

## OPEN ACCESS

# Python for Electrochemistry: A Free and All-In-One Toolset

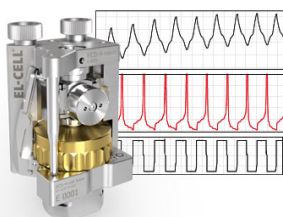
To cite this article: Weiran Zheng 2023 *ECS Adv.* 2 040502

View the [article online](#) for updates and enhancements.

## You may also like

- [Deposition of VO<sub>x</sub> Films by Reactive Sputtering and its Properties](#)  
Xiaoying Wei, Kailiang Zhang, Wang Fang et al.
- [A new calibration method for charm jet identification validated with proton-proton collision events at s = 13 TeV](#)  
The CMS collaboration, Armen Tumasyan, Wolfgang Adam et al.
- [Search for Multimessenger Sources of Gravitational Waves and High-energy Neutrinos with Advanced LIGO during Its First Observing Run, ANTARES, and IceCube](#)  
A. Albert, M. André, M. Anghinolfi et al.

## Measure the Electrode Expansion in the Nanometer Range. Discover the new ECD-4-nano!

  
electrochemical test equipment


- Battery Test Cell for Dilatometric Analysis (Expansion of Electrodes)
- Capacitive Displacement Sensor (Range 250  $\mu\text{m}$ , Resolution  $\leq 5$  nm)
- Detect Thickness Changes of the Individual Electrode or the Full Cell.

[www.el-cell.com](http://www.el-cell.com) +49 40 79012-734 [sales@el-cell.com](mailto:sales@el-cell.com)





# Python for Electrochemistry: A Free and All-In-One Toolset

Weiran Zheng<sup>z</sup> 

Department of Chemistry, Guangdong Technion-Israel Institute of Technology, Shantou 515063, People's Republic of China  
Guangdong Provincial Key Laboratory of Materials and Technologies for Energy Conversion, Guangdong Technion-Israel Institute of Technology, Shantou 515063, People's Republic of China  
Technion-Israel Institute of Technology, Haifa 32000, Israel

Python, an open-source, interpreted programming language, has emerged as a transformative force within the scientific community, captivating researchers with its rich ecosystem of packages and syntax that prioritizes readability and simplicity. In the rapidly evolving field of electrochemistry, where the analysis of complex data sets, custom analysis routines, and theoretical simulations are indispensable, Python's capabilities have garnered significant attention. This review serves as a general introduction to the utilization of Python in electrochemistry, focusing on beginners who are new to programming concepts.

© 2023 The Author(s). Published on behalf of The Electrochemical Society by IOP Publishing Limited. This is an open access article distributed under the terms of the Creative Commons Attribution 4.0 License (CC BY, <http://creativecommons.org/licenses/by/4.0/>), which permits unrestricted reuse of the work in any medium, provided the original work is properly cited. [DOI: 10.1149/2754-2734/acf0b]



Manuscript submitted September 2, 2023; revised manuscript received September 27, 2023. Published October 10, 2023.

Electrochemistry has gained significant momentum across various research sectors in recent years, from energy conversion and storage to organic synthesis and environmental analysis. Driven by advancements in experimental techniques and increasing parameters, electrochemical experiments become more intricate, often requiring combining multiple techniques (e.g., in situ experiments) while producing increasingly large datasets. Electrochemists face the challenge of deciphering complex electrochemical responses (e.g., current, potential, resistance, capacitance, etc.), understanding electrode processes, and extracting crucial parameters that govern electrochemical behaviors. To conduct such studies, one must analyze multidimensional datasets, develop customized analysis procedures, simulate and verify theoretical models, and synchronize multiple instruments with potentiostats.

The need for standardized and robust data analysis tools and methodologies has become paramount. In response to this growing demand, advanced data analysis techniques, such as statistical analysis, machine learning, and computational modeling, have become indispensable.<sup>1-3</sup> Learning and applying modern toolkits that integrate these analytical approaches can arm electrochemists with the ability to standardize data analysis, unravel the complexities of electrochemical systems, facilitate groundbreaking discoveries, and push the boundaries of electrochemical research.

Python, an open-source, general-purpose programming language, is one of the best platforms for scientific computation due to its ease of use, readability, and vast collection of scientific modules and libraries.<sup>4</sup> It was conceived in the late 1980s by Guido van Rossum with the intention of creating a language that prioritized code readability and expressiveness, enabling programmers to achieve more with fewer lines of code compared to languages like C++ or Java. Since its first release as Python 1.0 in 1994, Python has evolved continuously, with the current version at the time of writing being Python 3.11. Following its release and regular updates, Python has been gaining popularity in the scientific and engineering communities due to its advantages over traditional tools, primarily commercial software (free vs paid; open-source vs close-source). Moreover, its affordability and accessibility lead to widespread adoption in university education across various disciplines, equipping the younger chemists with basic programming skills.<sup>5,6</sup> Yet, despite its popularity and the vast and free availability of many scripts/modules/packages contributed by electrochemical groups worldwide, the power of Python as an all-in-one electrochemical toolset for data analysis and problem-solving still needs to be discovered by many researchers.

In this short review, we outline the general concepts and workflow related to using Python as a toolset for electrochemical research, from its advantages to the code structure and the electrochemical resources, aiming to provide an introduction to beginners who are new to programming.

## Why use Python for Electrochemical Analysis?

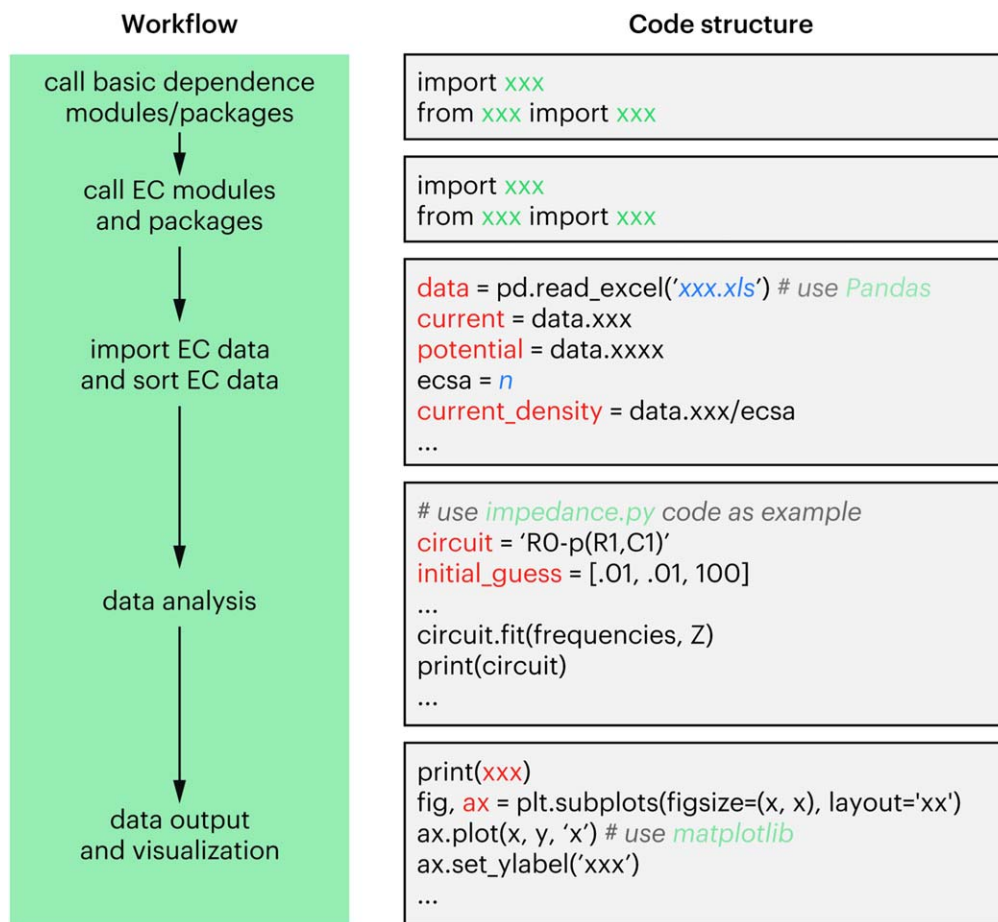
The most significant advantage of Python is code simplicity and readability, meaning it can be accessed and understood by researchers without a background in computer science. One can embrace programming readily after training for a few hours, unlike other languages like C++ or Java. Its syntax is designed to be readable and straightforward, simplifying code development, comprehension, and maintenance. Python's object-oriented nature allows researchers to organize their codes around data and their interactions while providing flexibility in variable typing. It supports multiple programming paradigms, including functional, procedural, and object-oriented styles. The ability to execute code line-by-line facilitates efficient debugging and interactive data analysis.<sup>7</sup>

The second advantage of Python is its rich ecosystem and active community. Apart from the essential functions, the extensive standard library of Python further enriches its capabilities, offering modules for file input/output (I/O), system calls, internet protocols, and graphical user interfaces. Researchers can leverage Python's scientific libraries (i.e., Python packages), such as "NumPy" for numerical operations, "SciPy" for scientific computing, "Matplotlib" for plotting, and "Pandas" for data analysis, to tackle the diverse computational tasks encountered in research. Moreover, users can easily develop and share their scripts in the community to provide more specific functionalities.

The advantages of Python extend beyond its accessibility and library ecosystem. As an open-source language, Python is freely available and continuously improved by a global community of developers, making it more affordable, flexible, and up-to-date than many commercial software packages.

The incorporation of Python programming into electrochemical research yields a multitude of specific advantages. A prime benefit is that researchers can augment their productivity by automating monotonous tasks. For instance, Python can be used to script routine procedures such as titration curve plotting or cyclic voltammetry analysis, which would otherwise consume a significant amount of time if done manually. Python's ability to handle and process voluminous datasets is another considerable benefit. With Python's powerful libraries like "NumPy" and "Pandas," researchers can analyze gigabytes of data from electrochemical impedance spectroscopy (EIS) or chronoamperometry experiments in a fraction of the time it would take using less efficient methods. Furthermore, Python

<sup>z</sup>E-mail: [weiran.zheng@gtit.edu.cn](mailto:weiran.zheng@gtit.edu.cn)



**Figure 1.** Workflow and corresponding code example of Python programming for electrochemical applications. The package “*impedance.py*” is an example in the data analysis.

can perform intricate computations, such as modeling complex electrochemical systems using libraries like “*SciPy*.” This liberates researchers to spend more time on crucial tasks, like experimental work and interpreting data for advancements in batteries, fuel cells, and bioelectrochemistry. Additionally, Python encourages reproducibility in research. This is indispensable in the scientific community, as it bolsters the credibility and verifiability of results. Other scientists can employ the same Python scripts for data pre-processing, analysis, and visualization to replicate and verify results effectively. For example, a Jupyter Notebook containing all the Python code and graphical output from an experiment on lithium-ion battery performance could be shared among researchers worldwide, enabling them to reproduce the study and validate the findings.

To summarize, Python expands the horizons of electrochemical research by empowering general researchers to develop customized analysis routines tailored to their specific needs, surpassing the limitations of commercial software. The language also enables the simulation of theoretical models, providing insights that may not be directly observable in experiments alone.

### General Structure of a Python Script for Electrochemistry

First, one must have a Python coding environment to write and run codes. Anaconda is one of the most popular platforms for scientific coding that runs locally. A full tutorial on how to install and configure Anaconda can be found on its website (<https://www.anaconda.com>), which we will not repeat here. Often, a more user-friendly front-end coding environment is installed based on Anaconda. In my view, using the “*JupyterLab*” as the development environment (i.e., coding environment) is strongly suggested to

ensure informative and sharable scripts with others because of its notebook-like style. Another emerging choice is Colab (<https://colab.google>) developed by Google, which provides a hosted service that requires little to no effort in installing individual environments and packages, making it easier to collaborate and share codes. After configuring the environment, one can start with the coding (i.e., Python scripts). A Python script for electrochemical analysis typically follows a structured approach divided into several essential parts, as shown in Fig. 1 with example codes.

**Call basic modules and packages.**—The script begins by importing basic Python modules and packages that are often dependencies of other packages. They are often hosted on GitHub together with detailed instructions and examples. These often include “*NumPy*” for numerical operations, “*Pandas*” for data handling, and “*Matplotlib*” for plotting. For example, you might see code like “*import numpy as np*” or “*import pandas as pd*” at the start of the script. These modules and packages provide the fundamental tools for data manipulation, mathematical operations, and plotting. Some of the most important ones are listed as basic dependencies in Fig. 2.

**Call electrochemical modules and packages.**—The script imports specific electrochemistry-focused packages. These could include modules like “*pints*” for time-series analysis, “*impedance*” for EIS analysis, “*electrochem*” for cyclic voltammetry (CV) data visualization, and “*lmfit*” for curve fitting. For example, “*from impedance.models.circuits import Randles*” would allow access to the Randles circuit model. These specialized packages offer more nuanced tools tailored to electrochemical research needs.

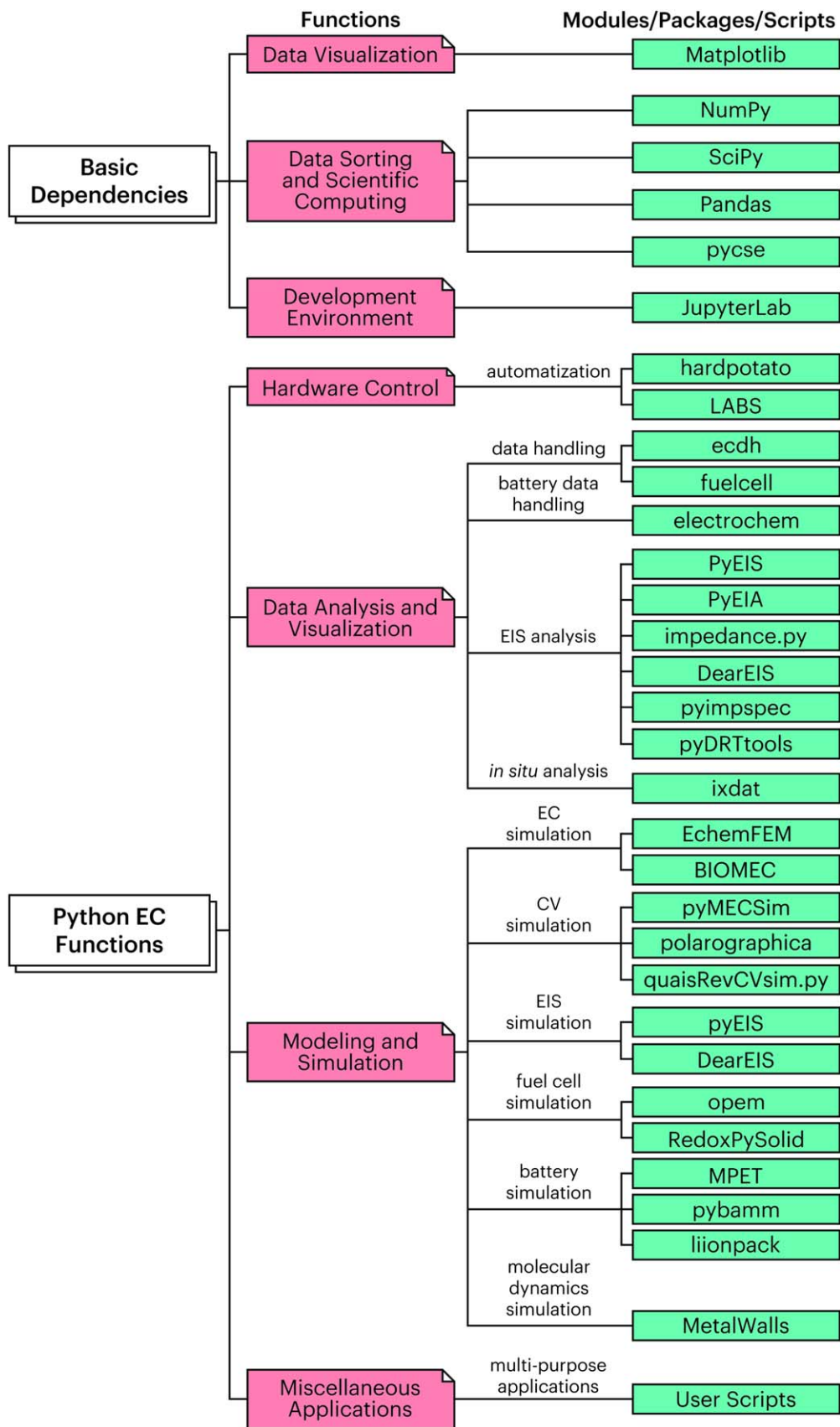


Figure 2. Selected Python modules, packages, and scripts for electrochemical applications. EC: electrochemistry.

**Data import and sorting.**—Following the setup, the script moves to data import and sorting. Typically, this involves using “Pandas” or “NumPy” functions to load data from files, often CSV or Excel, into Python’s data structures (like DataFrames or arrays). The data is then sorted or cleaned as necessary. For example, one might remove outliers, normalize current with electrochemical surface area (ECSA), or sort it based on specific criteria (e.g., current range, potential range). This part of the script sets up the data for subsequent analysis.

**Data analysis.**—The core of the Python script lies in the data analysis section. This section uses the previously imported modules to process the sorted data, applying statistical methods, fitting curves, generating models, or whatever else is needed for the specific analysis. For instance, you might use “lmfit” to fit an experimental current-voltage curve to a theoretical model or “ixdat” to correlate electrochemical data with mass spectroscopy data for in situ analysis.

**Data output and visualization.**—The final part of the script focuses on output and visualization. Here, the processed data is visualized using plotting libraries like “Matplotlib” or “Seaborn,” and the plots are customized to be clear and informative. Also, the processed data and analysis results are often written back to files for future reference using “Pandas” or “NumPy” functions.

Overall, a Python script for electrochemical data analysis provides a comprehensive approach to handling, processing, and visualizing data. Each part is crucial and contributes to the overall analysis, making Python a powerful tool for electrochemical research.

### Python Modules/Packages for Electrochemistry

In Python programming, modules are used to manage and reuse certain codes effectively. These modules come in the form of Python scripts that have a .py extension and contain functions, classes, and variables that can be called upon when needed. The Python language comes equipped with popular core modules such as *os*, *re*, and *sys*. Additionally, developers have the ability to create their own modules or use third-party modules available on the Python Package Index (PyPI) (<https://pypi.org>) through a package manager like *pip*. By using modules, developers can avoid rewriting common code and promote code reuse. Importing modules also helps to prevent naming conflicts and keeps the project organized. For larger sets of modules, developers can pack them together as a package (or library), which is simply a collection of modules organized within a directory.

Regarding modules and packages for electrochemistry, they can be classified into two main categories. First, there are the basic dependencies which are crucial for supporting basic functionalities and other modules/packages. Then there are the specific electrochemical tools that provide more specialized functions. To help clarify these categories, Fig. 2 lists some popular modules and packages, along with their relationships and general functions. All modules and packages can be found via PyPI with detailed instructions and examples. Users are strongly advised to read these documents for a comprehensive understanding. These modules and packages, developed by various electrochemical groups, provide essential tools for hardware control, data analysis and visualization, modeling and simulation, and other applications in daily research. Herein, only a brief glance is provided for general awareness.

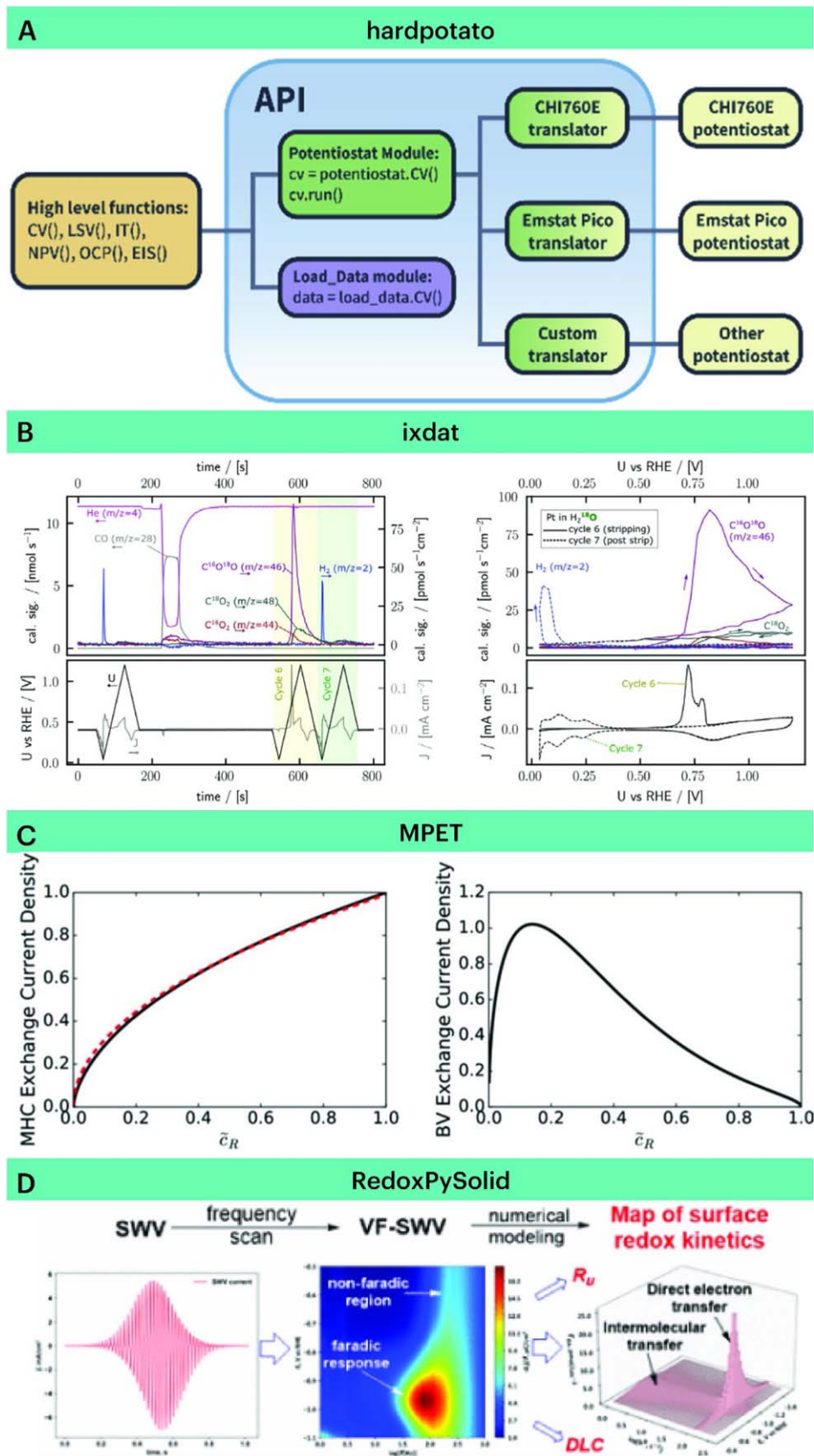
**Hardware control and automatization.**—A few examples include the “hardpotato” package (Python API to control programmable potentiostats, Fig. 3A)<sup>8</sup> and the “LABS” package (Laboratory Automation and Batch Scheduling).<sup>9</sup> The former is developed to standardize commands across different potentiostat models, enabling automated experiments on any instrument, from data acquisition to analysis and simulation. The latter is used for integrating multiple

lab devices and enabling users to orchestrate automated electrochemical synthesis with the ability of parameter input and system monitoring.

**Data analysis and visualization.**—Analyzing and plotting data are the major topics for Python, and most packages are developed to suit specific analytical methodologies. A few general current-voltage data handling packages are the “*ecdh*”<sup>10</sup> (for direct and simple handling of electrochemical data, from sorting to visualization), the “*fuelcell*”<sup>11,12</sup> (for streamlining electrochemical data analysis), and the “*electrochem*”<sup>13</sup> (for reading raw electrochemical data and generating plots and tables for battery researchers) packages. The packages for analyzing EIS data contribute a significant number to the PyPI and a few handy ones include the “*PyEIS*,”<sup>14</sup> “*PyEIA*,”<sup>15</sup> “*impedance.py*,”<sup>16,17</sup> “*DearEIS*,”<sup>18</sup> “*pyimpspec*,”<sup>19</sup> “*pymultipleis*,”<sup>20</sup> and so on, sharing similar basic functions (e.g., fitting and simulation) with different focuses on functionalities (e.g., distribution of relaxation times (DRT) analysis). A special package worth noting is “*ixdat*” devoted to in situ experimental data analysis (Fig. 3B).<sup>21</sup> Developed as a collaboration project by the interfacial electrochemistry group at Imperial College London, the package enables convenient correlation between results from operando/in situ analysis. So far, it supports the combination of electrochemistry, mass spectroscopy, and spectroelectrochemistry data, with X-ray diffraction and X-ray photoelectron spectroscopy in the development stage.

**Electrochemical modeling and simulation.**—Modeling and simulation play a vital role in understanding and predicting complex electrochemical phenomena, aiding in the design and optimization of electrochemical systems. It represents another significant application of Python in the realm of electrochemistry research. With the help of specific packages, researchers can create sophisticated models and run simulations to understand and predict electrochemical systems’ behavior.<sup>22</sup> Two notable examples of such packages are “*opem*” (open-source proton exchange membrane (PEM) fuel cell simulation tool)<sup>23,24</sup> and “*MPET*” (multi-physics electrochemical tool, Fig. 3C).<sup>25,26</sup> “*opem*” is designed for the modeling and simulation of PEM fuel cells, offering a comprehensive set of functions and modules to simulate various aspects, including performance analysis, polarization curve fitting, and system optimization. It incorporates advanced mathematical models considering mass transport, electrochemical reactions, and heat transfer. “*MPET*” package focuses on multi-physics modeling and simulation of electrochemical systems, providing a framework for coupling and simulating different physical phenomena, such as electrochemical reactions, heat transfer, fluid flow, and species transport in batteries, fuel cells, and electrolyzers. With *MPET*, researchers can investigate the performance, durability, and optimization of various electrochemical devices, facilitating advancements in energy storage and conversion technologies. Notably, the same group recently released Hybrid-MPET for hybrid electrodes.<sup>27</sup>

Other useful packages include “*pyMECSim*” (a Python cyclic voltammetry simulation software for advanced analysis of electrochemical kinetics, e.g., kinetic zone diagram and Langmuir isotherm),<sup>28</sup> “*Supycap*” (an analysis tool for supercapacitors),<sup>29</sup> “*MetalWalls*” (a molecular dynamics software dedicated to electrochemical system simulation),<sup>30,31</sup> “*RedoxPySolid*” (a simulation tool for statistical model of heterogeneous electrochemistry, Fig. 3D),<sup>32,33</sup> “*pybamm*” (Python battery mathematical modeling tool),<sup>34,35</sup> “*liionpack*” (a battery pack modeling tool based on *pybamm*),<sup>36,37</sup> and many more. These packages empower researchers to simulate and analyze complex electrochemical processes, facilitating the understanding of fundamental mechanisms and aiding in the design and optimization of electrochemical devices. As Python continues to evolve, it is expected that more specialized packages will emerge, further expanding the capabilities of electrochemical modeling and simulation and driving advancements in the field of electrochemistry.



**Figure 3.** Published examples of selected packages demonstrating their functions in electrochemistry research. (A) “hardpotato” package for unified control of commercial potentiostats; Reproduced with permission from Ref. 8 Copyright 2022, the authors. (B) “ixdat” package for correlation between mass spectroscopic data and electrochemical signal; Reproduced with permission from Ref. 45 Copyright 2021, Elsevier Ltd. (C) “MPET” package for modeling and verification of kinetic aspects of porous electrodes within a battery system. Reproduced with permission from Ref. 25 Copyright 2017, the authors. (D) “RedoxPySolid” package for data analysis and modeling from variable-frequency square wave voltammetry; Reproduced with permission from Ref. 33 Copyright 2021, American Chemical Society.

### User-Shared Python Scripts and Applications

Many functions are not necessarily published as modules or packages in the PyPI by the authors but as user-generated scripts (or

notebooks if “JupyterLab” is used) and Python applications. These scripts and applications, normally publicly shared as supporting information of papers or as GitHub/GitLab repositories, can be

reused by other researchers to reproduce the literature results and analyze their own data with ease. Beginners can import and preprocess raw electrochemical data, apply filters or corrections, and generate visual representations such as cyclic voltammograms, impedance spectra, Tafel plots, etc.

A few notable scripts are mentioned here. Wang et al. provided a collection of electrochemistry simulation scripts with a graphical user interface (GUI) to visualize the dynamic behaviors on the electrode through simulations of related equations, including electrical double layer (EDL) capacity, electrochemical polarization, potential step, current step, pulse voltammetry, cyclic voltammetry, and electrochemical impedance.<sup>38</sup> Thomas Roy et al. developed a software called EchemFEM to provide finite element solvers to simulate electrochemical transport, such as diffusion, advection, and migration.<sup>39,40</sup> The Monash electrochemical group released BIOMECS (Bayesian inference and optimization for the Monash electrochemical simulator) for calculating parameters using mathematical optimization and Bayesian inference.<sup>41,42</sup> The pyDRTtools developed by Francesco Ciucci et al. offers an intuitive Python application to run DRT analysis from EIS data.<sup>43,44</sup>

Overall, these scripts and applications, similar to modules and packages, provide more flexibility in implementing preferred data analysis algorithms and customizing visualizations to highlight specific features or trends of electrochemical data.

### Challenges and Future Perspectives

Using Python for electrochemical research offers numerous benefits, such as its extensive scientific modules and packages, versatility, and ease of use. However, there are certain challenges. The most significant is the steep learning curve for beginners in Python programming, requiring time and effort to gain proficiency in the language and its relevant libraries (some of them are not well documented). Moreover, often for the same applications, especially for electrochemical simulations and modeling, one may find multiple modules and packages doing a similar job, even producing conflicting results. It requires a more careful examination of the codes and variables to ensure that the functionalities suit the right purpose.

Despite the challenges, Python's increasing popularity as a scientific programming language is driving the development of specialized tools and packages for electrochemistry. With the field's progress, we can anticipate more user-friendly and standardized packages tailored to researchers' specific needs. Moreover, Python's open-source nature fosters collaboration and knowledge sharing among researchers, allowing them to collectively develop and enhance electrochemical analysis tools. Another future perspective lies in integrating Python with emerging technologies such as machine learning and artificial intelligence. The compatibility of Python with popular machine learning libraries, including Scikit-learn and TensorFlow, offers new possibilities for data-driven approaches in electrochemical research. By leveraging machine learning algorithms, patterns can be extracted, properties can be predicted, and experimental designs can be optimized, leading to accelerated discoveries and advancements in electrochemical systems.

Overall, Python stands as a powerful tool in the realm of electrochemical research, empowering researchers to advance their studies through its simplicity, flexibility, and wide-ranging capabilities. Whether researchers seek to understand the kinetic aspects of electrochemical reactions, or batch process a large dataset and visualization, Python offers a wealth of resources to enhance research capabilities and drive scientific progress. By embracing Python, the field of electrochemistry can unlock its true potential in unraveling complex phenomena and addressing critical challenges.

### Acknowledgments

Weiran Zheng is grateful for the support of the Guangdong Basic and Applied Basic Research Foundation (Grant Number: 2023A1515012277) and the Guangdong Technion-Israel Institute of Technology (Grant Number: ST2200002).

### ORCID

Weiran Zheng  <https://orcid.org/0000-0002-9915-6982>

### References

1. A. M. Bond, J. Zhang, L. Gundry, and G. F. Kennedy, *Curr. Opin. Electrochem.*, **34**, 101009 (2022).
2. H. Chen, E. Kätelhön, and R. G. Compton, *Curr. Opin. Electrochem.*, **38**, 101214 (2023).
3. J. M. Diaz-Cruz, N. Serrano, C. Perez-Rafols, C. Arino, and M. Esteban, *J. Solid State Electrochem.*, **24**, 2653 (2020).
4. J. Sundnes, *Introduction to Scientific Programming with Python* (Springer, Berlin) (2020).
5. K. A. Tanemura, D. Sierra-Costa, and K. M. Merz Jr., *Python for Chemists* (American Chemical Society, Washington) (2022).
6. A. R. McDonald and J. A. Nash, *Teaching Programming across the Chemistry Curriculum* (American Chemical Society, Washington) (2021).
7. C. J. Weiss, *J. Chem. Educ.*, **94**, 592 (2017).
8. O. Rodriguez, M. A. Pence, and J. Rodriguez-Lopez, *Anal. Chem.*, **95**, 4840 (2023).
9. M. M. Hielscher, M. Dorr, J. Schneider, and S. R. Waldvogel, *Chem. Asian J.*, **18**, e202300380 (2023).
10. A. M. Raniseti and M. Rødne, ElectroChemical Data Handler., (<https://github.com/amundmr/ecdh>) (2023).
11. S. Garg, A. Smith, and A. Limaye, Fuelcell: Data processing for fuel cell and electrolyzer experiments., (<https://github.com/samayarg/fuelcell>) (2021).
12. S. Garg, J. Fornaciari, A. Weber, and N. Danilovic, *J. Open Source Softw.*, **6**, 2940 (2021).
13. Vincent Wu, Arbin Electrochemical Tools., (<https://github.com/vince-wu/electrochem>) (2020).
14. K. Knudsen, PyEIS: A Python-based Electrochemical Impedance Spectroscopy simulator and analyzer., (<https://github.com/kbkknudsen/PyEIS>) (2019).
15. J. L. Vishart, J. Castillo-Leon, and W. E. Svendsen, *SoftwareX*, **15**, 100720 (2021).
16. M. Murbach, B. Gerwe, N. Dawson-Elli, and T. L.-k., *J. Open Source Softw.*, **5**, 2349 (2020).
17. M. Murbach, B. Gerwe, N. Dawson-Elli, and T. L.-k., impedance.py: A Python package for working with electrochemical impedance data., (<https://github.com/ECSHackWeek/impedance.py>) (2023).
18. V. Yrjänä, DearEIS: A GUI program for analyzing, simulating, and visualizing impedance spectra., (<https://github.com/vyrjana/DearEIS>) (2023).
19. V. Yrjänä, pyimpespec: A package for parsing, validating, analyzing, and simulating impedance spectra., (<https://github.com/vyrjana/pyimpespec>) (2023).
20. R. Chukwupymultipleis: A library for fitting a sequence of electrochemical impedance spectra., (<https://github.com/richinex/pymultipleis>) (2023).
21. S. B. Scott, ixdat: The In-situ Experimental Data Tool., (<https://github.com/ixdat/ixdat>) (2023).
22. E. L. Molel and T. F. Fuller, *J. Electrochem. Soc.* (2023).
23. S. Haghghi, K. Askari, S. Hamidi, and M. Mahdi, "Rahimi." *J. Open Source Softw.*, **3**, 676 (2018).
24. S. Haghghi, OPEM, (Open Source PEM Fuel Cell Simulation Tool). (<https://github.com/ecs/im/opem>) (2021).
25. R. B. Smith and M. Z. Bazant, *J. Electrochem. Soc.*, **164**, E3291 (2017).
26. D. Cogswell, MPET – Multiphase Porous Electrode Theory., (<https://github.com/TRI-AMDD/mpet>) (2023).
27. Q. Liang and M. Z. Bazant, *J. Electrochem. Soc.*, **170**, 093510 (2023).
28. K. VaddipyMECSim: A Python wrapper for MECSim., (<https://github.com/kiranvad/pyMECSim>) (2021).
29. Y. Chen, Supycap: Analysis tool for the CC and CV experiment of supercapacitors., (<https://github.com/AdaYuanChen/Supycap>) (2021).
30. A. Marin-Lafleche et al., *J. Open Source Softw.*, **5**, 2373 (2020).
31. M. Salanne, MetalWalls, (MW). (<https://gitlab.com/ampere2/metalwalls>) (2021).
32. A. Marianov, RedoxPySolid: Statistical model of heterogeneous electrochemistry., (<https://github.com/Aleksei-Marianov/RedoxPySolid>) (2022).
33. A. N. Marianov, A. S. Kochubei, T. Roman, O. J. Conquest, C. Stampfl, and Y. Jiang, *Anal. Chem.*, **93**, 10175 (2021).
34. V. Sulzer, F. B. Planella, P. Agarwal, S. Chopra, and A. Khetarpal, PyBaMM, (Python Battery Mathematical Modelling). (<https://github.com/pybamm-team/PyBaMM>) (2023).
35. V. Sulzer, S. G. Marquis, R. Timms, M. Robinson, and S. J. Chapman, *J. Open Res. Softw.*, **9**, 14 (2021).
36. T. Tranter et al., *J. Open Source Softw.*, **7**, 4051 (2022).
37. T. Tranter et al., (2023), liionpack: A battery pack simulation tool that uses the PyBaMM framework. (<https://github.com/pybamm-team/liionpack>).
38. X. Wang and Z. Wang, *J. Chem. Educ.*, **99**, 752 (2021).
39. T. Roy, V. Ehlinger, and N. Govindarajan, Finite Element Method for Electrochemical Transport, (EchemFEM). (<https://github.com/LLNL/echemfem>) (2023).
40. T. Roy, J. Andrej, and V. A. Beck, *J. Comput. Phys.*, **475**, 111859 (2023).
41. L. Gundry, G. Kennedy, J. Keith, M. Robinson, D. Gavaghan, A. M. Bond, and J. Zhang, *ChemElectroChem*, **8**, 2238 (2021).
42. L. Gundry, Bayesian Inference and Optimisation for the Monash Electrochemical Simulator., (<https://github.com/lukegun/BIOEMEC>) (2023).
43. B. Py, A. Maradesa, and F. Ciucci, *Electrochim. Acta*, **439**, 141688 (2023).
44. F. Ciucci, pyDRTtools: An intuitive python GUI to compute the DRT., (<https://github.com/ciuccilab/pyDRTtools>) (2023).
45. S. B. Scott, J. Kibsgaard, P. C. K. Vesborg, and I. Chorkendorff, *Electrochim. Acta*, **374**, 137842 (2021).