Yet Another Way to Analyze Noise
An updated Python-Based Framework of Merchant et al. 2015 for Long-Term Underwater Noise Processing

K.J. Palmer*[1]

[1]NOAA Pacific Island Fisheries Science Center Affiliate with Ocean Associates, Inc.

December 12, 2025

Abstract

Ambient noise is a key component of passive acoustic monitoring, influencing both the detectability of vocalizing animals and the acoustic conditions they experience. Modern monitoring programs routinely generate multi-terabyte recording archives, creating a need for scalable, reproducible tools to derive long-term noise metrics. Based on Merchant et al. 2015, this technical note presents a Python implementation of hybrid millidecade noise metrics within the YAWN framework and validates it against the established PyPAM implementation. After harmonizing the calibration conventions, the two approaches agree within approximately 1 dB across the full band, with a median offset of 0.45 dB. This confirms the Python implementation as a reliable option for long-term underwater noise assessment and provides a foundation for integration with broader modeling and machine learning workflows.

1 Introduction

The ambient noise in the ocean affects both the behavior of marine animals and the performance of passive acoustic monitoring (PAM) systems. Elevated noise levels can mask biologically important signals, reduce detection ranges for PAM deployments, and complicate efforts to detect and protect threatened populations. Regulatory frameworks, including national and international initiatives that treat underwater noise as an environmental indicator, further motivate consistent and transparent methods for estimating long-term noise metrics.

Over the last decade, long-term PAM programs have expanded in scope, duration, and spatial coverage. Multi-year deployments on fixed recorders, gliders, and other platforms routinely produce data volumes that are difficult to manage with legacy workflows. At the same time, the need for reproducible open methods has grown, both for scientific synthesis and for regulatory contexts where traceable methodologies are essential.

Hybrid millidecade bands are a useful compromise between spectral resolution and storage cost [1]. The approach preserves fine, approximately linear resolution at low frequencies and transitions to logarithmically spaced bands at higher frequencies, allowing long-term spectral statistics to be stored compactly without discarding biologically meaningful structure.

*kpalmer@coa.edu

There are a plethora of free and paid services used to calculate noise metrics including PAMGuard [2], Triton, and PAMGuide [3]. Each of these systems have their benefits and limitations and bioacousicians often find themselves in need of modification for their own specific data needs. For instance, Triton and its associate Remoras are also free and there is a compiled version that does not require MATLAB. However, any customization does require a MATLAB license which is frequently cost prohibitive, especially for researchers in developing countries. PAMGuard is both free and an industry standard. However, JAVA is not commonly known among biologists which makes it challenging to troubleshoot without involving small, but dedicated, team of maintainers. Pypam is another free, open-source, and written in python. It contains multiple useful features for measuring sounscapes. This polished repository has been profelssionally developed and has been used by SanctSound. However, the nested dependencies result in rigid data structure make it challenging to integrate custom calibration systems. It also cannot presently handle duty-cycled data. Finally, PAMGuide was written in both R and MATLAB and includes a Matlab-GUI, allowing for user-friendly interface. The paper was well received, with over 200 citations, in no small part because of the well-documented and published code provided by the author. However, it too has limitations that have required extensive modifications for many long-term noise projects. Principle among the limitations the speed of analysis and the storage options initially provided (either mat files or .csv files). These become ungainly at best and untenable at worst when working with large, multi-instrument, or multi-year arrays.

Here we present Yet Another Way to Analyze Noise, a package based on [3]. The present work builds on PAMGuide by 1) exporting metrics to HDF5 files 2) Implementing it in freely available Python software 3) saving hybrid millidecade bands rather than full PSD matrices and 4) Allowing raw data to be pulled from google storage. Similar to [3], the YAWN framework consists of a series of functions and applications in a single file with minimal dependencies; allowing for easy modification on the extremely off chance that anything breaks.

The Python implementation of YAWN is openly available at SPACIOUS-NoiseProcessing. To demonstrate that the YAWN implementation produces results that are scientifically interchangeable with existing tools, provided that calibration conventions are aligned, hybrid milidecade bands are directly compared to outputs from the PyPam package.

A versioned archival release will be made available via Zenodo once the manuscript is deposited on arXiv.

2 Methods

2.1 YAWN Architecture

YAWN processes long-term acoustic recordings by reading audio files sequentially, extracting time–frequency representations, converting them to acoustic pressure units, and aggregating the results into daily HDF5 files. The application identifies input files either locally or in cloud storage and infers their timestamps directly from filenames using configurable regular expressions. When timestamps cannot be extracted, the file metadata provides a fallback.

Each file is streamed in blocks to avoid loading multi-gigabyte recordings into memory. After optional DC removal, YAWN computes power spectral densities (PSDs) using the scipy.signal function with user-specified FFT size, overlap, and temporal averaging interval. PSDs are initially expressed in voltage units and are subsequently converted to acoustic pressure using a scalar or frequency-dependent sensitivity parameter, `Si`, which specifies the hydrophone response in dB re 1 V/µPa. Setting `Si = 0` yields a neutral calibration of 1 V/µPa, although a frequency–sensitivity CSV may also be provided when instrument responses vary with frequency.

Once calibrated, the PSDs are aggregated into broadband levels, third-octave bands, decade bands, and hybrid millidecade bands, all expressed in dB re 1 µPa^2/Hz. These metrics, together with their associated frequency axes and timestamps, are written to a daily HDF5 file named for the deployment. YAWN also includes utilities for visualization and diagnostics. Hybrid millidecade statistics can be summarized through percentile curves, RMS spectra, or spectral probability density, while long-term spectral averages can be generated directly from the HDF5 products without rereading the raw audio.

Each HDF5 file is organized under a single deployment group (e.g. CalCurCEAS_2024). Time information is stored in DateTime, while all processing and spectral estimation settings are captured as attributes within the Parameters group. Spectral products are stored as separate datasets, including broadband levels, third-octave bands, decade bands, and hybrid millidecade spectra, with corresponding frequency axes stored explicitly.

2.2 Flexible Datetime Handling

YAWN automatically infers recording timestamps from filenames using a small library of configurable regular expressions. Common logger conventions (e.g. AMAR, SoundTrap, and simple `yyyymmdd_HHMMSS` formats) are detected from the first file in the series and parsed into absolute datetimes; the same pattern is then applied consistently to the remaining files. When no known pattern is found, YAWN falls back to the file modification time to ensure that every averaged spectrum is associated with a valid timestamp. This flexible, pattern-based approach avoids hard-coded filename assumptions and reduces reliance on external log files, distinguishing YAWN from more rigid workflows such as PAMGuide and PyPAM when working with heterogeneous instruments or mixed archives.

Name	Regex pattern	Example fragment	`strptime` format
`yyyymmdd_HHMMSS_fff`	`\d{8}_\d{6}_\d{3}`	`20241001_123045_123`	`%Y%m%d_%H%M%S_%f`
`yyyymmdd_HHMMSS`	`\d{8}_\d{6}`	`20241001_123045`	`%Y%m%d_%H%M%S`
`AMAR`	`\d{8}T\d{6}`	`20241001T123045`	`%Y%m%dT%H%M%S`
`SoundTrap_1`	`\d{9}\.\d{12}`	`202401011.123456789012`	`%Y%m%d%H%M%S%f`
`SoundTrap_2`	`\d{4}\.\d{12}`	`2024.123456789012`	`%Y%m%d%H%M%S%f`
`yymmdd-HHMMSS.fff`	`\d{6}-\d{6}\.\d{3}`	`241001-123045.123`	`%y%m%d-%H%M%S.%f`
`yymmdd_HHMMSS`	`\d{6}_\d{6}`	`241001_123045`	`%y%m%d_%H%M%S`

Table 1: Pre-configured filename patterns used by YAWN to infer recording datetimes. The appropriate regular expression is inferred from the first file in the archive and applied to subsequent files; when no pattern matches, file modification time is used as a fallback.

2.3 Visualization and Long-Term Spectrograms

In addition to computing hybrid millidecade and other band-integrated noise metrics, YAWN provides plotting utilities through the `noiseapp` object. These functions operate directly on the daily HDF5 outputs, avoiding the need to reread raw audio. Long-term spectrograms (long-term spectral averages; LTSAs) can be generated with user-defined temporal averaging windows and frequency axes.

The `plot_ltsa` routine constructs LTSAs by aggregating calibrated power spectral densities over arbitrary averaging intervals (e.g. 15 minutes, 1 hour, 4 hours), returning spectral density in dB re 1 $\mu Pa^2/Hz$ on a user-specified frequency grid. The frequency axis can be displayed either in linear frequency or in $\log_{10}$ frequency space, allowing low-frequency structure and higher-frequency bands to be visualized on the same panel. Complementary functions, such as `plot_milidecade_statistics`, summarize hybrid millidecade bands through percentile spectra, RMS spectra, or spectral probability density functions, again computed directly from the HDF5 archives. Together, these tools support rapid visual inspection of long-term soundscapes and diagnostic checks on processing settings without duplicating the underlying computations.

Pandas time-offset strings (e.g., 15min, 1h, 4h, 1D). These strings allow averaging intervals to be set with arbitrary temporal resolution without modifying the underlying processing code. Figure 1 shows an example LTSA for a single day, constructed from calibrated spectra averaged over 15-minute intervals and plotted on a $\log_{10}$ frequency axis. Figure 2 extends the same approach to a 30-day period, using 4-hour averaging windows to highlight broader temporal patterns while retaining biologically relevant spectral detail. Both examples are generated directly from the HDF5 archives using the `plot_ltsa` function, illustrating how users can tailor averaging periods and frequency scaling to the temporal and spectral resolution required by a given application.

```
fig = plot_ltsa(groups,
                averaging_period='4h',
                titleText="LTSA 30 days 4 hr averaging",
                freq_scaled=True,
                log_freq=True)
```

2.4 Hybrid Millidecade Computation

Hybrid millidecade bands combine linear-frequency resolution at low frequencies with logarithmically spaced bands at higher frequencies. The PSD matrix is first defined on the FFT frequency grid. From the lower analysis limit up to a crossover frequency of 435 Hz, each FFT bin is treated as its own narrow band, preserving the original linear resolution. Above the crossover, the algorithm constructs logarithmically spaced bands using fixed millidecade spacing, defined by a base-10

exponential progression. Each band therefore has a well-defined center frequency and frequency extent in log space.

For a given band, all FFT bins with centers falling within the band edges are identified. Their powers are summed in linear units, scaled by the FFT bin width, and normalized by the effective bandwidth of the hybrid band. The resulting quantity is an average spectral density across the band, which is then converted back to decibels. If a band contains no FFT bins, the algorithm evaluates the PSD at the nearest FFT frequency; if only one bin is present, its PSD defines the band directly. The completed hybrid millidecade matrix contains one spectrum per averaging interval and one value per hybrid band, and is stored in the HDF5 output along with the corresponding band definitions.

Hybrid millidecade bands in YAWN are computed from the calibrated PSDs in dB re 1 $\mu Pa^2/Hz$, defined on the FFT frequency grid of the spectrogram.

For a given time-averaged PSD matrix $A(f, t)$ in dB, YAWN first defines a linear-frequency region from the lowest analysis frequency up to a crossover frequency f_{cross} (currently 435 Hz). In this region, each FFT bin is treated as its own band, with nominal centre frequency equal to the FFT bin center and band edges defined by $\pm\Delta f/2$, where Δf is the FFT bin width.

Above f_{cross}, the code constructs logarithmically spaced bands in "millidecades" of frequency. If the base is 10 and the number of bands per decade is N, the k-th band is defined by a centre frequency

$$f_k = f_0 \times 10^{k/N},$$

with lower and upper edges obtained by multiplying f_k by constant factors corresponding to half a band in log-space. The bands are truncated at the Nyquist frequency.

For each band with edges $[f_{\text{low}}, f_{\text{high}}]$, the algorithm identifies all FFT bins whose center frequencies fall within the band. Power is summed in linear units,

$$P_{\text{band}}(t) = \sum_{f \in [f_{\text{low}}, f_{\text{high}}]} 10^{A(f,t)/10}\, \Delta f,$$

and then normalized by the band width to obtain an average spectral density,

$$S_{\text{band}}(t) = \frac{P_{\text{band}}(t)}{\max(f_{\text{high}} - f_{\text{low}}, \Delta f)}.$$

Finally, the band-averaged spectral density is converted back to dB:

$$L_{\text{band}}(t) = 10 \log_{10}\left(\frac{S_{\text{band}}(t)}{p_{\text{ref}}^2}\right),$$

where $p_{\text{ref}} = 1$ µPa.

If a band contains no FFT bins, the algorithm falls back to the nearest FFT bin at the band centre. If a band contains exactly one bin, that bin's PSD is used directly. The resulting hybrid millidecade matrix has shape (time, band) and is stored in the HDF5 file alongside a matrix of band frequencies and edges.

2.5 PyPAM Comparison

To ensure that YAWN reproduces established hybrid millidecade metrics, its output was compared with the `PyPAM` reference implementation. A glider-mounted hydrophone recording was processed with both tools using matching FFT parameters, averaging intervals, and analysis bands. The comparison required careful alignment of calibration conventions. YAWN applies calibration through

`Si`, the hydrophone sensitivity in dB re 1 V/µPa. With `Si = 0`, voltage PSDs are converted using a neutral 1 V/µPa mapping.

PyPAM uses a different parameterization. Its `Hydrophone` object defines sensitivity and the peak-to-peak voltage corresponding to full-scale digital samples. When audio is read as floating-point values in $[-1, 1]$, the peak-to-peak range is two units; therefore, `Vpp = 2.0` must be specified. Using `Vpp = 1.0` would underestimate voltage by a factor of two and reduce acoustic power by 6 dB. For the comparison, PyPAM was configured with `sensitivity = 0` and `Vpp = 2.0`, yielding the same effective calibration as `Si = 0` in YAWN.

After processing, the hybrid millidecade spectra produced by each tool were averaged in linear space and converted back to decibels for comparison.

2.6 Calibration and comparison with PyPAM

To validate the Python implementation, hybrid millidecade spectra from YAWN were compared with those from the `PyPAM` package, using the two audio inputs with nominal analysis settings. The first audio file was from an underwater glider and sampled audio at 180,260 Hz and was 5 minutes long. The second audio file was the synthetic white noise file provided with the PAMGuide files. [3]. This file was 10 seconds long and sampled at 48 kHz.

The two systems differ in how they handle the relationship between digital sample values, volts, and acoustic pressure, so a fair comparison requires aligning their amplitude calibration conventions. In YAWN, the scalar `Si` (dB re 1 V/µPa) specifies the conversion from voltage PSD to acoustic PSD. Setting `Si = 0` enforces a neutral calibration of 1 V/µPa. For consistency, `Si = 0` was used in YAWN and PyPAM was configured with `sensitivity = 0` and `Vpp = 2.0`. Under these matched settings, the spectra produced by the two tools are nearly identical at the plotted scale, with a median offset of 0.45 dB and a 5th–95th percentile range of 0.06 dB to 1.04 dB.

PyPAM expresses calibration through a `Hydrophone` object. Floating-point audio is assumed to lie in the range $[-1, 1]$, which spans two units peak-to-peak and therefore corresponds to `Vpp = 2.0`. Assigning `sensitivity = 0` dB re 1 V/µPa reproduces the neutral calibration used in YAWN. If `Vpp` were instead set to 1.0, the voltage represented by the samples would be underestimated by a factor of two, shifting all PSD values downward by 6 dB.

With the calibration parameters harmonized, yPAM's hybrid millidecade bands method was run using the same FFT length, frequency bands, and time-bin sizes as YAWN. Time-averaged spectra for both implementations were then computed by averaging in the linear domain across time and converting the results to decibels.

3 Results

3.1 Visualization and long-term spectrograms

Example LTSAs are provided in Figure 1 and Example LTSAs are provided in Figure 2. Daily average was calculated by averaging 60 second Hybrid-milidecade bands over 15 minute segments.

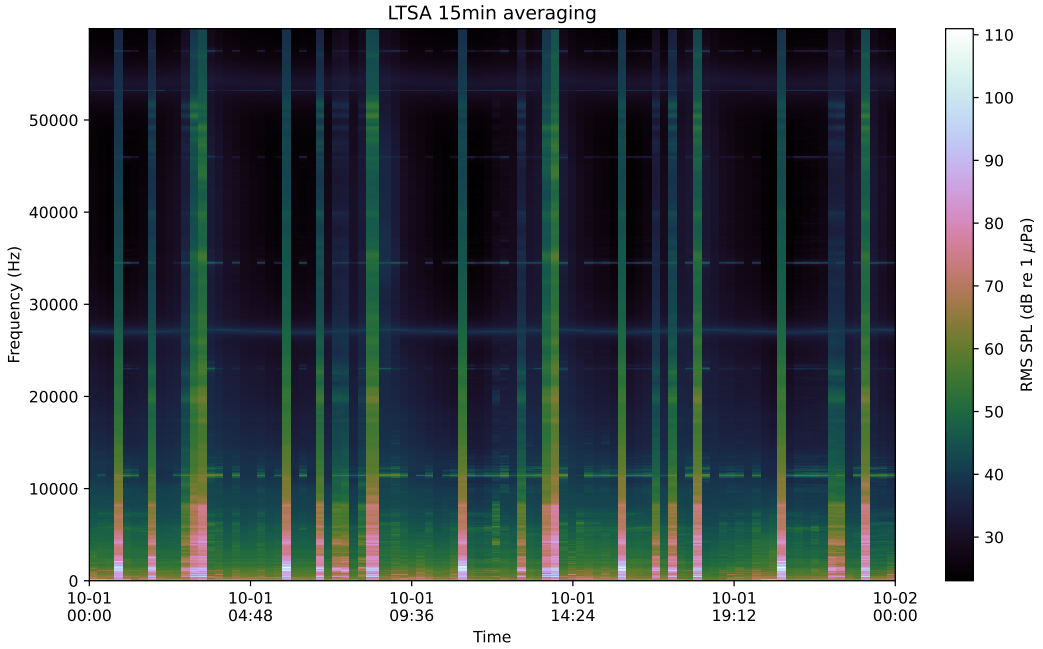

Figure 1: Example long-term spectral average (LTSA) for a single day, generated from calibrated spectra averaged over 15-minute intervals using the `plot_ltsa` function. Spectral density is shown in dB re 1 µPa^2/Hz on a $\log_{10}$ frequency axis, illustrating sub-daily variability in both level and spectral shape.

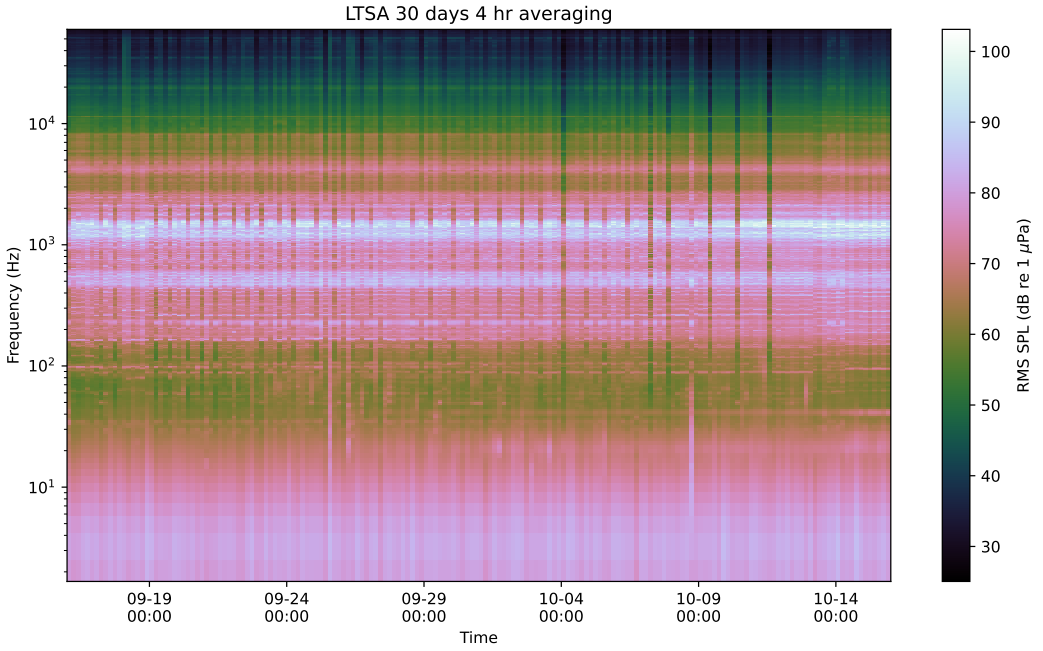

Figure 2: Example long-term spectral average (LTSA) for a 30-day period, with spectra averaged over 4-hour intervals. This panel highlights longer-term changes in ambient noise across the deployment while retaining sufficient spectral resolution for biologically relevant interpretation. As in Figure 1, the plot is generated directly from the HDF5 archives using the `plot_ltsa` function and displayed on a $\log_{10}$ frequency axis.

3.2 Calibration and comparison with PyPAM

Figure 3 and Figure 4 show the time-averaged hybrid millidecade spectra produced by YAWN and PyPAM for the glider and white noise recording, respectively. Binsize was set to 60 s, nfft was 180260 samples and obverlap was 50%.

The spectral shapes matched closely across the full analysis band for both audio files. To quantify the agreement, the difference between the two spectra (`YAWN` minus `PyPAM`) was computed for each millidecade band. For the glider audio file, median offset was 0.45 dB. The 5th–95th percentile interval of the offset distribution ranged from 0.06 dB to 1.04 dB. The residual difference was spectrally flat and small in magnitude, consistent with expected implementation details such as window normalization and effective noise bandwidth.

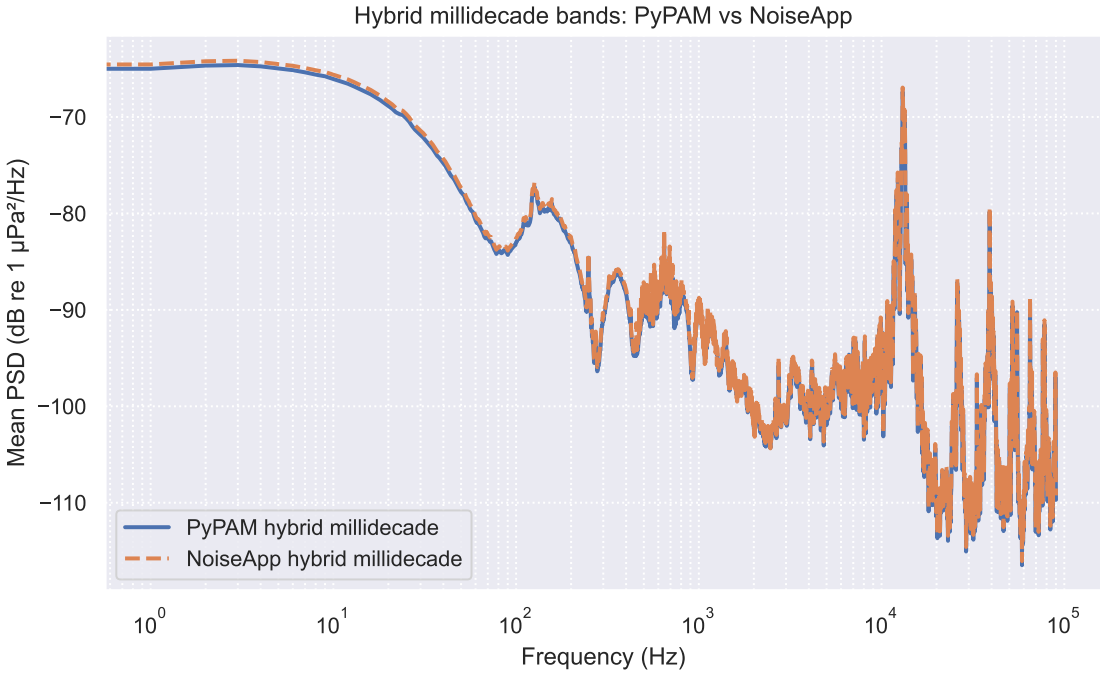

Figure 3: Time-averaged hybrid millidecade spectra for a glider-mounted hydrophone recording, computed with PyPAM and the Python-based YAWN implementation.

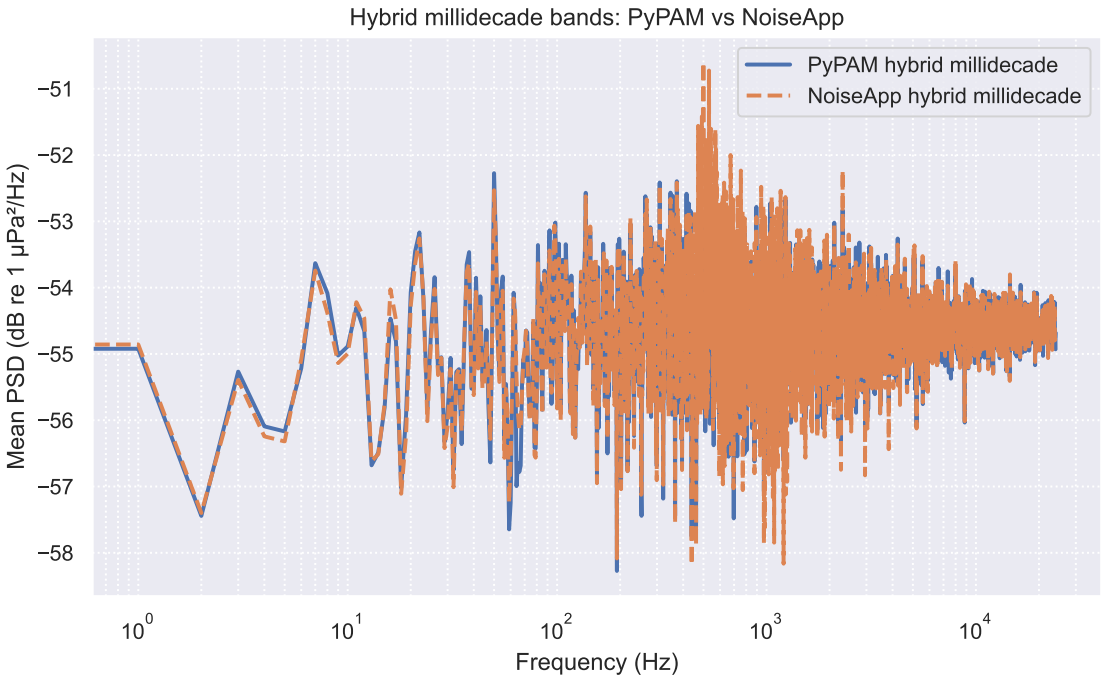

Figure 4: Time-averaged hybrid millidecade spectra for the synthetic white-noise recording, computed with PyPAM and the Python-based YAWN implementation.

4 Discussion

The direct comparison with PyPAM shows that the hybrid millidecade implementation in YAWN reproduces established results when equivalent analysis settings are used. After matching FFT length, overlap, averaging interval, frequency bands, and amplitude calibration, hybrid millidecade spectra from the two tools differed by less than 1 dB across the analysis band, with a median offset of 0.45 dB. The remaining differences were spectrally flat and small in magnitude, consistent with minor implementation details such as window normalization and effective noise bandwidth rather than systematic bias.

The comparison also highlights that differences between noise analysis tools often arise from calibration assumptions rather than from the spectral aggregation methods themselves. In particular, the handling of digital sample scaling and peak-to-peak voltage in PyPAM must be configured explicitly to match the voltage-to-pressure conversion used in YAWN. When these settings are not aligned, offsets on the order of several decibels can occur, even when FFT and band definitions are otherwise identical. Making these assumptions explicit is therefore essential for meaningful cross-tool comparisons and long-term consistency.

From a practical perspective, the YAWN implementation offers several advantages for large-scale monitoring:

- HDF5 storage of multiple metrics for space and easy sharing
- Local and cloud-hosted archive operation
- automatic calculations for hybrid milidecade, third octave, decade, and broadband calculations.
- Automatic datetime extraction from multiple audio types

- Easy modification with the noiseapp object being self contained in a single file requiring minimal dependencies
- Python integration widely used for machine learning, statistical modeling, and data visualization, facilitating integration with other components of analysis pipelines.
- Plotting options for long-term spectrograms with user-defined averaging periods and spectral density

5 Discussion

YAWN's design choices reflect practical constraints encountered in long-term passive acoustic monitoring. By processing audio sequentially and storing only band-integrated products rather than full PSD matrices, the framework reduces storage requirements while preserving spectral detail relevant to biological and environmental interpretation. Writing outputs to daily HDF5 files allows time series spanning months to years to be analyzed, summarized, and visualized without repeated access to raw audio, which is often impractical for large archives.

The internal structure of the HDF5 outputs further supports reproducibility and reuse. Each file contains a single deployment group that bundles spectral products, frequency axes, timestamps, and processing parameters in a self-describing format. This organization makes it straightforward to regenerate plots, compute additional summary statistics, or integrate noise metrics into downstream analyses without relying on external configuration files or undocumented defaults.

The hybrid millidecade representation implemented here is particularly useful for applications that require long-term spectral context rather than fine-scale temporal resolution. The linear-frequency region at low frequencies preserves structure associated with biologically important sounds and low-frequency noise sources, while the logarithmic spacing at higher frequencies provides compact coverage of the broadband soundscape. This balance makes the resulting products well suited for comparative studies, trend analysis, and coupling with propagation modeling or detection performance assessments.

6 Conclusion

This technical note introduces YAWN, a Python workflow for long-term underwater noise processing based on Merchant et al. (2015). YAWN streams audio in blocks, computes Welch PSDs, converts spectra to acoustic units, and summarizes them as broadband, third-octave, decade, and hybrid millidecade metrics. Outputs are stored as daily HDF5 files that include timestamps, frequency axes, and the processing settings needed to reproduce the analysis. The implementation is validated by comparing hybrid millidecade spectra to those produced by the established PyPAM package under matched analysis settings. With consistent calibration, the two tools agree within approximately 1 dB across the analysis band, with a median offset of 0.45 dB. The note also demonstrates plotting utilities that operate directly on the HDF5 archives, including long-term spectrograms with user-defined averaging windows.

References

[1] Samuel B. Martin, Brian J. Gaudet, Holger Klinck, Patrick J. Dugan, Jennifer L. Miksis-Olds, David K. Mellinger, David A. Mann, Olaf Boebel, Curtis C. Wilson, Douglas W. Ponirakis,

et al. Hybrid millidecade spectra: A practical format for exchange of long-term ambient sound data. *JASA Express Letters*, 1(1):011202, 2021. doi: 10.1121/10.0003249.

[2] Douglas Gillespie, DK Mellinger, JONATHAN Gordon, David Mclaren, PAUL Redmond, Ronald McHugh, PW Trinder, XY Deng, and A Thode. Pamguard: Semiautomated, open source software for real-time acoustic detection and localisation of cetaceans. *Journal of the Acoustical Society of America*, 30(5):54–62, 2008.

[3] Nathan D Merchant, Kurt M Fristrup, Mark P Johnson, Peter L Tyack, Matthew J Witt, Philippe Blondel, and Susan E Parks. Measuring acoustic habitats. *Methods in Ecology and Evolution*, 6(3):257–265, 2015.