Soft Potato: an open source electrochemical simulator and toolkit

Oliver Rodríguez*
Email: oliver.rdz@softpotato.xyz

1 Abstract

Here, I introduce Soft Potato, an open source electrochemical simulator and toolkit that contains typical electrochemical equations and simulations to aid researchers streamline their data analysis when dealing with large datasets. As examples, I show here how to use it to calculate steady state voltammmograms from ultramicrodisc electrodes and obtain a calibration curve from the limiting current as a function of the redox species concentration. This example, showcases the use of the library, as well as the flexibility given by interfacing with third-party libraries to perform data analysis. The second example consists on simulating macroelectrode voltammmograms with varying electron transfer rate constants to automate the simulations by iterating over any parameter. Overall, I believe this open source library can be of great help to the electrochemistry community to easily streamline data analysis and potentially obtain immediate feedback for following experiments.

2 Introduction

Electrochemistry, being a measurement science, requires heavy data analysis and even simulations. For the latter, some excellent tools are readily available, the commercial ones being used in research and the open source options being left as teaching aids in undergraduate and graduate introductory courses. Electrochemical data analysis however, is still mostly done in a manual way by using software such as Origin or even Excel to import data, perform operations and plot the results. Although this is arguably enough for simple experiments, handling data for more complex experiments or with larger quantities of files is cumbersome. This can be automated with the help of programming languages such as Python and Matlab, however, it requires good enough programming skills.

Here, I introduce the softpotato Python library and its Desktop application version. The former contains typical electrochemical equations and a simulator that can be easily used to corroborate or compare experimental results, while the latter is thought as an alternative for teaching and easy visualization. The goal is

*No affiliation.
to decrease the entry level barrier that poses learning a new programming language and help electrochemical researchers to automate data analysis and obtain immediate results after the experiments are finished. Data generation and collection is nowadays easily done via the potentiostat software, where most of them provide some sort of programmable capabilities to vary parameters and automate data extraction and saving. Doing this, an electrochemist can easily generate large data sets that require analysis, becoming a challenge if doing it manually. Instead, the softpotato library includes equations that are part of the electrochemistry toolbox such as Cottrell, Randles-Sevcik, limiting current of microdisc electrodes and their current responses, as well as the simulation of macroelectrodes. The library can be used for corroborating results or even fitting experiments to extract electrochemical parameters immediately after finishing a set of experiments, as the calculation is essentially instantaneous. This can help decrease the time spent by a researcher and improve their efficiency, as they will have immediate feedback to make quick decisions on the following experiments. In the following, I introduce the application programming interface (API) and two examples on how to use the library to help the reader on the implementation of typical case scenarios. For help on how to install the library and more information, please see Ref. [1, 2].

3 API implementation

The softpotato library contains four main modules that can be accessed by the user. The technique module contains classes to create potential and time arrays for potential sweeps and potential steps, it also contains the Custom class that allows us to construct a combination of potential waveforms. With the exception of electrochemical impedance spectroscopy, basically any other amperometric electrochemical technique can be constructed with a combination of sweeps and steps. For example, sampled current voltammetry consists on applying incremental potential steps from a base potential [3, 4, 5], stripping voltammetry consists on a potential step to adsorb a species of interest followed by a sweep to desorb it. The inclusion of the Custom class gives the user enough flexibility to build their own potential waveforms according to their needs.

The calculate module contains classes to calculate currents for macro, microdisc and microband electrodes. The Macro class can calculate the Cottrell and Randles-Sevcik responses, given the expected parameters. The MicroDisc class calculates the steady state voltammetry according to Equation 33 from reference [6] and the transient current for any potential according to Equation 5 from reference [5]. The MicroDisc class then takes into account electron transfer kinetics or mass transport only currents as needed. The MicroBand class on the other hand, can only calculate the expected quasi-steady state limiting current, as there is no known analytical solution or approximation to the current transient or voltammetry. In general, the calculate module contains equations for general current responses that can be easily included in a single script for data analysis and to corroborate experimental results.

The simulate module solves Fick’s laws of diffusion with finite differences, assuming planar diffusion and Butler-Volmer kinetics. At the time of writing, only the E mechanism is implemented, however, other
mechanisms are being developed (adsorption, capacitance, EC, EC', CE, ECE) as well as other diffusional regimes (sphere, disc and band). Finally, the plotting module contains functions for easy plotting and data visualization.

### 3.1 Graphical user interface

A desktop application has also been developed [7]. Figure 1 shows a screenshot on the Linux Desktop. The current version 2.0, can simulate voltammograms in the same manner as the simulate module of the softpotato Python library. Version 3.0 is being developed and it will wrap around the softpotato library to include the calculate module as well. Currently, the Desktop application can be run in Windows and Ubuntu-based Linux distributions. MacOS and other Linux-distribution users and developers can clone the GitHub repository [7] and run it as a typical Python program. The Desktop application is thought as an easy to use software for teaching and quick analysis, as the GUI does not require the user to be proficient in Python. It also includes a slide bar at the bottom of the application to control the time to easily visualize the change in concentration profile and its effect on the voltammetry.

![Figure 1: Screenshot of the Soft Potato Desktop application.](image)

### 3.2 Example: calibration curve with a microdisc electrode

To showcase the flexibility of having a script-based library of electrochemical equations, I am using the calculate module to obtain linear sweep voltammograms of an ultramicrodisc electrode with varying concentrations of the reduced species, Figure 2 shows the results (please see Section 6 for the script used here). I have included a value for the normally distributed random noise to include variation in the currents; this is done by giving a value to the noise parameter in the LSV function, by default this is set to zero. Figure 2.
shows the obtained voltammograms for each concentration, small variations in the current are due to the added noise. Figure 2a shows the voltammograms normalized by their respective limiting currents, where the added noise is more evident. Although the MicroDisc class calculates the expected limiting current by default, here it was obtained by averaging the currents from 0.2 to 0.5 V to account for the variation given by the noise. Figure 2b shows the limiting currents as a function of the bulk concentrations, as well as the linear fitting obtained using the linregress function of the scipy Python library. This showcases the flexibility provided by softpotato, as it can be interfaced with third-party libraries to perform data analysis, plotting, etc. The electrochemical equations implemented in softpotato can even be used to fit experimental results by using the curve_fit function from scipy.optimize, which requires the user to write their own objective function. In this way, any function from the softpotato.calculate library can be used as a parameter in curve_fit to estimate electrochemical parameters from experimental results.

Figure 2: Results of the calibration curve example. a) LSVs from microdisc electrodes with a radius of 5 μm, b) LSVs normalized by their respective limiting current and c) calibration curve with its fitting.

3.3 Example: simulation of cyclic voltammograms with macroelectrodes

Here, I show the results of a CV simulated with the simulate module. As previously mentioned, the module uses finite differences to solve Fick’s laws of diffusion assuming planar diffusion and Butler-Volmer kinetics. The simulation requires the waveform to be created with the technique module and passed along with the electrochemical paramters. Here, I am iterating on the electron transfer rate constant to see the effect that electrochemical kinetics has on the voltammogram. See Figure 3 for the results and Section 7 for the script used here. This showcases how powerful the library is compared to graphical user interface alternatives, including the Soft Potato Desktop application. Iterating over any parameter or combination of parameters is quite easy, provided the user has some minimum knowledge on Python programming. In contrast, GUI desktop applications are too limited in this regard, since the developer needs to manually implement variation on each of the parameters to account for all the possible combinations that a user may need.
Figure 3: Cyclic voltammograms simulated with softpotato with varying electron transfer rate constants.

4 Conclusions

Here, I have presented softpotato, a Python library that includes typical electrochemical equations and can simulate electrochemical processes assuming planar diffusion and Butler-Volmer kinetics. The library is thought as a tool to aid in the analysis and simulation of electrochemical experiments, everything on a single script. In this way, data analysis and simulations can be automated to obtain results immediately after the experiments are performed, opening the possibility of having live feedback for the next experiments. To the best of my knowledge, and at the time of writing, there is currently no other electrochemistry tool similar to softpotato that can enable a researcher to do this. Although open source general simulators already exist, they are used mostly as tools for teaching. With softpotato and the Soft Potato Desktop application, it is my goal to help turning electrochemistry into a data science, where computers are used in a more efficient manner to solve complex problems and perform data handling. It is my hope that softpotato can be embraced by the community at large to help them the same way it has helped me in my own research.

5 Acknowledgements

I want to acknowledge my PhD supervisor, Prof. Guy Denuault for all the long discussions regarding fundamental electrochemistry and simulations during my time as his trainee. Soft Potato would not exist without his guidance and help. I also want to acknowledge Marko M. Melander, Johan Hjelm and Steven Linfield for their valuable contributions to the Soft Potato Desktop and softpotato library.

References


6 Script of Example 1

```
import softpotato as sp
import numpy as np
import matplotlib.pyplot as plt
from scipy.stats import linregress

# Potential waveform in V:
E = np.linspace(-0.5, 0.5, 100)

# Electrochemical parameters:
a = 5e-4  # cm, electrode radius
n = 1  # number of electrons
DO = 1e-5  # cm2/s, diffusion coefficient of species O
DR = 1e-5  # cm2/s, diffusion coefficient of species R
cOb = 0  # mol/cm3, bulk concentration of species O
E0 = 0  # V, equilibrium potential
k0 = 1e8  # cm/s, electron transfer rate constant
alpha = 0.5  # transfer coefficient
noise = 0.01e-9  # A, normally distributed random noise

# Array of concentrations of R in mol/cm3:
cRb = np.array([0.5, 1, 2, 5, 10])*1e-6

# Calculation of LSVs:
nc = cRb.size
iLim = np.zeros(nc)
lsv = np.zeros([E.size, nc])
for x in range(nc):
    ume = sp.calculate.MicroDisc(a, n, DO, DR, cOb, cRb[x], E0, k0, alpha,
            noise=noise)
    lsv[:, x] = ume.LSV(E)
iLim[x] = np.mean(lsv[80:, x])

# Data analysis:
res = linregress(cRb, iLim)
```
fit = res.intercept + res.slope*cRb

# Plotting:
sp.plotting.plot(E, lsv*1e9, ylab='$i\omega/\mu A$', fig=1, show=0)
sp.plotting.plot(E, lsv/iLim, ylab='$i/i_{\text{lim}}$', fig=2, show=0)

plt.figure(3)
plt.plot(cRb*1e6, iLim*1e9, 'o', label='Simulation')
plt.plot(cRb*1e6, fit*1e9, '- ', label='Fitting')
sp.plotting.format(xlab='$C_R/\mu M$', ylab='$i/\mu A$', legend=[1], show=1)
7 Script of Example 2

```python
import softpotato as sp
import numpy as np

# Potential waveform parameters:
Eini = -0.5    # V, initial potential
Efin = 0.5     # V, final potential
sr = 0.1       # V/s, scan rate
dE = 0.01      # V, potential increment

# Potential waveform created with softpotato
wf = sp.technique.Sweep(Eini=Eini, Efin=Efin, sr=sr, dE=dE)

n = 1          # number of electrons
A = 1          # cm², geometrical area
E0 = 0         # V, equilibrium potential
cOb = 0        # mol/cm³, bulk concentration of species O
cRb = 1e-6     # mol/cm³, bulk concentration of species R
DO = 1e-5      # cm²/s, diffusion coefficient of species O
DR = 1e-5      # cm²/s, diffusion coefficient of species R
alpha = 0.5    # transfer coefficient

# Array of electron transfer rate constants in cm/s:
k0 = np.array([1e2, 1e1, 1e0, 1e-1, 1e-2, 1e-3, 1e-4, 1e-5])

# Simulation of CVs:
nk = k0.size
i = np.zeros([wf.t.size, nk])
for x in range(nk):
    sim = sp.simulate.E(wf, n, A, E0, cOb, cRb, DO, DR, k0[x], alpha)
    sim.run()
    i[:,x] = sim.i

# Plotting:
sp.plotting.plot(wf.E, i*1e3, ylab='$i$ $\mu$A', fig=1, show=1)
```