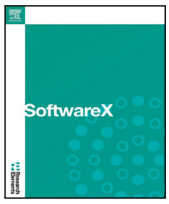
SoftwareX 31 (2025) 102187

Contents lists available at ScienceDirect

SoftwareX

journal homepage: www.elsevier.com/locate/softx



Original software publication

Spectrally regularised LVMs: A spectral regularisation framework for latent variable models designed for single-channel applications

Ryan Balshaw [a,*], P. Stephan Heyns [a], Daniel N. Wilke [a,b], Stephan Schmidt [a]

[a] Centre for Asset Integrity Management, Department of Mechanical and Aeronautical Engineering, University of Pretoria, Pretoria, South Africa
[b] School of Mechanical, Industrial & Aeronautical Engineering, University of the Witwatersrand, Johannesburg, South Africa

ARTICLE INFO

Keywords:
Latent variable models
Spectral regularisation
Python

ABSTRACT

Latent variable models (LVMs) are commonly used to capture the underlying dependencies, patterns, and hidden structures in observed data. Source duplication is a by-product of the data Hankelisation pre-processing step common to single-channel LVM applications, which hinders practical LVM utilisation. In this article, a Python package titled `spectrally-regularised-LVMs` is presented. The proposed package addresses the source duplication issue by adding a novel spectral regularisation term. This package provides a framework for spectral regularisation in single-channel LVM applications, thereby making it easier to investigate and utilise LVMs with spectral regularisation. This is achieved via symbolic or explicit representations of potential LVM objective functions, which are incorporated into a framework that uses spectral regularisation during the LVM parameter estimation process. This package aims to provide a consistent linear LVM optimisation framework incorporating spectral regularisation and caters to single-channel time-series applications.

Code metadata

Current code version	v0.1.4-post.1
Permanent link to code/repository used for this code version	https://github.com/ElsevierSoftwareX/SOFTX-D-23-00490
Permanent link to Reproducible Capsule	https://doi.org/10.5281/zenodo.14717880
Legal Code License	MIT license
Code versioning system used	git
Software code languages, tools, and services used	Python3
Compilation requirements, operating environments & dependencies	Python ≥ 3.11; Numpy $\geq 1.23.1, < 2.0$; Matplotlib $\geq 3.5.2$; SciPy $\geq 1.8.1$; scikit-learn $\geq 1.1.2$; tqdm $\geq 4.64.1$; and SymPy $\geq 1.11.1, < 1.13$
If available Link to developer documentation/manual	https://spectrally-regularised-lvms.readthedocs.io/en/latest/
Support email for questions	ryanbalshaw1@gmail.com

1. Introduction

Latent variable models (LVMs) represent a statistical methodology which aims to extract hidden sources of information, referred to as the latent variables $\mathbf{z} \sim p(\mathbf{z}), \mathbf{z} \in \mathbb{R}^d$, from an observed set of random variables $\mathbf{x} \sim p(\mathbf{x}), \mathbf{x} \in \mathbb{R}^D$. The overarching goal of the LVM framework, as shown in Fig. 1, is to *(i)* construct a generative model to generate observed samples and *(ii)* infer the latent variables, commonly referred to as the latent sources, from the observed data [1,2]. In the linear LVM formulation, a linear encoding and decoding step is used to obtain samples from the posterior distribution $p(\mathbf{z}|\mathbf{x})$ and the generative distribution $p(\mathbf{x}|\mathbf{z})$.

In common LVM frameworks, e.g., principal component analysis (PCA) [3–6], a Gaussian assumption is made for the prior $p(\mathbf{z})$, the generative distribution $p(\mathbf{x}|\mathbf{z})$ and posterior distribution $p(\mathbf{z}|\mathbf{x})$, and the latent sources are driven to capture components that best describe the dominant sources in the observed data via minimum mean-square error compression or variance maximisation [5]. In this way, these methodologies are sample generation-focused and the latent sources capture the dominant variance in the observed data. The PCA framework deals

* Corresponding author.
E-mail addresses: ryanbalshaw1@gmail.com (R. Balshaw), stephan.heyns@up.ac.za (P.S. Heyns), daniel.wilke@wits.ac.za (D.N. Wilke), stephan.schmidt@up.ac.za (S. Schmidt).

https://doi.org/10.1016/j.softx.2025.102187
Received 31 July 2023; Received in revised form 10 February 2025; Accepted 30 April 2025
Available online 22 May 2025

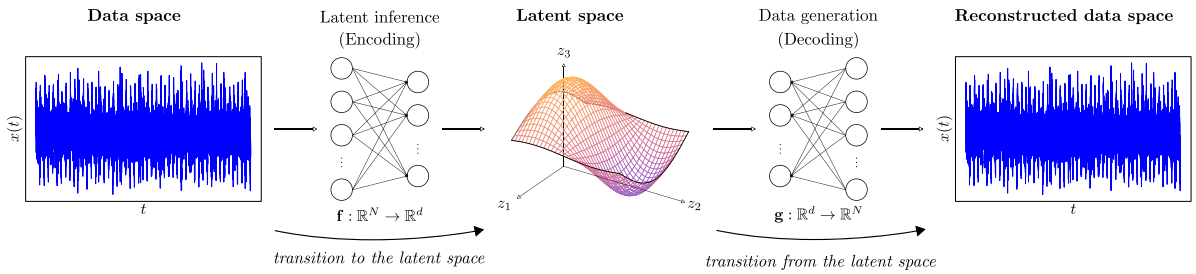


Fig. 1. An overview of a LVM framework applied to single-channel time-series data. Note that the dimensionality of the latent space is presented in $\mathbb{R}^3$, and no data pre-processing steps are considered between the data space and the latent space. Data pre-processing steps are discussed in Section 2.1.

with second-order statistics and results in uncorrelated latent sources, i.e., the covariance for two sources z_i and z_j, represented by $\text{cov}\left[z_i, z_j\right]$, for $i \neq j$ is zero and non-zero otherwise. The independent component analysis (ICA) formulation manifests as an LVM with the latent sources assumed to be non-Gaussian and mutually independent [7–10]. In ICA frameworks, statistical independence and non-Gaussianity assumptions are more rigorous and enforce that the latent sources should be maximally informative in a non-Gaussian setting [11,12]. The statistical independence assumption implies that the latent distribution factorises, i.e., $p(\mathbf{z}) = \prod_{i=1}^{d} p_i(z_i)$, where $\mathbf{z} \in \mathbb{R}^d$. In the assumed Gaussian setting, it is possible to show that the PCA linear transformation can produce a statistically independent latent distribution [5].

In applications when a single-channel time-series signal is of interest and available, certain processing steps, e.g., data Hankelisation, must be taken to enable the LVM framework fully [13]. The single-channel time-series signal of interest is represented as $x[n]$ indexed by $n \in \{0, 1, \ldots, N-1\}$. The signal $x[n]$ is discretely sampled over a duration of T at a sampling frequency of F_s, yielding $N = T \cdot F_s$ samples. In this setting, as noted in Hyvärinen [12], the linear LVM framework is strongly related to the blind-deconvolution problem [14]. The blind deconvolution problem aims to simultaneously recover the underlying latent sources and identify a single unknown inverse filter that recovers the underlying source signal [15]. Separating the underlying latent sources is complex and inherently results in an underdetermined system, as the observed data does not uniquely specify the sources.

When using LVM frameworks, the input data needs to be multi-dimensional. For single-channel applications, the vibration signal $x[n]$ is Hankelised to construct the required matrix, with the matrix's rows constructed from overlapping segments of the input signal. However, this process introduces redundancy: the columns of the Hankel matrix represent time-delayed versions of the observed time-series signal [13]. This representation indicates that the features of the observed data $\mathbf{x}$ characterise similar dynamical behaviour. Given the relationship to blind deconvolution for single-channel applications, the LVM parameter estimation process may inadvertently extract multiple latent sources with overlapping spectral content when estimating the linear encodings to obtain the latent variables $\mathbf{z}$ [12]. This process leads to source duplication [16]. Thus, while the linear LVM framework seeks a set of informative latent components, in single-channel applications the effective source signals may partially or completely overlap in their spectral support when using data Hankelisation [17].

Davies and James [17] demonstrated that the sources in the observed data must have disjoint spectral support to fully recover the latent sources. Additionally, if no modifications are made to the standard LVM objective functions, clustering algorithms must combine extracted sources that contain duplicate information [17]. Source duplication is an existing limitation that hinders the practicality of certain LVM objective functions in single-channel time-series applications as the full-rank solution, i.e., $d = D$, is required to avoid missing any interesting sources. Additionally, the model estimation step may be computationally extensive if $d \ll D$ as much computation is wasted on duplicate information. Hence, we seek to bypass this limitation to improve the practicality of linear LVMs. This work proposes a framework that offers ease of objective function implementation, and a spectral regularisation term is proposed and implemented to enforce the disjoint requirement.

The proposed framework caters to spectral regularisation for linear LVM formulations, and the `spectrally-regularised-LVMs` package makes spectral regularisation readily accessible for first-order and second-order optimisers. The package provides simplicity, ease of use, and efficiency to the single-channel LVM framework, whereby the burden of repetitive pre-processing and optimisation framework implementation tasks is no longer placed on the user. The proposed framework operates in a deflation-based optimisation setting, i.e., each latent source is solved for individually. Additionally, the methodology is objective function agnostic, which allows users to implement their cost functions and use the proposed framework. Users can implement their objective functions directly using the symbolic Python package SymPy [18] to obtain the gradient and Hessian of the model objective function automatically or using standard linear algebra operations enabled though the NumPy library [19]. Additionally, a finite-difference scheme is provided should users wish to obtain approximations of the gradient and Hessian of the model objective function. This provides generality to the `spectrally-regularised-LVMs` package and creates flexibility for future development and package utilisation.

1.1. Related work

In the literature, spectral regularisation typically refers to a regularisation technique for learning-based problems which utilises the spectra of an operator or matrix to control the properties of some model [20], e.g., Tikhonov regularisation [21]. For matrices, the spectra refer to their eigenvalue set [22]. Gerfon et al. [23] provide a succinct overview of spectral regularisation techniques for supervised learning tasks. Miyato et al. [24] use the spectral properties of deep generative models to improve model performance. Specifically, Miyato et al. [24] use the spectral norm of weight matrices to control the Lipschitz continuity of the discriminator network function for generative adversarial networks to improve model trainability. Xie et al. [25] use a spectral loss based on cosine similarity to encourage reconstruction consistency for anomaly detection using adversarial auto-encoders. In this work, spectral regularisation refers to the regularisation of the spectra of the LVM component vectors, where the spectra are obtained with the Fourier transform to ensure that it does not capture the same signal information as other component vectors. In this way, spectral orthogonality within the LVM components is encouraged, and the signal information captured by the LVM is driven to consist of unique components.

2. Model formulation

In this section, the pre-processing steps for single-channel time-series are detailed, the preliminary formulation of linear LVMs is presented, and the proposed spectral regularisation term is defined.

2.1. Time-series pre-processing

For single-channel time-series data, we obtain a signal $x[n]$ that must be pre-processed to fit into the standard LVM framework. This can be achieved through data Hankelisation to obtain a matrix $\mathbf{X} \in \mathbb{R}^{L_H \times L_w}$

$$\mathbf{X} = \begin{bmatrix} x[1] & x[2] & \cdots & x[L_w] \\ x[L_{sft}] & x[L_{sft}+1] & \cdots & x[L_{sft}+L_w] \\ x[2\cdot L_{sft}] & x[2\cdot L_{sft}+1] & \cdots & x[2\cdot L_{sft}+L_w] \\ \vdots & \vdots & \ddots & \vdots \\ x[L_{sft}\cdot(L_H-1)] & \cdots & \cdots & x[L_{sft}\cdot(L_H-1)+L_w] \end{bmatrix}, \tag{1}$$

where L_w is the window length, L_{sft} is the shift parameter and $L_H = \lfloor \frac{L-L_w}{L_{sft}} \rfloor + 1$ represents the number of rows in $\mathbf{X}$. Note that the shift parameter is typically set to $L_{sft} = 1$ to ensure that the LVM component vectors act as a set of finite impulse response (FIR) filters [17].

2.2. Linear latent variable models

The basic linear LVM generative model may be expressed for some observed data $\mathbf{x} \in \mathbb{R}^D$ as

$$\mathbf{x} = \sum_{i=1}^{d} \mathbf{a}_i z_i, \tag{2}$$

where $d \leq D$, z_i represents the i^{th} latent source, and $\mathbf{a}_i \in \mathbb{R}^D$ represents the i^{th} decoding transition vector. The matrix–vector notation of the generative model is

$$\mathbf{x} = \mathbf{A}\mathbf{z}, \tag{3}$$

where $\mathbf{A} \in \mathbb{R}^{D\times d} = \left[\mathbf{a}_1, \ldots, \mathbf{a}_d\right]$. In the latent encoding step, a matrix $\mathbf{W} \in \mathbb{R}^{d\times D} = [\mathbf{w}_1, \ldots, \mathbf{w}_d]^T$ can be used to infer the latent vector $\mathbf{z}$ by

$$\mathbf{z} = \mathbf{W}\mathbf{x}, \tag{4}$$

where the rows in $\mathbf{W}$ represent the encoding transition vectors. In the LVM model, samples $\mathbf{x} \sim p(\mathbf{x})$ are observed from some unknown distribution $p(\mathbf{x})$. The goal is to estimate $\mathbf{A}$ or $\mathbf{W}$. Common methodologies prefer to utilise characteristics of the latent sources to estimate the model parameters, e.g., maximum latent variance [4] or maximum non-Gaussianity through measures such as kurtosis or negentropy [11]. The objective function of interest is henceforward generally denoted by $\mathcal{L}_{model}$.

2.3. Spectral regularisation

In this work, we propose using a spectral regularisation term for linear LVM objective functions to combat source duplication and use a general optimisation framework to estimate the model parameters. The assumed form of the general LVM objective function may be written as

$$\begin{aligned} \min_{\mathbf{w}_i} \quad & \mathcal{L}_{model}(\mathbf{w}_i) + \mathcal{L}_{sr}(\mathbf{w}_i) \\ \text{such that} \quad & \mathbf{w}_i^T\mathbf{w}_i = 1, \end{aligned} \tag{5}$$

where $\mathcal{L}_{model}(\mathbf{w}_i)$ represents the objective function to be minimised, $\mathcal{L}_{sr}(\mathbf{w}_i)$ represents the spectral regulariser that acts as an explicit regularisation term on $\mathcal{L}_{model}(\mathbf{w}_i)$, and the optimisation constraint $\mathbf{w}_i^T\mathbf{w}_i = 1$ is used to ensure that the parameter estimation process focuses on the direction of $\mathbf{w}_i$ and not its magnitude. The spectral regularisation term encourages the $\mathbf{w}_i$ solution to be spectrally unique with respect to the previous solution vectors $\mathbf{w}_j$, where $j < i$. The general objective function can be reformulated into a Lagrangian expression

$$\begin{aligned} \mathcal{L}(\mathbf{w}_i, \lambda_{eq}) = {} & \mathcal{L}_{model}(\mathbf{w}_i) + \mathcal{L}_{sr}(\mathbf{w}_i) \\ & + \lambda_{eq}\left(\mathbf{w}_i^T\mathbf{w}_i - 1\right), \end{aligned} \tag{6}$$

where λ_{eq} represents the Lagrange multiplier. The regularisation term proposed in this work enforces spectral orthogonality. It uses the squared modulus of the Fourier domain representation of a vector $\mathbf{w}_i$, denoted as $\mathbf{b}(\mathbf{w}_i)$, to minimise the dot product between $\mathbf{b}(\mathbf{w}_i)$ and the squared Fourier representation for the previously solved $\mathbf{w}_j$ vectors, $j = 1, \ldots, i-1$. This is given as

$$\mathcal{L}_{sr}(\mathbf{w}_i) = \alpha \sum_{j=1}^{i-1} \mathbf{b}(\mathbf{w}_i)^T\mathbf{b}(\mathbf{w}_j), \quad i > 1 \tag{7}$$

where α represents a penalty enforcement parameter that controls the importance of the regularisation term. Note that the spectral regularisation term is only applied after estimating the first projection vector $\mathbf{w}_1$. To remove ambiguity behind the choice of α, the sequential unconstrained minimisation technique (SUMT) is used within the model optimisation step to iteratively increase the prominence of the regularisation term during each optimisation iteration [26]. The important terms related to the implemented optimisation algorithm to incorporate the regularisation term in Eq. (7) into the overall LVM objective are detailed in the documentation. For brevity, the full derivation is not included here. Using the spectral regularisation term in Eq. (7) allows the basic LVM model to apply to single-channel time-series data effectively. There is no reliance on carefully selected pre-processing strategies, e.g., the spectrogram, and source duplication is discouraged during the parameter estimation process.

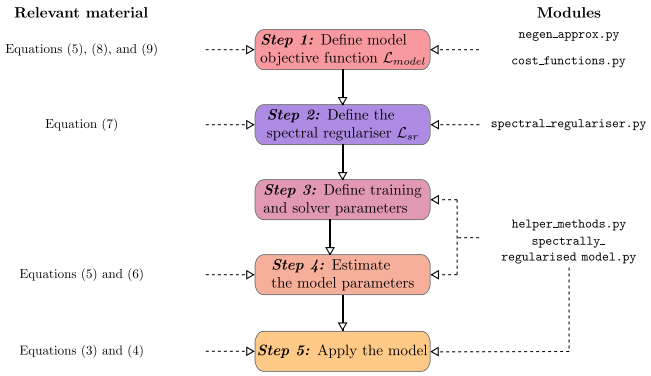


Fig. 2. A summary of the important mathematical aspects providing context to the software implementation and architecture, the associated package sub-modules, and any relevant material in this document.

3. Software description

This section provides a basic description of the software used in the `spectrally-regularised-LVMs` Python package. Fig. 2 provides context for how the relevant mathematical components of the package can be sequentially organised.

3.1. Introduction to spectrally-regularised-LVMs

The `spectrally-regularised-LVMs` package provides a framework for linear LVMs with the proposed spectral regularisation term using the Python API. In this framework, the objective function for the LVM can be implemented symbolically or explicitly as a Python function that uses NumPy objects. Alternatively, the objective function for two common LVMs, PCA and ICA, can be called through the sub-modules present in the package. The choice of a symbolic objective function implementation ensures that users do not need to spend effort on deriving the gradient and Hessian of their objective function, as this can be done symbolically. For the explicit implementation, users can provide the necessary first and second-order information or use a finite-difference approximation scheme to estimate the gradient and Hessian. As detailed in Fig. 2, five steps are required to formalise the proposed LVM framework in single-channel applications. These five steps are

```python
import sympy as sp
import spectrally_regularised_lvms as srLVMs

# Symbolic cost function implementation
cost_inst = srLVMs.SymbolicCost()

z, j, n = cost_inst.get_symbolic_parameters()

loss = -1/n * sp.Sum((z[j])**2, (j))

cost_inst.set_cost(loss)
```

(a) Symbolic objective function.

```python
import numpy as np
import spectrally_regularised_lvms as srLVMs

# Explicit cost function implementation
cost_inst = srLVMs.ExplicitCost()

obj = lambda X, w, z: -1 * np.mean((X @ w)**2, axis = 0)
grad = lambda X, w, z: -2 * np.mean((X @ w) * X, axis = 0,
                                    keepdims=True).T
hess = lambda X, w, z: -2 / X.shape[0] * (X.T @ X)

cost_inst.set_cost(obj)
cost_inst.set_gradient(grad)
cost_inst.set_hessian(hess)
```

(b) Explicit objective function.

Fig. 3. The code listings detailing how the PCA objective function given in Eq. (8) can be implemented symbolically or explicitly using functions that use NumPy objects [19].

combined within the package to create a framework from which users can utilise LVMs with spectral regularisation under a general model objective function $\mathcal{L}_{model}$. This makes it easier to implement LVMs with spectral regularisation and apply them to single-channel time-series applications. Given some time-series signal $x[n]$, the framework handles the signal pre-processing, spectral regularisation, and model estimation steps, thereby automating the involved aspects of the LVM task.

3.2. Software architecture

The `spectrally-regularised-LVMs` module comprises five sub-modules, which were designed to facilitate the steps detailed in Fig. 2. To facilitate step one from Fig. 2, the `cost_functions` sub-module was implemented. This provides four cost function classes to the user, of which two are dedicated to two simple LVM formulations, one which allows users to explicitly encode their cost function and its associated derivatives, and one which allows users to define their cost function symbolically using the SymPy package [18] and internally determines the associated gradient and Hessian of the objective function. Additionally, a finite-difference approximation scheme can approximate the gradient and Hessian. The `negen_approx` sub-module supports step one by providing estimator classes, which provide access to a set of approximation functions for negentropy [27]. The `spectral_regulariser` sub-module supports step two from Fig. 2 by implementing a spectral regulariser class that can be used to determine the regularisation term given in Eq. (7) alongside its gradient and Hessian. The `helper_methods` sub-module enables step three from Fig. 2, by providing a set of classes which assist with data pre-processing, data batch sampling, quasi-Newton update strategies [26], Gram–Schmidt orthogonalisation [28], and time-series signal Hankelisation [29]. The `spectrally_regularised_model` sub-module captures steps three and four from Fig. 2, respectively, and provides the linear LVM model which allows users to define their training parameters and estimate the model parameters using a `.fit`$(\cdot)$ method call to an instance of the `LinearModel` class [30]. Finally, to apply the LVM as detailed in step five from Fig. 2, methods from the `LinearModel` class facilitate the transition to and from the latent space, as detailed in Fig. 1. A `.transform`$(\cdot)$ method call computes the transformation from the data space to the latent space, and an `.inverse_transform`$(\cdot)$ method call performs the transformation from the latent space back to the data space. These method calls are available to instances of the `LinearModel` class. This provides a complete framework that automatically performs the necessary steps to estimate the parameters of linear LVMs with the proposed spectral regularisation term and apply the LVM encoding and decoding steps.

3.3. Software functionalities

The primary functionality of the `spectrally-regularised-LVMs` package is to provide a complete framework that estimates the linear LVM model parameters that are regularised via the spectral orthogonality term. This is achieved through a cost function initialisation stage and a parameter estimation stage. Each stage is discussed in turn and code listings are given. The cost function initialisation is generalised by offering a symbolic or explicit representation to cater to a wide variety of objective functions. For this example, the PCA objective function is used

$$\begin{aligned}\mathcal{L}_{model}(\mathbf{w}_i) &= -\mathbb{E}_{\mathbf{x}\sim p(\mathbf{x})}\{\left(\mathbf{w}_i^T\mathbf{x}\right)^2\} \\ &= -\frac{1}{N}\sum_{j=1}^{N} z_j^2, \end{aligned} \tag{8}$$

where $\{\mathbf{x}_1, \ldots, \mathbf{x}_N\} \overset{\text{i.i.d.}}{\sim} p(\mathbf{x})$ are independent and identically distributed (i.i.d) samples from the distribution $p(\mathbf{x})$, $\mathbb{E}_{\mathbf{x}\sim p(\mathbf{x})}\{\mathbf{x}\} = 0$, i.e., the data has zero mean, and $z_j = \mathbf{w}_i^T\mathbf{x}_j$ denotes the projection of sample $\mathbf{x}_j$ onto the direction specified by unit vector $\mathbf{w}_i$. In a symbolic representation, the PCA objective function can be implemented following the example in Fig. 3(a). In an explicit representation, the PCA objective function can be implemented using functions that use NumPy objects as detailed in Fig. 3(b).

The parameter estimation stage provides the user alternative functionalities and advanced options regarding the underlying pre-processing and model estimation steps. The main component is to create an instance of the `LinearModel` class and use the `.fit`$(\cdot)$ method to estimate the model parameters. Further information related to the arguments for the parameter estimation stage is given in the documentation. A code listing example of this stage is shown in Fig. 4.

3.4. Software dependencies

The `spectrally-regularised-LVMs` package is free software distributed under the MIT license, and it is open to contributions on GitHub. The package is hosted on PyPi [31] to enable ease of installation, broader accessibility to the scientific community, and to avoid dependency issues when using the package in a user's local Python environment. The package is dependent on the following Python packages: NumPy [19], Matplotlib [32], scikit-learn [33], tqdm [34] and SymPy [18].

4. Illustrative example

To demonstrate the capabilities and detail the main functionalities of the `spectrally-regularised-LVMs` package, an illustrative example is used. In this example, steps one through five from Fig. 2 are applied to the negentropy-based LVM objective function [12]. This objective function is given as

$$\mathcal{L}_{model}(\mathbf{w}_i) = -\left(\mathbb{E}_{\mathbf{x}\sim p(\mathbf{x})}\{G(\mathbf{w}_i^T\mathbf{x})\} - \mathbb{E}_{\nu\sim p(\nu)}\{G(\nu)\}\right)^2, \tag{9}$$

where $G(\cdot)$ is a non-quadratic function and ν represents a Gaussian variable that is zero mean and unit variance [27]. The objective function

```python
import numpy as np
import spectrally_regularised_lvms as srLVMs

# Define a toy signal
x_signal = np.random.randn(10000)

# Define the cost function instance
cost_inst = ...

# Define the model
model_inst = srLVMs.LinearModel(n_sources = 5,
                                cost_instance = cost_inst,
                                Lw = 256,
                                Lsft = 1)

# Estimate the model parameters
model_inst.fit(x_signal)

# Obtain the latent representation Z
Z = model_inst.transform(x_signal)

# Obtain the recovered representation of X
X_recon = model_inst.inverse_transform(Z)
```

Fig. 4. The code listing detailing how to perform parameter estimation in the spectrally regularised framework.

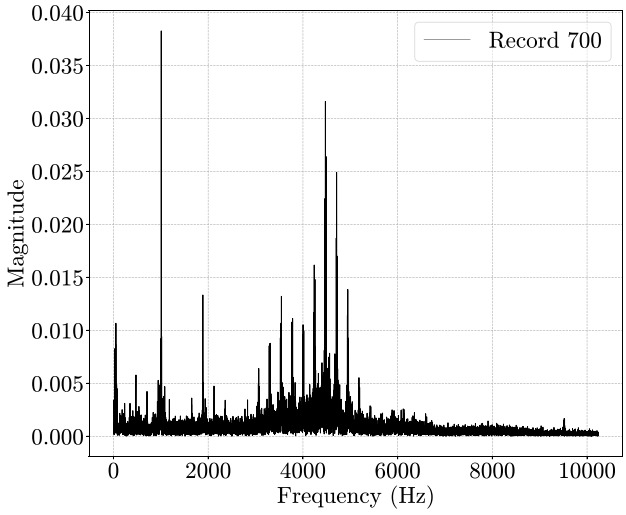


Fig. 5. The signal spectrum of record 700 from the second IMS dataset.

choice of non-Gaussianity is supported by its established effectiveness to extract useful diagnostic information in vibration signals from rotating machinery [35].

A time-series signal from the Intelligent Maintenance Systems (IMS) bearing dataset is used in this example [36]. The intention is to provide evidence of the applicability of the spectrally regularised framework on real-world data. Record 700 from the second IMS dataset is used from the first bearing channel as it is expected to contain a vibration signature with a faulty component [37]. The 'Examples' directory in the Github repository contains a notebook detailing this step-by-step use of the proposed package for this example and contains all code for result reproducibility. In Fig. 5, the spectra of the record 700 are shown. The sources from the estimated linear LVM were analysed, and their respective spectra with and without spectral regularisation are presented in Fig. 6. Evidently, without spectral regularisation, there is clear source duplication (left plot), and this issue is addressed with the addition of the proposed spectral regularisation term (right plot).

5. Impact

The `spectrally-regularised-LVMs` package provides a framework for linear LVMs, which utilises spectral regularisation for single-channel time-series data applications. This was done to streamline the application process by readily adapting the parameter estimation framework to depend on a generic objective function class instance, where the user can use different model objective functions depending on their required application. Furthermore, explicit control over the model parameter estimation process is provided to ensure that different pre-processing and parameter estimation approaches can be used. This allows for efficient iteration for a given application through different optimisation strategies, e.g., steepest gradient descent, stochastic gradient descent, or constrained Newton/quasi-Newton methods [26].

The Python package aims to provide researchers with a framework that provides a general parameter estimation methodology for linear LVMs and also removes latent redundancy via the proposed spectral regularisation term. The proposed regularisation term is objective function agnostic in its formulation, and providing access to a Python package for LVM parameter estimation under this term is expected to benefit the broader community. This allows the utilisation of the proposed methodology for various linear LVM objective formulations without re-implementing all aspects of linear LVMs from scratch.

Finally, while many popular linear LVM applications are available in Python, e.g., PCA and ICA in scikit-learn [33], these implementations cater to a general class of LVM applications and are not focused on applying the methodologies to single-channel time-series data. The `spectrally-regularised-LVMs` package provides direct access to the linear LVM framework and caters to the referenced methodology application, which is expected to offer fruitful avenues for future research.

6. Conclusion

The paper proposes and implements a modular linear LVM framework in Python. This framework offers customisable objective functions, facilitates the estimation of the model parameters, and uses a novel spectral regularisation term to combat source duplication. By making the `spectrally-regularised-LVM` framework accessible to the broader community, it makes the application and development of new methods easier, and we firmly believe it will benefit future research into LVMs for single-channel time-series data. Finally, the `spectrally-regularised-LVMs` package is envisaged to stimulate and benefit future research into single-channel time-series data applications.

CRediT authorship contribution statement

Ryan Balshaw: Conceptualization, Formal analysis, Investigation, Methodology, Software, Validation, Visualization, Writing – original draft. **P. Stephan Heyns:** Conceptualization, Funding acquisition, Methodology, Project administration, Supervision, Writing – review & editing. **Daniel N. Wilke:** Conceptualization, Methodology, Resources, Supervision, Writing – review & editing. **Stephan Schmidt:** Conceptualization, Methodology, Resources, Supervision, Writing – review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

The authors gratefully acknowledge the support received from AngloGold Ashanti in the execution of this research.

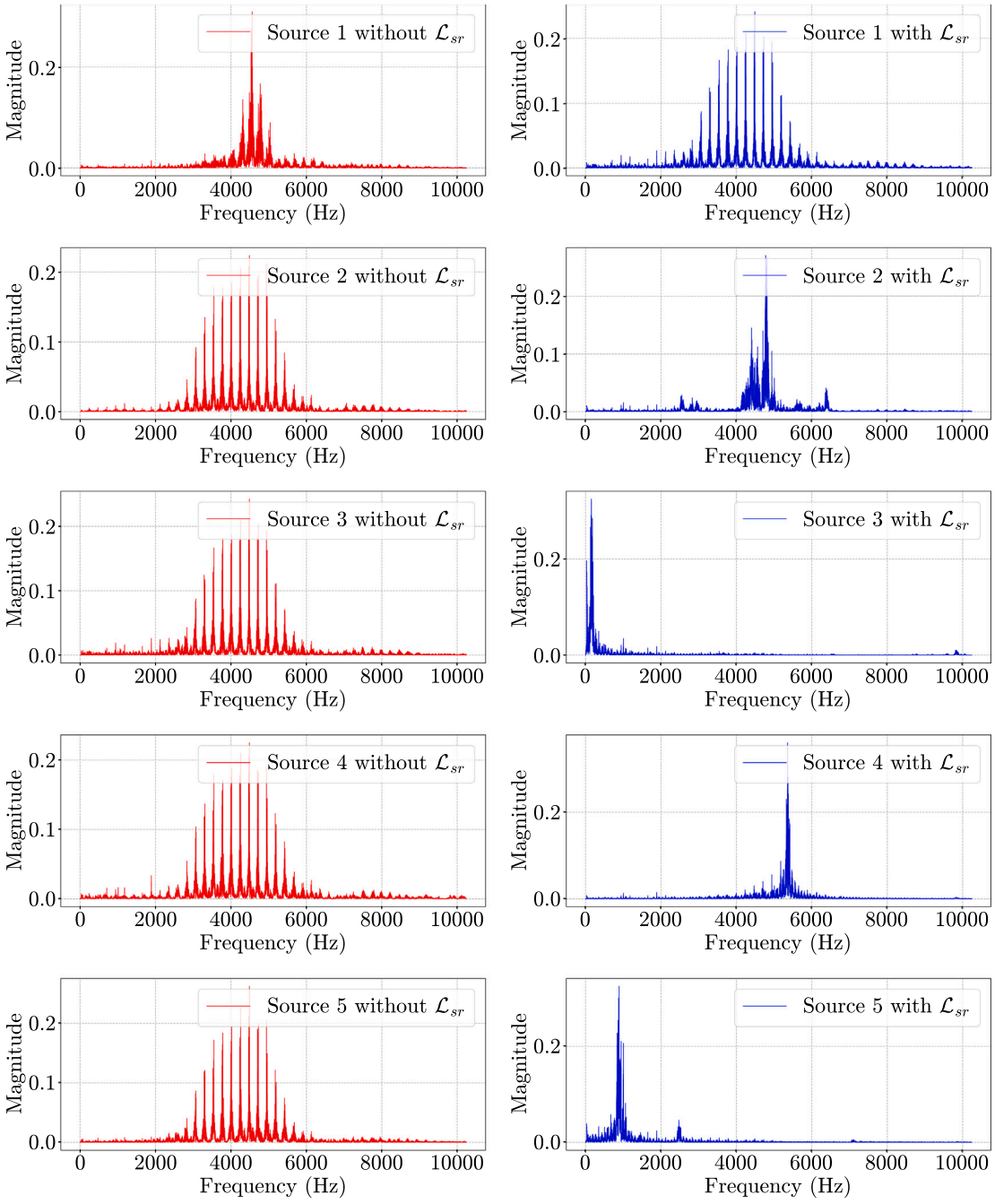


Fig. 6. The spectra of the latent source signals of record 700 from the second IMS dataset without (left) and with (right) the addition of the proposed spectral regularisation term. Several latent components display nearly identical spectral signatures without regularisation, indicating source duplication. Source duplication is discussed in Section 1. In contrast, coupling the negentropy-based LVM objective with spectral regularisation effectively prevents duplication, resulting in distinct latent components on the right. Note that the latent sources are sorted by kurtosis, a measure of non-Gaussianity related to the objective in Eq. (9), to highlight the components that deviate most from Gaussianity.

References

[1] Blei DM. Build, compute, critique, repeat: Data analysis with latent variable models. Annu Rev Stat Appl 2014;1(1):203–32. http://dx.doi.org/10.1146/annurev-statistics-022513-115657, URL https://www.annualreviews.org/doi/10.1146/annurev-statistics-022513-115657.

[2] Blei DM, Kucukelbir A, McAuliffe JD. Variational inference: A review for statisticians. J Amer Statist Assoc 2017;112(518):859–77. http://dx.doi.org/10.1080/01621459.2017.1285773, arXiv:arXiv:1601.00670v9, URL https://www.tandfonline.com/doi/full/10.1080/01621459.2017.1285773.

[3] Pearson K. LIII. On lines and planes of closest fit to systems of points in space. Lond Edinb Dublin Philos Mag J Sci 1901;2(11):559–72. http://dx.doi.org/10.1080/14786440109462720, URL https://www.tandfonline.com/doi/full/10.1080/14786440109462720.

[4] Hotelling H. Analysis of a complex of statistical variables into principal components. J Educ Psychol 1933;24(6):417–41. http://dx.doi.org/10.1037/h0071325, URL http://doi.apa.org/getdoi.cfm?doi=10.1037/h0071325.

[5] Bishop C. Pattern recognition and machine learning, 1st ed.. NY: Springer New York; 2006, p. 738, URL https://www.microsoft.com/en-us/research/publication/pattern-recognition-machine-learning/.

[6] Tipping ME, Bishop CM. Probabilistic principal component analysis. J R Stat Soc Ser B Stat Methodol 1999;61(3):611–22. http://dx.doi.org/10.1111/1467-9868.00196, URL https://onlinelibrary.wiley.com/doi/10.1111/1467-9868.00196.

[7] Jutten C, Herault J. Blind separation of sources, part I: An adaptive algorithm based on neuromimetic architecture. Signal Process 1991;24(1):1–10. http://dx.doi.org/10.1016/0165-1684(91)90079-X, URL https://linkinghub.elsevier.com/retrieve/pii/016516849190079X.

[8] Comon P. Independent component analysis, A new concept? Signal Process 1994;36(3):287–314. http://dx.doi.org/10.1016/0165-1684(94)90029-9, URL https://linkinghub.elsevier.com/retrieve/pii/0165168494900299.

[9] Bell AJ, Sejnowski TJ. An information-maximization approach to blind separation and blind deconvolution. Neural Comput 1995;7(6):1129–59. http://dx.doi.org/10.1162/neco.1995.7.6.1129, URL https://direct.mit.edu/neco/article/7/6/1129-1159/5909.

[10] Pham DT, Garat P. Blind separation of mixture of independent sources through a quasi-maximum likelihood approach. IEEE Trans Signal Process 1997;45(7):1712–25. http://dx.doi.org/10.1109/78.599941, URL http://ieeexplore.ieee.org/document/599941/.

[11] Hyvärinen A, Oja E. Independent component analysis: Algorithms and applications. Neural Netw 2000;13(4–5):411–30. http://dx.doi.org/10.1016/S0893-6080(00)00026-5, URL https://linkinghub.elsevier.com/retrieve/pii/S0893608000000265.

[12] Hyvärinen A, Karhunen J, Oja E. Independent component analysis. Adaptive and learning systems for signal processing, communications, and control, New York, USA: John Wiley & Sons, Inc.; 2001, http://dx.doi.org/10.1002/0471221317, URL http://doi.wiley.com/10.1002/0471221317.

[13] Balshaw R, Heyns PS, Wilke DN, Schmidt S. Importance of temporal preserving latent analysis for latent variable models in fault diagnostics of rotating machinery. Mech Syst Signal Process 2022;168(November 2021):108663. http://dx.doi.org/10.1016/j.ymssp.2021.108663, URL https://linkinghub.elsevier.com/retrieve/pii/S0888327021009870.

[14] Haykin S. Adaptive filter theory, 4th ed.. Upper Saddle River, N.J.: Prentice Hall; 2002, p. 920.

[15] Miao Y, Zhang B, Lin J, Zhao M, Liu H, Liu Z, et al. A review on the application of blind deconvolution in machinery fault diagnosis. Mech Syst Signal Process 2022;163:108202. http://dx.doi.org/10.1016/j.ymssp.2021.108202, URL https://www.sciencedirect.com/science/article/pii/S088832702100577X.

[16] Casey M, Westner A. Separation of mixed audio sources by independent subspace analysis. 2000, URL https://www.semanticscholar.org/paper/Separation-of-Mixed-Audio-Sources-By-Independent-Casey-Westner/f552b5991d14db64fe3e381bf25a90524f4d16d6.

[17] Davies M, James C. Source separation using single channel ICA. Signal Process 2007;87(8):1819–32. http://dx.doi.org/10.1016/j.sigpro.2007.01.011, URL https://linkinghub.elsevier.com/retrieve/pii/S0165168407000151.

[18] Meurer A, Smith CP, Paprocki M, Čertík O, Kirpichev SB, Rocklin M, et al. SymPy: Symbolic computing in Python. PeerJ Comput Sci 2017;3(1):e103. http://dx.doi.org/10.7717/peerj-cs.103, URL https://peerj.com/articles/cs-103.

[19] Harris CR, Millman KJ, van der Walt SJ, Gommers R, Virtanen P, Cournapeau D, et al. Array programming with NumPy. Nature 2020;585(7825):357–62. http://dx.doi.org/10.1038/s41586-020-2649-2, arXiv:2006.10256, URL https://www.nature.com/articles/s41586-020-2649-2.

[20] Vito ED, Rosasco L, Verri A. Spectral methods for regularization in learning theory. 2006.

[21] Tikhonov AN, Arsenin VI. Solutions of ill-posed problems. Winston; 1977, URL https://api.semanticscholar.org/CorpusID:122072756.

[22] Golub GH, Loan CFV. Matrix computations. In: Johns hopkins studies in the mathematical sciences, Johns Hopkins University Press; 2013, p. 694. http://dx.doi.org/10.56021/9781421407944, URL https://doi.org/10.56021/9781421407944.

[23] Gerfon L, Rosasco L, Odone F, De Vito E, Verri A. Spectral algorithms for supervised learning. Neural Comput 2008;20(7):1873–97. http://dx.doi.org/10.1162/neco.2008.05-07-517.

[24] Miyato T, Kataoka T, Koyama M, Yoshida Y. Spectral normalization for generative adversarial networks. In: 6th international conference on learning representations, ICLR 2018 - Conference track proceedings. 2018, URL http://arxiv.org/abs/1802.05957.

[25] Xie W, Lei J, Liu B, Li Y, Jia X. Spectral constraint adversarial autoencoders approach to feature representation in hyperspectral anomaly detection. Neural Netw 2019;119:222–34. http://dx.doi.org/10.1016/j.neunet.2019.08.012, https://linkinghub.elsevier.com/retrieve/pii/S0893608019302291.

[26] Snyman JA, Wilke DN. Practical mathematical optimization. In: Springer optimization and its applications. Springer optimization and its applications, 2nd ed.. Vol. 133, Cham: Springer International Publishing; 2018, p. 1–364. http://dx.doi.org/10.1007/978-3-319-77586-9, URL http://link.springer.com/10.1007/978-3-319-77586-9.

[27] Hyvärinen A. New approximations of differential entropy for independent component analysis and projection pursuit. Adv Neural Inf Process Syst 1997;(1):273–9, URL https://api.semanticscholar.org/CorpusID:11359216.

[28] Burden RL, Faires JD, Burden AM. Numerical analysis. 10th ed.. Boston, MA: Cengage Learning; 2016, p. 896.

[29] Balshaw R, Heyns PS, Wilke DN, Schmidt S. Latent indicators for temporal-preserving latent variable models in vibration-based condition monitoring under non-stationary conditions. Mech Syst Signal Process 2023;199(May):110446. http://dx.doi.org/10.1016/j.ymssp.2023.110446, https://linkinghub.elsevier.com/retrieve/pii/S0888327023003540.

[30] Buitinck L, Louppe G, Blondel M, Pedregosa F, Mueller A, Grisel O, et al. API design for machine learning software: experiences from the scikit-learn project. 2013, p. 1–15, arXiv:1309.0238, URL http://arxiv.org/abs/1309.0238.

[31] Python package index - PyPI. In: Python software foundation. URL https://pypi.org/.

[32] Hunter JD. Matplotlib: A 2D graphics environment. Comput Sci Eng 2007;9(3):90–5. http://dx.doi.org/10.1109/MCSE.2007.55, URL http://ieeexplore.ieee.org/document/4160265/.

[33] Pedregosa F, Varoquaux G, Gramfort A, Michel V, Thirion B, Grisel O, et al. Scikit-learn: Machine learning in Python. J Mach Learn Res 2011;12(85):2825–30, URL http://jmlr.org/papers/v12/pedregosa11a.html.

[34] da Costa-Luis C, Larroque SK, Altendorf K, Mary H, Richardsheridan, Korobov M, et al. tqdm: A fast, extensible progress bar for Python and CLI. 2023, http://dx.doi.org/10.5281/zenodo.7697295.

[35] Antoni J. The spectral kurtosis: A useful tool for characterising non-stationary signals. Mech Syst Signal Process 2006;20(2):282–307. http://dx.doi.org/10.1016/j.ymssp.2004.09.001.

[36] Qiu H, Lee J, Lin J, Yu G, Rexford Technical Services (2007). IMS, University of Cincinnati. Bearing data set, NASA ames prognostics data repository. 2007, URL http://ti.arc.nasa.gov/project/prognostic-data-repository.

[37] Gousseau W, Antoni J, Girardin F, Griffaton J. Analysis of the rolling element bearing data set of the center for intelligent maintenance systems of the University of Cincinnati. In: 13th international conference on condition monitoring and machinery failure prevention technologies, CM 2016/MFPT 2016. 2016, URL https://hal.science/hal-01715193/file/216-Gousseau.pdf.