Last document update: March, 2026

Application Note

1 What is Electrochemical Noise?

Electrochemical noise (EN) results from changes in current and potential caused by corrosion. The noise is assumed to be random, i.e. from random low-frequency stochastic processes. Sources of noise include sudden events like film rupture, cracking and crack propagation, metal dissolving, formation of hydrogen gas bubbles, and other mechanical erosion processes.

Most potentiostats can be set up to measure EN by simultaneously reading out the potential and current as a function over time using two nominally identical electrodes. A Zero-Resistance Ammeter (ZRA) connects two electrodes, so that the potential difference between the two electrodes is zero. Likewise, the potential can be measured with respect to a reference electrode. Unlike other electrochemical techniques, EN measurements are entirely passive, requiring no external excitation signal.

There are many methods for analyzing the EN data (Xia et al. 2020). Corrosion studies employing EN typically use Fourier Transforms (FT) or Power Spectral Densities (PSD) to analyze the frequency domain of the noise signals. In this application note we show how to use Python to measure, process, and analyze EN data using PalmSens instruments.

Supported instruments:

- Nexus series
- PalmSens 4 series
- EmStat4 series

2 Measuring Electrochemical Noise

For this application note, we used an EmStat4S with a dummy cell consisting of a Li-ion battery. The Working Electrode (WE) was connected to the positive terminal of the battery (+), the Reference Electrode (RE) to the negative terminal (-), and ground (GND) in series with a 1 GΩ resistor to the negative terminal (-). Placing the resistor in series with the GND is essential to avoid overload and possible damage to the potentiostat. See the cell setup for a conventional electrochemical cell in Figure 1.

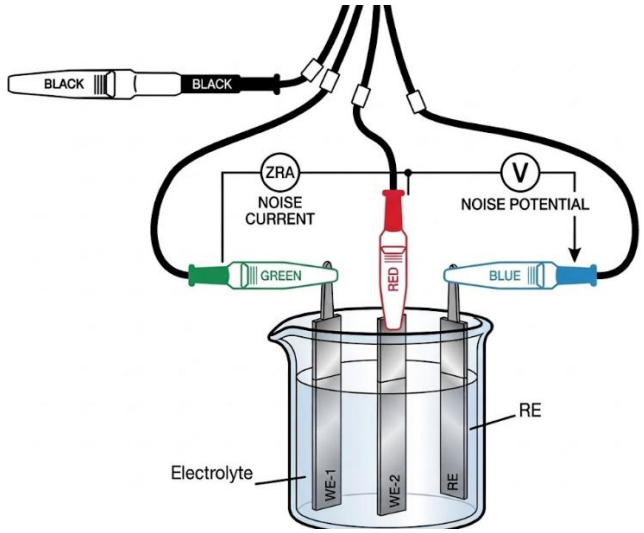

Figure 1. Electrochemical Noise setup showing how to make the connections. For instruments with a Sense lead, connect it along with the WE lead (stacked).

> **In the ZRA / EN setup, the potentiostat cannot disconnect the cell in case of an overload. Any damage caused by excessive current under this connection scheme is not covered by the warranty. Be careful when connecting self-powered samples with this setup.**

To collect EN data, we use the Chronoamperometry method in PyPalmSens. PyPalmSens is a free and open-source Python library for automating electrochemistry experiments using PalmSens instruments.

We set the measurement interval to 1 second to have a sampling rate of 1 Hz, and we allow the measurement to run for 1 hour. We record the WE potential in addition to the current. See the documentation for the ChronoAmperometry class for more information on how to customize the parameters.

After the measurement, the data are stored to the file `EN_1Hz_3600s.pssession`.

```
import pypalmsens as ps

ca = ps.ChronoAmperometry(
    current_range={'min': '100nA', 'max': '100mA', 'start': '100nA'},
    interval_time=1.0,
    run_time=3600,
    record_we_potential=True,
)

measurement = ps.measure(ca)

ps.save_session_file('EN_1Hz_3600s.pssession', [measurement])
```

3 Analyzing Electrochemical Noise

Several papers describe the procedure to analyze the data and extract it. For the data analysis, we follow the procedure described by Barrozo et al. (2020). The data analysis consists of 3 steps:

1. Detrend the data.
2. Multiply data by a 'window'.
3. Calculating the power spectrum or power spectral density (PSD).

Although the first two are technically optional, treating data improves the quality of the EN analysis. First, we use PyPalmSens to load and extract the data previously measured using the `load_session_file()` function. The data are stored in a measurement class that contains the data, measurement details, and device metadata. We extract the time, current, and potential arrays.

```
import pypalmsens as ps
import numpy as np

measurement = ps.load_session_file('EN_1Hz_3600s.pssession')[0]

t = np.array(measurement.dataset['Time'])
E = np.array(measurement.dataset['PotentialExtraRE'])
I = np.array(measurement.dataset['Current'])
```

3.1 Detrending

Detrending removes drift that may occur for long measurements. The challenge is to choose the right method that attenuates the low-frequency components without eliminating useful information or introducing artifacts (Bertocci et al. 2002).

We use the polynomial module from the numpy library to fit a polynomial trendline through the data. To remove the drift, subtract the trendline from the data.

```
pI = np.polynomial.Polynomial.fit(t, I, 10)
pE = np.polynomial.Polynomial.fit(t, E, 15)

I_corr = I - pI(t)
E_corr = E - pE(t)
```

We choose to use 10 and 15 as the degrees of the fitting polynomials. These resulted visually in a good fit, without introducing too many artifacts. Different datasets may require different values or different detrending methods.

Figure 2 shows the measured data, the polynomial fit, and the residual data.

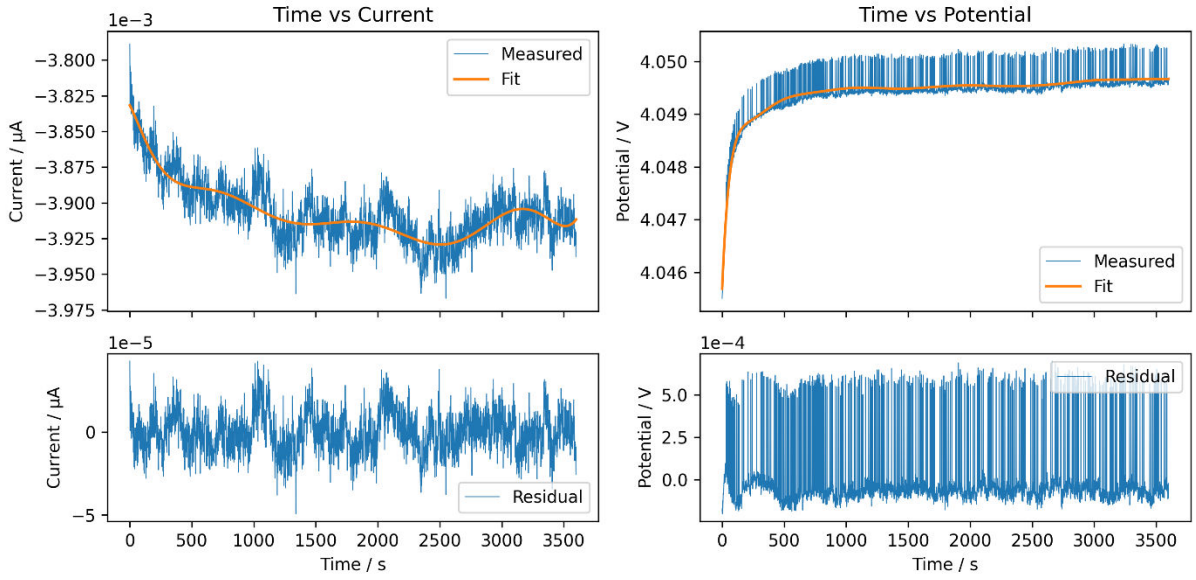

Figure 2. Polynomial fit of the current and potential curves

3.2 Applying a Window

Before applying a Fourier transform to the data, improve the quality of the data by applying a window (Bertocci et al. 1998). Windows are often 'bell-shaped' functions equal to 1 in the middle that taper to 0 at the edges. In signal processing, they help reduce spectral leakage of the low frequencies to the higher frequencies in Fourier transform. Many different types of windows exist. The Hann window is cited as one of the most suitable choices for electrochemical noise.

Generate the Hann window using the `signal.windows.hann` function from the scipy library. The Hann window can be applied by simply multiplying with the current and potential arrays of the same length.

```
from scipy import signal

Hw = signal.windows.hann(len(t))

I_h = I_corr * Hw
E_h = E_corr * Hw
```

Figure 3 shows the data with the windowed data, as well as the window envelope.

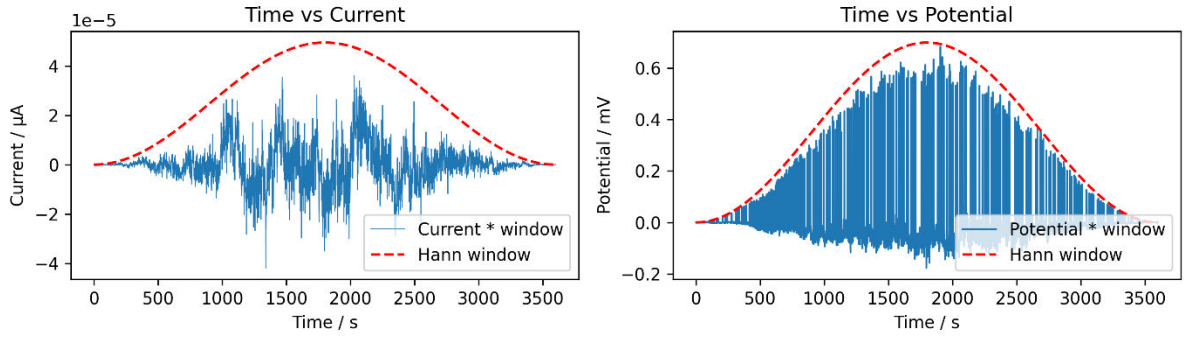

Figure 3. Hann window applied to the current and potential arrays.

3.3 Calculate the Power Spectrum

The Power Spectral Density (PSD) describes the distribution of power (or any timeseries) into frequency components composing that signal.

To calculate the PSD, we used the `signal.periodogram` function from scipy. This performs the Fourier transform of the current or potential and calculates the spectral densities.

We pass in the current and potential arrays with the Hann window applied and the sampling frequency (1 Hz). This returns the frequency and PSD arrays for the current and potential with the units $\mu A^2\ Hz^{-1}$ and $V^2\ Hz^{-1}$.

```
freq, E_psd = signal.periodogram(E_h, fs=1.0)
freq, I_psd = signal.periodogram(I_h, fs=1.0)
```

Figure 4 shows the resulting power spectra, with the low frequency vibrations to the left, and the high frequency vibrations to the right.

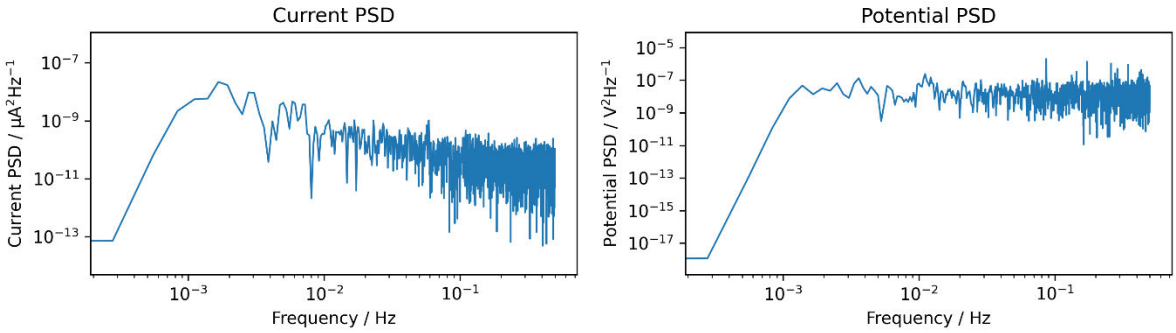

Figure 4. Power spectral densities of the current and potential.

3.4 Calculate Noise Resistance

One way to interpret the noise is using electrochemical noise resistance (R_{sn}). It corresponds to the low frequency noise impedance. Calculate R_{sn} by applying Ohm's law to the PSDs.

$$R_{sn}(f) = \left|\frac{PSD_E(f)}{PSD_I(f)}\right|^{1/2}$$

This can be easily translated to Python using:

```
Rsn = np.sqrt(E_psd / I_psd)
```

The spectral electrochemical noise resistance, R_{sn}^0, is defined as the dc limit of a spectral noise plot (Mansfeld and Lee, 1997). It can be calculated by fitting a linear regression model to $\log(R_{sn})$ vs $\log(f)$. With the obtained slope and intercept, plug in the lowest valule of the experimental frequency to get $\log(R_{sn}^0)$ and, subsequently, R_{sn}^0.

We use the variables `i` and `j` to delimit the region to fit the linear model. This helps to remove the 0-frequency that results in a 'divide-by-zero' error, and high-frequency vibrations outside the region of interest.

```
i = 3  # index of initial frequency
j = int(len(t) / 10)  # index of final frequency

x = np.log10(freq[i:j])
y = np.log10(Rsn[i:j])

ffit = np.polynomial.Polynomial.fit(x, y, 1).convert()
intercept, slope = ffit.coef
Rsn0 = 10**ffit(freq[0])
```

We used the 10% lowest frequencies for the fit. We discarded the first few low frequency points that are highly dependent on the quality of the detrending and therefore do not correspond to physically reasonable amplitudes.

The table below shows the values we obtained for the examined data.

Parameter	Value	Code
Initial index	3	`i`
Final index	360	`j`
Initial frequency	0.0008 Hz	`freq[i]`
Final frequency	0.1000 Hz	`freq[j]`
y-intercept	1.778	`ffit.coef[0]`
Slope	0.518	`ffit.coef[1]`
$\log\,(R_{sn}^{0})$	1.778 Ω	`ffit(freq[0])`
R_{sn}^{0}	59.9 Ω	`10**ffit(freq[0])`

Figure 5 shows a linear model fitted to the data.

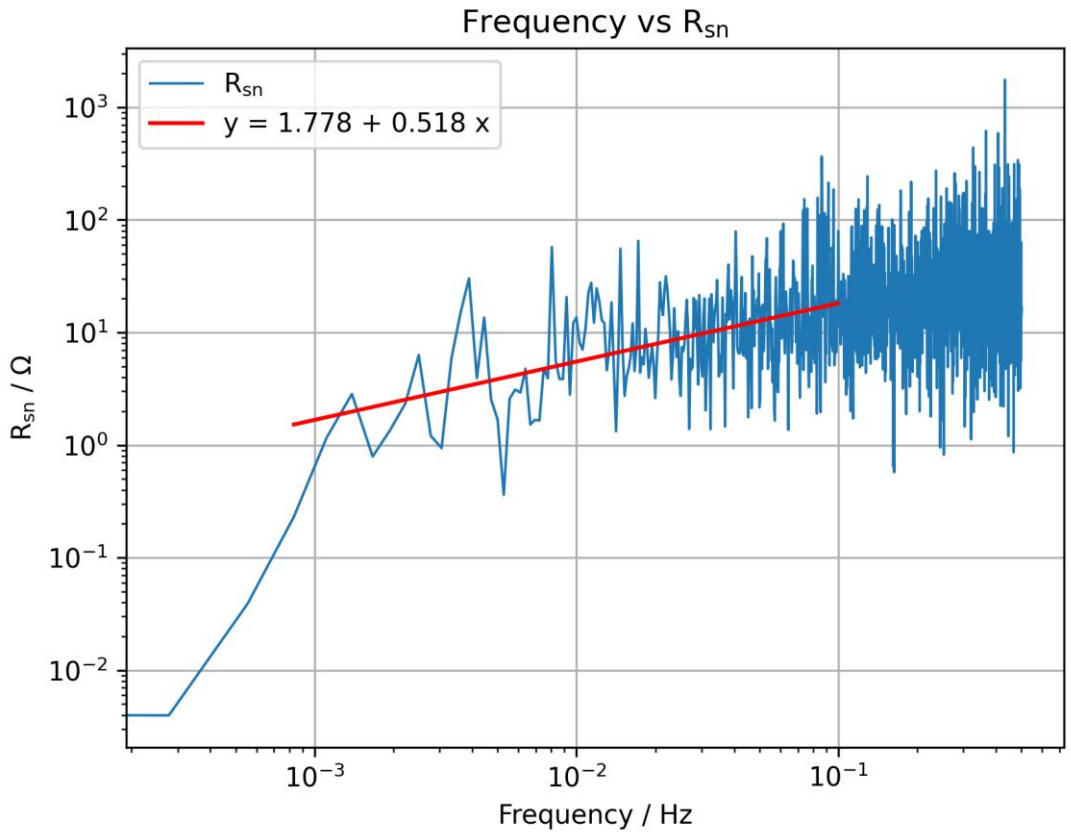

Figure 5. Spectral electrochemical noise resistance

4 Conclusion

With this application note we demonstrated how to process the data for electrochemical noise analysis using functions from the widely used numpy and scipy libraries. We based our implementation on the

routine described by Barrozo et al. (2020). However, many alternative methods for studying EN exist, see for example this review by Xia et al. (2020).

We used PyPalmSens to show how to set up a chrono-amperometry method to measure noise data using PalmSens instruments with Python. The data were exported to numpy arrays with a few lines of code. This makes it straightforward to integrate in existing Python-based data analysis workflows, using familiar libraries from the vast scientific Python ecosystem.

The full code for data processing, including sample data, are available as a Jupyter notebook on our code repository.

5 References

- Barrozo, Sidineia, Riberto Nunes Peres, Marcus José Witzler, Assis Vicente Benedetti, and Cecílio Sadao Fugivara. 2020. "Electrochemical Noise Analysis to Obtain the Rsn Value via FFT Using Excel." *Eclética Química* 45(4):57–70. doi:10.26850/1678-4618eqj.v45.4.2020.p57-70.
- Bertocci, U., J. Frydman, C. Gabrielli, F. Huet, and M. Keddam. 1998. "Analysis of Electrochemical Noise by Power Spectral Density Applied to Corrosion Studies: Maximum Entropy Method or Fast Fourier Transform?" *Journal of The Electrochemical Society* 145(8):2780. doi:10.1149/1.1838714.
- Bertocci, U., F. Huet, R. P. Nogueira, and P. Rousseau. 2002. "Drift Removal Procedures in the Analysis of Electrochemical Noise." *Corrosion* 58(4):337–47. doi:10.5006/1.3287684.
- Mansfeld, F., and C. C. Lee. 1997. "The Frequency Dependence of the Noise Resistance for Polymer-Coated Metals." *Journal of The Electrochemical Society* 144(6):2068. doi:10.1149/1.1837743.
- Xia, Da-Hai, Shizhe Song, Yashar Behnamian, Wenbin Hu, Y. Frank Cheng, Jing-Li Luo, and François Huet. 2020. "Review—Electrochemical Noise Applied in Corrosion Science: Theoretical and Mathematical Models towards Quantitative Analysis." *Journal of The Electrochemical Society* 167(8):081507. doi:10.1149/1945-7111/ab8de3.