NOTICE: This is the author's version of a work accepted for publication by Elsevier. Changes resulting from the publishing process, including peer review, editing, corrections, structural formatting and other quality control mechanisms, may not be reflected in this document. Changes may have been made to this work since it was submitted for publication. A definitive version will subsequently be published in BIOELECTROCHEMISTRY and available at http://dx.doi.org/10.1016/j.bioelechem.2009.02.010

# SOAS: a free program to analyze electrochemical data and other one-dimensional signals

Vincent Fourmond[a,b], Kevin Hoke[c], Hendrik A. Heering[d], Carole Baffert[a,b], Fanny Leroux[a,b], Patrick Bertrand[a,b], Christophe Léger[*,a,b]

[a] *Unité de Bioénergétique et Ingénierie des Protéines, UPR 9036, CNRS. 31 Chemin Joseph Aiguier, F-13402 Marseille Cedex 20, France.*
[b] *Aix-Marseille Université, 3 Place Victor Hugo, F-13333 Marseilles Cedex 3, France. CNRS, Marseille, F.*
[c] *Department of Chemistry, P.O. Box 5016, Berry College, Mt. Berry, Georgia, 30149 USA.*
[d] *Leiden Institute of Chemistry, Leiden University, Einsteinweg 55, 2333 CC Leiden, The Netherlands.*

**Abstract**

This paper describes an open source program called SOAS, which we developed with the aim of analysing one-dimensional signals. It offers a large set of commands for handling voltammetric and chronoamperometric data, including smoothing signals, differentiation, subtracting baselines, fitting current responses, measuring limiting currents, and searching for peak positions. Although emphasis is on the analysis of electrochemical signals, particularly protein film voltammetry data, SOAS may also prove useful for processing spectra. This free program is available by download from the internet, and can be installed on computers running any flavor of Unix or Linux, most easily on MacOS X.

*Key words:*
open source software, electron transfer, protein film voltammetry, data analysis

[*]Corresponding author.
*Email address:* `christophe.leger@ibsm.cnrs-mrs.fr` ()
*URL:* `http://bip.cnrs-mrs.fr/bip06` ()

## 1. Introduction

Electrochemistry programs, both free and commercial, are available for several purposes. Data analysis software allow the user to load data files and with some degree of user manipulation find peak positions and other parameters. Such programs are typically included with electrochemical instrumentation (e.g., BAS, EcoChemie, CHI), but often provide only a limited range of options. These limitations have led us and others to develop programs for more efficient and effective processing of experimental data. SOAS, the program described here, is a free command-driven interactive utility for processing and analyzing one-dimensional signals, particularly electrochemical data.

Several programs for data analysis are currently available to electrochemists. For example, the "Electrochemical Science and Technology Information Resource" (ESTIR) website maintains a list of electrochemical freewares (http://electrochem.cwru.edu/estir/). A program with advanced utilities for voltammetric data analysis ("UTILS"), running under a DOS environment, was written by Dr. H.A. Heering and is available as freeware on request (h.a.heering@chem.leidenuniv.nl). Certain options of this program originally inspired SOAS. In addition to data analysis, many programs use a variety of algorithms to simulate processes at the electrode [for an overview, see ref 1], and some simulators also incorporate data-fitting capabilities. Most simple electrochemical simulators follow procedure similar to that outlined by Gosser [2], though more sophisticated simulations of diffusion kinetics incorporate approaches similar to those reported by Rudolph and Feldberg [3]. Of other programs we have recently examined, SIMSENSOR is available from the laboratory of Dr. Jean-Michel Savéant, and simulates responses for catalysis performed by enzymes adsorbed on an electrode surface with their cosubstrates in solution [4] (`http://www.lemp7.cnrs.fr/simsensor/Download-instructions.htm`). Dr. Carlo Nervi's Electrochemical Simulations Package is also available for free download (`http://lem.ch.unito.it/chemistry/esp_manual.html`) and runs under a DOS environment. It can simulate staircase cyclic voltammetry and square wave voltammetry for species in solution, with a mechanism taken from file or keyboard. Dr. Leslaw K. Bieniasz has also published a electrochemical simulator [5], available by email from the author (`http://home.arcor.de/aerbe/en/prog/spectraltools.html`). Dr. W. Huang has developed Electrochemist.com (previously named POLAR) which is available under a shareware/subscription license, with fees scaled according to the number of software features desired. It simulates and fits several mechanisms and techniques, including adsorption mechanisms. It has been reviewed elsewhere [6, 7]. Elchsoft's DigiElch (`http://www.elchsoft.com/DigiElch/Default.aspx`) is also available under a subscription license and performs simulation and fitting. Lars Jeuken's Jellyfit program can fit trumpet plots (peak position against scan rate) for adsorbed redox species using various models (`http://www.mnp.leeds.ac.uk/ljcjeuken/Jellyfit.htm`). Each of these programs runs on IBM PC-compatible systems, but with varying degrees of support. Of freely available programs, the open source Echem++ project has been more frequently updated [8] (`http://www.echem.uni-tuebingen.de/~bs/echem/software/EChem++/echem++.shtml`). Running under Linux, it can model and simulate species in solution and adsorbed on an electrode. It is based on a modular, open-source, object-oriented programming model, where reaction mechanisms and other functions can be customized using the C++ programming language.

Certainly, users may find several of these programs adequate (or superior) for their specific needs. We are careful to note, however, that most of these programs are used for simulating and sometimes fitting electrochemical responses for species in solution (though a few support adsorbed species as well), whereas SOAS has been developed primarily for the rapid and efficient processing of any electrochemical signals, and for fitting protein film voltammetry (PFV) data. Here we will describe some of the features of SOAS, with examples to show its utility in a variety of situations.

From the point of view of the user, the program SOAS can load the ASCII files exported by GPES, CHI, and EG&G software, but it is also possible to use its built-in interpreter of commands to load ASCII data files of any particular format. Although the program can be used to handle any 1-dimensional signal, such as kinetic traces or spectra, it embeds a number of routines that are specifically oriented towards electrochemical data analysis. The user can select any portion of the signal, remove spikes or regular noise, and correct it by subtracting a blank or a linear or polynomial baseline (a number of options are available for either catalytic and non-catalytic data). The user can also measure limiting currents or slopes, search for peaks and measure their characteristics (e.g., position, intensity, area, half-widths). The data sets can be transformed (e.g., for correcting a potential scale, normalizing or taking the log of the current, or applying any other simple modification), and fitted to either built-in or user-defined functions.

The user interacts with the program by typing the name of pre-defined commands (as in other programs, like GNUPLOT for example, `http://www.gnuplot.info/`), and many routines let the user change certain parameters by using the mouse. The results are exported as ASCII files. For example, a series of cyclic voltammograms at varying scan rates can be processed rapidly to characterize the shape of oxidation and reduction peaks. The resulting output file would contain the key metrics for peaks in each voltammogram, including peak positions, heights, widths and areas. Thus, the user can rapidly assess changes in peak position as a function of scan rate, or find peak areas in order to calculate electroactive surface coverage.

From a technical point of view, SOAS uses the core of another program called GILDAS (`http://www.iram.fr/IRAMFR/GILDAS/`, this acronym stands for "Grenoble Image and Line Data Analysis Software"). GILDAS is a collection of software oriented towards radioastronomical applications, and is free for non-profit organizations. Its kernel consists of a command line interpreter called SIC and an all-purpose graphical program called GREG ("Grenoble Graphics"), which we used to draw the figures of this paper. GREG can be used either interactively, or in a non-interactive mode using scripts, or in the form of a library of subroutines callable from a third-party application like SOAS. After SOAS is started, the user can either handle the data with the commands provided by SOAS, or enter interactive GREG and use its command line interpreter, which extends the capabilities of SOAS. SOAS also interfaces with certain software packages available on the NETLIB repository (`http://www.netlib.org`), including a fast Fourier transform utility (FFTPACK) and a very efficient software routine for non-linear regression (ODRPACK) [9].

Unlike GREG or GNUPLOT, SOAS is not intended to be a plotting program: it does not let the user define how the data are displayed, nor does it export figures, although screen snapshots can be sent to the printer or saved as Postscript files. The normal output is an interactive plot window or ASCII files. The latter are readily opened in a user's preferred graphing/spreadsheet applications for additional processing or presentation.

SOAS is available for download (`http://bip.cnrs-mrs.fr/bip06/software.html`), and can be installed on computers running any flavor of Unix. A one-click installer for Apple computers running Mac-OS 10.3 and above is also available. As SOAS is an open source program, users can freely edit, modify and redistribute the code under the terms of the GNU General Public License. We will welcome and incorporate in the version we distribute the contributions we receive.

Here, our goal is to give an idea of what the program can do by describing its most useful features, rather than explaining the syntax of every command; the latter information can be found in the online manual (`http://bip.cnrs-mrs.fr/bip06/soas/`).

## 2. An overview of selected features

### *2.1. The basics*

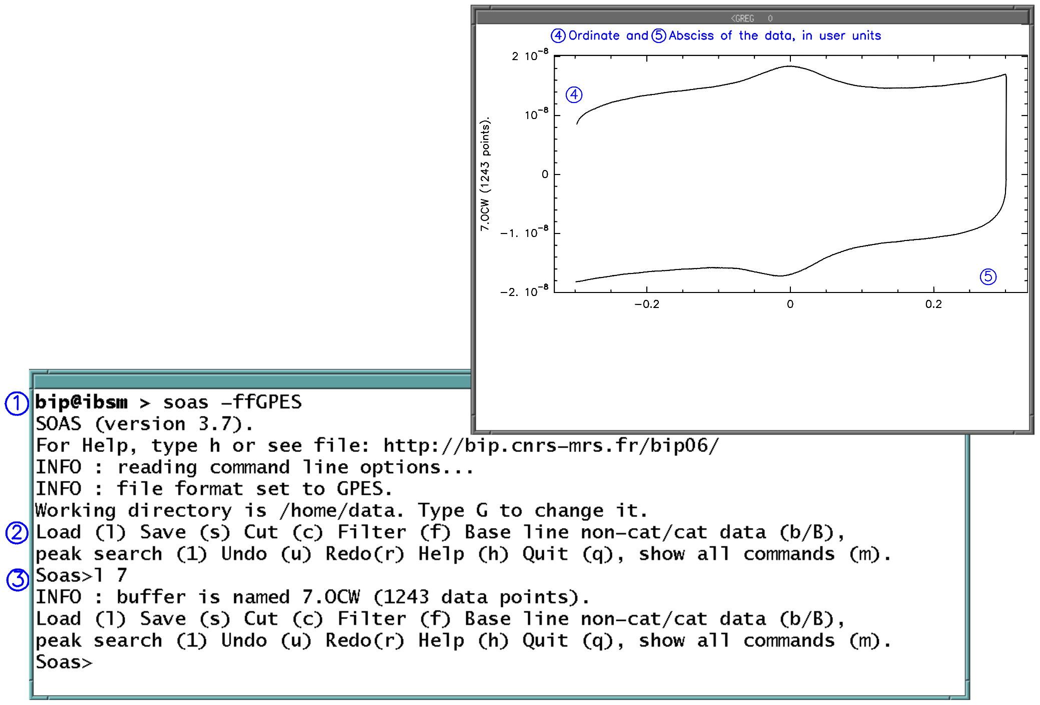
#### *2.1.1. Starting the program*

A user launches the program by typing "`soas -ffGPES`" in an X terminal (the "`-ffGPES`" option is specific to EcoChemie Autolab users). The prompt becomes "`Soas>`" and a graphic window pops-up (Figure 1). The commands will be typed in the terminal. A command history log allows previous commands to be retrieved with the up and down arrows of the keyboard, and then edited for execution. The readline function used in SOAS is identical to that used in GNU Bash and GNU Emacs: for example `ctrl K` deletes from the current word to the end of line, etc. Automatic completion of command names and filenames (e.g. when the user is loading a file) is possible with the TAB key. Many commands let the user select options and parameters by clicking with the mouse in the plot window and/or using keys on the keyboard. The list of actions that are available while any command is being used is displayed in the terminal, unless the user does not need such prompting and chooses to work in "expert" mode.

#### *2.1.2. The stack*

SOAS internally stores the data sets as "buffers" and manages them in a "buffer stack". Every time an operation changes a data set (such as file loading, baseline subtraction, filtering), the former data set is pushed onto an "undo stack". These data can be recovered with an "undo" function. The undo stack has a limited size of 10 operations, with a "first-in first-out" replacement procedure when the stack is full. Older operations cannot be recovered. We are planning to lift this limit in a future release of SOAS. Complementary to the "undo" stack, there is a "redo" stack, onto which newer data get pushed whenever older operations are undone.

The data sets in the entire stack (undo + redo) can be used for various operations, such as overlaying another buffer on the current buffer or subtracting one buffer from another. The entire contents of the stack can also



```
bip@ibsm > soas -ffGPES
SOAS (version 3.7).
For Help, type h or see file: http://bip.cnrs-mrs.fr/bip06/
INFO : reading command line options...
INFO : file format set to GPES.
Working directory is /home/data. Type G to change it.
Load (l) Save (s) Cut (c) Filter (f) Base line non-cat/cat data (b/B),
peak search (1) Undo (u) Redo(r) Help (h) Quit (q), show all commands (m).
Soas>l 7
INFO : buffer is named 7.OCW (1243 data points).
Load (l) Save (s) Cut (c) Filter (f) Base line non-cat/cat data (b/B),
peak search (1) Undo (u) Redo(r) Help (h) Quit (q), show all commands (m).
Soas>
```

Figure 1: Screen shot showing the appearance of SOAS at start up. (The annotations in blue were added after the screen was grabbed.) The first flag, on the left side of Figure 1, points to the command that was typed in the X-terminal in order to launch the program (the option "`-ffGPES`" indicates that the program will load files that were exported by GPES software). A short menu, pointed to by the second flag, lists the most useful commands. The line "`l 7`", at the third flag, loads the file 7.OCW, which is then plotted in the pop-up window.

be displayed in the graphic window, as well as be saved to or loaded from a single data file.

### *2.1.3. The central concept of markers*

For performing a variety of operations, including defining baselines and measuring data coordinates, the user will first place "markers" on the data. This is achieved by clicking with the mouse in the graphic window, and the coordinates of the marker are calculated according to one of three protocols, as illustrated in figure 2. "Exact" uses as marker coordinates the data point that is nearest to the position of the mouse cursor. "Off data" places the marker at the coordinates specified by the mouse position in the graphics window. The "Smooth" protocol is for noisy data sets: in that case the $y$-

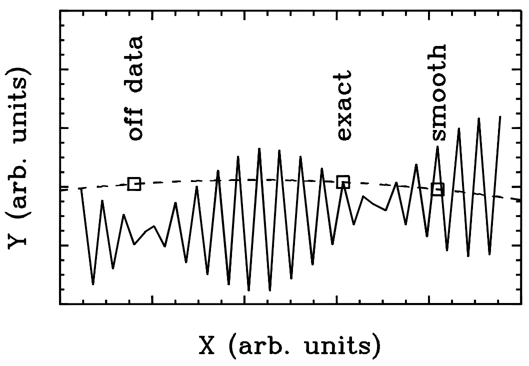


Figure 2: The three types of markers used in SOAS. For certain operations, the user will click with the mouse to define the coordinates of these particular data points (shown as empty squares). The exact position of the marker will depend on whether it is "exact" (exactly on a data point), "off data" (exactly where the mouse was clicked), or "smooth" (calculated after averaging data). For example, these markers will be used to interpolate baselines (dashed line).

coordinate of the cursor is calculated by averaging data on either side of the position of the mouse. Technically speaking, the $y$-coordinate of the marker is calculated from the average of a maximum of 10% of the dataset around the cursor position; a line is fitted through the range of points to be averaged, and the range is decreased until less than range/4 consecutive residuals with the same sign are found. That is, the range must contain at least 2 periods of noise and must not span a significantly non-linear part of the data.

### *2.2. Removing spikes and regular noise*

Spikes are erroneous data points that are clearly offset from the data. SOAS offers separate commands for removing them manually (simply by clicking close to where the wrong data point is) or automatically. In the latter case, the program calculates the average and variance of the absolute increments $|y(i+1) - y(i)|$ over the entire data set. A threshold value is calculated from the value of average plus 200 times the variance. A spike is defined as a point $y(i)$ such that $|y(i) - y(i-1)|$ and $|y(i+1) - y(i-1)|$ are greater and lower than the threshold, respectively. The $y$- value of the erroneous data point is replaced with the average of $y(i-1)$ and $y(i+1)$. This definition is generalized to allow for spikes consisting of up to 5 data points.

In Figure 3, we illustrate the effect of removing irregular noise from the raw voltammogram plotted with a black line in panel A. Panel B shows what is subtracted from the data by the command that automatically detects and removes spikes. Panel C shows the contribution that can be further subtracted when the user selects with the mouse the data points that should be deleted. The bottom panel is the regular noise subtracted by using a

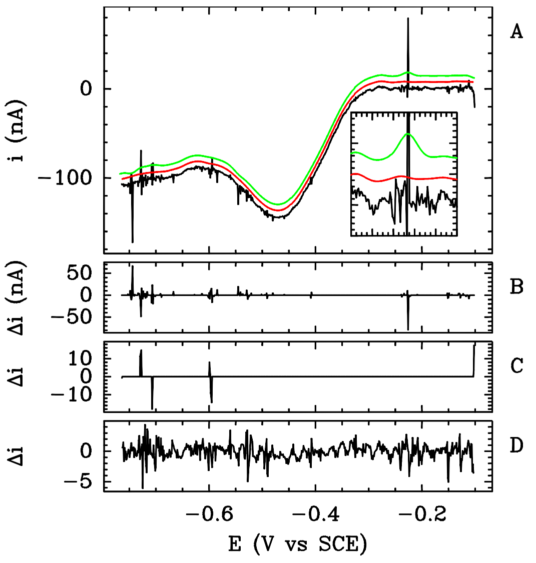


Figure 3: Illustration of the smoothing capabilities of SOAS. Starting from the raw signal plotted in black in Panel A, the command that automatically detects spikes was applied first. This removed the contribution shown in Panel B. More spikes were removed "by hand", using the command that lets the user clicks near the data points that should be deleted. The corresponding residue is shown in Panel C. Last, the regular noise shown in Panel D was removed using the Fourier procedure. This returned the filtered signal shown in red and slightly offset from the raw data in Panel A for clarity. The inset in panel A is a magnified portion of the voltammogram. The green signal is the result of smoothing regular noise without having removed the spikes first.

Fourier transform procedure (see below). The resulting, smoothed signal is shown in red on the top panel. Performing this series of corrections won't take more than 30 sec using SOAS. The green data show the result of an unsuccessful attempt to remove regular noise *before* taking out the spikes.

Regarding regular noise, SOAS offers two main options for smoothing the data. One is to convolve the signal with a Gaussian curve, the width of which is adjusted so as to remove the high frequency noise. This is achieved by using a Fourier transform. The interface is friendly in that the user can continuously adjust the size of the filter and see at the same time the raw data, the filtered data and the difference between the two. The latter should be used as a control, to decrease the risk of over-filtering and distorting the data. Regular noise should oscillate evenly over the course of the signal and the displayed difference should reflect this. Overfiltering is readily identified when the difference between the raw and smoothed data displays peaks or valleys. In this extreme, the filter is operating on frequencies comparable to that of the actual signal, and relevant data is being suppressed. Alternatively, rather than examining the difference between the raw and smoothed data, the user can choose to visualise the power spectrum of the raw and smoothed signals (Fig. 4). The reciprocal size of the optimized filter is easily read from

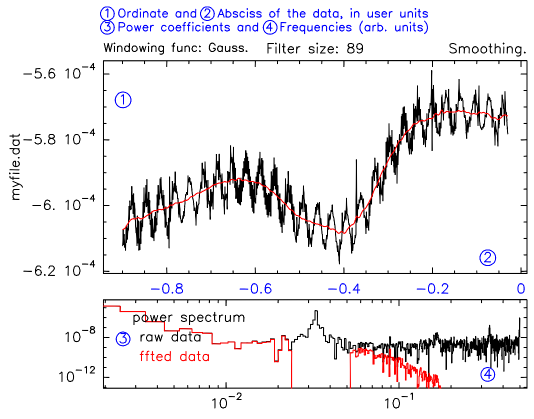

① Ordinate and ② Abscíss of the data, in user units
③ Power coefficients and ④ Frequencies (arb. units)
Windowing func: Gauss.
Filter size: 89
Smoothing.
myfile.dat
power spectrum
raw data
ffted data

Figure 4: The plot window of SOAS while the user is using the Fourier transform procedure to smooth data. (The annotations in blue were added after the screen was grabbed.) The raw data are shown on the upper panel (in black), together with the smoothed signal (in red). The lower box may be used to display either the difference between the raw and smoothed data or the power spectra of the two signals, as shown here. In this example, the parameters are set so as to remove the contributions of both the high frequencies and an intermediate range of frequencies.

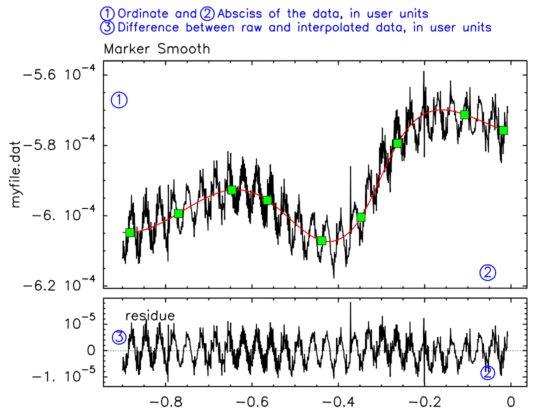
this plot of power against frequency, and it can be selected by the user simply by clicking with the mouse. An option also makes it possible to set to zero the Fourier coefficients that correspond to a user-defined intermediate range of frequencies (e.g. to remove 50Hz mains interference).

Alternatively, the user can approximate the data by setting a few carefully chosen points (using the markers discussed above) and let the program calculate the smooth function that passes through all those points, as illustrated by the screen capture in Figure 5. For this, SOAS uses a spline procedure [10] to interpolate a third order polynomial in between each pair of consecutive markers.

## *2.3. Differentiating*

The above two methods for smoothing the data can also be used within SOAS to differentiate a signal. One of the two available commands is based on the Fourier transform routine (Figure 4). This is because, from a technical point of view, it is very easy to compute a derivative from the Fourier transform: SOAS only has to multiply the Fourier coefficient by the frequency to the power of $n$ to obtain the $n$-th order derivative of the signal. Therefore, the commands for differentiating and filtering the signal have the same interface (Figure 4) and options, and indeed they are based on the same code. This FFT-based procedure was used to differentiate the catalytic data for hydrogen oxidation by *A. vinosum* NiFe hydrogenase in fig 2C of ref [11].

Alternatively, while the user is smoothing the data by fitting them with a spline function (Figure 5), the program can return the derivative of the spline



① Ordinate and ② Absciss of the data, in user units
③ Difference between raw and interpolated data, in user units
Marker Smooth
myfile.dat
residue

Figure 5: The graphic window of SOAS when the spline command is being used. (The annotations in blue were added after the screen was grabbed.) Spline interpolation is useful for smoothing data (as shown here) or for removing baselines. In either case, the user sets markers (green squares in the top panel) that approximate the data (in black). The calculated spline function, shown in red, passes through all markers. The lower panel displays the difference between the raw data and the spline function. Smoothing the data may be achieved by setting evenly spaced "smoothed" markers over the entire range of data (as illustrated) and replacing the data with the spline function. To subtract a baseline, the markers would instead be placed before and after a peak, and the resulting spline function subtracted from the data.

function, which approximates in a crude way the derivative of the data (since the spline function contains only intervals of cubic polynomials, its derivative consists of intervals of second order polynomials).

## *2.4. Baseline corrections*

A command in SOAS calculates a linear baseline (dashed blue line in Figure 6) by fitting a straight line to a portion of data that has been selected by the user, and extrapolating through the entire range. This command also returns the equation of the line that is fitted through the data, and therefore it can also be used for measuring slopes.

The following 2nd-order polynomial baseline will be useful for voltammetric data obtained with a rotating disk electrode (RDE) and for some chronoamperometric data. As an alternative to a simple linear baseline guided by a part of the sweep where the faradaic current is zero (the $i = 0$ range), SOAS provides a simple non-linear baseline, guided by both the $i = 0$ range and by a part of the sweep where the faradaic current has reached a limiting value (the $i = i_{\text{lim}}$ range). In Figure 6A, each pair of red markers defines an interval, one in the $i = 0$ region, and the other in the $i = i_{\text{lim}}$ part of the voltammogram (empty and filled symbols, respectively). For

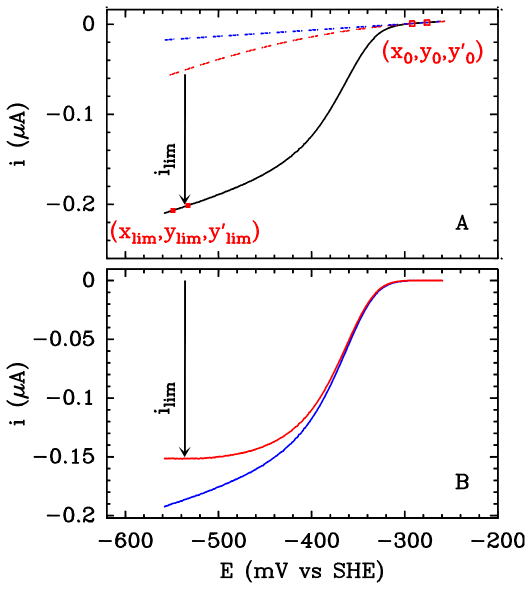


Figure 6: The black line in Panel A is a voltammogram recorded at a rotating disc electrode. The dashed blue line is the straight baseline calculated and extrapolated from the $i = 0$ range. The dashed red line is the non linear baseline which is defined by the slope of the current response at $i = 0$ as well as at $i = i_{\mathrm{lim}}$ (see text). By construction, the latter baseline forces the corrected current to reach a well-defined plateau at high driving-force. The signals corrected by subtracting either baseline are shown in panel B.

both ranges, the average potentials, $x_0$ and $x_{\mathrm{lim}}$, currents, $y_0$ and $y_{\mathrm{lim}}$, and slopes, $y'_0$ and $y'_{\mathrm{lim}}$, are measured. The value of ilim and the three parameters that define the 2nd order polynomial baseline in the intermediate range of potential ($x_0 > x > x_{\mathrm{lim}}$) are calculated from the following set of four conditions, namely (i) it goes through the points of coordinates ($x_0$,$y_0$) and ($x_{\mathrm{lim}}$,$y_{\mathrm{lim}} - i_{\mathrm{lim}}$), and (ii) its slopes at these positions match the values of $y'_0$ and $y'_{\mathrm{lim}}$. In the ranges $x > x_0$ and $x < x_{\mathrm{lim}}$, the baseline is linearly extrapolated. This arithmetic is described in the online manual, under the heading “Subtract polynomial baseline from catalytic wave”. The dashed red line in Figure 6A shows the resulting composite baseline. The catalytic current left by subtracting this baseline (red line in Figure 6B) reaches a limiting current at high driving force, whereas if a linear baseline is subtracted (dashed blue line) the corrected current keeps increasing linearly at high driving force without reaching a plateau (blue line in Figure 6B). Only the user can decide whether subtracting a baseline that forces the corrected data to reach a true limiting current is the right thing to do. We shall only note that the lack of a strict limiting current is often observed with adsorbed electrocatalysts; it has been attributed to the existence of a distribution of orientations of the catalyst on the electrode surface [12, 13]. According to this model, some information is embedded in the value of the residual slope at high driving force; the latter can be measured with SOAS using the tool that is also used for subtracting linear baselines. The polynomial baseline described above was used to correct the data for succinate oxidation by *E. coli* fumarate reductase in fig. 2 of ref [14].

The spline procedure of SOAS can be used in a straightforward manner for defining baselines. As soon as the user has set three or more markers on the data, the program will interpolate the smooth baseline consisting of cubic polynomial intervals (Figure 5). Technically speaking, cubic spline interpolation between $n$ user-defined markers involves solving the equations for $n-1$ cubic polynomials, such that both the $y$-values, the first derivatives, and the 2nd derivatives of two adjacent polynomials are equal at the position of the marker, thus yielding a quasi-continuous curve [10]. This procedure is mostly useful in situations where the baseline is interpolated from regions of the signal where the current reaches the background: this is so for voltammetric peaks resulting from adsorbed redox species (see e.g. fig. 2 in ref [15]), or in certain chronoamperometric experiments with adsorbed enzymes which are transiently inhibited (see e.g. supplementary fig. S3 in ref [16]). A cubic spline baseline provides virtually unlimited flexibility, thus needs to be used carefully and with sound judgment, using as few markers as possible. To estimate a baseline under a voltammetric peak, it is usually sufficient to define two or three markers on either side of the peak. We advise not to set the markers too close to the wings of the peak, because this will distort the signal.

In each of the three cases above, the difference between the data and the baseline is plotted while the baseline is being adjusted (see e.g. Figure 5), and the user can choose whether the data should be corrected by subtracting the baseline or by dividing the data by the baseline.

The user can also use the buffers in the stack to perform simple operations related to baseline corrections: this includes subtracting a blank or dividing the main signal by another data set. The latter operation may be very useful for correcting chronoamperometric data for film loss, as described in ref [17].

### *2.5. Peak search*

Following selection of a single sweep for a voltammogram, baseline correction, and noise correction (if necessary), the user can with a single keystroke have SOAS find one or two peaks. The peak position and other key details, such as widths and areas, are automatically appended to a cumulative ASCII output file for the current working file directory. By using the stack and file manipulation commands in SOAS, a user can rapidly work through peak assignment for a large set of voltammograms, with remarkably few keystrokes and mouse clicks. Further processing of peaks can be performed, including fitting and filtering. Processed peaks can also be output as ASCII files, for use in other programs.

## *2.6. Using interactive GREG*

Most users should find the commands of SOAS enough for their needs. Should this not be the case, using the command line interpretor of GREG, as described below, greatly pushes back the limits of SOAS. Without exiting SOAS, the user can modify the data by entering "interactive GREG" (in which case the prompt changes from "`Soas>`" to "`GreG>`") and using its command line editor. GREG has its own syntax, documented under the heading "GREG1 Language Internal Help" in the manual of GREG (`http://www.iram.fr/IRAMFR/GILDAS/doc/html/greg-html/greg.html`). Typing `HELP` will return a description of the available commands and their arguments. A simple command like "`LET Y LOG10(Y)`" can be used to transform the current before returning to SOAS to keep working on the data set. The list of elementary functions recognized by SIC, the command interpretor of GILDAS, is listed under the heading "Functions and Operators" in the manual of SIC (`http://www.iram.fr/IRAMFR/GILDAS/doc/html/sic-html/sic.html`). Any user-defined transformation can be performed in this manner, and the data can be passed along when the user quits GREG to return to SOAS. The user may also use interactive GREG to load a data file in an ASCII format that is not supported by SOAS. For example the GREG command "`COLUMN X 1 Y 2 /FILE Myfile.dat /LINES 3 100`" loads into the arrays X and Y the 1st and 2nd columns of the text file "Myfile.dat". The "`/LINES`" option limits the range of lines read in the input file. This may be useful to avoid non-data lines. The lines of commands are stored and can be retrieved using the up/down arrows of the keyboard, then edited (making use of the typical GNU control keys) before execution. There is no auto-completion with the TAB key but the commands can be abbreviated: e.g. "`COL`" is understood like "`COLUMN`", but "`CO`" is ambiguous because there is also a "`CONNECT`" command. All this is described in the online manual of SIC.

### *2.7. Fitting*

SOAS embeds ODRPACK (`http://www.netlib.org/odrpack/`), a FORTRAN package which can fit data to a model function by finding the parameters that minimise the sum of the square weighted orthogonal distances from a set of observations to a curve determined by parameters. The algorithm implemented by ODRPACK is an efficient and stable trust region Levenberg-Marquardt procedure, as described in ref [9]. The ODRPACK manual is available from the NETLIB repository, or from the online manual of SOAS under the heading "Fits/Preamble." It is a very efficient and robust procedure to solve ordinary nonlinear least squares problems. Of course, using good starting values for the parameters to be adjusted is as important as with all other nonlinear least squares software: a poor initial approximation may prevent a solution (or worse, *the right* solution) from being found.

In SOAS, a number of built-in model functions can be used for fitting Nernstian non-catalytic peaks for adsorbed redox species to the model of Laviron [12], catalytic waves, Nernstian sigmoids, exponential and multi-exponential decays. The user who needs a function that is not yet built-in can either edit the source code of SOAS and recompile the program to incorporate the new model (that is the hard way) or use a SOAS command that makes it possible to type in simple model functions , as in "`f(x)=a*exp(-x/b)+c`", where `a`, `b` and `c` are the parameters to be adjusted (this functionality is available provided the program was linked to scalc libraries at the time of compilation. See installation notes). The model function can call any of the following elementary functions: exp, ln, sin, cos, tan, sqrt, asin, acos, atan, cosh, sinh, tanh, erf, gamma, heavy; "**" is the symbol for exponentiation. SOAS will attempt to guess the starting parameters only when the data are fitted to a built-in function. In all cases, the user can set or modify the starting parameters and decide whether they will be adjusted or held fixed at their input values. The user may also specify whether the fit is to be by orthogonal distance regression or by ordinary least squares, and whether SOAS displays detailed iteration reports or only the final result. The software returns the best values of the model parameters and a covariance matrix, which is useful for constructing confidence intervals for the parameters.

Extensive code writing will be needed before SOAS can fit voltammetric data obtained with diffusing species. This is because the corresponding problems rarely have closed-form solutions; that is, the current equations cannot be expressed analytically in terms of certain elementary functions. Rather, they are obtained by numerically solving a certain set of differential equations, but the open source version of SOAS which we distribute does not include a routine for evaluating integrals, or an ordinary differential equation integrator to be called within the fitting procedure. For fitting diffusion-limited voltammetry, users may wish to investigate one of several alternative programs mentioned earlier.

## 3. Downloading, installing, and acknowledging SOAS

SOAS can be downloaded from our web site (http://bip.cnrs-mrs.fr/bip06/software.html). It is written in Fortran, with a few parts in C/C++, and embeds public domain NETLIB numerical libraries, FFTPACK (http://www.netlib.org/fftpack/), ODRPACK [9] (http://www.netlib.org/odrpack/), spline (the http://www.netlib.org/sfmm/spline.f and sfmm/seval.f routines from the netlib repository), SSORT [18] (http://www.netlib.org/slatec/src/ssort.f), and the GILDAS suite, which is "freely available to non-profit institutes on an as-is basis" (http://www.iram.fr/IRAMFR/GILDAS/). Redistribution and modifications of SOAS are possible under the terms of the GNU General Public License (http://www.gnu.org/licenses/old-licenses/gpl-2.0.html).

Soas can be compiled on any system where Gildas can be compiled. Great effort has been spent in making Soas compatible with as many versions of Gildas as possible. To make the most of SOAS, it is recommended to have development files for the readline library (version 5), and the SCalc library for fits with arbitrary functions. No precompiled binary packages are available, except for MacOS.

A self-extracting binary distributions of SOAS for Mac OS X (10.4 and above) can also be downloaded from our web site. This version was compiled on a PPC-based Macintosh using gfortran 4.3, on a system augmented by libraries for scalc 0.2 and readline 5.2. Successful compiling required temporarily disabling Apple-provided readline libraries to avoid conflicts during compilation. It was also necessary to compile the GILDAS software package from source as well, using the same compiler environment. In our experience, finding a suitable environment (compiler, source, libraries) to compile GILDAS and other supporting software was the most difficult aspect of bringing SOAS to new platforms. However, we have compiled, and made available, SOAS and GILDAS binaries that run on default MacOS systems without the end-user requiring additional software beyond X11 graphics support (available from Apple as an optional system installation). Currently, SOAS runs on Intel-based Macintosh systems through Apple's Rosetta emulation; as gfortran development improves, native-Intel releases are under development.

We welcome an acknowledgement in publications using SOAS software to analyse data: if you find this program useful, thank you for letting others know by citing this paper.

## 4. Acknowledgement

SOAS and UTILS were born while some of us (CL, KRH, HAH) were in the group of Fraser A. Armstrong, whom we acknowledge for his contaminating enthusiasm for electrochemistry, enzymes and pubs. The work in Marseilles is funded by the CNRS, Aix-Marseille Université, ANR and the City of Marseilles. KRH is grateful to Berry College for the support of a Faculty Development Grant. HAH is funded by the Netherlands Organization for Scientific Research (NWO).

## References

[1] D. Britz, Digital Simulation in Electrochemistry, Springer, Berlin, 2005.

[2] D. K. Gosser, Cyclic Voltammetry: Simulation and Analysis of Reaction Mechanisms, Wiley-VCH, New York, 1993.

[3] M. Rudolph, D. P. Reddy, S. W. Feldberg, A simulator for cyclic voltammetric responses, Analytical Chemistry 66 (1994) 589A–600A, doi:10.1021/ac00082a002.

[4] C. Andrieux, B. Limoges, D. Marchal, J.-M. Savéant, A general procedure of time-resolved catalytic responses, Anal. Chem. 78 (2006) 3038–3143, doi:10.1021/ac052176v.

[5] L. K. Bieniasz, Elsim - a problem solving environment for electrochemical kinetic simulations. version 3.0 - solution of governing equations associated with interfacial species, independent of spatial coordinates or in one-dimensional space geometry, Comput. Chem. 21 (1997) 1–12.

[6] R. Lowry, Polar, Physical Sciences Educational Reviews 3 (2) (2002) 26–27.

[7] G. W. H. Potter, Polarograph.com, Physical Sciences Educational Reviews 5 (1) (2002) 41–43.

[8] K. Ludwig, L. Rajendran, B. Speiser, Echem++ - an object oriented problem solving environment for electrochemistry. part 1. a C++ class collection for electrochemical excitation functions, J. Electroanal. Chem. 568 (2004) 203–214, doi:10.1016/j.jelechem.2004.01.024.

[9] P. T. Boggs, J. R. Donaldson, R. H. R. H. Byrd, R. B. Schnabel, Algorithm 676: Odrpack: software for weighted orthogonal distance regression, ACM Transactions on Mathematical Software (TOMS) 15 (4) (1989) 348–364, doi:10.1145/76909.76913.

[10] W. H. Press, S. A. Teukolsky, W. T. Vetterling, B. P. Flannery, Numerical Recipes in Fortran 90, second edition, Cambridge university press, 1996.

[11] A. K. Jones, S. E. Lamle, H. R. Pershad, K. A. Vincent, S. P. J. Albracht, F. A. Armstrong, Enzyme electrokinetics: electrochemical studies of the anaerobic interconversions between active and inactive states of allochromatium vinosum nife hydrogenase, J. Am. Chem. Soc. 125 (28) (2003) 8505–8514, doi:10.1021/ja035296y.

[12] C. Léger, P. Bertrand, Direct electrochemistry of redox enzymes as a tool for mechanistic studies, Chem. Rev. 108 (7) (2008) 2379–2438, doi:10.1021/cr0680742.

[13] C. Léger, A. K. Jones, S. P. J. Albracht, F. A. Armstrong., Effect of a dispersion of interfacial electron transfer rates on steady state catalytic electron transport in [NiFe]-hydrogenase and other enzymes., J. Phys. Chem. B 106 (2002) 13058–13063, doi:10.1021/jp0265687.

[14] C. Léger, K. Heffron, H. R. Pershad, E. Maklashina, C. Luna-Chavez, G. Cecchini, B. A. C. Ackrell, F. A. Armstrong, Enzyme electrokinetics: energetics of succinate oxidation by fumarate reductase and succinate dehydrogenase., Biochemistry 40 (2001) 11234–11245, doi:10.1021/bi010889b.

[15] S. E. Chobot, H. H. Hernandez, C. L. Drennan, S. J. Elliott, Direct electrochemical characterization of archael thioredoxins, Angew. Chem. Int. Edit. 46 (2007) 4145–4147, doi:10.1002/anie.200604620.

[16] M. G. Almeida, B. Guigliarelli, P. Bertrand, J. J. G. Moura, I. Moura, C. Léger, A needle in a haystack: the active site of the membrane-bound complex cytochrome *c* nitrite reductase, FEBS Letts. 581 (2007) 284–288, doi:10.1016/j.febslet.2006.12.023.

[17] V. Fourmond, T. Lautier, C. Baffert, F. Leroux, P.-P. Liebgott, S. Dementin, M. Rousset, P. Arnoux, D. Pignol, I. Meynial, P. Soucaille, P. Bertrand, C. Léger, Correcting for electrocatalyst desorption and inactivation in chronoamperometry experiments, Anal. Chem. ? (2009) ?, in press http://dx.doi.org/10.1021/ac8025702.

[18] R. C. Singleton, Algorithm 347, an efficient algorithm for sorting with minimal storage, Communications of the ACM 12 (3) (1969) 85–187, http://dx.doi.org/10.1145/362875.362901.