Communications Project

# Periodogramsaa

Homework Results

*Authors:*

| | | |
|---|---|---|
| Saqib Faraz | 458317 |
| Ankita Kannan Iyer | 457620 |
| Harsh Godhani | 461986 |
| Richard Grünert | 289427 |
| Kanaiyalal Thummar | 435534 |

4.5.2023

## 1 Introduction

A periodogram is an estimation of the power spectral density (PSD) of a random process such as a measured signal that is superimposed by noise.

### 1.1 Power Spectral Density

The power spectral density (PSD) describes the distribution of a signal's power over frequency. This is in contrast to the regular fourier transform (or DFT) of the signal which describes its harmonic content based on amplitude and phase. It can be used to uncover hidden periodicities of a random process (or an estimation of such).[4]

The autocorrelation of a signal will provide a measure of similarity of the signal to a time-shifted version of itself. It will be in units of power ($\mathrm{V}^2$). If the signal has any periodic components then the autocorrelation will also show this periodicity. Therefore, we can analyze the harmonic content of the signal's power by performing the fourier transform of its autocorrelation, which will result in the power spectral density (PSD) and be in units of "power per frequency" ($\mathrm{V}^2\,\mathrm{Hz}^{-1}$ or $\mathrm{V}^2\,\mathrm{s}$). For a wide-sense-stationary (WSS) random process (such as white gaussian noise), the autocorrelation does not change with time. Hence it is used as a method of estimating the power spectral density. [4][7]

### 1.2 Periodograms

Since real-world signals are usually time-limited, the exact analytical solutions for the power spectral density cannot be obtained because we can never measure *all* of the infinite realizations of a random process, i.e. we discretely sample the process. Therefore, an estimation has to be made that, ideally, increases in quality as more of the process is measured. [7][4]

Methods for spectral estimation can generally be split into the two categories *parametric* and *non-parametric*. [7]

The simplest estimator of the PSD is the standard non-parametric periodogram. It is defined as the square of the DFT sequence normalized by the sequence length. If $x[n]$ is a sequence of $N$ samples of the random process and its DFT is $X[m]$, then the PSD-estimation computes as [4]

$$\tilde{S}_m = \frac{1}{N} \cdot |X[m]|^2$$

Which can be shown to equal the DFT of the autocorrelation function. [5][7]

There are extensions to the standard periodogram that aim to reduce its relatively poor variance, e.g. *Bartlett's Method* or *Welch's Method* which will be discussed below. [5]

## 2 Types of Periodograms

The following will look at "Welch" and "Lomb-Scargle" periodograms and some python implementations with `numpy` and `scipy`.

Two test signals, $A$ and $B$, are defined for this purpose:

$$A(t) = 1\,\mathrm{V} \sin\left(2\pi \cdot 100.5\,\mathrm{Hz} \cdot t\right) + 0.6\,\mathrm{V} \cdot \sin\left(2\pi \cdot 199.5\,\mathrm{Hz} \cdot t\right)$$

$$B(t) = 1\,\mathrm{V} \sin\left(2\pi \cdot 100.5\,\mathrm{Hz} \cdot t\right) + 0.6\,\mathrm{V} \cdot \sin\left(2\pi \cdot 109.5\,\mathrm{Hz} \cdot t\right)$$

Gaussian noise with a standard deviation of $\sigma = 1\,\mathrm{V}$ is added to each of these signals and their standard periodograms will be compared with the Welch and Lomb-Scargle periodograms.

## 2.1 Welch Periodograms

Welch's method is a modified type of periodogram. Instead of transforming the whole input sequence into the frequency domain, the sequence is split into multiple overlapping[1] intervals (windowed). Then, all the individual power spectra of these intervals are calculated and averaged to produce the final periodogram. The overlapping of the segments can help to restore some of the power content on the edges of the segments that was lost due to windowing. Figure 1 shows an illustration of this method.

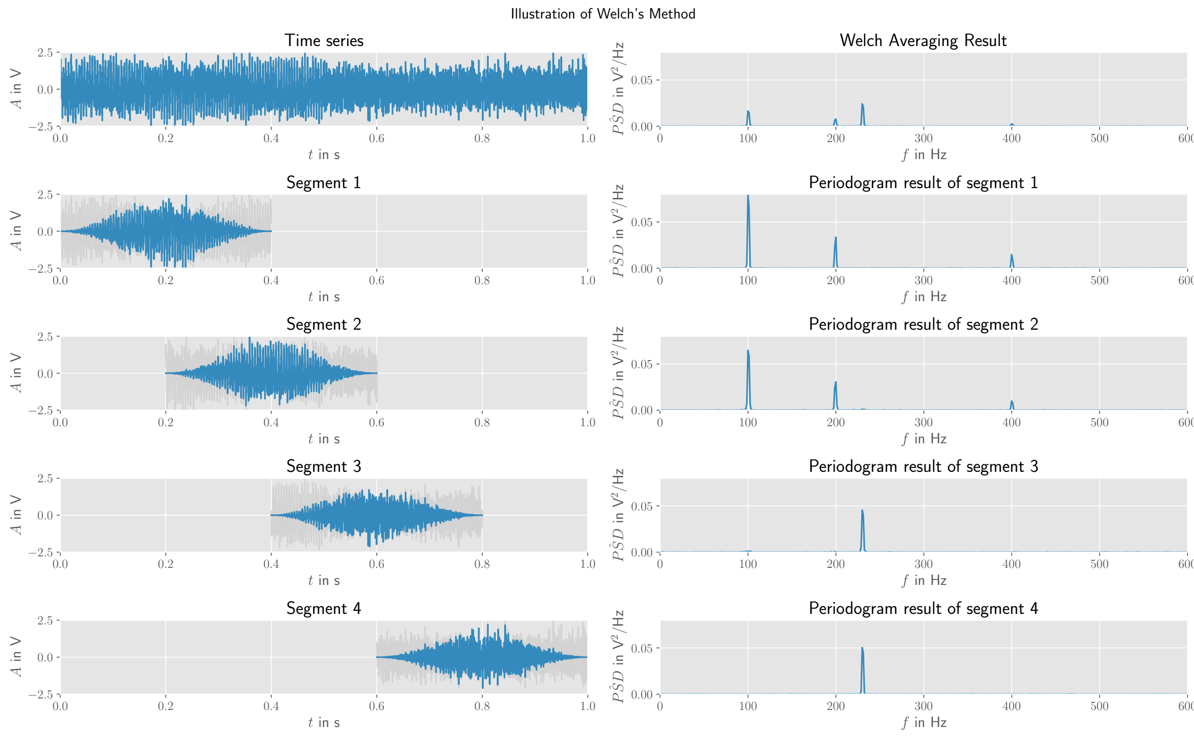

Figure 1: Illustration of Welch's method. The time series is split into multiple windowed segments which are then individually transformed into periodograms that are averaged to obtain the overall periodogram. In this case, the 4 segments were windowed with a Hann window. The total size is $N = 6000$.

Due to the averaging approach of Welch's method, the resulting periodogram might be less sensitive to noise in the time series. E.g. think about a short-duration noise spike in the data: In the full FFT, it might introduce large high frequency content but in the Welch periodogram it will only be evaluated in a fraction of the segments and therefore might not be visible after averaging.

However, since the splitting of the sequence into segments reduces the time window of each individual FFT, the frequency resolution will be smaller than the FFT of the whole sequence ($\frac{1}{T_w}$). This can become a problem when trying to distinguish harmonics of the signal that are relatively closely spaced in frequency.

Adjustable parameters of this method include the segment size, the overlapping size, and the window applied.

Figure 2 shows the periodograms of both test signals and compares them to the standard periodogram. The noise level has been reduced by the Welch periodogram. However, the frequencies of signal $B$ are not distinguishable anymore due to the reduced frequency resolution. The code for this can be found in appendix A.1. Figure 3 further shows that the application of the Welch window requires a tradeoff of its parameters.

[1]without overlapping, this method is called "Bartlett's Method"

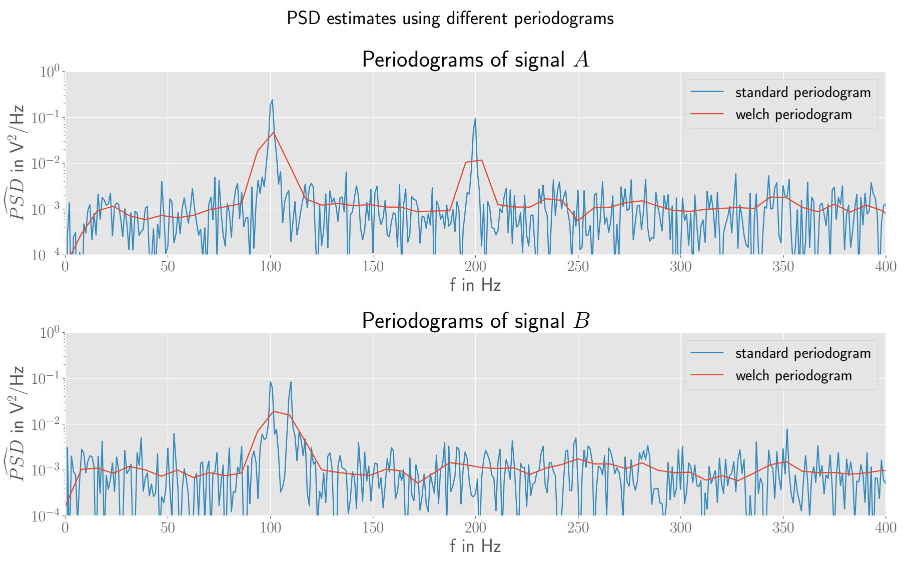

Figure 2: Comparison of standard and welch periodograms. `scipy`'s default parameters have been used for the Welch periodograms.

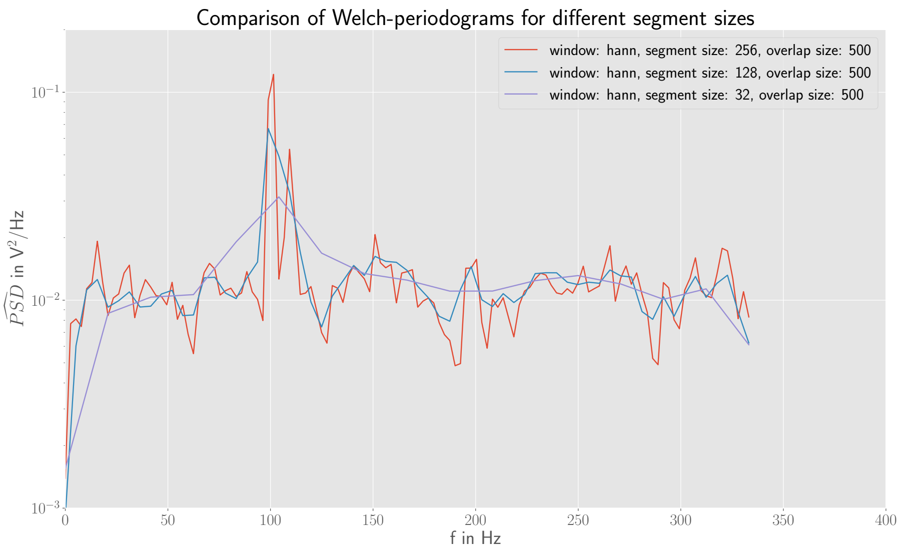

Figure 3: Comparison of different Welch segment sizes when applied to signal $B$. For code see appendix A.2.

## 2.2 Lomb-Scargle Periodograms

The *Lomb-Scargle* method of PSD-estimation can be classified as a method of "Least-squares spectral analysis" (LSSA). LSSA tries to extend the periodogram estimation to data sequences that are not necessarily sampled in an equally-spaced manner. This also applies if individual samples of the data are lost or could not be aquired and will remove the need to resample or interpolate the data. [3][2]

Instead of explicitly performing the Fourier transform, the Lomb-Scargle periodogram estimates the PSD using a sinusoidal model that matches the data. The Lomb-Scargle technique looks for the model that best explains the data as a sum of sinusoids with various frequencies and amplitudes.

The algorithm first determines a range of frequencies for measuring the PSD. These frequencies, which might be evenly or logarithmically separated, are determined by the frequency range anticipated in the signal. The program then uses a weighted sum of the data points, where the weights vary depending on how well the model fits the data, to compute a power estimate at each frequency.

Least-squares fitting is employed to determine the weights. In order to achieve this, one must identify the set of parameters that minimizes the total of the squared discrepancies between the model and the data at each moment. These weights are then used by the algorithm to determine the power estimate for each frequency.

However, the method has significant drawbacks. It's assumption that the signal is stationary, which may not be the case for all signal types, is one of its limitations. The method may also be sensitive to the selection of model parameters, such as the quantity of frequencies employed in the model or the frequency range assessed. [8][1][6]

To simulate this method, signal $A$ was sampled at uniformly distributed random times. As seen in Figure 4, the Lomb-Scargle periodogram approximately follows the standard periodogram.

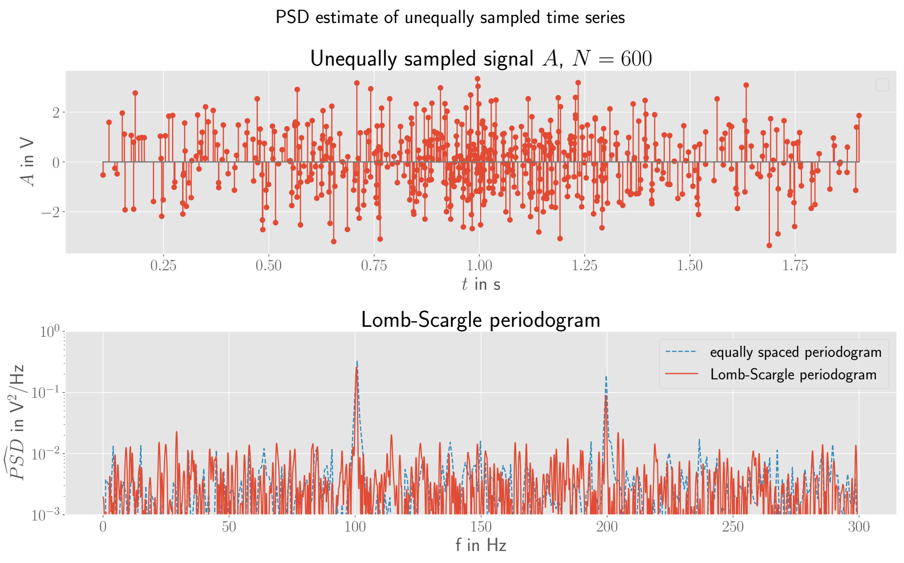

Figure 4: Lomb-Scargle PSD estimate of an unevenly spaced

## 3 Conclusions

Periodograms estimate the power spectral density of a random process. There are numerous different estimation methods of which two have been analyzed. Each of these comes with advantages and drawbacks so that one has to both evaluate which method is best suited as well as tune its parameters to fit the specific application needs.

## References

### Literature

[1] G. Larry Bretthorst. "Generalizing the Lomb-Scargle Periodogram". In: *Washington University, Dept. of Chemistry* ().

[2] LSSTC Data Science Fellowship. *Session 13: Introduction To The Lomb-Scargle Periodogram (Lecture III)*. https://www.youtube.com/watch?v=2EwtD3Nhazs. Last Access 05/2023. Oct. 2021.

[3] Ingudam Gomita, Sandeep Chauhan, and Raj Kumar Sagar. “A Brief Overview on Least Square Spectral Analysis and Power Spectrum”. In: *Proceedings of Fifth International Conference on Soft Computing for Problem Solving, Advances in Intelligent Systems and Computing.* https://doi.org/10.1007/978-981-10-0451-3_2, 2016.

[4] David Hysell. “Antennas and Radar for Environmental Scientists and Engineers”. In: https://doi.org/10.1017/9781108164122.012: Cambridge University Press, 2018. Chap. Weather Radar, pp. 288–322.

[5] K. Deergha Rao and M. N. S. Swamy. *Digital Signal Processing.* Vol. Springer Nature Singapore Pte Ltd. ISBN 978-981-10-8081-4. https://doi.org/10.1007/978-981-10-8081-4: Springer, 2018.

[6] J.O Smith. *Mathematics of the Discrete Fourier Transform (DFT) with Audio Applications.* https://ccrma.stanford.edu/~jos/mdft/mdft-citation.html. Last Access 05/2023.

[7] Jill Stewart et al. “An Application of The Lomb-Scargle Periodogram To Investigate Heart Rate Variability During Haemodialysis”. In: (June 2020). DOI: 10.36227/techrxiv.12442181.v1. URL: https://www.techrxiv.org/articles/preprint/An_Application_of_The_Lomb-Scargle_Periodogram_To_Investigate_Heart_Rate_Variability_During_Haemodialysis/12442181.

[8] Anthony Zaknich. *Principles of adaptive filters and self-learning systems.* Ed. by Michael A. Johnson Michael J. Grimble. Advanced Textbooks in Control and Signal Processing ISBN 978-1-84628-121-1. https://doi.org/10.1007/b138890: Springer London, 2005.

[9] M. Zechmeister and M. Kürster. *The generalised Lomb-Scargle periodogram: A new formalism for the floating-mean and Keplerian periodograms.* https://www.aanda.org/articles/aa/full_html/2009/11/aa11296-08/aa11296-08.html. Last Access: 05/2023. 2009.

## Software Used

[1] Charles R. Harris et al. “Array programming with NumPy”. In: *Nature* 585.7825 (Sept. 2020), pp. 357–362. DOI: 10.1038/s41586-020-2649-2. URL: https://doi.org/10.1038/s41586-020-2649-2.

[2] J. D. Hunter. “Matplotlib: A 2D graphics environment”. In: *Computing in Science & Engineering* 9.3 (2007), pp. 90–95. DOI: 10.1109/MCSE.2007.55.

[3] Python Core Team. *Python: A dynamic, open source programming language.* Python version 3.7. Python Software Foundation. 2019. URL: https://www.python.org/.

[4] Pauli Virtanen et al. “SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python”. In: *Nature Methods* 17 (2020), pp. 261–272. DOI: 10.1038/s41592-019-0686-2.

# A Python Code

## A.1 Welch Periodogram

```python
# welch.py

import scipy
import numpy as np
from matplotlib import pyplot as plt

def my_signal(t, param_list):
    signal = np.zeros(t.size)
    noise = np.random.normal(scale=1, size=t.size)
    for pair in param_list:
        amp = pair[0]
        freq = pair[1]
        signal += amp * np.sin(2 * np.pi * freq * t)
    return signal + noise


def plot_periodogram(ax, x, y, **kwargs):
    ax.semilogy(x, y, **kwargs)
    ax.set_ylim([10e-5, 1])
    ax.set_xlim([0, 400])
    ax.set_xlabel("f in Hz")
    ax.set_ylabel("$\widehat{PSD}$ in V$^2$/Hz")


def main():
    N = 2000
    t_disc = np.linspace(0, 1, N)
    Ts = t_disc[1] - t_disc[0]

    signal_A = my_signal(t_disc, [(1.0, 100.5), (0.6, 199.5)])
    signal_B = my_signal(t_disc, [(0.6, 100.5), (0.6, 109.5)])

    freq_A, per_A = scipy.signal.periodogram(signal_A, fs=1/Ts)
    freq_welch_A, per_welch_A = scipy.signal.welch(signal_A, fs=1/Ts)

    freq_B, per_B = scipy.signal.periodogram(signal_B, fs=1/Ts)
    freq_welch_B, per_welch_B = scipy.signal.welch(signal_B, fs=1/Ts)


    fig = plt.figure()
    axs = fig.subplots(2, 1)
    plot_periodogram(axs[0], freq_A, per_A,
                     label="standard periodogram", color="C1")
    plot_periodogram(axs[0], freq_welch_A, per_welch_A,
                     label="welch periodogram", color="C0")
    axs[0].set_title("Periodograms of signal $A$")

    plot_periodogram(axs[1], freq_B, per_B,
                     label="standard periodogram", color="C1")
    plot_periodogram(axs[1], freq_welch_B, per_welch_B,
                     label="welch periodogram", color="C0")
    axs[1].set_title("Periodograms of signal $B$")

    for ax in axs:
        ax.legend()
    fig.suptitle("PSD estimates using different periodograms")
    fig.tight_layout()
    plt.savefig("outputs/welch.pdf")
    plt.show()

if __name__ == "__main__":
    main()
```

## A.2 Welch Periodogram: Varying Segment Size

```python
# welch_segments.py

import scipy
import numpy as np
from matplotlib import pyplot as plt


class WelchConfig:
    def __init__(self, win='hann', nperseg=None, noverlap=None):
        self.win = win
        self.nperseg = nperseg
        self.noverlap = noverlap

    def periodogram(self, data, fs):
        return \
            scipy.signal.welch(data, fs=fs,
                               window=self.win,
                               nperseg=self.nperseg,
                               noverlap=self.noverlap)

    def get_label(self):
        # cheats (only for N=2000)
        nperseg_l = 1000 if self.nperseg is None else self.nperseg
        noverlap_l = 500 if self.noverlap is None else self.noverlap
        return "window: %s, segment size: %d, overlap size: %d"\
            % (self.win, nperseg_l, noverlap_l)


def my_signal(t, param_list):
    signal = np.zeros(t.size)
    noise = np.random.normal(scale=2, size=t.size)
    for pair in param_list:
        amp = pair[0]
        freq = pair[1]
        signal += amp * np.sin(2 * np.pi * freq * t)
    return signal + noise


def main():
    N = 2000
    t_disc = np.linspace(0, 3, N)
    Ts = t_disc[1] - t_disc[0]

    signal_B = my_signal(t_disc, [(1, 100.5), (0.6, 109.5)])

    welch_configs = [
        WelchConfig('hann', 256),
        WelchConfig('hann', 128),
        WelchConfig('hann', 32),
    ]

    plt.figure()

    for config in welch_configs:
        freq_welch_B, per_welch_B = config.periodogram(signal_B, fs=1/Ts)
        plt.semilogy(freq_welch_B, per_welch_B, label=config.get_label())

    plt.ylim([10e-4, 0.2])
    plt.xlim([0, 400])
    plt.xlabel("f in Hz")
    plt.ylabel("$\widehat{PSD}$ in V$^2$/Hz")
    plt.legend()
    plt.title("Comparison of Welch-periodograms for different segment sizes")
    plt.tight_layout()
    plt.savefig("outputs/welch_comp.pdf")
    plt.show()


if __name__ == "__main__":
    main()
```

## A.3 Lomb-Scargle Periodogram

```python
#!/usr/bin/env python3

import scipy
import numpy as np
from matplotlib import pyplot as plt


def my_signal(t, param_list):
    signal = np.zeros(t.size)
    noise = np.random.normal(scale=1, size=t.size)
    for pair in param_list:
        amp = pair[0]
        freq = pair[1]
        signal += amp * np.sin(2 * np.pi * freq * t)
    return signal + noise


def plot_periodogram(ax, x, y, **kwargs):
    ax.semilogy(x, y, **kwargs)
    ax.set_ylim([10e-5, 1])
    ax.set_xlim([0, 400])
    ax.set_xlabel("f in Hz")
    ax.set_ylabel("$\widehat{PSD}$ in V$^2$/Hz")


def main():
    N = 600
    t_disc = np.linspace(0, 1, N)
    Ts = t_disc[1] - t_disc[0]
    # add noise to sample times
    t_noise = np.random.uniform(0, 1, N)
    t_disc_noisy = t_disc + t_noise

    signal_A = my_signal(t_disc, [(1.0, 100.5), (0.6, 199.5)])
    signal_A_uneq = my_signal(t_disc_noisy, [(1.0, 100.5), (0.6, 199.5)])

    freq_A, per_A = scipy.signal.periodogram(signal_A, fs=1/Ts)

    # rad. frequency vector at which to calculate the lomb scargle estimations
    w = np.linspace(0.01, 2*np.pi*300, 10000)
    ls_pgam = scipy.signal.lombscargle(t_disc_noisy, signal_A_uneq, w, normalize=True)

    fig = plt.figure()
    axs = fig.subplots(2, 1)
    axs[0].stem(t_disc_noisy, signal_A_uneq)
    axs[0].set_title("Unequally sampled signal $A$, $N=%s$" % (str(N)))
    axs[0].set_ylabel("$A$ in V")
    axs[0].set_xlabel("$t$ in s")
    axs[1].semilogy(freq_A, per_A, linestyle="dashed", label="equally spaced periodogram", color="C1")
    axs[1].semilogy(w/(2*np.pi), ls_pgam, color="C0", label="Lomb-Scargle periodogram")
    axs[1].set_ylim([10e-4, 1])
    axs[1].set_title("Lomb-Scargle periodogram")
    axs[1].set_ylabel("$\widehat{PSD}$ in V$^2$/Hz")
    axs[1].set_xlabel("f in Hz")

    for ax in axs:
        ax.legend()

    fig.suptitle("PSD estimate of unequally sampled time series")
    fig.tight_layout()
    plt.savefig("outputs/lomb_scargle.pdf")
    plt.show()


if __name__ == "__main__":
    main()
```