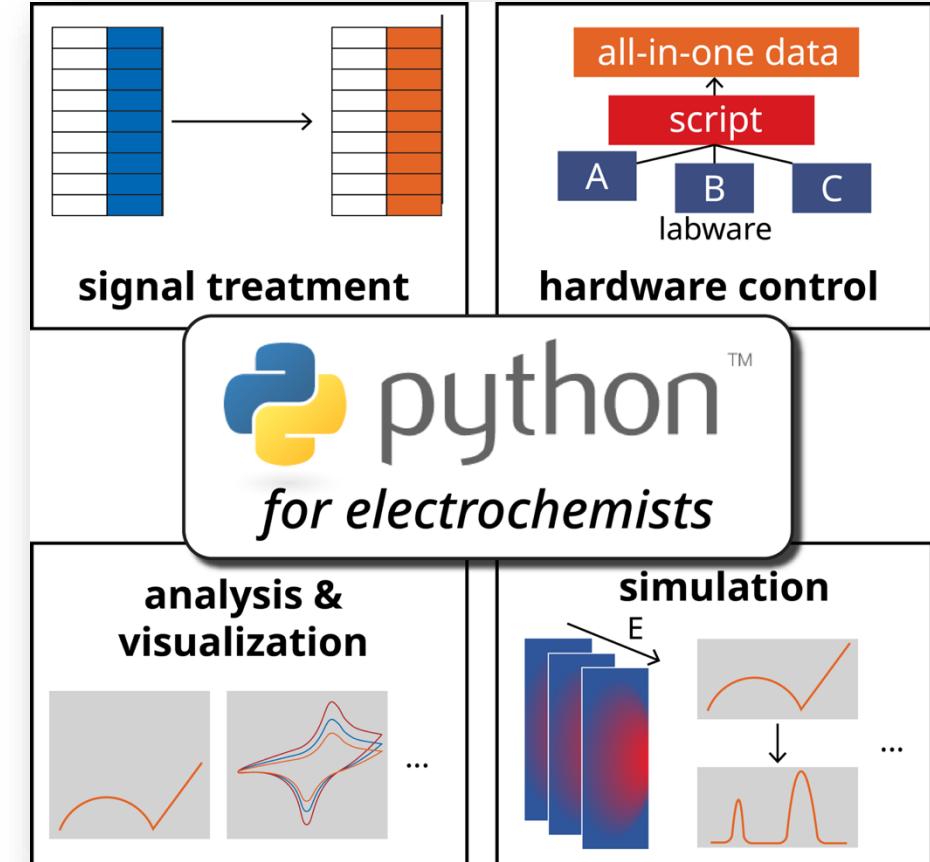




Introducing Python for Electrochemistry Research

Weiran Zheng 郑蔚然
weiran.zheng@gtiit.edu.cn

Guangdong Technion-Israel Institute of Technology, Shantou, China
Technion-Israel Institute of Technology, Haifa, Israel



0. Target of this webinar

This webinar is for:

- Any electrochemists who **do not know Python** but are interested in its capability;
- Any electrochemists who spend lots of time **struggling with repeated data processing, plotting, and format adjustment**;
- Any electrochemists who **do not want to involve lots of commercial software in daily research**;

This webinar is not for:

- Anyone who is already using Python in daily research
- Who wants to know advanced Python applications (deep learning, etc.)

0. Overview of the webinar

1. Introduction

- What is Python, and what can we benefit from using Python

2. Python basics

- Coding environment
- Packages

3. Electrochemical Data Handling

4. Applications of Python in Electrochemical Research (w. case demonstration)

- Data analysis visualization
- Simulation and modeling
- Hardware control

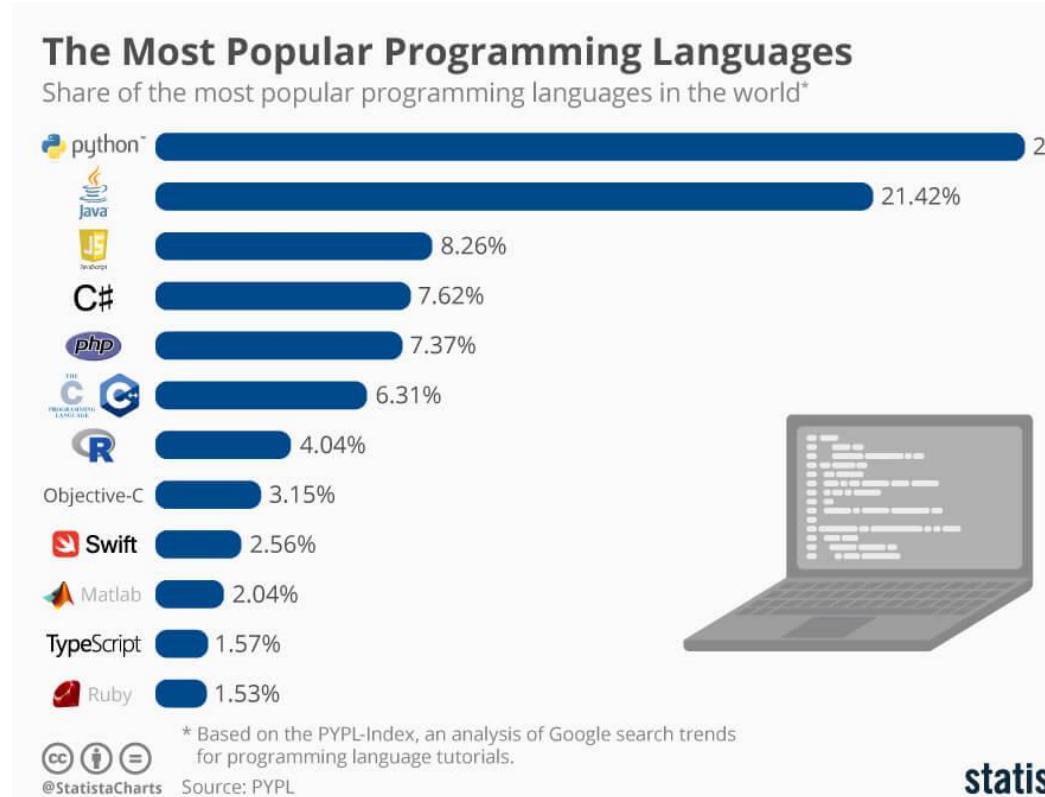
5. Q and A



This Webinar is mostly based on:
W. Zheng, *ECS Advances*, 2023, 2, 040502

1.1 Introduction

- Python: **high-level programming language known for its simplicity and readability.**
 - web development, **data analysis**, **scientific computing**, artificial intelligence, and more;
 - object-oriented, functional, and procedural programming;
 - a rich standard library along with **numerous third-party libraries (packages)**.



Other languages:

R (statistical computing and graphics)
Matlab (computational language)



1.2 Python vs. Matlab

Python

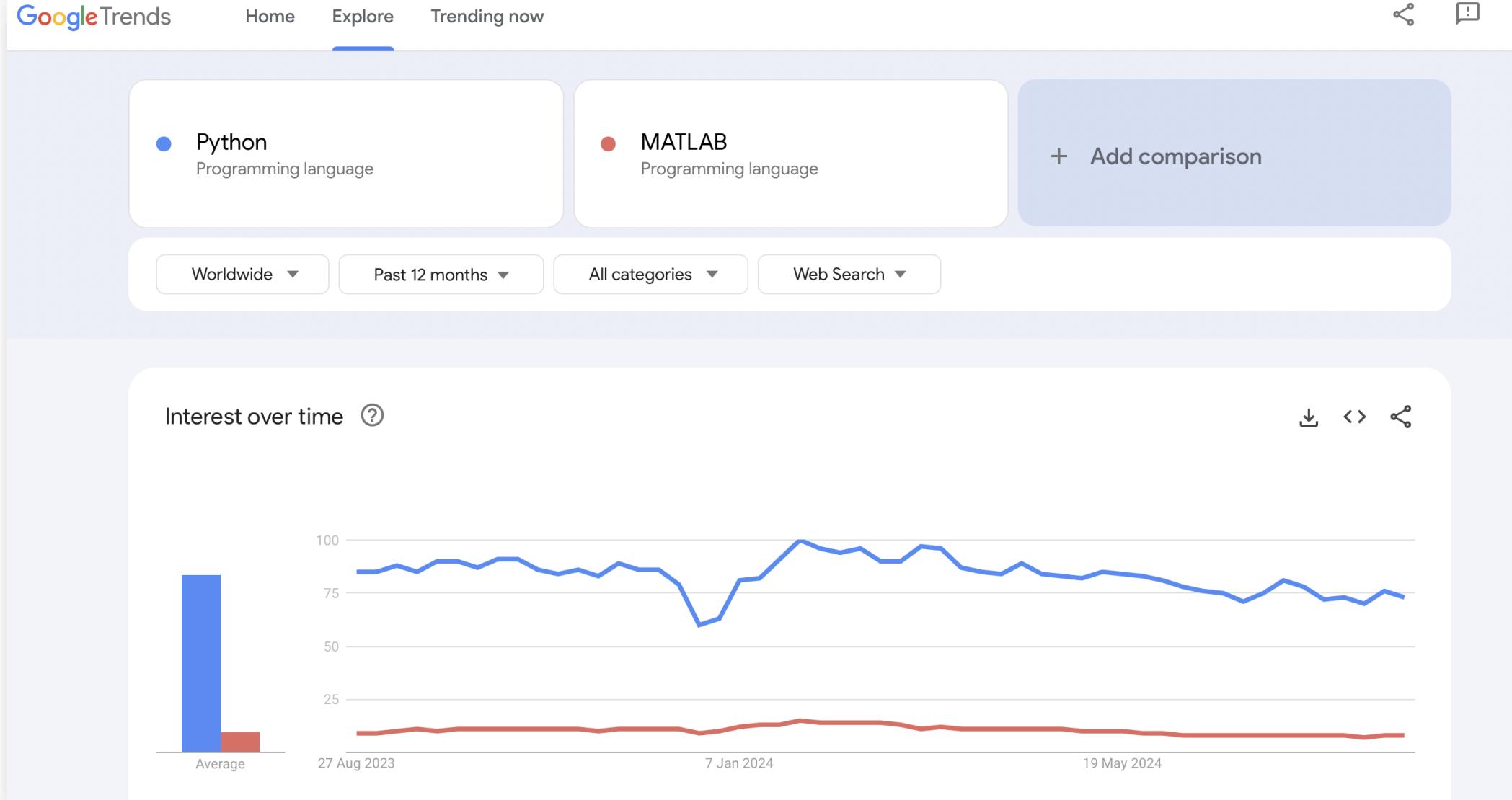
- **General-Purpose Language (all-in-one)**: Versatile and used in web development, data analysis, machine learning, etc.
- **Open Source (code is available and free)**: Free to use with numerous libraries like NumPy, pandas, and TensorFlow.
- **Community Support (continuously developing packages/scripts)**: Large community with extensive resources and tutorials.
- **Syntax (understandable if reading other's code)**: Known for its readable and clean syntax.

MATLAB

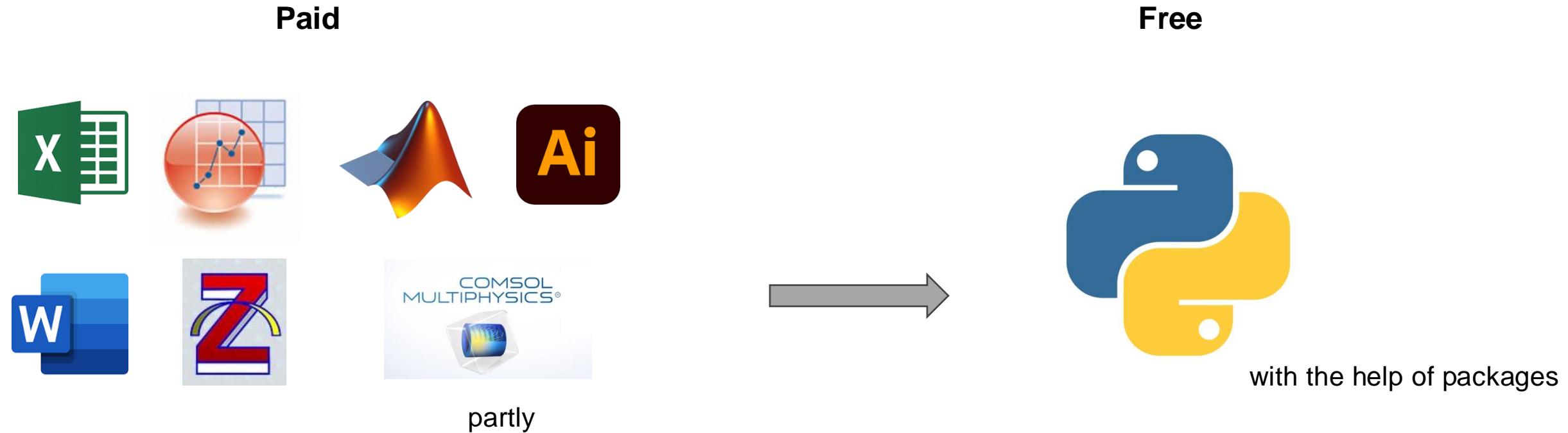
- **Specialized Tool**: Primarily used for numerical computing and simulations.
- **Commercial Software**: Requires a license, with specialized toolboxes for different applications.
- **Built-In Functions**: Extensive built-in functions for matrix operations and data visualization.
- **User Interface**: Offers a user-friendly interface with integrated plotting capabilities.

Key Differences

- **Cost**: Python is free; MATLAB is commercial.
- **Versatility**: Python is more versatile for different applications.
- **Study curve**: MATLAB is easier for beginners focused on mathematical and engineering tasks due to its specialized toolboxes.



1.3 All-in-one toolset for electrochemistry



- License can be expensive
- A complicated process from data to results
- Hard to share/collaborate
- Difficult for PI to manage

- Repeated data import/format conversion/sorting avoided
- Analyzing and plotting in one place
- Data summary in a notebook style

2.1 Python basics – Where to write the code

A screenshot of a Google search results page for "python coding environment". The results are displayed in a grid format with three columns. Each result includes the logo, name, developer, and license information. The results are as follows:

Logo	Name	Developer	License
	PyCharm	Proprietary software	
	Thonny	MIT License	
	Atom	MIT License	
	PyScripter	MIT License	
	Komodo IDE	Mozilla Public License	
	KDevelop	GNU General Public License	
	IDLE	Python Software Foundation...	
	PyDev	Eclipse Public License	
	eric	GNU General Public License	
	Eclipse	Common Public License	
	NetBeans	Freeware	
	Komodo Edit	GNU General Public License	
	Spyder	MIT License	
	Visual Studio	Proprietary software	
	Wing IDE	Proprietary software	
	Python Tools for Visual St...	Apache License	
	Geany	GNU General Public License	

There are lots of choices...

Integrated development environment (IDE) - PyCharm

The screenshot shows the PyCharm IDE interface with the file `paginator.py` open. The code editor displays the implementation of the `Paginator` class. A tooltip is visible over the line `return self._count`, showing the documentation and several suggestions for completing the method name. The suggestions include `_get_count`, `_get_num_pages`, `_get_page`, and `_get_page_range`. The code editor also shows the `Structure` tool window on the left, listing all methods and attributes of the `Paginator` class.

```
def page(self, num):
    """
    Create a page object for the given page number.
    """
    Returns
    =====
    num = self._validate_number(self, num)
    bot = num // self.per_page
    top = bot + self._orphan_threshold
    if top < num:
        top += 1
    if not self.allow_empty_first_page:
        if num < top:
            raise EmptyPage(num)
    if num < bot:
        num = bot
    if num > top:
        num = top
    return self._get_page(*args, **kwargs)

    def _get_page(*args, **kwargs):
        This method is deprecated. It stands for an old API.
        Instead, use the paginator.get_page() method.
        This method is only here to support Django's internal
        PageNumberInput widget.
        return self._get_page(*args, **kwargs)

    def _get_num_pages(self):
        """
        Returns the total number of pages.
        """
        if self._count is None:
            try:
                self._count = self.object_list.count()
            except (AttributeError, TypeError):
                # AttributeError if object_list has no count() method.
                # TypeError if object_list.count() requires arguments
                # (i.e. is of type list).
                self._count = len(self.object_list)
        return self._count

    count = property(_get_count)

    def _get_num_pages(self):
        """
        Returns the total number of pages.
        """
        if self._count is None:
            try:
                self._count = self.object_list.count()
            except (AttributeError, TypeError):
                # AttributeError if object_list has no count() method.
                # TypeError if object_list.count() requires arguments
                # (i.e. is of type list).
                self._count = len(self.object_list)
        return self._count

    count = property(_get_count)
```

but sometimes, they look scary for beginners...

2.2 Recommendations – option 1

1. install Anaconda:



<https://www.anaconda.com>

2. Coding in Jupyterlab:



<https://jupyter.org>

- 1. Comprehensive Package Management**
- 2. Environment Management:** Allows **isolated environments** for different project dependencies.
- 3. Large Collection of Packages:** Comes with over **1,500 pre-installed packages**, including popular ones like NumPy, pandas, and Matplotlib.
- 4. Simplified Setup:** Provides a **straightforward installation** process, making it easy to set up Python environments.
- 5. Cross-Platform Support:** Works on Windows, **macOS**, and Linux, ensuring compatibility across different systems.
- 6. Integrated Tools:** Includes **Jupyterlab** and Spyder for interactive computing and development.

2.2 Recommendations – option 2

Jupyterlab desktop

- Simplest way
- Notebook style
- App-like experience

Windows (10, 11)	Mac (macOS 10.15+)	Linux
x64 Installer	arm64 Installer (Apple silicon)	Snap Store [recommended]
	x64 Installer (Intel chip)	.deb x64 Installer (Debian, Ubuntu)
		.rpm x64 Installer (Red Hat, Fedora, SUSE)

1. Interactive Computing: Allows for interactive data exploration and visualization.

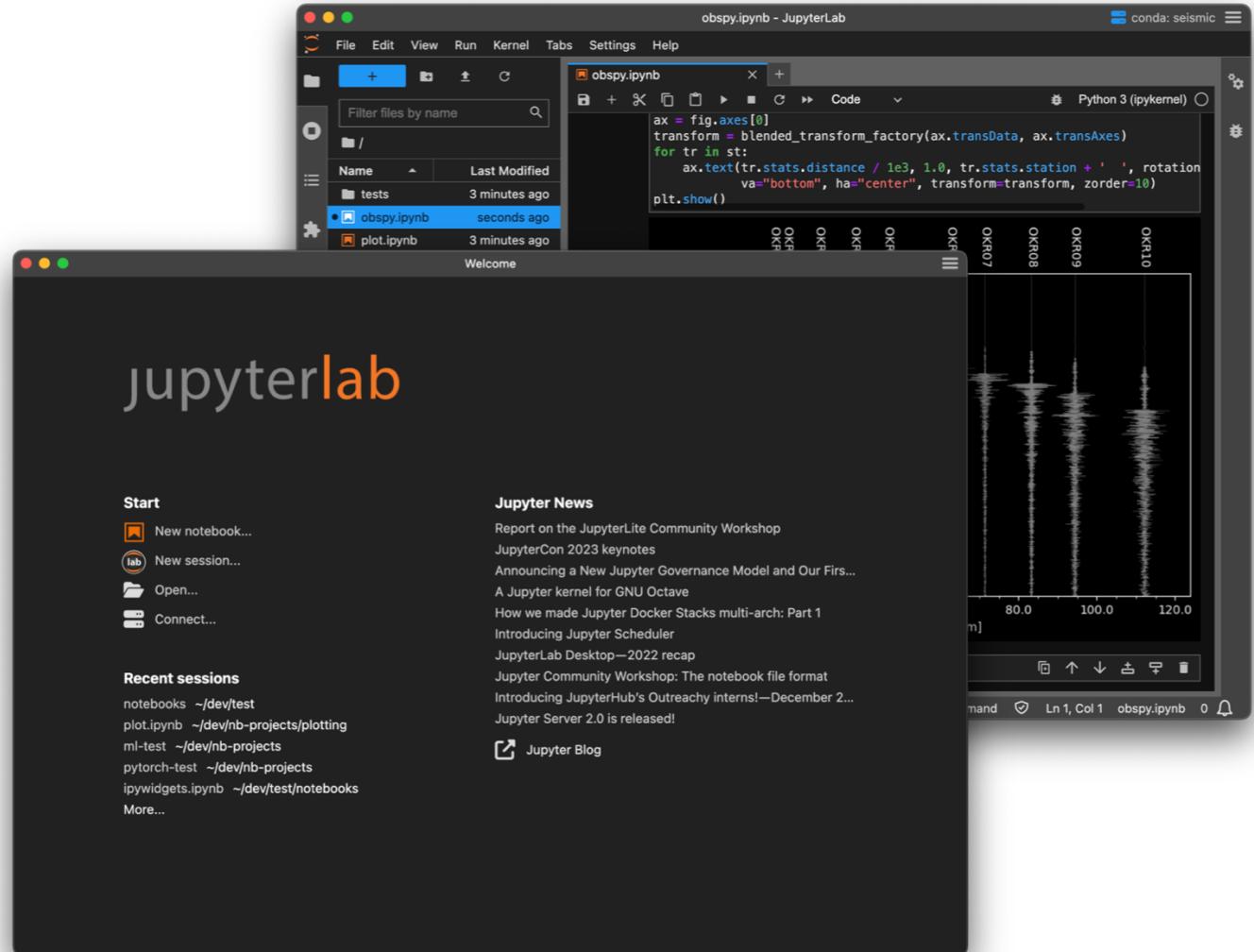
2. Integrated Environment: Combines notebooks, code, and data visualization in one interface.

3. Supports Multiple Languages: Not just Python, but also Markdown, R, Julia, and more through different kernels.

4. Real-Time Collaboration: Facilitates teamwork with shared notebooks.

5. Rich Output: Supports text, equations, visualizations, and rich media outputs.

6. Extensibility: Can be customized with extensions to enhance functionality.



<https://github.com/jupyterlab/jupyterlab-desktop>

2.2 Recommendations – option 3

Google colab

<https://colab.research.google.com>

The screenshot shows the Google Colab interface. The top navigation bar includes File, Edit, View, Insert, Runtime, Tools, Help, and a message stating "Cannot save changes". Below the navigation bar is a "Table of contents" sidebar with sections like Getting started, Data science, Machine learning, More Resources, and Featured examples. A "Section" button is also present. The main content area displays a "Data science" section with text explaining Colab's capabilities and how to import data. A code cell contains the following Python code:

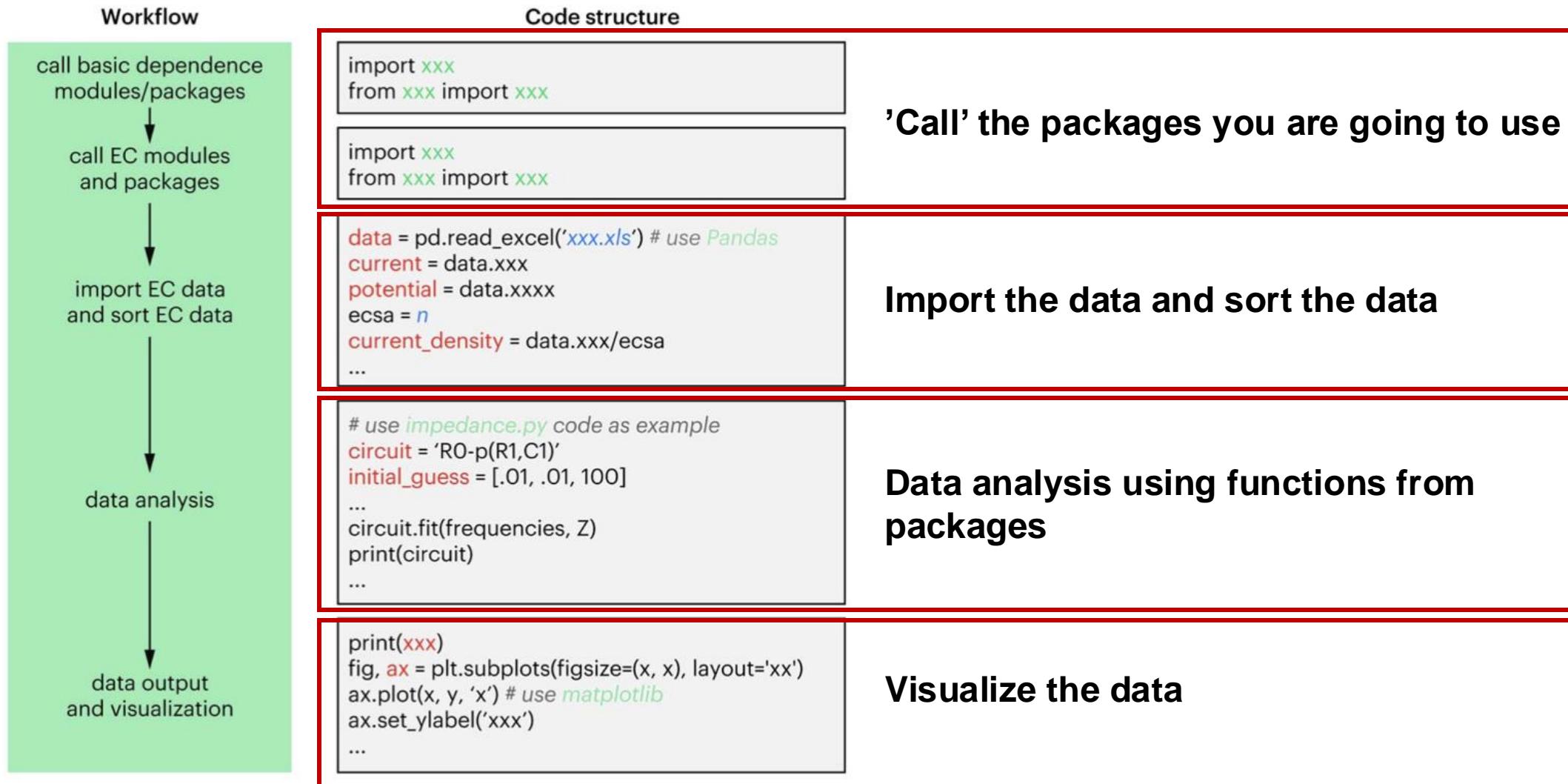
```
import numpy as np
import IPython.display as display
from matplotlib import pyplot as plt
import io
import base64

ys = 200 + np.random.randn(100)
x = [x for x in range(len(ys))]

fig = plt.figure(figsize=(4, 3), facecolor='w')
plt.plot(x, ys, '-')
plt.fill_between(x, ys, 195, where=(ys > 195), facecolor='g', alpha=0.6)
plt.title("Sample Visualization", fontsize=10)

data = io.BytesIO()
plt.savefig(data)
image = F"data:image/png;base64,{base64.b64encode(data.getvalue()).decode()}"
alt = "Sample Visualization"
display.display(display.Markdown(F"""\!{{alt}}\!{{image}}"""))
plt.close(fig)
```

2.3 Python basics – How to write the code



Code structure of a typical electrochemical task

2.4 Basic grammar and packages

(example 1 in Jupyterlab)

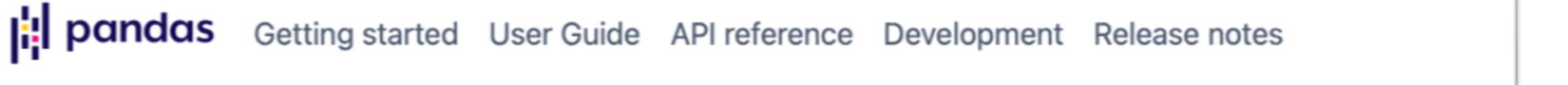
Functions in Python

- **Variables**
- **Basic operations**
- Lists
- Dictionaries
- Conditional Statements
- Loops
- **Functions**
- Classes

Must-have functions enabled by packages

 Pandas Data analysis and manipulation	 NumPy Mathematical functions
 Matplotlib Data visualisations	 Seaborn Data visualisations
 SciPy Scientific computing	

2.5 Basic packages - pandas



<https://pandas.pydata.org/docs/index.html>

Format	Type	Data Description	Reader	Writer
text	CSV		read_csv	to_csv
text	Fixed-Width Text File		read_fwf	
text	JSON		read_json	to_json
text	HTML		read_html	to_html
text	LaTeX			Styler.to_latex
text	XML		read_xml	to_xml
text	Local clipboard		read_clipboard	to_clipboard
binary	MS Excel		read_excel	to_excel
binary	OpenDocument		read_excel	
binary	HDF5 Format		read_hdf	to_hdf

df1				df4			Result									
	A	B	C	D	B	D	F	A	B	C	D	B	D	F		
0	A0	B0	C0	D0	2	B2	D2	F2	0	A0	B0	C0	D0	NaN	NaN	NaN
1	A1	B1	C1	D1	3	B3	D3	F3	1	A1	B1	C1	D1	NaN	NaN	NaN
2	A2	B2	C2	D2	6	B6	D6	F6	2	A2	B2	C2	D2	B2	D2	F2
3	A3	B3	C3	D3	7	B7	D7	F7	3	A3	B3	C3	D3	B3	D3	F3

- **Read/sort/write datasets**
 - This is especially useful when data sharing similar structures (instrumental data)
- **Simple calculation**
 - ECSA normalization, iR compensation, Reference electrode correction, etc.

2.5 Basic packages - matplotlib

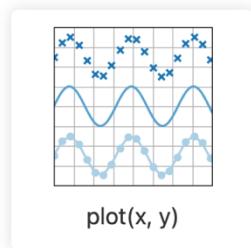


Plot types User guide Tutorials Examples Reference Contribute Releases

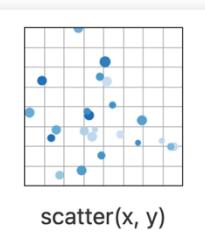
<https://matplotlib.org>

Pairwise data

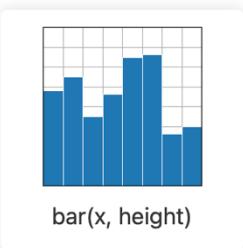
Plots of pairwise (x, y) , tabular (var_0, \dots, var_n) , and functional $f(x) = y$ data.



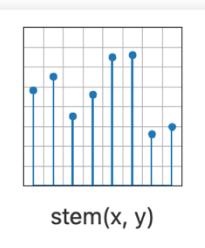
`plot(x, y)`



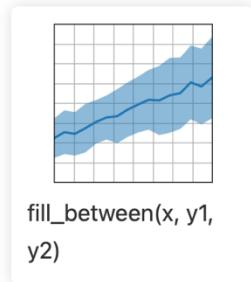
`scatter(x, y)`



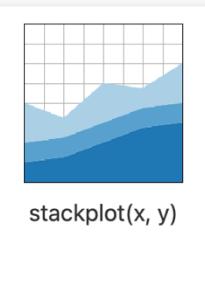
`bar(x, height)`



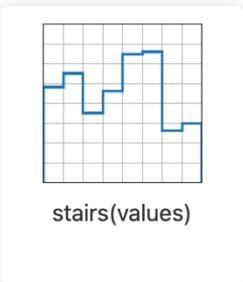
`stem(x, y)`



`fill_between(x, y1, y2)`



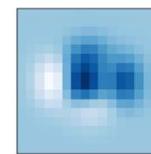
`stackplot(x, y)`



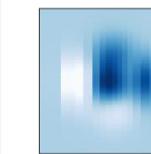
`stairs(values)`

Gridded data

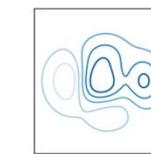
Plots of arrays and images $Z_{i,j}$ and fields $U_{i,j}, V_{i,j}$ on [regular grids](#) and corresponding coordinate grids $X_{i,j}, Y_{i,j}$.



`imshow(Z)`



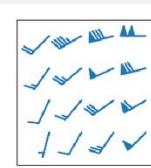
`pcolormesh(X, Y, Z)`



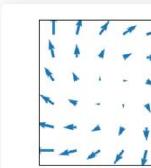
`contour(X, Y, Z)`



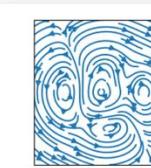
`contourf(X, Y, Z)`



`barbs(X, Y, U, V)`



`quiver(X, Y, U, V)`



`streamplot(X, Y, U, V)`

[Download all examples in Python source code: `plot_types_python.zip`](#)

(example 2 in Jupyterlab)

[Download all examples in Jupyter notebooks: `plot_types_jupyter.zip`](#)

2.5 Basic packages - SciPy



Installing User Guide API reference Building from source Development Release notes

Section Navigation

User guide

Special functions ([scipy.special](#))

Integration ([scipy.integrate](#))

Optimization ([scipy.optimize](#))

Interpolation ([scipy.interpolate](#))

Fourier Transforms ([scipy.fft](#))

Signal Processing ([scipy.signal](#))

Linear Algebra ([scipy.linalg](#))

Sparse Arrays ([scipy.sparse](#))

Sparse eigenvalue problems with ARPACK

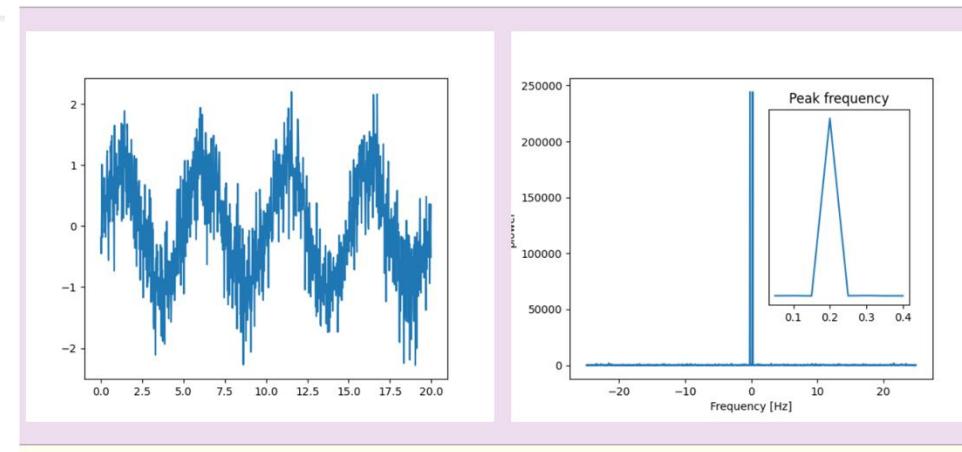
Compressed Sparse Graph Routines
([scipy.sparse.csgraph](#))

Spatial data structures and algorithms
([scipy.spatial](#))

Statistics ([scipy.stats](#))

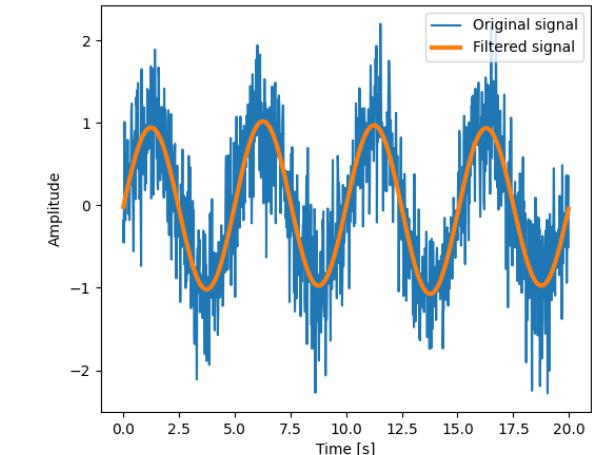
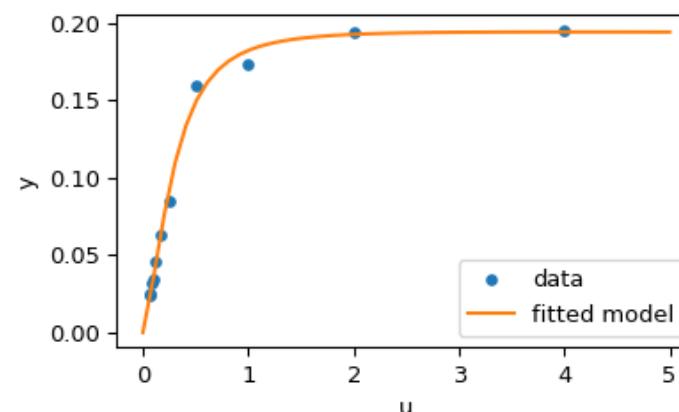
Multidimensional image processing
([scipy.ndimage](#))

File IO ([scipy.io](#))

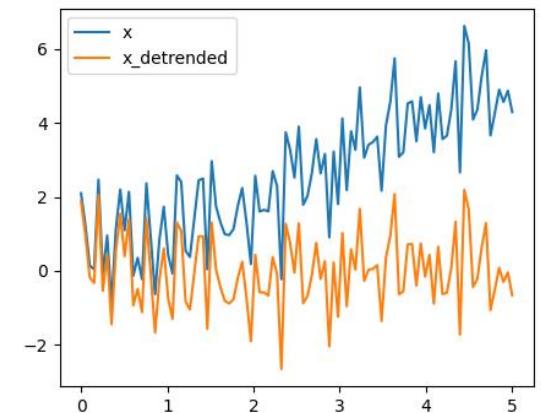


Signal

FFT



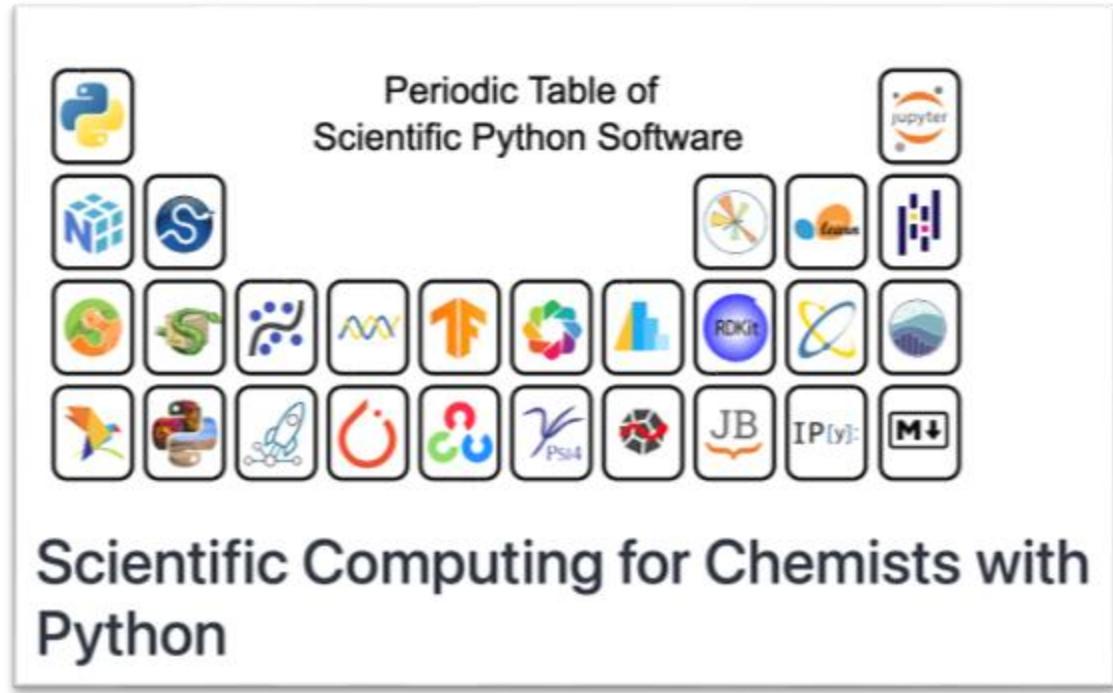
noise filtering



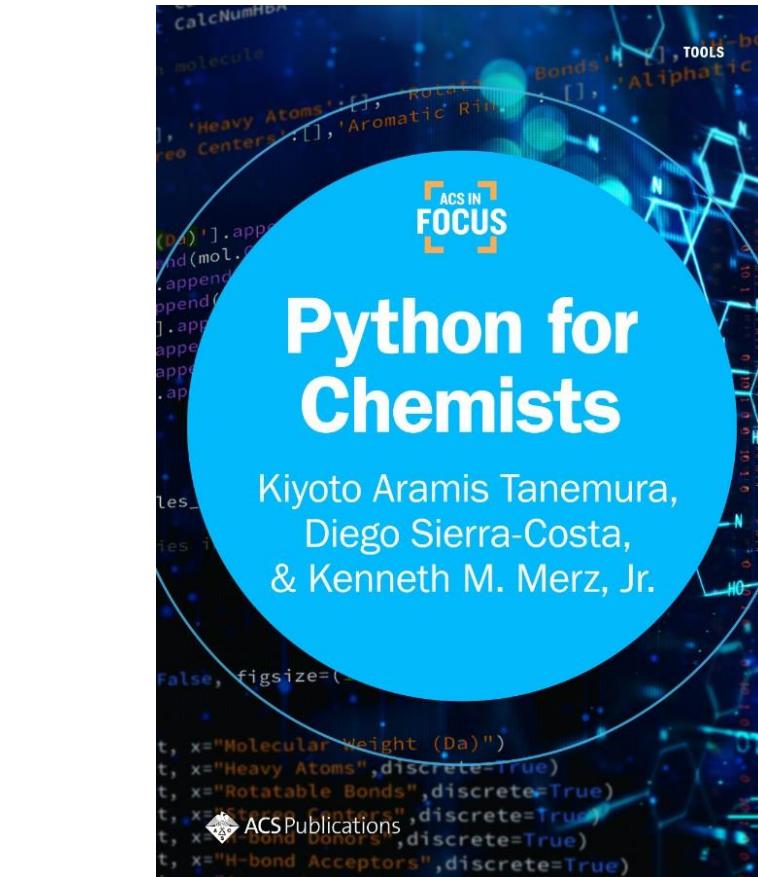
model fitting

detrend

2.6 Learning resources



<https://weisscharlesj.github.io/SciCompforChemists/notebooks/introduction/intro.html>



Python for Chemists; American Chemical Society, 2022. DOI: 10.1021/acsinfocus.7e5030

2.6 Learning resources



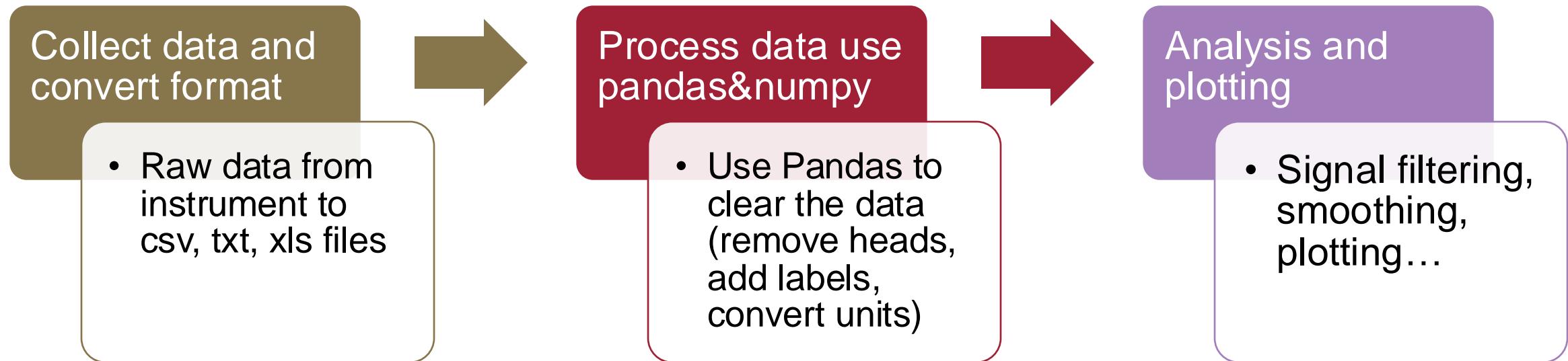
<https://electrochemistry.uoregon.edu/resources/>

Plotting and Data Analysis in Python

- [CodeAcademy offers free trials of their pro membership which includes many interactive Python courses](#)
- [Similar courses are available from JetBrains](#)
- [Introduction to Python and Jupyter for Materials Science \(Prof. João Fonseca at University of Manchester\)](#)
- [Well made Python video tutorials from Keith Galli](#)
- [Helpful tips for plotting with matplotlib \(Electrochemical Society Student Chapter at the University of Washington\)](#)

3.1 Electrochemical data handling - General

In general:

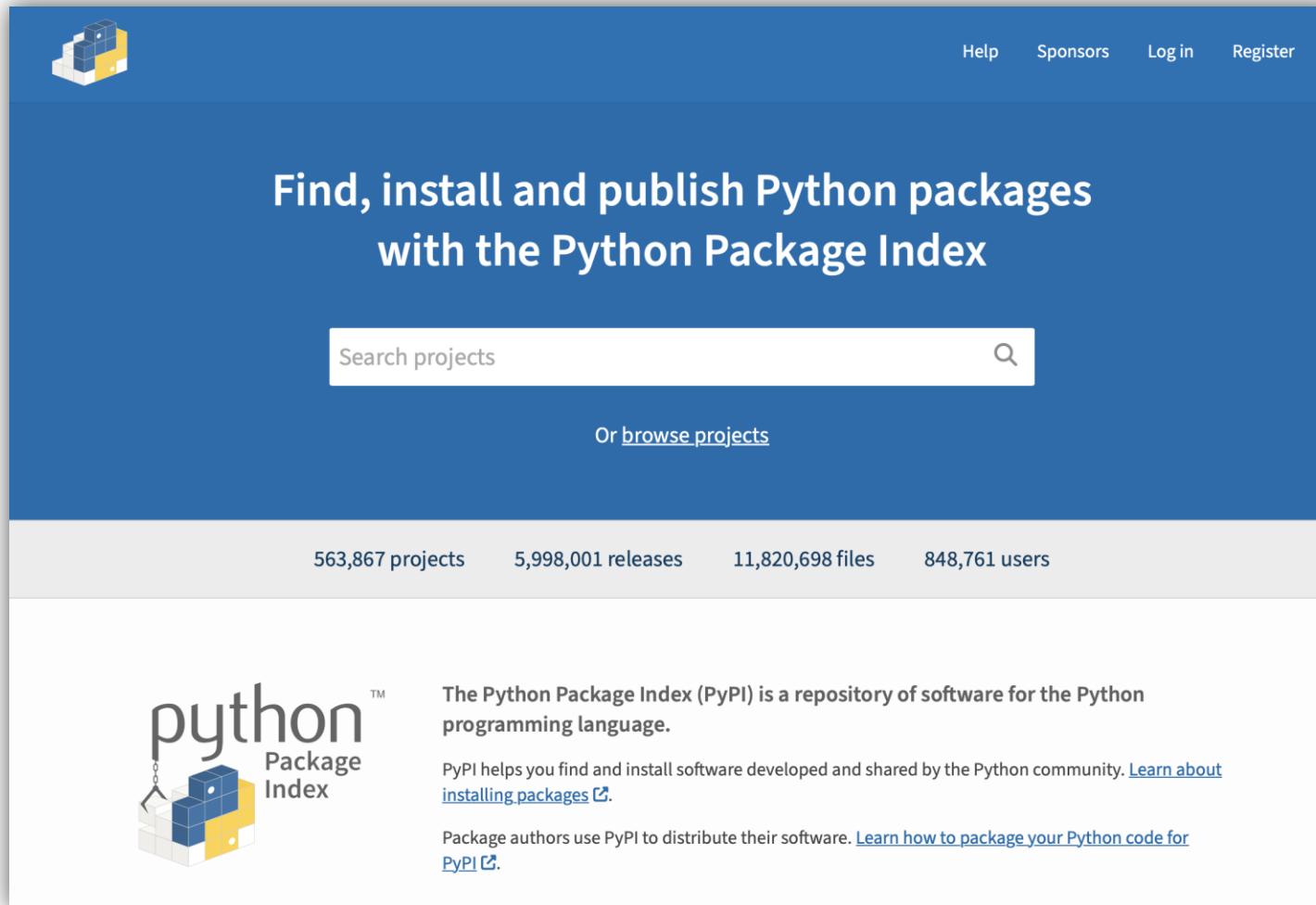


(example 3 and 4 in Jupyterlab)

- No change on the raw data (delete no information)
- Once written, you can copy-paste to share it with others
- Easier for PI to manage

3.2 Electrochemical data handling - Packages

Sometimes, you can find packages for direct import and read of instrumental files:

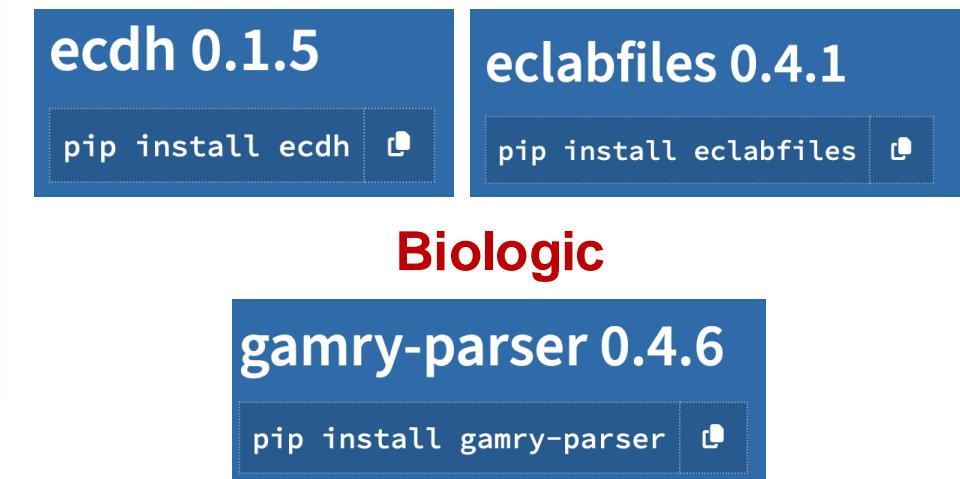


The screenshot shows the PyPI homepage with a blue header bar. On the left is the Python logo icon. In the center, the text reads "Find, install and publish Python packages with the Python Package Index". Below this is a search bar with the placeholder "Search projects" and a magnifying glass icon. Underneath the search bar is a link "Or [browse projects](#)". At the bottom of the header, there are statistics: "563,867 projects", "5,998,001 releases", "11,820,698 files", and "848,761 users". The main content area has a white background. On the left, there's a "python Package Index" logo with a stylized 3D cube icon. To the right of the logo, text explains what PyPI is: "The Python Package Index (PyPI) is a repository of software for the Python programming language." It also describes how to use PyPI: "PyPI helps you find and install software developed and shared by the Python community. [Learn about installing packages](#)." Another section says "Package authors use PyPI to distribute their software. [Learn how to package your Python code for PyPI](#)."

<https://pypi.org>: released packages



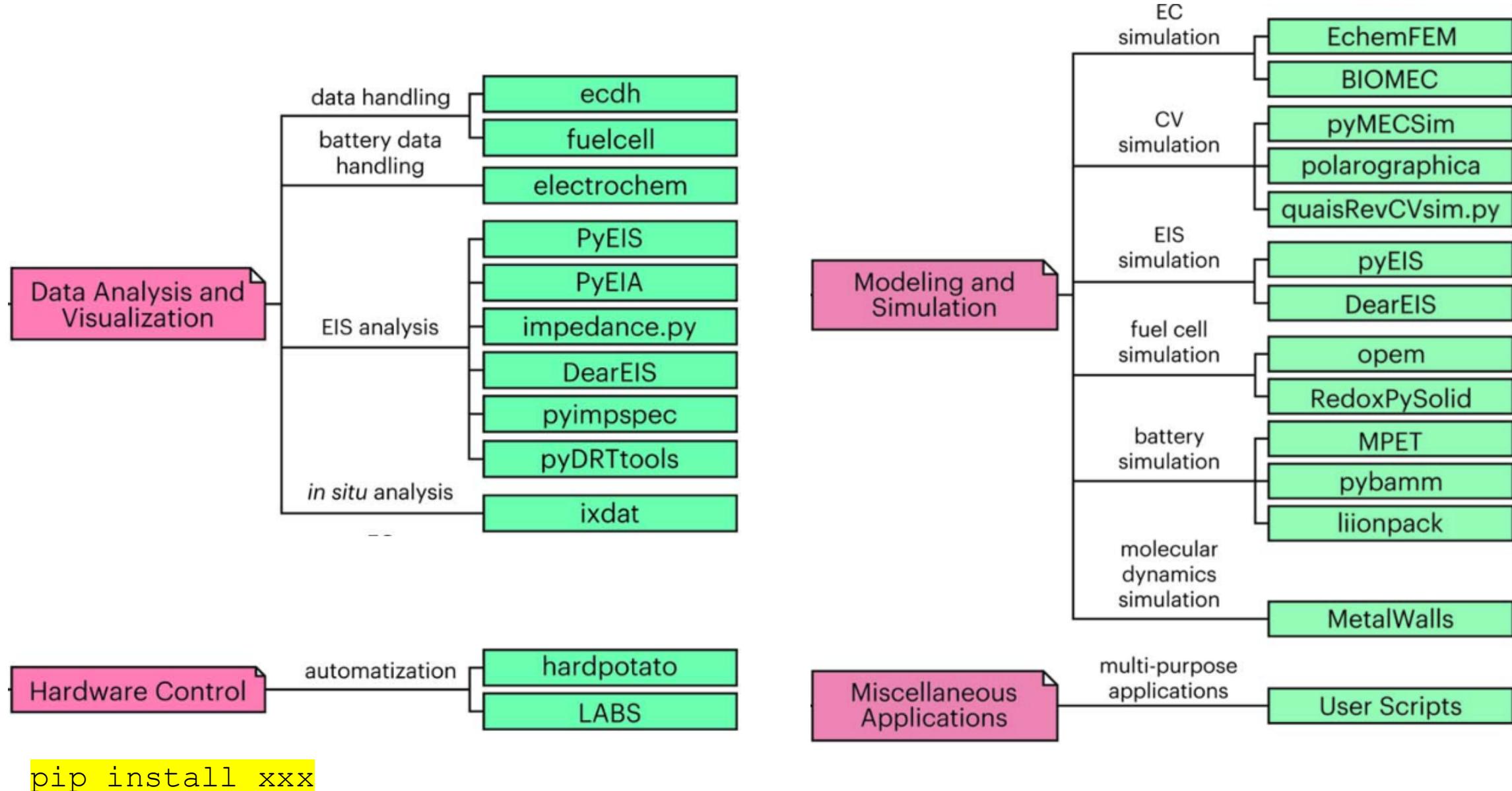
<https://github.com>: scripts written by others



The screenshot shows GitHub with several package repositories listed. Each repository has a blue header with the package name and version, followed by a "pip install" command and a download link icon. The packages shown are: "ecdh 0.1.5", "eclabfiles 0.4.1", "Biologic", "gamry-parser 0.4.6", and "Gamry".

Repository	Version	Action
ecdh	0.1.5	pip install ecdh
eclabfiles	0.4.1	pip install eclabfiles
Biologic		
gamry-parser	0.4.6	pip install gamry-parser
Gamry		

4.1 Python for analysis/visualization, modeling, and hardware control



4.2 Python for analysis/visualization

Some tools must be used in a coding environment, and some have GUI.

CV analysis

MADAP: Modular and Autonomous Data Analysis Platform (**with GUI**)



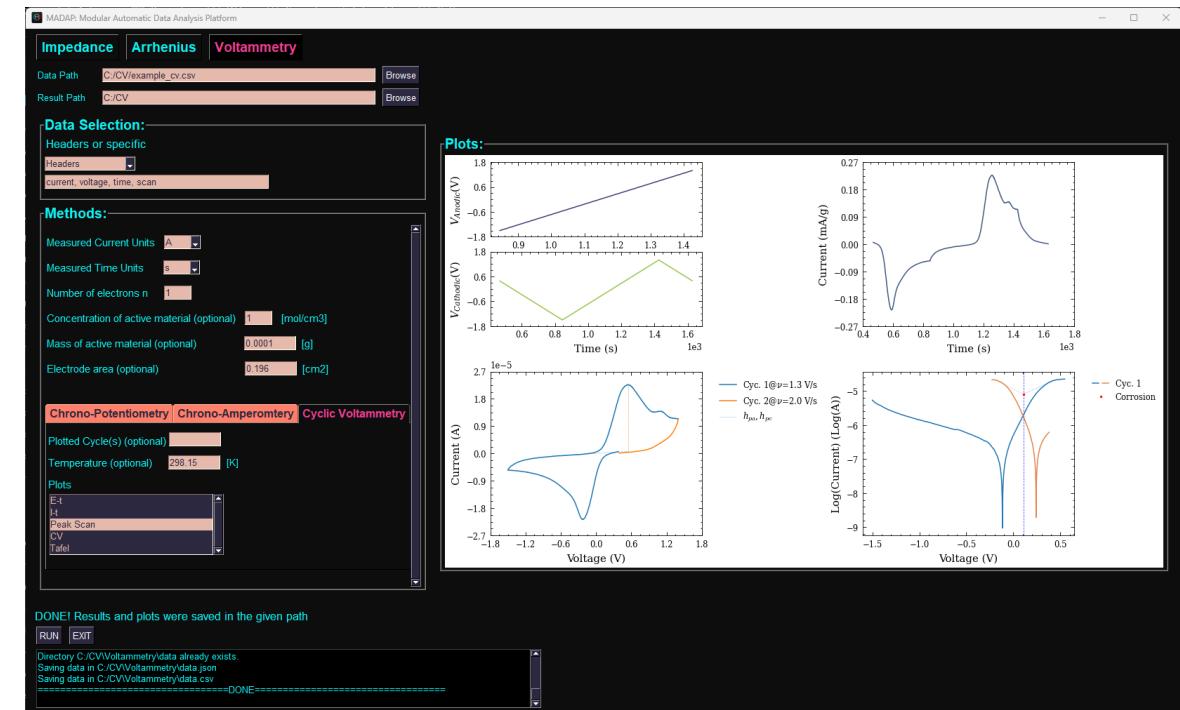
```
from madap.echem.arrhenius import arrhenius
from madap.echem.e_impedance import e_impedance
from madap.data_acquisition import data_acquisition as da

# Load the data
data = da.acquire_data('data.csv')
# Define the desired plots for Arrhenius analysis
plots_arr = ["arrhenius", "arrhenius_fit"]
# Define the desired plots for impedance analysis
plots_eis = ["nyquist", "nyquist_fit", "bode", "residual"]
# Define a save location#
save_dir = "/results"

### Arrhenius
# Instantiate the Arrhenius class for analysis (column names do not have to match exactly, this
Arr = arrhenius.Arrhenius(da.format_data(data["temperature"], da.format_data(data["conductivity"])))
# Perform analysis and plotting
Arr.perform_all_actions(save_dir, plots = plots_arr)

### Impedance
# Initialize the Impedance class for analysis (column names do not have to match exactly, this
Im = e_impedance.EImpedance(da.format_data(data["freq"]), da.format_data(data["real"])), da.format_data(data["imag"]))
# Initialis the EIS procedure. The initial value is the initial guess for the equivalent circuit
Eis = e_impedance.EIS(Im, suggested_circuit = "R0-p(R1,CPE1)", initial_value =[860, 3e+5, 1e-6])
# Analyze the data
Eis.perform_all_actions(save_dir, plots = plots_eis)

# More usages and options can be found in the documentation.
```



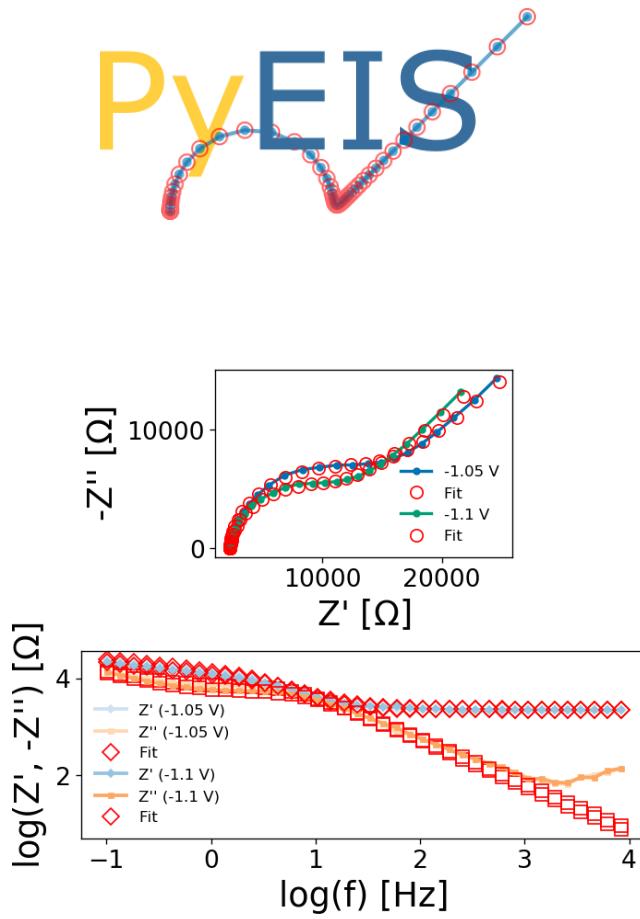
python -m madap_gui

<https://pypi.org/project/MADAP/>

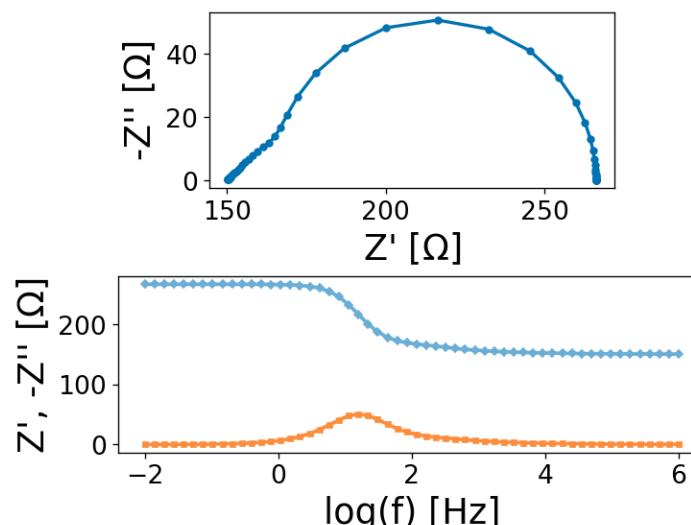
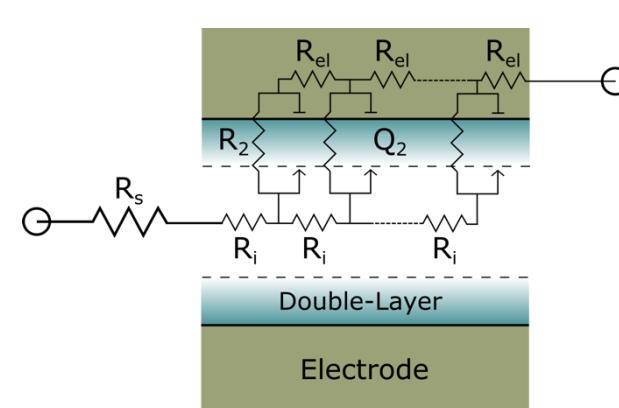
10.5281/zenodo.10357192

4.2 Python for analysis/visualization

EIS analysis



PyEIS: A Python-based Electrochemical Impedance Spectroscopy simulator and analyzer (no GUI)



[PyEIS / Tutorials](#) /

kbknudsen dev ops

Name

..

.ipynb_checkpoints

data

PyEIS_command_overview.pdf

PyEIS_experimental-data_tutorial.ipynb

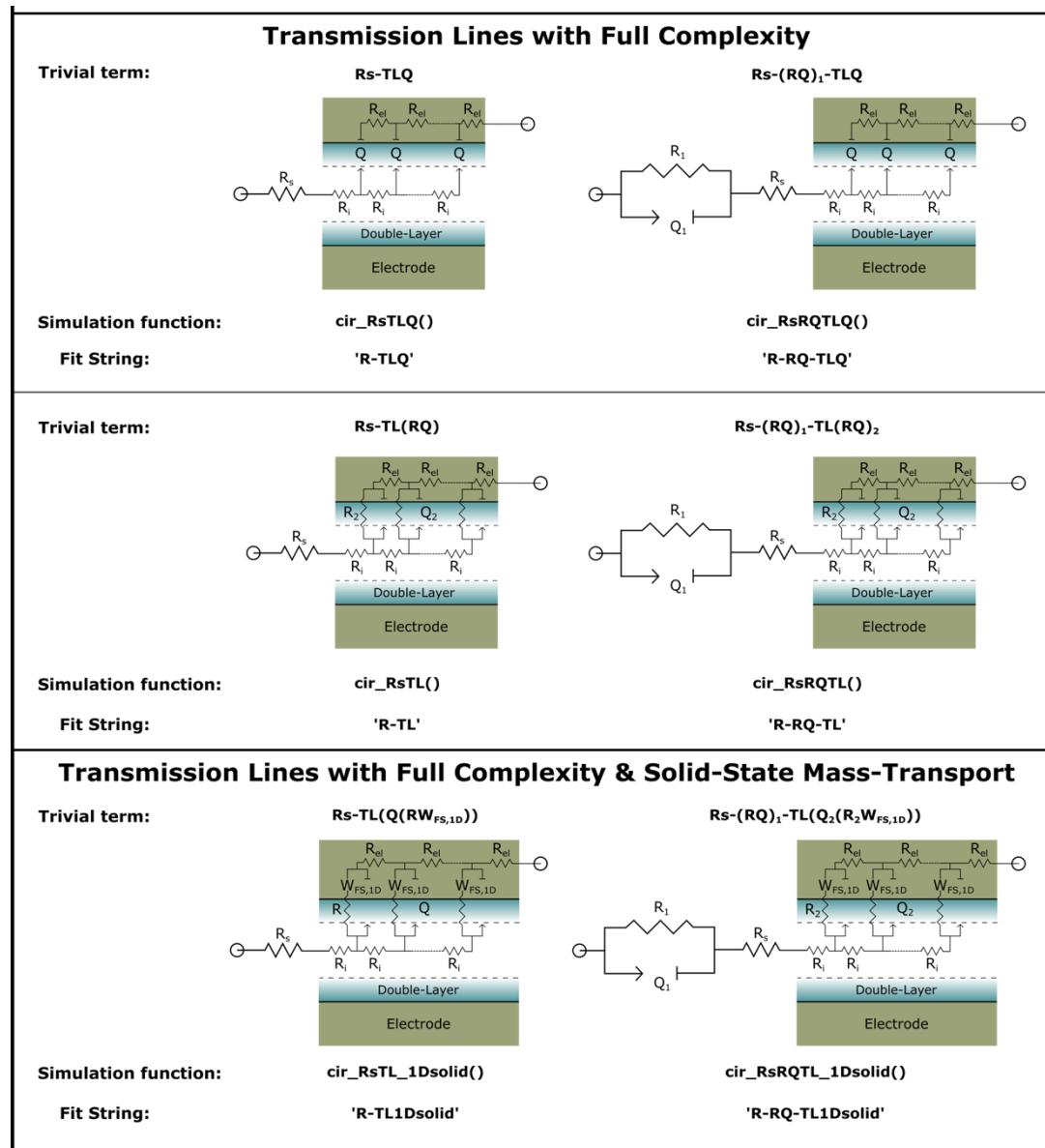
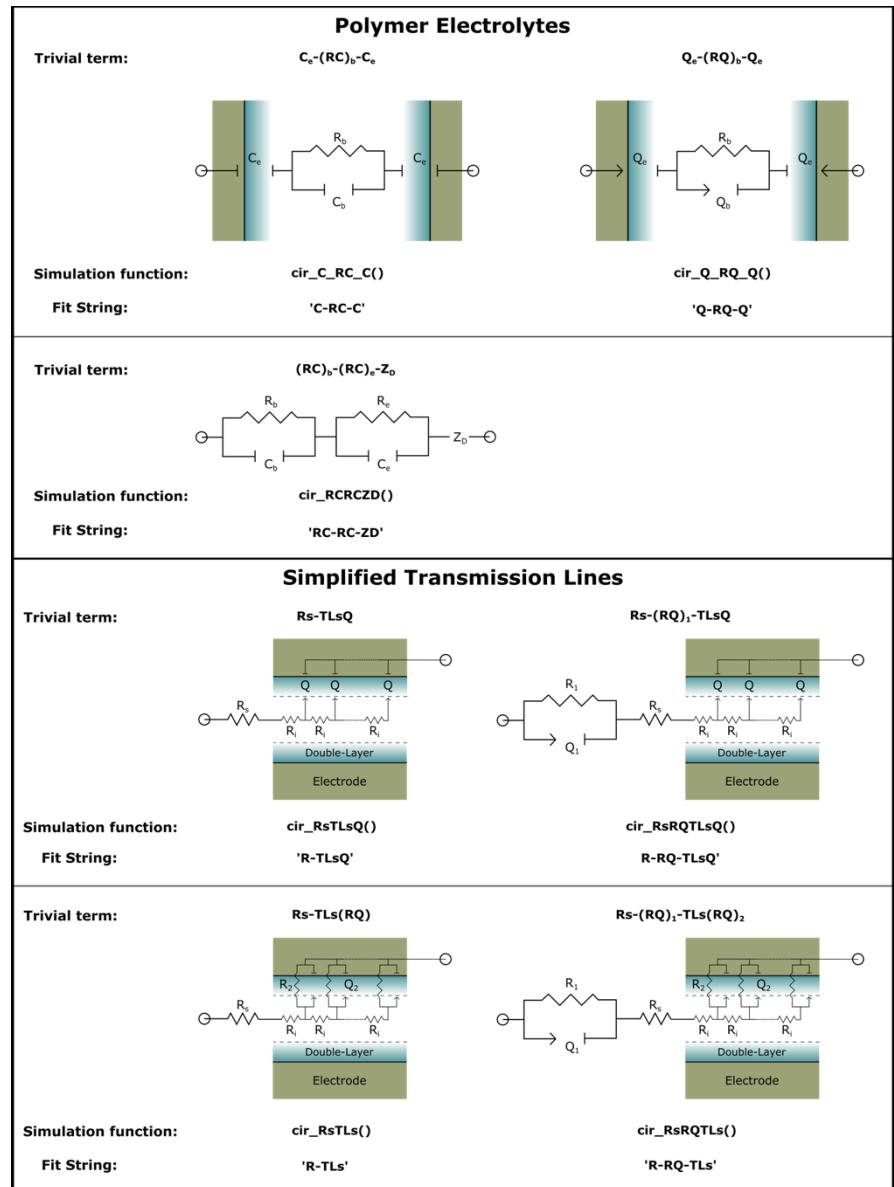
PyEIS_simulation_tutorial.ipynb

RQ example.png

<https://github.com/kbknudsen/PyEIS/tree/master>

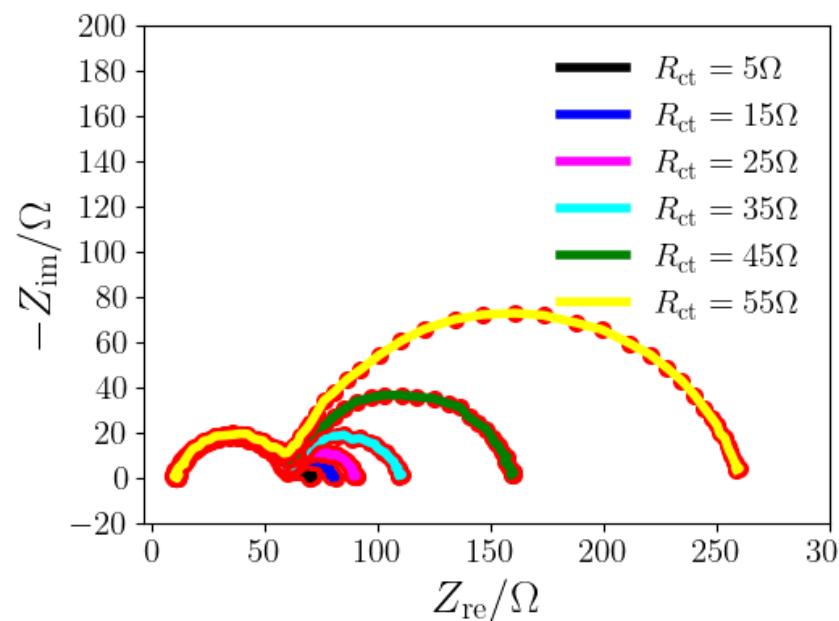
4.2 Python for analysis/visualization

PyEIS models



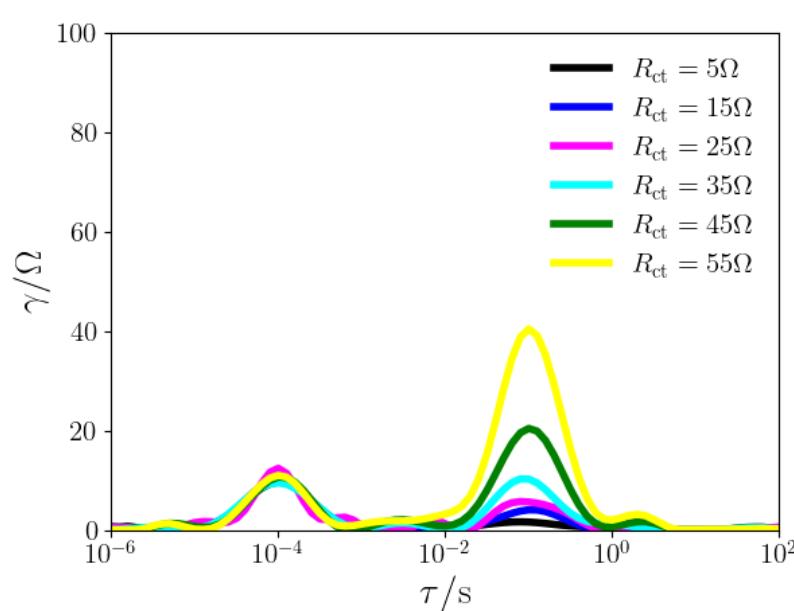
4.2 Python for analysis/visualization

EIS analysis **pyDRTtools**: A Python-based DRTtools to Deconvolve the Distribution of Relaxation Times from Electrochemical Impedance Spectroscopy Data (**with GUI**)



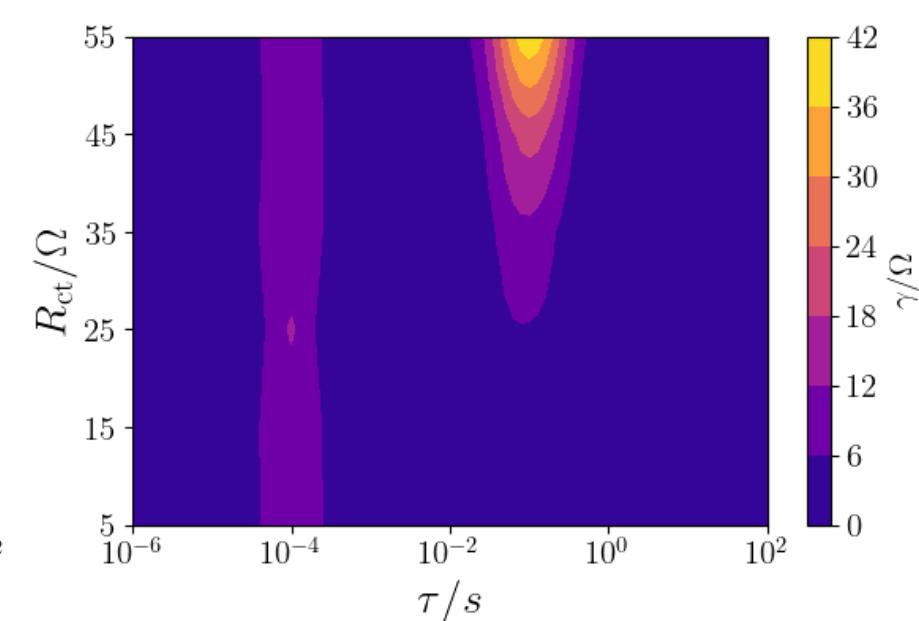
Batch fitting of
Nyquist plots

Ciucci group



Batch DRT
analysis

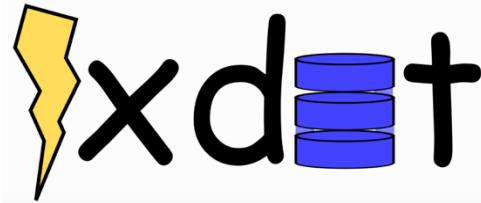
<https://github.com/ciuccislab/pyDRTtools>



Contour plot

4.2 Python for analysis/visualization

In situ data analysis ixdat: The in-situ experimental data tool

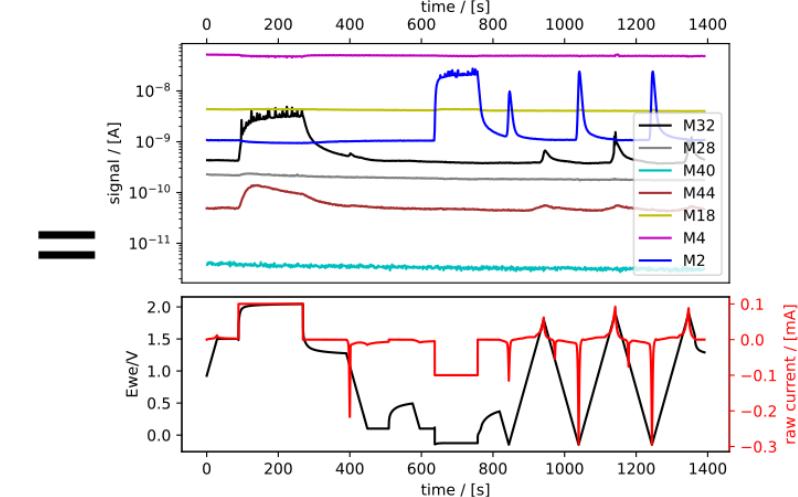
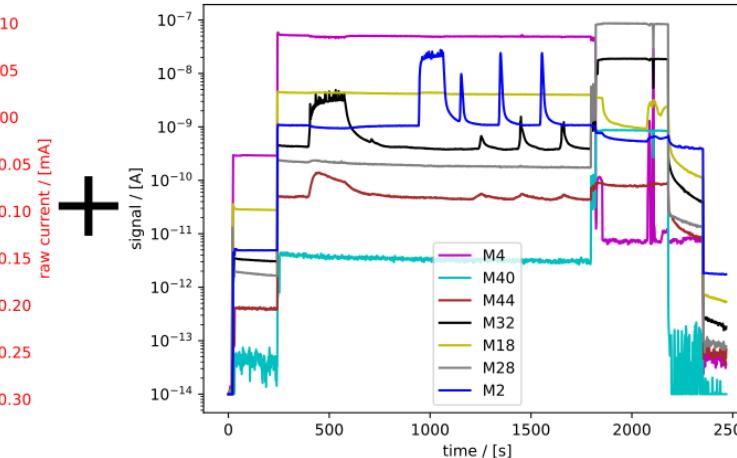
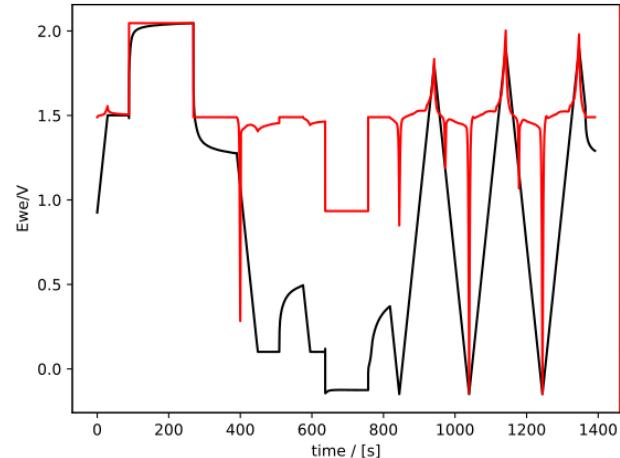


```
ec = Measurement.read_set("awesome_EC_data", reader="biologic")
ec.plot_measurement()

ms = Measurement.read("2021-03-30 16_59_35 MS data.tsv", reader="zilien")
ms.plot_measurement()

ecms = ec + ms
ecms.plot_measurement()

ecms.export("my_combined_data.csv")
```



the ixdat team – ICL etc.

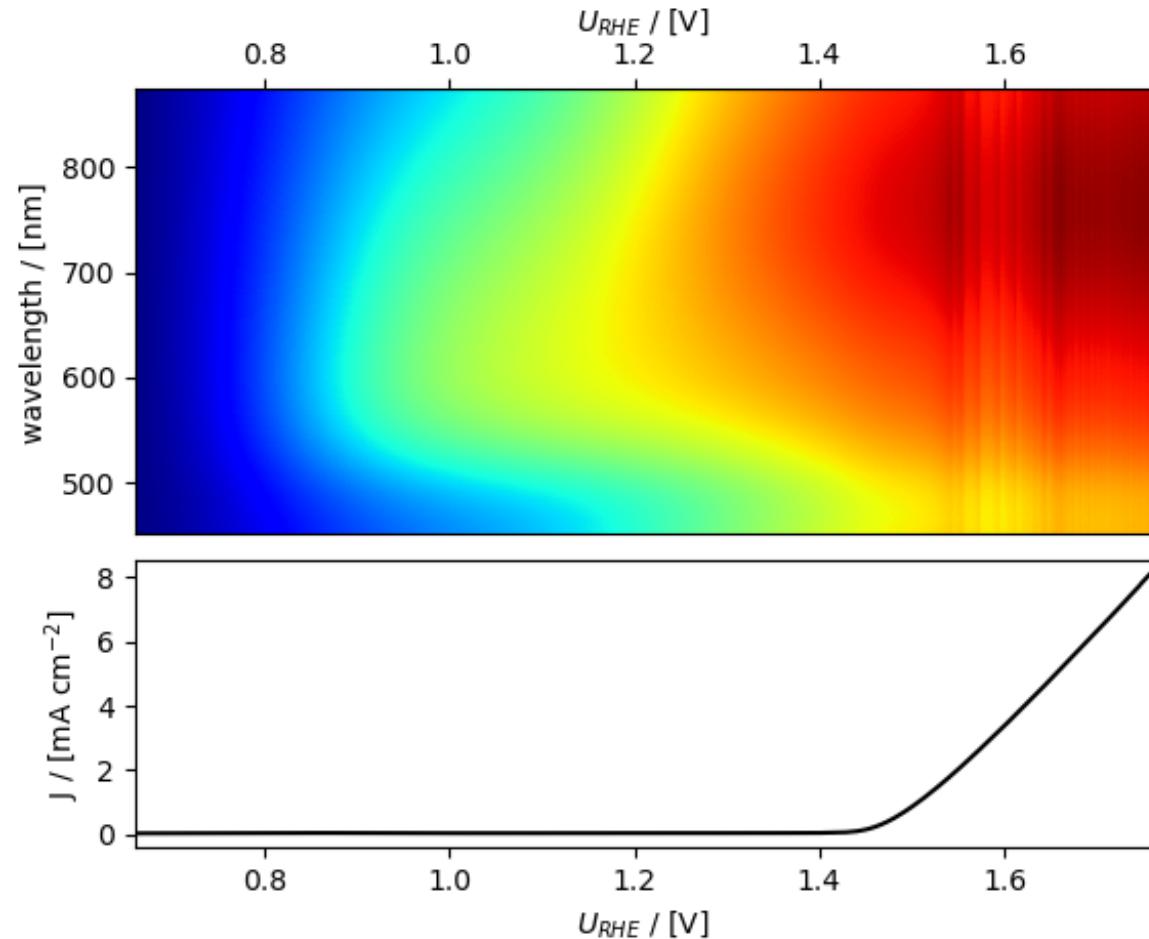
<https://ixdat.readthedocs.io/en/latest/index.html>

4.2 Python for analysis/visualization

In situ data analysis ixdat: The in-situ experimental data tool



Measurement technique	Status	Readers
Electrochemistry	Released	<ul style="list-style-type: none">biologic: .mpt files from Biologic's EC-Lab softwareautolab: ascii files from AutoLab's NOVA softwareivium: .txt files from Ivium's IviumSoft software
Mass Spectrometry	Released	<ul style="list-style-type: none">pfeiffer: .dat files from Pfeiffer Vacuum's PVMassSpec softwarecinfdata: text export from DTU Physics' cinfdata systemzilien: .tsv files from Spectro Inlets' Zilien software
Electrochemistry - Mass Spectrometry	Released	<ul style="list-style-type: none">zilien: .tsv files from Spectro Inlets' Zilien softwareEC_MS: .pkl files from the legacy EC_MS python package
Spectroelectrochemistry	Released	<ul style="list-style-type: none">msrh_sec: .csv file sets from Imperial College London's SEC system
X-ray photoelectron spectroscopy (XPS)	Development	<ul style="list-style-type: none">avantage: .avg files from Thermo Scientific's Avantage software
X-ray diffraction (XRD)	Development	<ul style="list-style-type: none">xrdml: .xrdml files from e.g. PanAnalytical's Empyereon



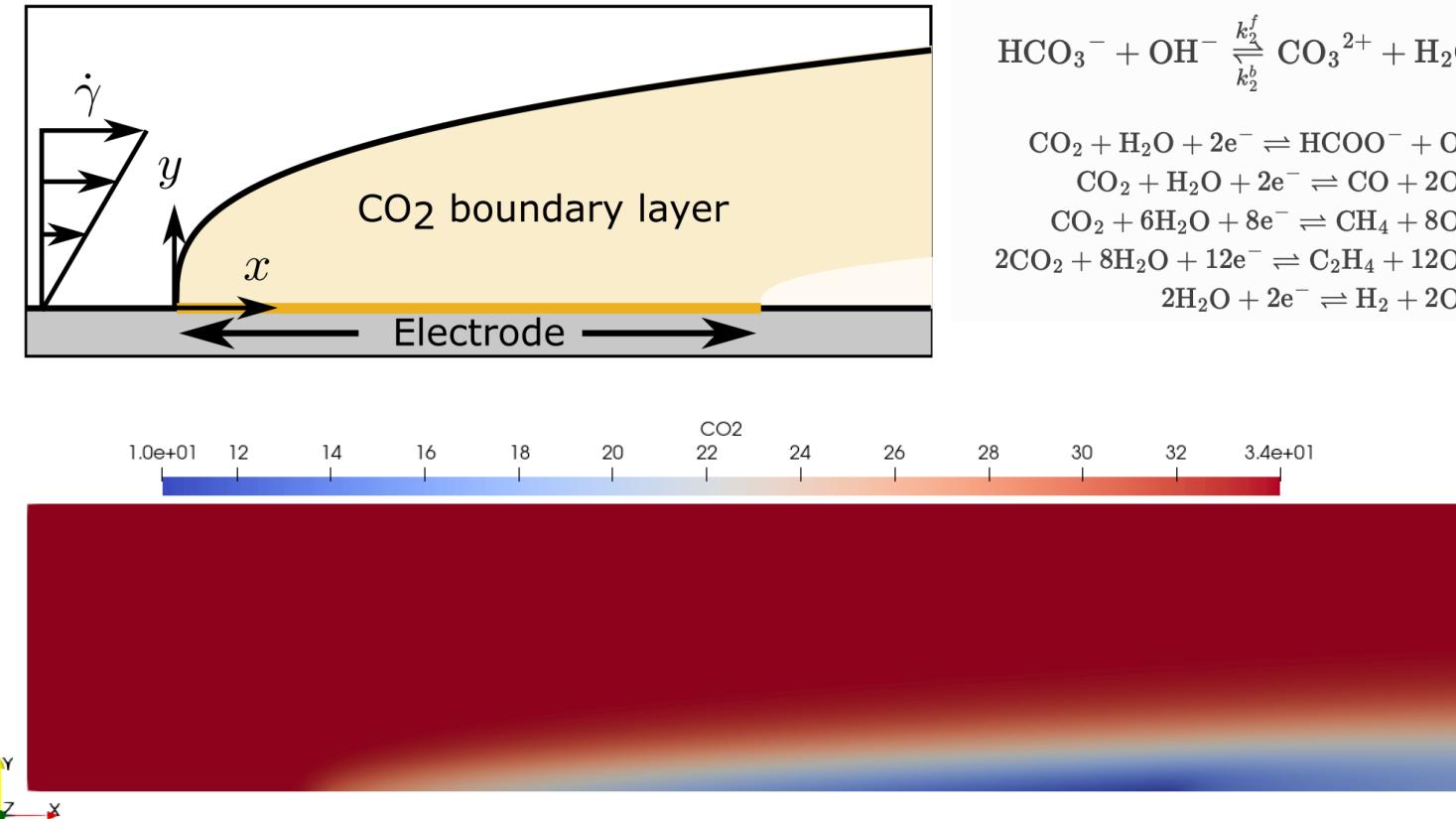
the ixdat team – ICL etc.

<https://github.com/ixdat/ixdat>

4.3 Python for simulation

Simulation

Flow reactor for CO₂ electrolysis



echemFEM: Finite Element solvers for electrochemical transport

[echemfem / examples /](#)

tlroy solve for bulk concentrations ✓

Name



BMCSL.py

bicarb.py

bicarb_Ag_Cu_tandem_example.py

bortels_structuredquad.geo

bortels_structuredquad_nondim.geo

bortels_structuredquad_nondim_coarse1664.geo

bortels_threeion.py

bortels_threeion_extruded_3D_nondim.py

bortels_threeion_nondim.py

bortels_twoion.py

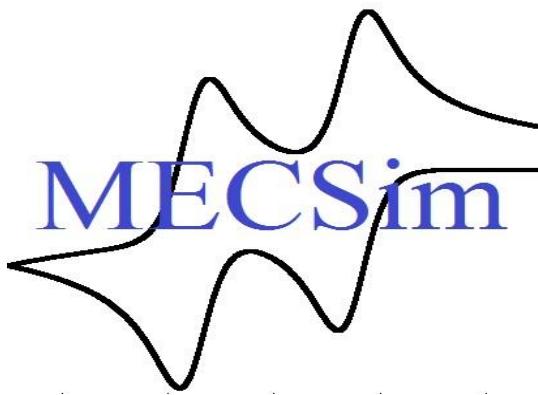
bortels_twoion_nondim.py

bortels_unstructuredquad_nondim_coarse.geo

carbonate.py

4.3 Python for simulation

CV Simulation



MECSim by the Monash
Electrochemistry group

<https://garethkennedy.net/MECSim.html>

pyMECSim: A Python wrapper of MECSim software

```
from pymecsim import *

A = Specie('A', C0=1e-6)
B = Specie('B')
C = Specie('C')
D = Specie('D')
species = [A, B, C, D]

R1 = ChargeTransfer([(A,1),('e',1)],[(B,1)],E0=0.0, ks=1.0e1)
R2 = ChemicalReaction([(B,1),(C,1)],[(A,1),(D,1)], kf=1e8, kb=1e-5)

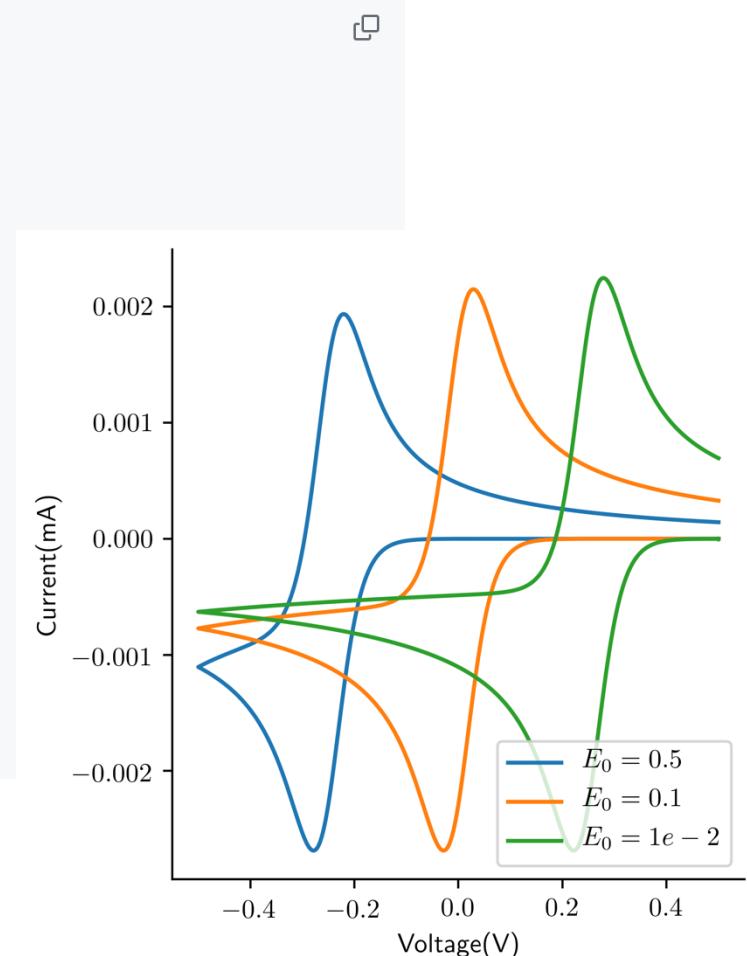
rxn = [R1, R2]

mech = Mechanism(rxn)
print(mech)

cv = DCVoltammetry(E_start = 0.5, E_rev=-0.5, nu=1.0e0)
volt = Voltammetry(objs=[cv])

exp = Experiment(mech, voltammetry=volt)

sim = MECSIM(exp=exp)
sim.solve()
sim.plot() # plot the simulated CV curve
```

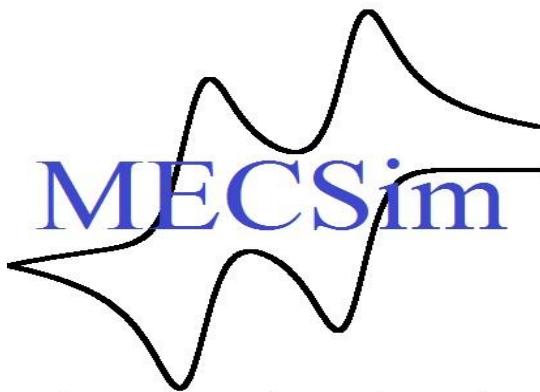


pyMECSim: Kiran Vaddi
MECSim: Gareth Kennedy

<https://github.com/kiranvad/pyMECSim>

4.3 Python for simulation

CV Simulation



MECSim by the Monash
Electrochemistry group

<https://garethkennedy.net/MECSim.html>

pyMECSim: A Python wrapper of MECSim software

[pyMECSim](#) / [notebooks](#) /

Kiran Vaddi update readme and setup.py

Name



..

Basic Usage.ipynb

Kinetic Zone Diagram.ipynb

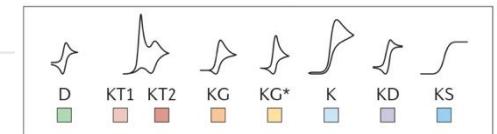
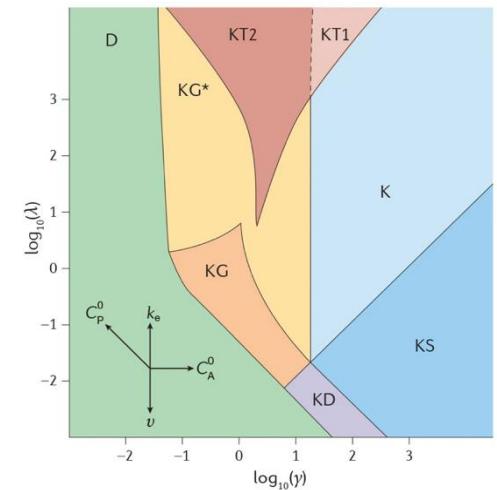
Langmuir Isotherm.ipynb

Notebooks

This repository is shipped few useful [notebooks](#).

- [Cyclic Voltammetry Simulation Example for Single Electron Transfer Mechanism](#) -- that explains all the possible use cases of the simulator on a simple one electron transfer mechanism
- [Kinetic Zone Diagram](#) -- example explaining how to generate various CV curves in the [kinetic zone diagram](#)
- [Langmuir isotherm](#)-- example with modelling surface reactions and reproducing the Langmuir isotherm assumption

<https://github.com/kiranvad/pyMECSim>



Nature Reviews | Chemistry

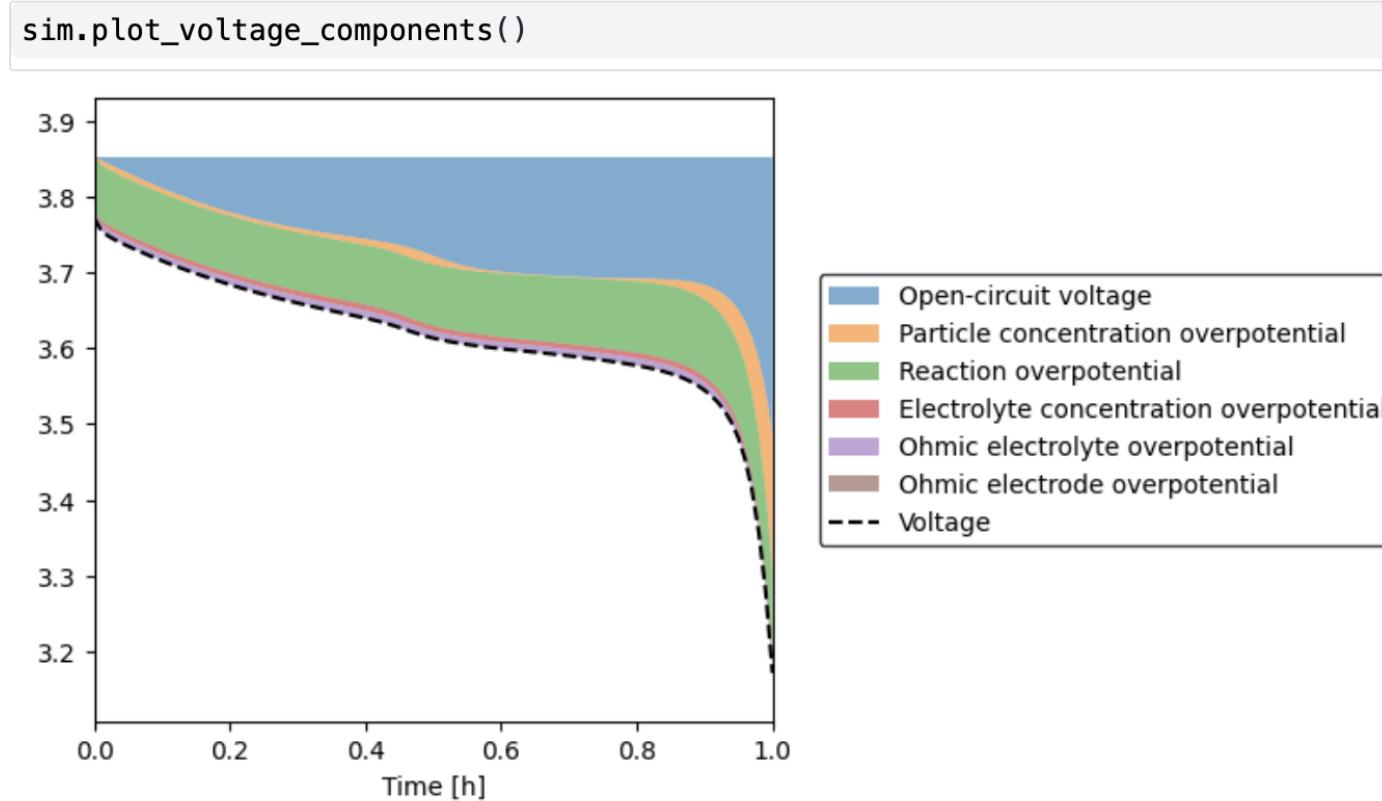
4.3 Python for simulation

Battery Simulation

PyBaMM 

<https://pybamm.org>

pyBaMM: Fast and flexible physics-based battery models in Python



voltage plot showing the contribution of the various overpotentials and losses

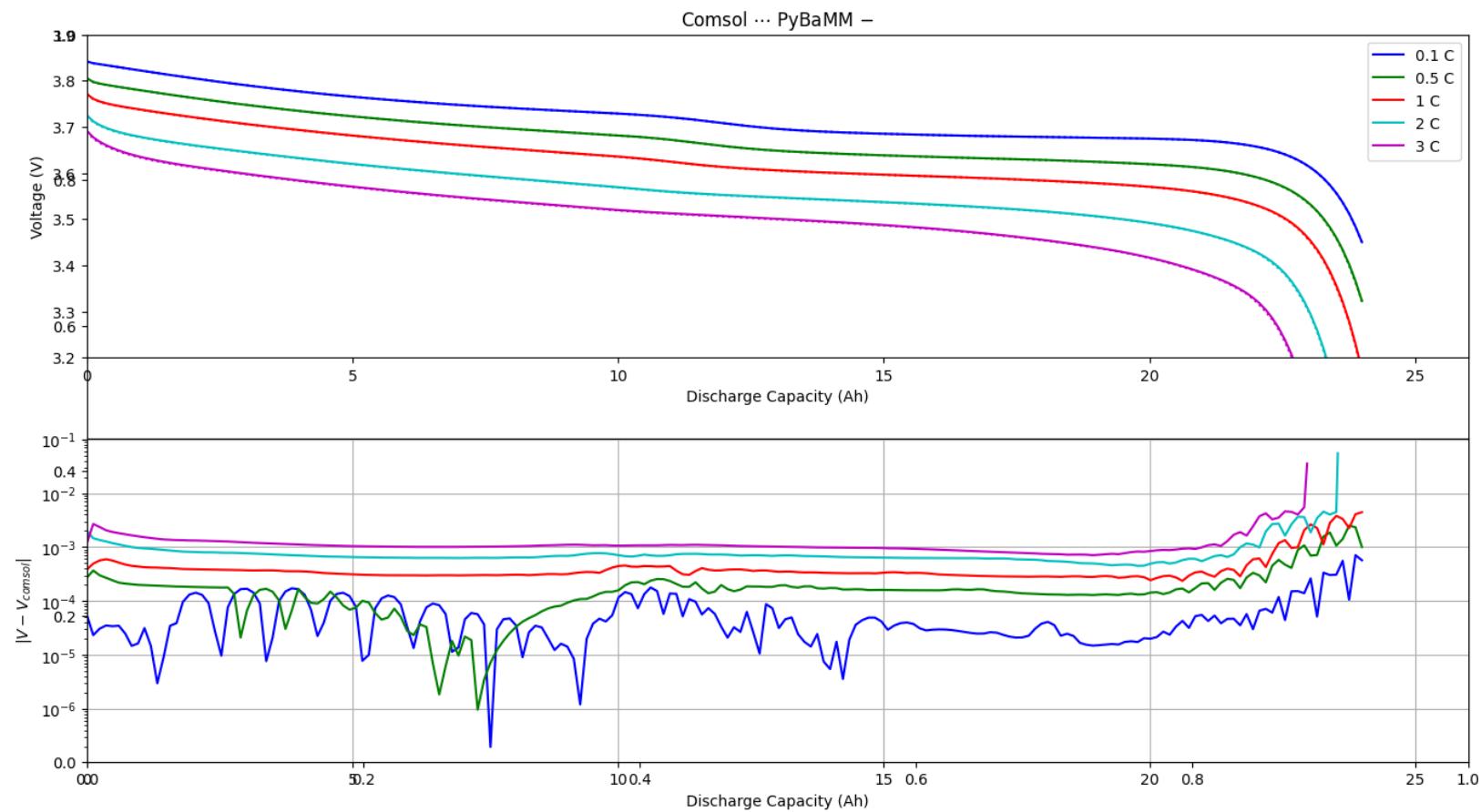
4.3 Python for simulation

Simulation

PyBaMM 

<https://pybamm.org>

pyBaMM: Fast and flexible physics-based battery models in Python



Comparing PyBaMM and COMSOL

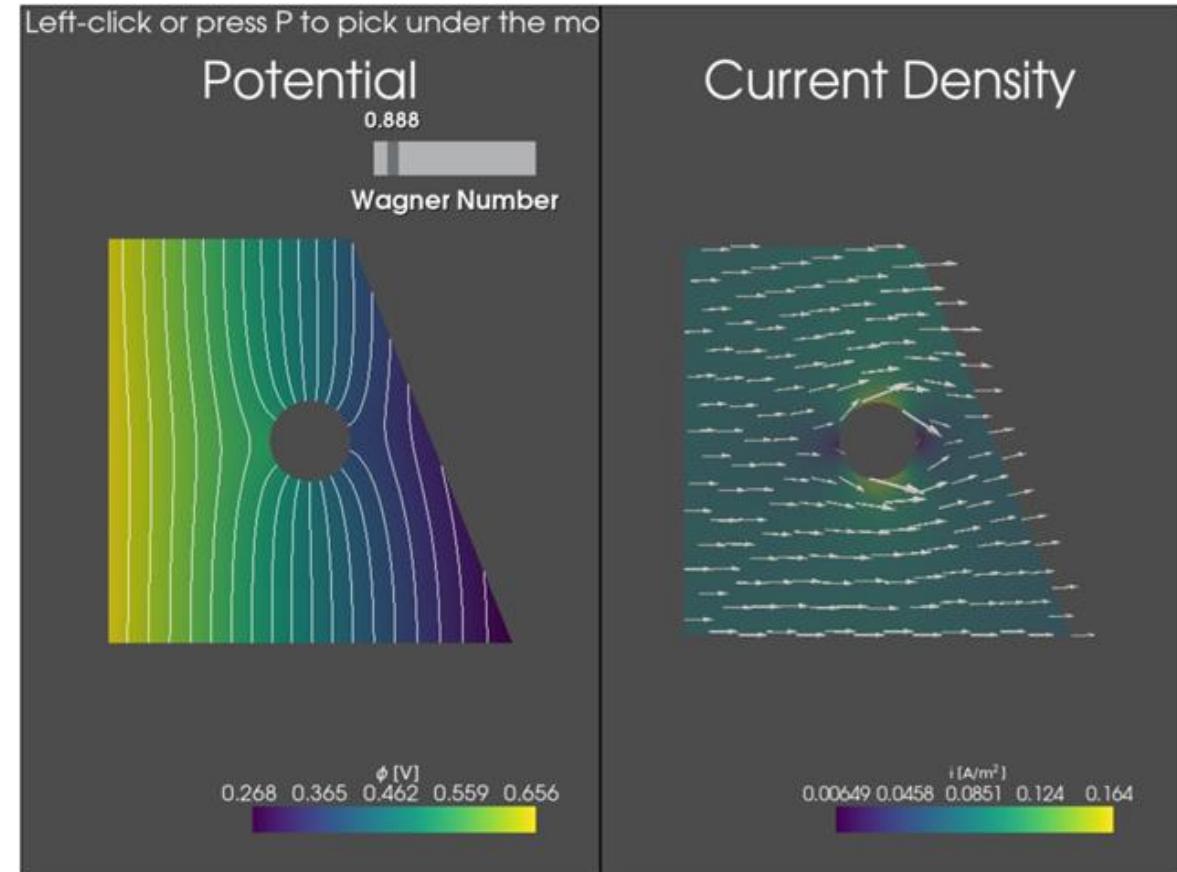
4.3 Python for simulation

PDE Simulation



<https://docs.fenicsproject.org>

FEniCSx: open-source computing platform for solving partial differential equations (PDEs) with the finite element method (FEM)



Journal of The Electrochemical Society

OPEN ACCESS

Application of Open-Source, Python-Based Tools for the Simulation of Electrochemical Systems

Evans Leshinka Molel¹ and Thomas F. Fuller¹

Published 9 October 2023 • © 2023 The Author(s). Published on behalf of The Electrochemical Society by IOP Publishing Limited

[Journal of The Electrochemical Society, Volume 170, Number 10](#)

[Focus Issue on Multiscale Modeling, Simulation and Design: In Honor of Ralph E. White](#)

Citation Evans Leshinka Molel and Thomas F. Fuller 2023 *J. Electrochem. Soc.* **170** 103501

DOI 10.1149/1945-7111/acfcdb

4.3 Python for simulation

MD Simulation

MetalWalls: A classical molecular dynamics software dedicated to the simulation of electrochemical systems

<https://gitlab.com/ampere2/metalwalls>

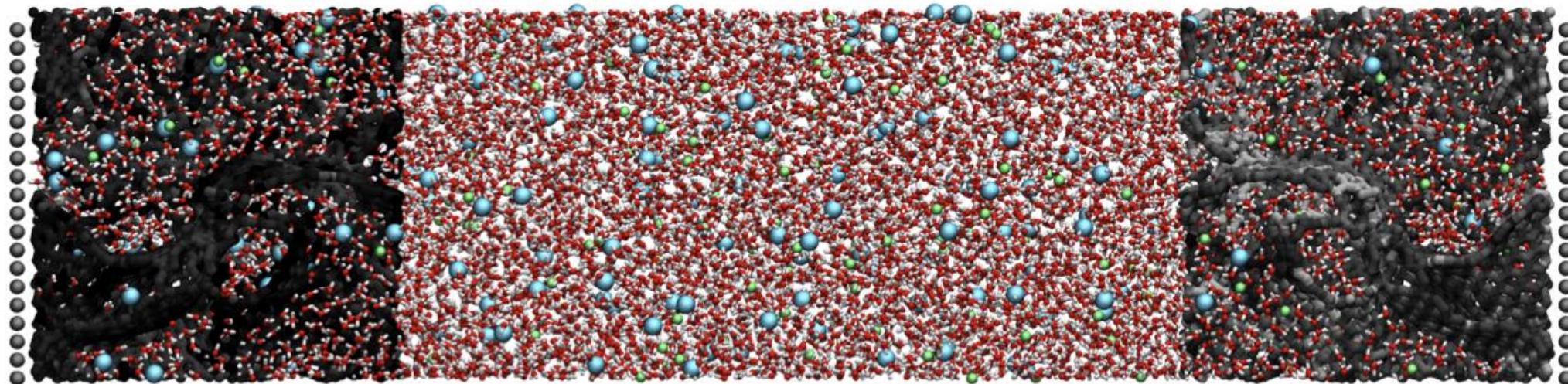


Figure 1: Typical system simulated using MetalWalls. An aqueous NaCl electrolyte is put in contact with two electrified nanoporous carbon electrodes (left: positive electrode, right: negative electrode). Water molecules are in red and white, Cl^- ions in cyan, Na^+ ions in green and the carbon atoms are colored according to their charge (from light grey to black when the charge increases).

4.4 Python for hardware control

Open-Source Scripting

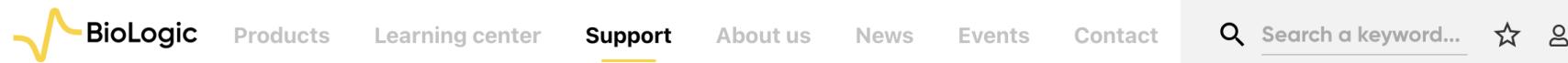
Custom Applications and Open-Source Scripting

Gamry's [software](#) includes all of the traditional techniques and then some. Our software includes over 85 standard techniques and numerous custom techniques. Our own Open-Source Scripting language allows you to modify any technique available. We also have eChem Toolkits that provide you with the ability to utilize development environments such as LabVIEW, C, C++, C+, Python and Visual Basic to develop your own applications. Numerous examples are provided to get you started.

Writing the experimental techniques in an open-source scripting language means that Gamry can easily modify any experiment to your needs. Perhaps you're only interested in acquiring data for the last 100 seconds of a 10 hour potentiostatic hold. Maybe you want to automatically calculate a corrosion rate and tabulate the results for a series of repetitive experiments. Maybe you want to automatically write the results to a database. Gamry's flexibility allows you to do all of these and more.

<https://www.gamry.com/open-source-scripting/>

4.4 Python for hardware control



Information >

EC-Lab® Developer Package

Download

Version history

A variant version of BioLogic's benchmark electrochemistry control and analysis software, designed for programmers/researchers with experience of coding, wishing to utilize the package in highly specialized, customized environments.

Some highly specialized projects require the development of proprietary software to manage Potentiostat / Galvanostats and FRA. For such projects, BioLogic has developed EC-Lab ® Developer Version, which allows programmers to integrate the software into their own software.

This OEM package supports the following instruments:

SP50/SP-150/SP-50e/SP-150e/SP-200/SP-240/SP-300,

VSP/VSP-3e/VSP-300, VMP3/VMP-3e/VMP-300,

MPG-2 series, HCP-803/HCP-1005

The EC Lab® Developer package includes a Dynamic Link Library (DLL) with specific functions for:

- connection/disconnection to a selected instrument,
- initialization of the channels by loading the firmware,
- loading techniques on the channels,
- start/stop the selected channels,
- retrieving data.

The package also includes USB drivers and a library with almost 30 DC and AC techniques (*.ecc files). Different examples with test programs are available in four languages: LabVIEW (including Vis), C+/C++, C# (not for VMP3 instruments), DELPHI, VEEPRO & Python.

Tested configurations:

- C#/C/C++ (32/64 bits)
- LabVIEW (32/64 bits)
- Python (32/64 bits) – example given here https://pyexplabsys.readthedocs.io/drivers/bio_logic.html#

4.4 Python for hardware control

easy-biologic 0.4.0

```
pip install easy-biologic
```

Controller class for communicating with BioLogic devices.

Brian Carlsen

Methods

- **BiologicDevice(address, timeout = 5)**: Creates a new Biologic Device representing the device connected at `address`.
- **connect(bin_file, xlx_file)**: Connects to the device, loading the bin and xlx file if provided.
- **disconnect()**: Disconnects from the device.
- **is_connected()**: Whether the device is connected or not.
- **load_technique(ch, technique, params, index = 0, last = True, types = none)**: Loads a technique onto the given device channel.
- **load_techniques(ch, techniques, parameters, types = None)**: Loads a series of techniques onto the given device channel.
- **update_parameters(ch, technique, parameters, index = 0, types = None)**: Update the parameters of the given technique on the specified device channel.
- **start_channel(ch)**: Starts the given channel.
- **start_channels(chs = None)**: Starts multiple channels.
- **stop_channel(ch)**: Stops the given channel.
- **stop_channels(chs = None)**: Stops the given channels.
- **channel_info(ch)**: Returns information about the given channel.
- **channel_configuration(ch)**: Returns configuration information of the channel.
- **set_channel_configuration(ch, mode, connection)**: Sets the channel's hardware configuration.
- **get_values(ch)**: Returns current values of the given channel.
- **get_data(ch)**: Returns buffered data of the given channel.

4.4 Python for hardware control

Hardpotato: Python API to control programmable potentiostats.

analytical
chemistry

pubs.acs.org/ac



Technical Note

Hard Potato: A Python Library to Control Commercial Potentiostats and to Automate Electrochemical Experiments

Oliver Rodríguez,* Michael A. Pence, and Joaquín Rodríguez-López*

Cite This: *Anal. Chem.* 2023, 95, 4840–4845

Read Online

Currently, the following potentiostats are included in the library:

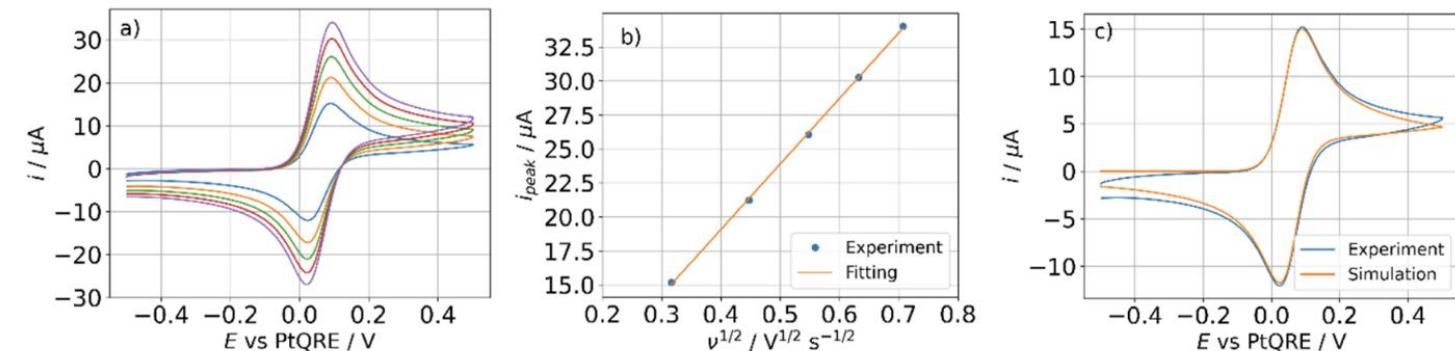
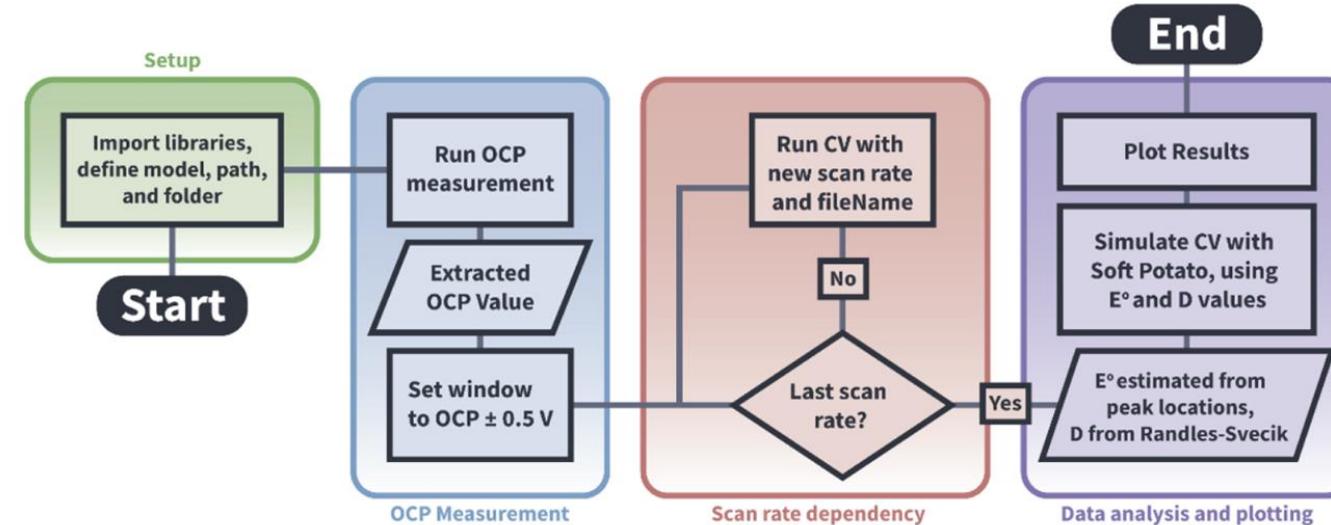
- CHI1205B from CH Instruments (chi1205b)
- CHI1242B from CH Instruments (chi1242b)
- CHI601E from CH Instruments (chi601e)
- CHI760E from CH Instruments (chi760e)
- Emstat Pico from PalmSens (emstatpico)

with the following techniques:

- Cyclic voltammetry, CV
- Chronoamperometry, CA
- Linear sweep voltammetry, LSV
- Open circuit potential, OCP

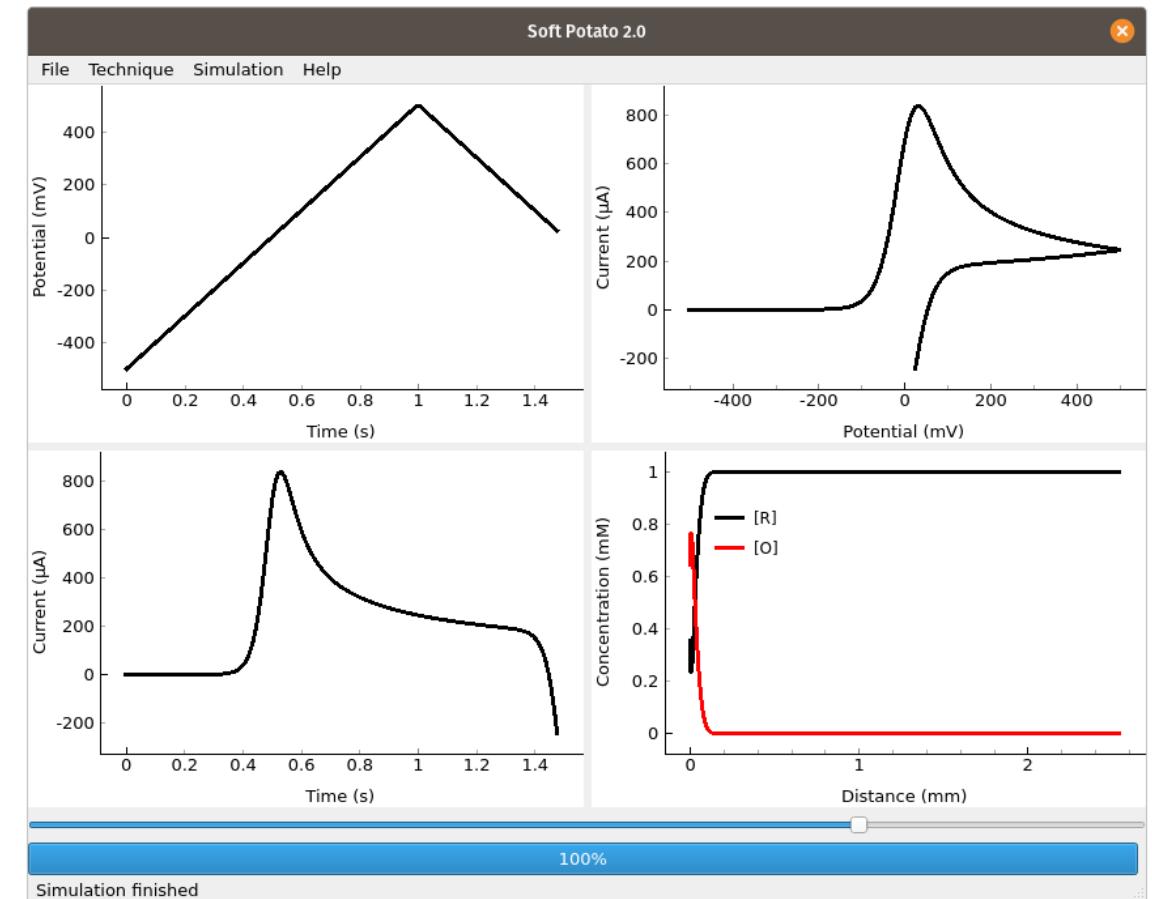
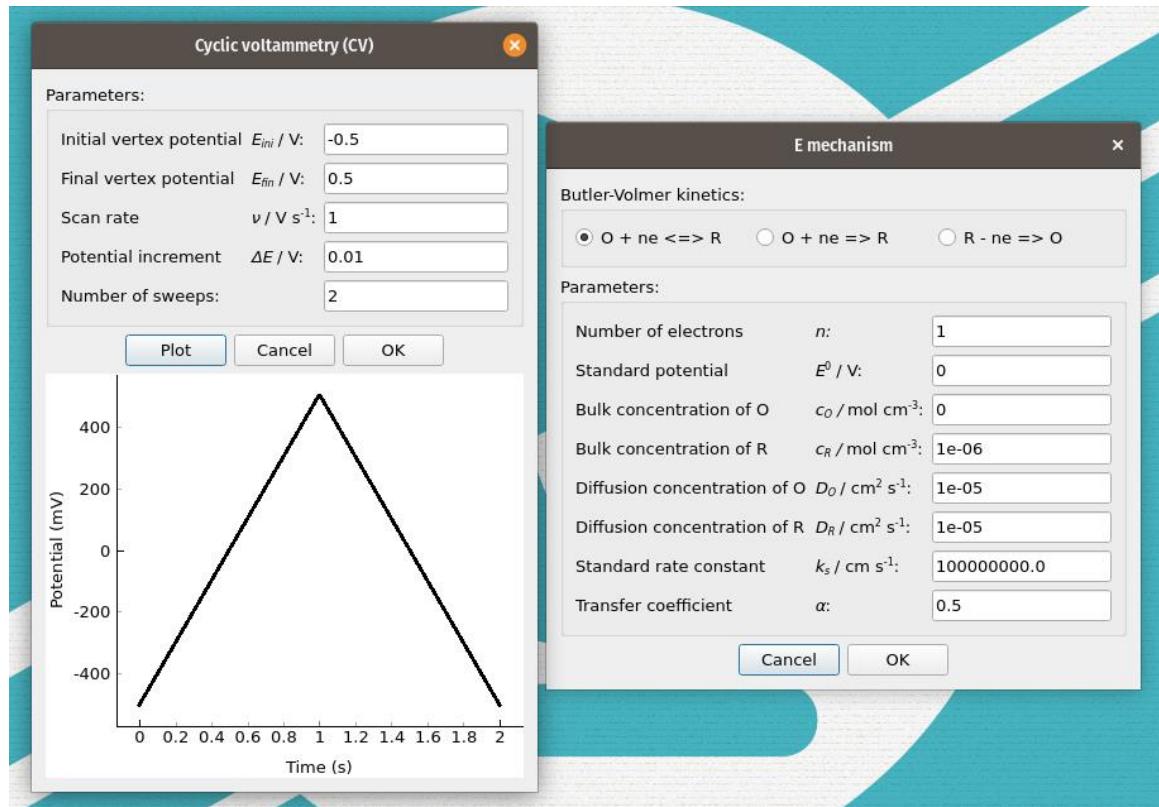
For the CHI601E and CHI760E only:

- Normal pulse voltammetry, NPV



4.4 Python for hardware control

Softpotato: Python tools for CV simulation (with GUI)



4.4 Python for hardware control

LABS: Python-based hardware control for electrochemical synthesis



Research Article | Open Access | CC BY

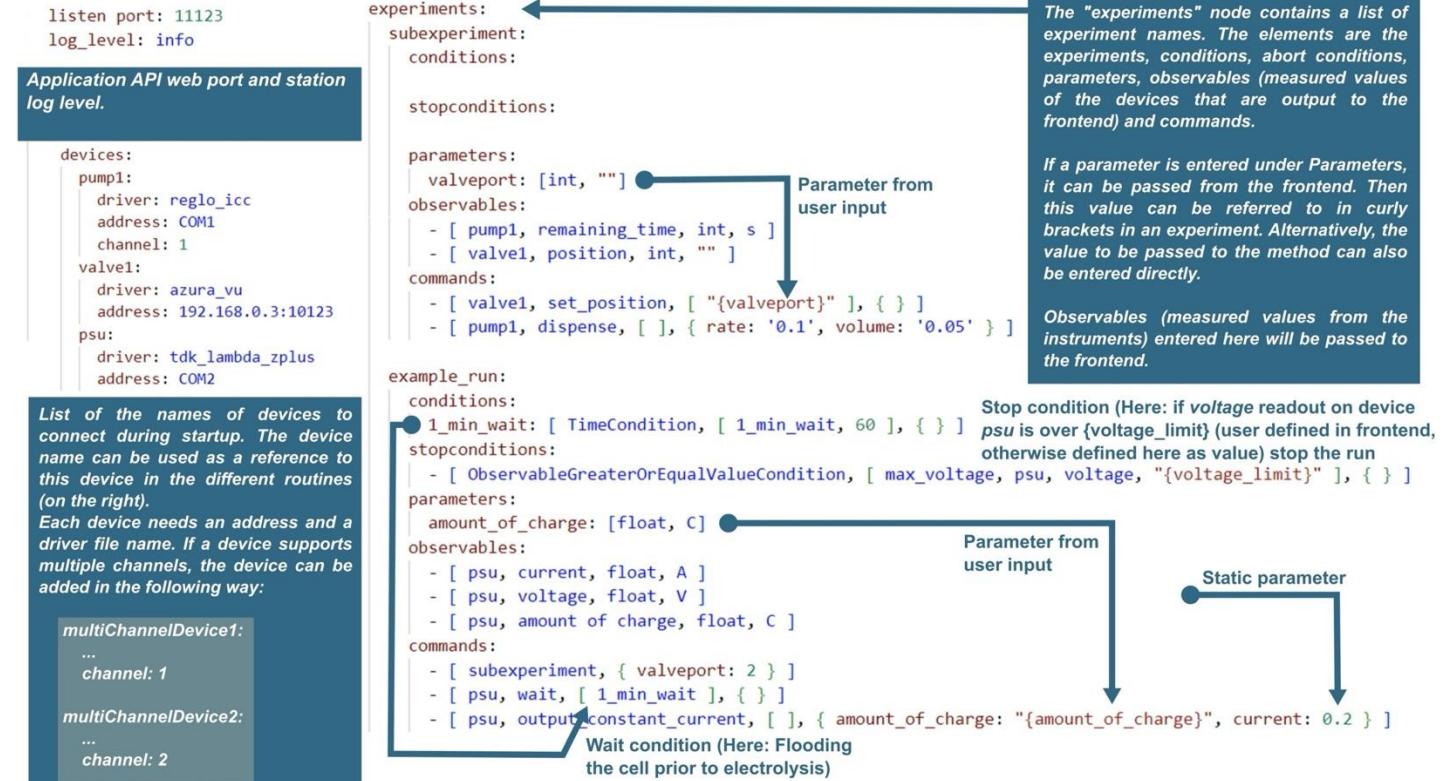
LABS: Laboratory Automation and Batch Scheduling – A Modular Open Source Python Program for the Control of Automated Electrochemical Synthesis with a Web Interface

Maximilian M. Hielscher, Maurice Dörr, Johannes Schneider, Prof. Dr. Siegfried R. Waldvogel✉

Dedicated to Prof. Dr. Dennis Curran on the occasion of his 70th birthday.

First published: 03 June 2023 | <https://doi.org/10.1002/asia.202300380>

Methods Config File Example



5.1 Advice on learning Python – User scripts

- **Read package documents**
- **Practice**
 - Scripts in github.com
 - Scripts in papers

 ScottSoren	part 1 figure 1 with ixdat exported data	c784ba4 · 2 years ago	25 Commits
 data_export	part 1 figure 1 with ixdat exported data	2 years ago	
 data_processing	debug for EC-MS figures (ixdat v0.2.x)	2 years ago	
 data_raw	READMEs of subfolders	2 years ago	
 figures	part 1 figure 1 with ixdat exported data	2 years ago	
 src/pyOER	part 1 figure 1 with ixdat exported data	2 years ago	
 tables	Merge branch 'main' into move_leis	2 years ago	
 .gitignore	part 1 figure 1 with ixdat exported data	2 years ago	
 LICENSE	Initial commit	2 years ago	
 README.rst	part 1 figure 1 with ixdat exported data	2 years ago	

<https://github.com/ixdat/LowOverpotentialRegime>

Journal of Chemical Education > Vol 100/Issue 8 > Article

Editors' Choice

LABORATORY EXPERIMENT | July 10, 2023

Teaching Heterogeneous Electrocatalytic Water Oxidation with Nickel- and Cobalt-Based Catalysts Using Cyclic Voltammetry and Python Simulation

Jingjing Qiu*, Anneke Moeller, Janet Zhen, Hansen Yang, Lily Din, and Nicole Adelstein

Cite Share Jump to Expand

Practice code in the supporting information

5. Advice on learning Python – User scripts

 TomFuller-electrochemistry	Update readme.md	144796b · 3 years ago	 97 Commits	
 Charge screening and Debye length	Update readme.md	4 years ago		
 Classic CV	Update readme.md	4 years ago		
 Continuous CV with infinite kinetics	Update readme.md	4 years ago		
 Cottrell Equation	Update readme.md	3 years ago		
 Current distribution in porous electrode	Updat	 Ragone Plot for Battery	Add files via upload	4 years ago
 EIS for Randles Circuit	Updat	 Secondary Current Distribution	Update readme.md	3 years ago
 Equilibrium potential for lead-acid cell	Updat	 Separator Li-ion Battery	Update readme.md	3 years ago
 Getting started with Python	Updat	 Simplifications to Butler-Volmer Equation	Add files via upload	4 years ago
 Kinetic expressions in full cells	Updat	 Transport properties binary concentrated el...	Add files via upload	4 years ago
 Polarization Curve, Fuel Cell	Updat	 Transport properties dilute binary electrolyte	Add files via upload	4 years ago
		 README.md	Update README.md	4 years ago

Learn by
practicing!

Tom Fuller (Georgia Tech) et al.

<https://github.com/TomFuller-electrochemistry/Python-Simulations-for-the-Education-of-Electrochemists-and-Electrochemical-Engineers>

5.2 Advice on learning Python – ChatGPT

Using Python, provide a script for a 3D surface plot using variables x, y, and z.

GPT-4o

Certainly! Below is a Python script using Matplotlib to create a 3D surface plot with the variables x, y, and z.

```
python
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D

# Generate data
x = np.linspace(-5, 5, 100)
y = np.linspace(-5, 5, 100)
x, y = np.meshgrid(x, y)
z = np.sin(np.sqrt(x**2 + y**2))

# Create 3D plot
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')

# Plot surface
ax.plot_surface(x, y, z, cmap='viridis')

# Labels and title
ax.set_xlabel('X axis')
ax.set_ylabel('Y axis')
ax.set_zlabel('Z axis')
ax.set_title('3D Surface Plot')

# Show plot
plt.show()
```

GPT-4o

When running the code, I got an error: "numpy not installed", help me to debug.

To resolve the "numpy not installed" error, you need to install the numpy package. You can do this using pip. Open your terminal or command prompt and run the following command:

```
bash
pip install numpy
```

If you also need to ensure matplotlib is installed, run:

```
bash
pip install matplotlib
```

After installing, try running your script again.

Debug and learning

Write scripts for certain tasks

**Thanks for listening
and
Happy coding in Python**