iterative Hirshfeld partitioning. *The Journal of Chemical Physics*, 141(3):034114, 2014.

[100] Alexandre Tkatchenko, Robert A. DiStasio, Roberto Car, and Matthias Scheffler. Accurate and Efficient Method for Many-Body van der Waals Interactions. *Phys. Rev. Lett.*, 108:236402, Jun 2012.

[101] Alberto Ambrosetti, Anthony M. Reilly, Robert A. Distasio, and Alexandre Tkatchenko. Long-range correlation energy calculated from coupled atomic response functions. *The Journal of Chemical Physics*, 140(18):18A508, 2014.

[102] Stephan N. Steinmann and Clemence Corminboeuf. Comprehensive Benchmarking of a Density-Dependent Dispersion Correction. *Journal of Chemical Theory and Computation*, 7(11):3567–3577, 2011.

[103] F. Ortmann, F. Bechstedt, and W. G. Schmidt. Semiempirical van der Waals correction to the density functional description of solids and molecular structures. *Phys. Rev. B*, 73:205101, May 2006.

[104] Petr Jurečka, Jiří Černý, Pavel Hobza, and Dennis R. Salahub. Density functional theory augmented with an empirical dispersion term. Interaction energies and geometries of 80 noncovalent complexes compared with ab initio quantum mechanics calculations. *Journal of Computational Chemistry*, 28(2):555–569, 2007.

[105] M. Dion, H. Rydberg, E. Schröder, D. C. Langreth, and B. I. Lundqvist. Van der Waals Density Functional for General Geometries. *Phys. Rev. Lett.*, 92:246401, Jun 2004.

[106] Jiří Klimeš, David R Bowler, and Angelos Michaelides. Chemical accuracy for the van der Waals density functional. *Journal of Physics: Condensed Matter*, 22(2):022201, dec 2009.

[107] Jiří Klimeš, David R. Bowler, and Angelos Michaelides. Van der Waals density functionals applied to solids. *Phys. Rev. B*, 83:195131, May 2011.

[108] Kyuho Lee, Éamonn D. Murray, Lingzhu Kong, Bengt I. Lundqvist, and David C. Langreth. Higher-accuracy van der Waals density functional. *Phys. Rev. B*, 82:081101, Aug 2010.

[109] Ikutaro Hamada. van der Waals density functional made accurate. *Phys. Rev. B*, 89:121103, Mar 2014.

[110] Haowei Peng, Zeng-Hui Yang, John P. Perdew, and Jianwei Sun. Versatile van der Waals Density Functional Based on a Meta-Generalized Gradient Approximation. *Phys. Rev. X*, 6:041005, Oct 2016.

[111] Oleg A. Vydrov and Troy Van Voorhis. Nonlocal van der Waals density functional: The simpler the better. *The Journal of Chemical Physics*, 133(24):244103, 2010.

[112] Riccardo Sabatini, Tommaso Gorni, and Stefano de Gironcoli. Nonlocal van der Waals density functional made simple and efficient. *Phys. Rev. B*, 87:041108, Jan 2013.

[113] Torbjörn Björkman. van der Waals density functional for solids. *Phys. Rev. B*, 86:165109, Oct 2012.

[114] G. P. Kerker. Efficient iteration scheme for self-consistent pseudopotential calculations. *Phys. Rev. B*, 23:3082–3084, Mar 1981.

[115] C. G. Broyden. A class of methods for solving nonlinear simultaneous equations. *Math. Comp.*, 19:577–593, 1965.

[116] D. D. Johnson. Modified broyden's method for accelerating convergence in self-consistent calculations. *Phys. Rev. B*, 38:12807–12813, Dec 1988.

[117] Péter Pulay. Convergence acceleration of iterative sequences. the case of scf iteration. *Chemical Physics Letters*, 73(2):393–398, 1980.

[118] H Akai and P H Dederichs. A simple improved iteration scheme for electronic structure calculations. *Journal of Physics C: Solid State Physics*, 18(12):2455–2460, apr 1985.

[119] CP2K repository in GitHub. https://github.com/cp2k/cp2k/tree/master/data.

[120] Gnuplot. http://www.gnuplot.info/.

[121] Grace. https://plasma-gate.weizmann.ac.il/Grace/.

[122] STFC GitLab: Open source software to collaborate on code. https://gitlab.stfc.ac.uk/.

[123] CMake. https://cmake.org/.

[124] Michael Metcalf, John Reid, and Malcolm Cohen. *Modern Fortran Explained*. Oxford University Press, Inc., USA, 4th edition, 2011.

[125] S. Reeves and P. Bartlett, August 2020. Internal correspondence.

[126] E. Lust, A. Jänes, V. Sammelselg, P. Miidla, and K. Lust. Surface roughness of bismuth, antimony and cadmium electrodes. *Electrochimica Acta*, 44(2):373 – 383, 1998.

[127] Structure of the input coordinates consistent with the vasp code. https://www.vasp.at/wiki/index.php/POSCAR.

[128] OPIUM - pseudopotential generation project. http://opium.sourceforge.net/.

[129] ONETEP FAQ: where can i obtain reliable pseudopotentials? https://www.onetep.org/Main/FAQ.