The Journal of Open Source Software

mcsm-benchs: Benchmarking methods for multi-component signal processing

Juan M. Miramont [1,2¶], **Rémi Bardenet** [1], **Pierre Chainais** [1], and **François Auger** [2]

1 Université de Lille, CNRS, Centrale Lille, UMR 9189 Centre de Recherche en Informatique, Signal et Automatique de Lille (CRIStAL), Lille, France. **2** Nantes Université, Institut de Recherche en Énergie Électrique de Nantes Atlantique (IREENA, UR 4642), Saint-Nazaire, France. ¶ Corresponding author

DOI: 10.21105/joss.08175

Software

- Review
- Repository
- Archive

Editor: Marcel Stimberg

Reviewers:

- @nmy2103
- @tsbinns

Submitted: 20 February 2025
Published: 22 September 2025

Summary

Time-frequency (TF) representations are natural encodings of non-stationary time series, also termed signals, helping to discern patterns that reveal their time-varying frequency structure (Flandrin, 1998). The model one usually has in mind when discussing TF representations is a so-called multi-component signal (MCS), where the latter is thought to be the sum of several components of individual interest.

`mcsm-benchs` is an open-source Python library for creating reproducible benchmarks of methods that aim at extracting information from MCSs.

`Benchmark` objects can be created by simply passing a series of simulation parameters, a dictionary of methods and a set of performance metrics, while a `SignalBank` instance can be used to standardize comparisons across benchmarks. The `SignalBank` class can synthesize several MCSs as objects from a custom `Signal` class that behave like regular arrays in Python but also contain information about the components.

Additionally, `mcsm-benchs` includes a `ResultsInterpreter` class that can produce human-readable reports, the functionality of which underpins *collaborative* benchmarks (Moreau et al., 2022). These are based on online repositories (see here for an example) and can be periodically updated by members of the research community, fostering open and collaborative science. Several examples are given in the documentation, as well as a GitHub repository template that relies heavily on `mcsm-benchs` and continuous integration/deployment workflows, in order to automatize the process of publishing new benchmarks.

Statement of need

MCS processing is an area with a long history, and is still a field of methodological innovation (Bardenet et al., 2020; Bardenet & Hardy, 2021; Colominas et al., 2020; Ghosh et al., 2022; Krémé et al., 2020; Legros et al., 2024; Legros & Fourer, 2022; Pascal & Bardenet, 2022a, 2022b). In such a context, systematically comparing existing methods while keeping a record of how they performed on a predefined set of representative problems, i.e., benchmarking, can shed light on novel avenues of research and set clear baselines for new approaches. This is a widely adopted strategy in neighboring fields such as optimization (Bartz-Beielstein et al., 2020; Hansen et al., 2021; Moreau et al., 2022) and machine learning (Mattson et al., 2020), yet, to the best of our knowledge, there are no benchmarking tools for MCS processing.

Ideally, researchers would show the superior performance of their novel techniques by 1) selecting a number of competing approaches to compare with, 2) choosing synthetic or real-world signals that pose a challenge to the selected methods, 3) contaminating the signals with white or

colored noise, and 4) comparing the performance of the novel approaches under different signal-to-noise ratios. There are two problems with this pipeline. First, it is sometimes not clear which the state-of-the-art methods are. Second, the comparisons are commonly limited to very few signals selected to show the strengths of the new method.

`mcsm-benchs` brings a common framework to the table to easily carry out extensive comparisons between MCS-based approaches in a unified and *objective* way. It can be used to benchmark any number of approaches and create clear baselines for new methods that are accessible to the whole research community.

The toolbox is versatile enough to allow comparisons between many kinds of methods. For instance, `mcsm-benchs` was used to compare statistical tests for signal detection (Juan M. Miramont et al., 2022) and for denoising of synthetic and realistic signals under different scenarios, such as white noise or even real-world noises (Juan M. Miramont et al., 2024). As another example, many approaches within MCS processing focus instead on retrieving individual components and estimating their instantaneous frequencies. These methods can be easily benchmarked using `mcsm-benchs` as well (see Juan M. Miramont et al., 2023).

While the aforementioned cases illustrate the most typical applications in MCS processing, methods are in fact always treated like *black boxes* by `mcsm-benchs`. The only constraint imposed by the software is that the outputs of a method should match the inputs of the performance metrics given by the user. Thanks to this feature, `mcsm-benchs` has significant potential, as it could be used to systematically compare any signal processing algorithm. Large studies of methods for specific applications can thus be created and kept updated by the signal processing community using `mcsm-benchs`, hopefully leading to widely adopted procedures for evaluating new approaches that are transparent, less time-consuming, and straightforward for researchers to use.

Finally, in order to ease the adoption of this package by the community, `mcsm-benchs` also supports methods coded in `Octave`/`Matlab`, so that these can be seamlessly integrated into Python-based benchmarks.

Acknowledgements

This work was supported by grant ERC-2019-STG-851866, and French projects ANR20-CHIA-0002 and ASCETE-ANR19-CE48-0001. JMM would like to thank Guillaume Gautier and Yusuf Yiğit Pilavcı for their valuable insight.

References

Bardenet, R., Flamant, J., & Chainais, P. (2020). On the zeros of the spectrogram of white noise. *Applied and Computational Harmonic Analysis*, *48*(2), 682–705. https://doi.org/10.1016/j.acha.2018.09.002

Bardenet, R., & Hardy, A. (2021). Time-frequency transforms of white noises and Gaussian analytic functions. *Applied and Computational Harmonic Analysis*, *50*, 73–104. https://doi.org/10.1016/j.acha.2019.07.003

Bartz-Beielstein, T., Doerr, C., Berg, D. van den, Bossek, J., Chandrasekaran, S., Eftimov, T., Fischbach, A., Kerschke, P., La Cava, W., Lopez-Ibanez, M., & others. (2020). *Benchmarking in optimization: Best practice and open issues*. arXiv. https://doi.org/10.48550/arXiv.2007.03488

Colominas, M. A., Meignen, S., & Pham, D.-H. (2020). Fully adaptive ridge detection based on STFT phase information. *IEEE Signal Processing Letters*, *27*, 620–624. https://doi.org/10.1109/lsp.2020.2987166

Flandrin, P. (1998). *Time-frequency/time-scale analysis*. Academic Press. ISBN: 9780080543031

Ghosh, S., Lin, M., & Sun, D. (2022). Signal analysis via the stochastic geometry of spectrogram level sets. *IEEE Transactions on Signal Processing*, *70*, 1104–1117. https://doi.org/10.1109/tsp.2022.3153596

Hansen, N., Auger, A., Ros, R., Mersmann, O., Tušar, T., & Brockhoff, D. (2021). COCO: A platform for comparing continuous optimizers in a black-box setting. *Optimization Methods and Software*, *36*, 114–144. https://doi.org/10.1080/10556788.2020.1808977

Krémé, A. M., Emiya, V., Chaux, C., & Torresani, B. (2020). Filtering out time-frequency areas using Gabor multipliers. *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 5790–5794. https://doi.org/10.1109/icassp40776.2020.9053482

Legros, Q., & Fourer, D. (2022). *Time-frequency ridge estimation of multi-component signals using sparse modeling of signal innovation*. https://doi.org/10.48550/arXiv.2212.11343

Legros, Q., Fourer, D., Meignen, S., & Colominas, M. A. (2024). Instantaneous frequency and amplitude estimation in multicomponent signals using an EM-based algorithm. *IEEE Transactions on Signal Processing*, *72*, 1130–1140. https://doi.org/10.1109/TSP.2024.3361713

Mattson, P., Cheng, C., Diamos, G., Coleman, C., Micikevicius, P., Patterson, D., Tang, H., Wei, G.-Y., Bailis, P., Bittorf, V., & others. (2020). MLPerf training benchmark. In *Proceedings of Machine Learning and Systems* (Vol. 2, pp. 336–349). https://doi.org/10.48550/arXiv.1910.01500

Miramont, Juan M., Bardenet, R., Chainais, P., & Auger, F. (2022). A public benchmark for denoising and detection methods. *XXVIIIème Colloque Francophone Du GRETSI*, 1–4.

Miramont, Juan M., Bardenet, R., Chainais, P., & Auger, F. (2024). *Benchmarking multi-component signal processing methods in the time-frequency plane*. arXiv. https://doi.org/10.48550/arXiv.2402.08521

Miramont, Juan M., Legros, Q., Fourer, D., & Auger, F. (2023). Benchmarks of multi-component signal analysis methods. *EUSIPCO*, 1783–1787. https://doi.org/10.23919/eusipco58844.2023.10290093

Moreau, T., Massias, M., Gramfort, A., Ablin, P., Bannier, P.-A., Charlier, B., Dagréou, M., Dupré la Tour, T., Durif, G., F. Dantas, C., Klopfenstein, Q., Larsson, J., Lai, E., Lefort, T., Malézieux, B., Moufad, B., T. Nguyen, B., Rakotomamonjy, A., Ramzi, Z., … Vaiter, S. (2022). Benchopt: Reproducible, efficient and collaborative optimization benchmarks. *NeurIPS*. https://doi.org/10.48550/arXiv.2206.13424

Pascal, B., & Bardenet, R. (2022a). Une famille de représentations covariantes de signaux discrets et son application à la détection de signaux à partir de leurs zéros. *XXVIIIème Colloque Francophone Du GRETSI*, 1–4.

Pascal, B., & Bardenet, R. (2022b). A covariant, discrete time-frequency representation tailored for zero-based signal detection. *IEEE Transactions on Signal Processing*, *70*, 2950–2961. https://doi.org/10.1109/tsp.2022.3181342