Control Engineering Practice 147 (2024) 105923

Contents lists available at ScienceDirect

Control Engineering Practice

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

Local regularization assisted split augmented Lagrangian shrinkage algorithm for feature selection in condition monitoring

Yufei Gui [a], Xiaoquan Tang [b], Zepeng Liu [c,*]

[a] Department of Automatic Control and Systems Engineering, University of Sheffield, Sheffield, S1 3JD, UK
[b] School of Data Science and Shenzhen Research Institute of Big Data, The Chinese University of Hong Kong, Shenzhen, 518172, China
[c] School of Engineering, Newcastle University, NE1 7RU, UK

ARTICLE INFO

Keywords:
Condition monitoring
Feature selection
Local regularization
Split augmented Lagrangian shrinkage algorithm
Sparse representation
Bayesian evidence framework

ABSTRACT

Feature selection plays a vital role in improving the efficiency and accuracy of condition monitoring by constructing sparse but effective models. In this study, an advanced feature selection algorithm named the local regularization assisted split augmented Lagrangian shrinkage algorithm (LR-SALSA) is proposed. The feature selection is realized by solving a l_1-norm optimization problem, which usually selects more sparse and representative features at less computational costs. The proposed algorithm operates in two stages, namely variable selection and coefficient estimation. In the stage of variable selection, the primal problem is converted into three subproblems which can be solved separately. Then individual penalty parameters are applied to every coefficient of the model when dealing with the first subproblem. Under the Bayesian evidence framework, an iterative algorithm is derived to optimize these hyperparameters. During the optimization process, redundant variables will be pruned to guarantee model sparsity and improve computational efficiency at the same time. In the second stage, the coefficients for the selected model terms are determined using the least squares technique. The superior performance and efficiency of the proposed LR-SALSA method are validated through two numerical examples and a real-world cutting tool wear prediction case study. Compared with the existing methods, the proposed method can generate a sparse model and ensure a good trade-off between estimation accuracy and computational efficiency.

1. Introduction

Data-driven condition monitoring techniques have been extensively applied in complex industrial systems to ensure process safety and economic efficiency (Kong, Dyer, Payne, Hamerton, & Weaver, 2023; Zhu, Li, & Zhang, 2020). The conditions of a system are usually evaluated by analyzing a large number of process variables (Yang, Karimi and Pawelczyk, 2023). When the dimension of available data or features becomes extremely high, a critical issue known as the curse of dimensionality arises, leading to overfitting and huge computational costs (Yang, Wang et al., 2023). Besides, it is common that multiple sensors measure similar aspects of the process, bringing about the problem of multicollinearity. The estimation of model coefficients could be very unstable, making the model less robust to noise in the data. To build an accurate condition monitoring model, it is essential to develop an effective and efficient feature selection method to select a subset of important features and address the challenges mentioned above.

Feature selection has shown necessity and benefits in a wide range of areas including machine learning, pattern recognition, multimedia, and condition monitoring (Ardakani et al., 2016; Yang, Ma, Hauptmann, & Sebe, 2013). Its main objective is to select the most important features from all candidates such that a simpler and more comprehensible model is built and the data-mining performance is improved (Zhang, Cao, Sun, & Su, 2023). In literature, feature selection methods can be broadly categorized as filter, wrapper, and embedded methods (Li et al., 2018). Filter methods are independent of the subsequent learning stage. These methods assess the importance of features relying on some criteria such as feature similarity (Zhao, Wang, Liu, & Ye, 2013), mutual information (Lucke, Mei, Stief, Chioua, & Thornhill, 2019), and relative contribution (Mei, Yuan, Cui, Dong, & Zhao, 2022). For example, in Zhu, Hong, and Wong (2008), the Fisher's discriminant ratio was used to score the features' discriminability of cutting tool wear phases in advanced machining. In Tang, Bo, Liu, Sun and Wei (2018), the affinity propagation clustering and RReliefF algorithms were combined to build an unsupervised feature selection method, which selected significant features related to the health status of rolling bearings. Although filter methods are computationally efficient, the

* Corresponding author.
E-mail address: zepeng.liu@newcastle.ac.uk (Z. Liu).

https://doi.org/10.1016/j.conengprac.2024.105923
Received 8 December 2023; Received in revised form 16 March 2024; Accepted 18 March 2024
Available online 22 March 2024

selected features may not be optimal for the target learning algorithms due to the lack of guidance from a specific learning algorithm. Wrapper methods evaluate the quality of features based on the predictive performance of a predefined learning algorithm. The feature subset that yields the best predictive performance is used as the selected ones. However, the size of the search space increases exponentially as the feature candidates increase (Zhang, Nie, Li, & Wei, 2019). The existing searching strategies such as genetic algorithm and particle swarm optimization cannot fundamentally overcome this challenge (Dameshghi & Refan, 2019; Tang, Chai, Yu, & Zhao, 2012). As a result, wrapper methods are seldom used in practice. Embedded methods are a trade-off between filter and wrapper methods, which embed the feature selection into model learning. Firstly, the introduction of learning algorithms improves the performance of selected features. Besides, compared with wrapper methods, they are far more efficient as the repeated feature evaluation procedure is no longer needed.

The regularization technique is a representative embedded feature selection method that aims to build a sparse model by minimizing the fitting errors and forcing model coefficients to be small enough at the same time (Li et al., 2018). To achieve this, the l_p-norm penalty term is added on a machine learning model, where $0 \leq p \leq 1$. When $p = 0$, the l_0-norm penalizes the number of non-zero entries of the coefficient vector. Such an integer programming problem is usually difficult to solve. Therefore, the l_1-norm regularization is usually used as a relaxation of the l_0-norm. In this case, many feature coefficients become smaller but not exactly zero. The l_1-norm regularized feature selection method has attracted lots of attention in recent years (Dai, 2023). Furthermore, to cope with feature selection requirements in multi-class classification or multivariate regression problems, the regularization method is improved by adding the $l_{p,q}$-norm penalty term, where $p > 1$ and $0 \leq p \leq 1$. Different settings of p and q provide fine adjustments of the regularization effect (Li, Nie, Bian, Wu and Li, 2021; Quattoni, Carreras, Collins, & Darrell, 2009). Researchers have also proposed some feature selection methods for specific models, for example, the orthogonal forward selection (OFR) method is usually exploited in linear regression problems (Chen, Billings, & Luo, 1989) and the automatic relevance determination kernel functions are often used in feature selection of Bayesian regression problems (Li, Chen, Zhao and Wang, 2021).

As mentioned above, the l_1-norm optimization method is an attractive topic in feature selection and sparse model estimation. These methods aim to minimize not only the prediction error but also the sum of the absolute values of the model coefficients (Tang, Zhang, & Wang, 2019). One of the most popular l_1-norm optimization algorithms is the least absolute shrinkage and selection operator (LASSO), which can be used to prune redundant features for condition monitoring. However, extensive research shows that LASSO often produces an insufficiently sparse model leading prediction results to be inaccurate (Tang et al., 2019). To achieve a more sparse and accurate solution, the split augmented Lagrangian shrinkage algorithm (SALSA) is proposed. SALSA converts the l_1-norm optimization problem into several subproblems, which can be addressed separately without the need for a third-party solver (Afonso, Bioucas-Dias, & Figueiredo, 2010). Based on SALSA, recently, a new approach called sparse augmented Lagrangian (SAL) has been developed to build more compact models (Tang et al., 2019). SAL combines the random subsampling technique with the SALSA method to produce intermediate models. Then, the highly selected terms from these intermediate models are added to the final model. Finally, SALSA is applied again to estimate the model coefficients. This leads to a better performance in model sparsity compared with LASSO and SALSA. In Qin, Huang, Wang, Tang, and Liu (2022), the SAL method was used to select the significant features for tool wear prediction in machining. However, SAL suffers from low computational efficiency due to the application of the random subsampling technique, which involves conducting numerous repetitions of the SALSA algorithm. In Pan, Yuan, Gonçalves, and Stan (2016), the authors proposed a re-weighted l_1-minimization algorithm under the sparse Bayesian learning (SBL) framework. Nevertheless, the solution has to be calculated using a third-party solver. To cope with this limitation, the Bayesian augmented Lagrangian (BAL) algorithm is developed (Tang, Zhang, & Li, 2018), which solves the l_1-minimization problem in SBL using the SALSA method. Even though the computation efficiency has been improved, BAL still takes much longer time than LASSO and SALSA algorithms. From the aforementioned, it is still an open question of developing a l_1-norm optimization-based feature selection method with high sparsity, fast computation speed, and high model accuracy.

To fundamentally address the challenges, this paper proposes a novel algorithm referred to as local regularization assisted SALSA (LR-SALSA). The key idea is to embed a local regularization strategy into the framework of SALSA. Firstly, the l_1-minimization problem is converted to three subproblems that can be solved separately. The first subproblem is a ridge regression problem, which does not involve the pruning of redundant terms originally. In LR-SALSA, the local regularization strategy is employed to solve the first subproblem, which means that each candidate term has an associated individual regularization parameter. During the optimization process, many of these penalty parameters are driven to large values, resulting in successive removal of corresponding terms from the pool of candidate terms. This strategy improves the computational efficiency and enhances the sparsity of the final model by applying SALSA only once. After determining the important terms, the model coefficients are re-estimated using the least squares method. As a result, the main contributions of this study are as follows,

1. The LR-SALSA method is proposed under the Bayesian evidence framework, introducing an individual penalty parameter for every coefficient of the model. An iterative algorithm is derived to optimize these hyperparameters relying on the dataset only based on the Bayesian evidence framework (MacKay, 1992). This optimization process shows a quicker convergence in practice.
2. The optimization of local regularization hyperparameters is embedded into the iterative algorithm of SALSA. At each step, according to the values of the penalty parameters, redundant variables can be pruned in time, which significantly reduces the overall computing complexity. As a result, LR-SALSA exhibits better model sparsity and higher computational efficiency when compared to existing l_1-norm optimization algorithms, such as SAL and BAL.

To validate the performance of LR-SALSA, both simulation and experimental studies were conducted. The results of these studies show that LR-SALSA successfully produces a more compact model while maintaining a high level of prediction accuracy and fast computation speed. This demonstrates the effectiveness and practicality of the proposed feature selection method in real-world condition monitoring applications.

2. Preliminary

2.1. Problem description

Consider the following l_1-norm optimization problem

$$\min_{\boldsymbol{\theta}\in\mathbb{R}^M} \quad f_1(\boldsymbol{\theta}) + f_2(\boldsymbol{\theta}) \tag{1}$$

with $f_1(\boldsymbol{\theta}) = \frac{1}{2}\|\boldsymbol{y} - \mathbf{P}\boldsymbol{\theta}\|_2^2$ and $f_2(\boldsymbol{\theta}) = \lambda\|\boldsymbol{\theta}\|_1$, where $\boldsymbol{y} \in \mathbb{R}^N, \mathbf{P} \in \mathbb{R}^{N\times M}$ represent the output and regression matrix composed of multivariate inputs, respectively. λ denotes the l_1 regularization parameter. To convert the original problem to a constrained problem, the coefficient vector $\boldsymbol{\theta}$ in function f_2 is replaced with a new variable $\boldsymbol{\nu}$, which gives

$$\begin{aligned} \min_{\boldsymbol{\theta},\boldsymbol{\nu}\in\mathbb{R}^M} \quad & f_1(\boldsymbol{\theta}) + f_2(\boldsymbol{\nu}) \\ \text{s.t.} \quad & \boldsymbol{\theta} = \boldsymbol{\nu} \end{aligned} \tag{2}$$

In order to find the optimal solution to this problem, the Lagrangian function can be defined as

$$L(\boldsymbol{\theta}, \boldsymbol{\nu}, \boldsymbol{\varphi}) = f_1(\boldsymbol{\theta}) + f_2(\boldsymbol{\nu}) + \boldsymbol{\varphi}^T(\boldsymbol{\theta} - \boldsymbol{\nu}) \tag{3}$$

where $\boldsymbol{\varphi} \in \mathbb{R}^M$ is the Lagrangian multiplier. The solution to Eq. (2) is always a saddle point of $L(\boldsymbol{\theta}, \boldsymbol{\nu}, \boldsymbol{\varphi})$ in (3).

Typically, the dual ascent algorithm can be used to optimize the primal and dual problems, respectively. However, this algorithm may not always converge, especially when applied to non-convex or non-smooth optimization problems. So the Augmented Lagrangian method (ALM) is introduced by incorporating a penalty term

$$L_\mu(\boldsymbol{\theta}, \boldsymbol{\nu}, \boldsymbol{\varphi}) = f_1(\boldsymbol{\theta}) + f_2(\boldsymbol{\nu}) + \boldsymbol{\varphi}^T(\boldsymbol{\theta} - \boldsymbol{\nu}) + \frac{\mu}{2}\|\boldsymbol{\theta} - \boldsymbol{\nu}\|_2^2 \tag{4}$$

where $\mu \geq 0$ denotes the penalty parameter. The penalty term helps to enforce the constraints, leading to a more stable and efficient algorithm. Compared with the dual ascent method, the convergence is guaranteed without assumptions like strict convexity of objective functions.

Then, considering sometimes the objective function is separable, the alternating direction method of multipliers (ADMM) algorithm (Boyd, 2010) is proposed. The primal variable optimization step can be further split into subproblems that can be solved in parallel. Let $\boldsymbol{d} = \boldsymbol{\varphi}/\mu$, with ADMM algorithm, problem (4) is solved by optimizing the following subproblems,

$$\hat{\boldsymbol{\theta}}_{k+1} = \arg\min_{\boldsymbol{\theta}} f_1(\boldsymbol{\theta}) + \frac{\mu}{2}\|\boldsymbol{\theta} - \boldsymbol{\nu}_k + \boldsymbol{d}_k\|_2^2 \tag{5}$$

$$\boldsymbol{\nu}_{k+1} = \arg\min_{\boldsymbol{\nu}} f_2(\boldsymbol{\nu}) + \frac{\mu}{2}\|\hat{\boldsymbol{\theta}}_{k+1} - \boldsymbol{\nu} + \boldsymbol{d}_k\|_2^2 \tag{6}$$

$$\boldsymbol{d}_{k+1} = \boldsymbol{d}_k + (\hat{\boldsymbol{\theta}}_{k+1} - \boldsymbol{\nu}_{k+1}) \tag{7}$$

2.2. SALSA method

The SALSA algorithm can be regarded as an application of the ADMM algorithm to the l_1-norm regularized linear regression problem (1). It converts the original problem into several subproblems as shown in (5)–(7). Then these problems are addressed separately as shown in Algorithm 1. To elaborate further:

- For (5), $\hat{\boldsymbol{\theta}}_{k+1}$ can be calculated by taking the derivative of the cost function with respect to $\boldsymbol{\theta}$ to be zero.
- Concerning (6), the update for $\boldsymbol{\nu}_{k+1}$ incorporates the utilization of the soft thresholding operator (Pan, Sootla, & Stan, 2014) $S_{\lambda/\mu}(z) = \max(0, z - \lambda/\mu) - \max(0, -z - \lambda/\mu)$, where $z = \hat{\boldsymbol{\theta}}_{k+1} + \boldsymbol{d}_k$.
- At each iteration, the sum-of-squares error is evaluated through $e_{k+1} = (\boldsymbol{y} - \mathbf{P}\hat{\boldsymbol{\theta}}_{k+1})^T(\boldsymbol{y} - \mathbf{P}\hat{\boldsymbol{\theta}}_{k+1})$. The iteration stops when the sequence of errors converges.

Algorithm 1 SALSA

Require: Predefine μ, λ and δ_e.
Ensure: $\boldsymbol{\nu_0} = \boldsymbol{d_0} = \mathbf{0}$
repeat
  $\hat{\boldsymbol{\theta}}_{k+1} = (\mathbf{P}^T\mathbf{P} + \mu\mathbf{I})^{-1}(\mathbf{P}^T\boldsymbol{y} + \mu(\boldsymbol{\nu}_k - \boldsymbol{d}_k))$
  $\boldsymbol{\nu_{k+1}} = \max(0, \hat{\boldsymbol{\theta}}_{k+1} + \boldsymbol{d}_k - \lambda/\mu) - \max(0, -(\hat{\boldsymbol{\theta}}_{k+1} + \boldsymbol{d}_k) - \lambda/\mu)$
  $\boldsymbol{d}_{k+1} = \boldsymbol{d}_k + (\hat{\boldsymbol{\theta}}_{k+1} - \boldsymbol{\nu}_{k+1})$
  $e_{k+1} = (\boldsymbol{y} - \mathbf{P}\hat{\boldsymbol{\theta}}_{k+1})^T(\boldsymbol{y} - \mathbf{P}\hat{\boldsymbol{\theta}}_{k+1})$
  $k \leftarrow k + 1$
until stopping criterion $|e_{k+1} - e_k| \leq \delta_e$ is satisfied

2.3. SAL method

The SAL method, inspired by SALSA, aims to construct a parsimonious model. This approach encompasses two main stages: variable selection and parameter estimation.

- During the variable selection stage, the random subsampling technique is used to randomly generate a variety of datasets, each comprising system output $\widetilde{\boldsymbol{y}}_j$ and input $\widetilde{\mathbf{P}}_j$. For each subsampled dataset, SALSA executes the procedure outlined in Algorithm 1, resulting in an intermediate model. The terms that are present frequently in all intermediate models will be selected as the dominant terms.
- Subsequently, in the parameter estimation stage, SALSA will be run again, but on a subsampled dataset that only contains the previously selected primary terms. This step determines the coefficients of the final model.

Within this algorithm, both pruning and stability selection play vital roles in ensuring model sparsity. While the pruning process removes terms with minimal impact on model performance based on the subsampling dataset, stability selection discards redundant terms. The comprehensive procedure of the SAL method is presented in Algorithm 2.

Algorithm 2 SAL

Variable selection stage:

Require: Predefine μ, λ, δ_θ and δ_{thr}
Ensure: $\boldsymbol{\nu_0} = \boldsymbol{d_0} = \mathbf{0}$
for $j = 1$ to n: **do**
  Random subsampling $\rightarrow \widetilde{\boldsymbol{y}}_j$ and $\widetilde{\mathbf{P}}_j$
  Run SALSA and stop when $sign(\hat{\boldsymbol{\theta}}_{k+1}) = sign(\hat{\boldsymbol{\theta}}_k)$
  if $\hat{\theta}_i^2/\|\hat{\boldsymbol{\theta}}\|_2^2 \leq \delta_\theta$ **then**
    Term $\boldsymbol{p}_i$ is removed
  end if
  Record the selected terms of the intermediate model
end for
if $sp(\boldsymbol{p}_i)/n \geq \delta_{thr}$ where $sp(\boldsymbol{p}_i)$ is the occurrences of $\boldsymbol{p}_i$, **then**
  Term $\boldsymbol{p}_i$ is selected
end if
Choose the highly selected terms as the final model

Parameter estimation stage:

Random subsampling $\rightarrow \widetilde{\boldsymbol{y}}$ and $\widetilde{\mathbf{P}}$
Run SALSA and stop when $sign(\hat{\boldsymbol{\theta}}_{k+1}) = sign(\hat{\boldsymbol{\theta}}_k)$
Determine the coefficients in the final model

3. Local regularization assisted SALSA

3.1. Solving the subproblems

The original l_1-norm optimization problem (1), is converted to subproblems (5)–(7) using the ADMM framework. In solving subproblem (5), a local regularization strategy is employed to penalize each element of $\boldsymbol{\theta}$ for the purpose of producing a more compact model. Let $\boldsymbol{e} = \boldsymbol{y} - \mathbf{P}\boldsymbol{\theta}$, $\boldsymbol{\eta} = \boldsymbol{\nu} - \boldsymbol{d}$, the new cost function is written as

$$J_R(\boldsymbol{\theta}) = \boldsymbol{e}^T\boldsymbol{e} + \sum_{i=1}^{M} \mu_i \left(\theta_i - \eta_i\right)^2 \tag{8}$$

where $\mu_i \geq 0$ is the penalty parameter of element θ_i in $\boldsymbol{\theta}$. Compared with the 'global' regularization method, where a uniform intensity of penalty is applied to all elements in $\boldsymbol{\theta}$ in SALSA and SAL, this novel design of 'local' regularization assisted SALSA allows independent and flexible control of the penalty intensity of every coefficient in the optimization problem. Thus the function (8) consists of the sum-of-squares error function and a so-called locally quadratic regularization term.

The Bayesian evidence approximation can be used to optimize the regularization parameters μ_i automatically. From the Bayesian viewpoint, a regularization parameter is equivalent to the ratio of the

coefficient precision and the noise precision (MacKay, 1992). In this sense, $J_R(\boldsymbol{\theta})$ is equivalent to the following cost function,

$$J_B(\boldsymbol{\theta}) = \beta \boldsymbol{e}^T \boldsymbol{e} + \sum_{i=1}^{M} \alpha_i \left(\theta_i - \eta_i\right)^2 = \beta \boldsymbol{e}^T \boldsymbol{e} + (\boldsymbol{\theta} - \boldsymbol{\eta})^T \mathbf{A} (\boldsymbol{\theta} - \boldsymbol{\eta}) \tag{9}$$

where β and α_i govern the precision parameters of noise and θ_i, and $\mathbf{A} = \mathrm{diag}\{\alpha_1, \alpha_2, \ldots, \alpha_M\}$ denotes the precision matrix. The regularization parameter in (8) holds the relationship with the hyperparameters as $\mu_i = \alpha_i / \beta$.

Using the evidence approximation framework (Tipping, 2001), the unknown parameters $\boldsymbol{\alpha}$ and β are determined by maximizing the log marginal likelihood function

$$\begin{aligned} \ln p(\boldsymbol{y}|\mathbf{P}, \boldsymbol{\eta}, \boldsymbol{\alpha}, \beta) = & \frac{N}{2} \ln \frac{\beta}{2\pi} + \frac{1}{2} \sum_{i=1}^{M} \ln \alpha_i - \frac{1}{2} \beta \boldsymbol{y}^T \boldsymbol{y} \\ & - \frac{1}{2} \boldsymbol{\eta}^T \mathbf{A} \boldsymbol{\eta} + \frac{1}{2} \boldsymbol{m}^T \mathbf{S}^{-1} \boldsymbol{m} + \frac{1}{2} \ln |\mathbf{S}| \end{aligned} \tag{10}$$

where $\boldsymbol{m}$ and $\mathbf{S}$ define the mean and covariance matrix of the posterior distribution for $\boldsymbol{\theta}$, and are given by

$$\boldsymbol{m} = \mathbf{S}(\mathbf{A}\boldsymbol{\eta} + \beta \mathbf{P}^T \boldsymbol{y}) \tag{11}$$

$$\mathbf{S} = (\mathbf{A} + \beta \mathbf{P}^T \mathbf{P})^{-1} \tag{12}$$

Taking the derivatives of $\ln p(\boldsymbol{y}|\mathbf{P}, \boldsymbol{\eta}, \boldsymbol{\alpha}, \beta)$ with respect to $\boldsymbol{\alpha}$ and β to be zero yields the updating formulas for $\boldsymbol{\alpha}$ and β, respectively.

$$\alpha_i^{\mathrm{new}} = \frac{\gamma_i}{(m_i - \eta_i)^2} \tag{13}$$

$$(\beta^{\mathrm{new}})^{-1} = \frac{\|\boldsymbol{y} - \mathbf{P}\boldsymbol{m}\|_2^2}{N - \sum_{i=1}^{M} \gamma_i} \tag{14}$$

where $\gamma_i = 1 - \alpha_i S_{ii}$, S_{ii} is the ith diagonal component of $\mathbf{S}$. The detailed derivation of (13) and (14) is presented in Appendices A–D.

The optimization begins by setting initial values for $\boldsymbol{\alpha}$ and β. Using these initial values, the posterior mean and covariance are computed according to (11) and (12), respectively. Subsequently, the hyperparameters are re-estimated using (13) and (14); and re-estimating the posterior mean and covariance, using (11) and (12), until a suitable convergence criterion is satisfied. It is worth mentioning that the optimization results in a proportion of α_i that are close to large values, and so the coefficients θ_i corresponding to these hyperparameters have posterior distributions with a mean of η_i and variance of zero. Thus those variables are removed from the model and a sparse structure is found.

Giving values $\boldsymbol{\alpha}_k^*$ and β_k^* for the hyperparameters that maximize the marginal likelihood and solutions of $\boldsymbol{\nu}_k$ and $\boldsymbol{d}_k$, at the $(k+1)$th iteration, $\boldsymbol{\alpha}_k^*$ and β_k^* are set as the initial values, and $\boldsymbol{\eta}_k = \boldsymbol{\nu}_k - \boldsymbol{d}_k$. Running the optimization procedure yields optimal values $\boldsymbol{\alpha}_{k+1}^*$ and β_{k+1}^*. At the same time, $\hat{\boldsymbol{\theta}}_{k+1}$ and $\boldsymbol{\mu}_{k+1}$ are updated

$$\hat{\boldsymbol{\theta}}_{k+1} = (\mathbf{A}_{k+1}^* + \beta_{k+1}^* \mathbf{P}^T \mathbf{P})^{-1} (\mathbf{A}_{k+1}^* \boldsymbol{\eta} + \beta_{k+1}^* \mathbf{P}^T \boldsymbol{y}) \tag{15}$$

$$\mu_{i,k+1} = \alpha_{i,k+1} / \beta_{k+1} \tag{16}$$

Similar to SALSA, the subproblem (6) is also solved using the soft thresholding operator $S_{\lambda/\mu}(z)$. Each element of the variable $\boldsymbol{\nu}_{k+1}$ is updated as

$$\begin{aligned} \nu_{i,k+1} = & \max(0, \hat{\theta}_{i,k+1} + d_{i,k} - \lambda/\mu_{i,k+1}) \\ & - \max(0, -\hat{\theta}_{i,k+1} - d_{i,k} - \lambda/\mu_{i,k+1}) \end{aligned} \tag{17}$$

With $\hat{\boldsymbol{\theta}}_{k+1}$ and $\boldsymbol{\nu}_{k+1}$, $\boldsymbol{d}_{k+1}$ can be calculated by

$$\boldsymbol{d}_{k+1} = \boldsymbol{d}_k + (\hat{\boldsymbol{\theta}}_{k+1} - \boldsymbol{\nu}_{k+1}) \tag{18}$$

In terms of the stopping criterion, the sum-of-squares error is computed by $e_{k+1} = (\boldsymbol{y} - \mathbf{P}\hat{\boldsymbol{\theta}}_{k+1})^T (\boldsymbol{y} - \mathbf{P}\hat{\boldsymbol{\theta}}_{k+1})$ and the iteration stops when $|e_{k+1} - e_k| \le \delta_e$, where δ_e denotes the convergence tolerance.

3.2. Variable selection

The pruning of variables is carried out at each iterative step. Initially, $\boldsymbol{\eta}$ equals zero, so the terms that have very small estimated coefficients, i.e., $\hat{\theta}_i \to 0$, will be removed. This is because corresponding hyperparameters α_i tend to become extremely large. At the $(k+1)$th iteration, if the estimated coefficients are close to $\boldsymbol{\eta}_k$, i.e., $\hat{\theta}_{i,k+1} \to \eta_{i,k}$, the corresponding terms will be removed. This operation can guarantee the convergence of optimizing hyperparameters $\boldsymbol{\alpha}_{k+1}$ and β_{k+1}. Besides, pruning helps to build a sparse model. Intuitively, the terms with coefficients that converge to $\boldsymbol{\eta}_k$ will not be selected. Substituting $\boldsymbol{\nu}_k$ in the expression of $\boldsymbol{\eta}_k$ with (17), the following equation holds

$$\eta_{i,k} = \begin{cases} \hat{\theta}_{i,k} + d_{i,k-1} - 2\lambda/\mu_{i,k}, & \hat{\theta}_{i,k} + d_{i,k-1} > \lambda/\mu_{i,k} \\ -\hat{\theta}_{i,k} - d_{i,k-1}, & |\hat{\theta}_{i,k} + d_{i,k-1}| \le \lambda/\mu_{i,k} \\ \hat{\theta}_{i,k} + d_{i,k-1} + 2\lambda/\mu_{i,k}, & \hat{\theta}_{i,k} + d_{i,k-1} < -\lambda/\mu_{i,k} \end{cases} \tag{19}$$

In principle, if $|\hat{\theta}_{i,k} + d_{i,k-1}|$ is too large relative to $\lambda/\mu_{i,k}$, $\eta_{i,k}$ tends to take a value that is close to $\hat{\theta}_{i,k}$. At the $(k+1)$th step, $\hat{\theta}_{i,k+1}$ remains unchanged as $\hat{\theta}_{i,k}$, driving $\alpha_{i,k+1}$ to a large value, which further reduces $\lambda/\mu_{i,k+1}$. As a result, the ith variable will be discarded. Thus the selected terms should be those meeting convergence requirements and also contributing significantly to the prediction accuracy.

Suppose that M' variables remain with estimated coefficients $\hat{\theta}_i, i = 1, 2, \ldots, M'$ when the iteration converges. Some coefficients might be very small compared with others, i.e., $\hat{\theta}_i^2 \ll \|\hat{\theta}\|_2^2$. In this case, those small weights lower than a threshold will be further pruned to obtain a compact model.

3.3. Coefficients estimation

The coefficients of the selected variables will be re-estimated. Let $\mathrm{ST} = \{st_1, \ldots, st_{M''}\}$ denotes the set of indices of selected terms. These terms form a new candidate matrix $\mathbf{P}_{\mathrm{ST}} = [\boldsymbol{p}_{st_1}, \ldots, \boldsymbol{p}_{st_{M''}}] \in \mathbb{R}^{N \times M''}$. Related coefficients are $\boldsymbol{\theta}_{\mathrm{ST}} = [\theta_{st_1}, \ldots, \theta_{st_{M''}}]^T \in \mathbb{R}^{M''}$. Now the regression model becomes $\boldsymbol{y} = \mathbf{P}_{\mathrm{ST}} \boldsymbol{\theta}_{\mathrm{ST}} + \boldsymbol{e}$, where $\boldsymbol{y}$ denotes system output and $\boldsymbol{e}$ denotes the residual. Using the ordinary least squares estimator, the model coefficients are given by

$$\hat{\boldsymbol{\theta}}_{\mathrm{ST}} = (\mathbf{P}_{\mathrm{ST}}^T \mathbf{P}_{\mathrm{ST}})^{-1} \mathbf{P}_{\mathrm{ST}}^T \boldsymbol{y} \tag{20}$$

To improve the generalizability of the final model, ridge regression needs to be utilized. The model parameter estimation becomes

$$\hat{\boldsymbol{\theta}}_{\mathrm{ST}} = (\mathbf{P}_{\mathrm{ST}}^T \mathbf{P}_{\mathrm{ST}} + \kappa \mathbf{I})^{-1} \mathbf{P}_{\mathrm{ST}}^T \boldsymbol{y} \tag{21}$$

where κ is the penalty parameter.

Note that the identifiability of the estimation algorithm is guaranteed by the variable selection step. Firstly, the introduction of l_1-norm penalty term forces many of the model coefficients to zero, leaving only the important ones. This can be explained by the diamond-shaped constraint region in the space of coefficients. The corners represent situations in which other coefficients are exactly zero. Secondly, when multicollinearity exists in the candidate terms, the optimization algorithm tends to choose one variable that provides unique information and set the parameters of others that have similar information to zero, which means the redundant variables no longer exist in $\mathbf{P}_{\mathrm{ST}}$ (Herawati, Nisa, Setiawan, Nusyirwan, & Tiryono, 2018). After the variable selection step, the matrix $\mathbf{P}_{\mathrm{ST}}$ has full column rank. Hence, the model coefficients can be uniquely determined by the least squares algorithm.

3.4. The implemented algorithm

The proposed approach consists of two stages, namely variable selection and coefficient estimation. To obtain a sparse model structure, local regularization is introduced to solve the subproblem (5). In each iterative step, the evidence framework runs in a nested loop, estimating the coefficients $\boldsymbol{\theta}$ and optimal local regularization parameters $\boldsymbol{\mu}$. Within

this loop, terms corresponding to large μ_i are removed. After that, variables $\boldsymbol{\nu}$, $\boldsymbol{d}$ and $\boldsymbol{\eta}$ are sequentially updated. The iteration stops when the prediction error converges. After the iteration, terms with small coefficients are pruned, resulting in a more compact model.

In the coefficient estimation stage, the coefficients are re-estimated using the system output and input of the selected terms. The complete procedure is outlined in Algorithm 3.

Algorithm 3 LR-SALSA

Variable selection stage:

Require: Predefine $\boldsymbol{\alpha}_0$, β_0, λ, $tol_\alpha, \delta_\alpha, \delta_e$ and δ_θ
Ensure: $\boldsymbol{\nu_0} = \boldsymbol{d_0} = \boldsymbol{\eta_0} = \mathbf{0}$
for $k = 1$ to n: **do**
  repeat
    Update $\boldsymbol{m}$ and $\mathbf{S}$ by (11) and (12)
    Update $\boldsymbol{\alpha}$ and β by (13) and (14)
    if $|\alpha_i| \geq tol_\alpha$ **then**
      Term $\boldsymbol{p}_i$ is removed
    end if
  until $\max\{\alpha_i^{last} - \alpha_i^{new}\} \leq \delta_\alpha$
  $\hat{\boldsymbol{\theta}}_k = \boldsymbol{m}$, $\boldsymbol{\mu} = \boldsymbol{\alpha}/\beta$
  $\boldsymbol{\nu_k} = \max(0, \hat{\boldsymbol{\theta}}_k + \boldsymbol{d}_{k-1} - \lambda/\boldsymbol{\mu}) - \max(0, -(\hat{\boldsymbol{\theta}}_k + \boldsymbol{d}_{k-1}) - \lambda/\boldsymbol{\mu})$
  $\boldsymbol{d}_k = \boldsymbol{d}_{k-1} + (\hat{\boldsymbol{\theta}}_k - \boldsymbol{\nu}_k)$
  $\boldsymbol{\eta}_k = \boldsymbol{\nu}_k - \boldsymbol{d}_k$
  $e_k = (\boldsymbol{y} - \mathbf{P}\hat{\boldsymbol{\theta}}_k)^T(\boldsymbol{y} - \mathbf{P}\hat{\boldsymbol{\theta}}_k)$
  if $|e_k - e_{k-1}| \leq \delta_e$ **then**
    break
  end if
end for
if $\hat{\theta}_i^2/\|\hat{\boldsymbol{\theta}}\|_2^2 \leq \delta_\theta$ **then**
  Term $\boldsymbol{p}_i$ is removed
end if
Determine the terms of the final model

Coefficient estimation stage:

Select the variables $\boldsymbol{y}$ and $\mathbf{P}_{\mathrm{ST}}$
Determine the coefficients by (21)

The complexity of the proposed method is discussed here. In each iteration of the LR-SALSA algorithm, solving the first subproblem (5) takes most of the computing time, and the other two subproblems (6) and (7) are solved directly. For (5), the update rules for the hyperparameters $\boldsymbol{\alpha}$ and β depend on computing the posterior mean $\boldsymbol{m}$ and covariance matrix $\mathbf{S}$ of $\boldsymbol{\theta}$. The computation of $\boldsymbol{m}$ only involves matrix multiplications, whose amount is $O(M^2 + M^2 + MN)$. The computation of $\mathbf{S}$ requires an inverse operation of order $O(M^3)$ complexity and $O(M^2)$ memory storage. The computational complexity of $\boldsymbol{\alpha}$ and β is $O(M)$ and $O(MN)$, respectively. Therefore, the dominant cost in one iteration of hyperparameter updating should be $O(M^3)$.

In practice, the convergence is usually achieved in about tens of iterations. Even though there is a nested loop in the LR-SALSA algorithm, the solution can be obtained quickly as the pruning operation at each step rapidly reduces M to a very limited size. The computational efficiency improved by timely discarding redundant terms will be more significant when the initial value of M is very large. Based on the analysis above, the whole algorithm needs several times the cost of a standard SALSA algorithm, which is much less compared with the SAL method that needs about 25 times SALSA (Tang et al., 2019). The complexity of the proposed algorithm is lower than the BAL method as well. In BAL, the computational complexity is $O(N^3)$ and the pruning procedure does not improve the efficiency (Tang et al., 2018).

3.5. The convergence

The convergence of the proposed LR-SALSA algorithm can be analyzed under the framework of SALSA.

Theorem 1 (*Eckstein & Bertsekas, 1992*). *Consider problem (1), where f_1 and f_2 are closed, proper, convex functions. Consider arbitrary $\mu > 0$ and $\boldsymbol{\nu}_0, \boldsymbol{d}_0 \in \mathbb{R}^M$. Let $\{a_k \geq 0, k = 0, 1, \ldots, \infty\}$ and $\{b_k \geq 0, k = 0, 1, \ldots, \infty\}$ be two sequences such that*

$$\sum_{k=0}^{\infty} a_k < \infty \quad \text{and} \quad \sum_{k=0}^{\infty} b_k < \infty \tag{22}$$

Consider three sequences $\{\hat{\boldsymbol{\theta}}_k \in \mathbb{R}^M, k = 0, 1, \ldots\}$, $\{\boldsymbol{\nu}_k \in \mathbb{R}^M, k = 0, 1, \ldots\}$, and $\{\boldsymbol{d}_k \in \mathbb{R}^M, k = 0, 1, \ldots\}$ that satisfy

$$a_k \geq \left\| \hat{\boldsymbol{\theta}}_{k+1} - \arg\min_{\boldsymbol{\theta}} \left\{ f_1(\boldsymbol{\theta}) + \frac{\mu}{2} \|\boldsymbol{\theta} - \boldsymbol{\nu}_k + \boldsymbol{d}_k\|_2^2 \right\} \right\| \tag{23}$$

$$b_k \geq \left\| \boldsymbol{\nu}_{k+1} - \arg\min_{\boldsymbol{\nu}} \left\{ f_2(\boldsymbol{\nu}) + \frac{\mu}{2} \|\hat{\boldsymbol{\theta}}_{k+1} - \boldsymbol{\nu} + \boldsymbol{d}_k\|_2^2 \right\} \right\| \tag{24}$$

$$\boldsymbol{d}_{k+1} = \boldsymbol{d}_k + (\hat{\boldsymbol{\theta}}_{k+1} - \boldsymbol{\nu}_{k+1}) \tag{25}$$

If (1) has a solution, the sequence $\{\hat{\boldsymbol{\theta}}_k\}$ converges, $\hat{\boldsymbol{\theta}}_k \to \boldsymbol{\theta}^*$, where $\boldsymbol{\theta}^*$ is a solution of (1). However, if (1) does not have a solution, then at least one of the sequences $\{\boldsymbol{\nu}_k\}$ or $\{\boldsymbol{d}_k\}$ diverges.

This theorem shows that if the subproblems (5) and (6) can be solved exactly, or the sequence of errors is absolutely summable, convergence is guaranteed. As shown in Afonso et al. (2010), the objective function of (5) in SALSA is a strictly convex quadratic function, leading to the following solution:

$$\hat{\boldsymbol{\theta}}_{k+1} = (\mathbf{P}^T\mathbf{P} + \mu\mathbf{I})^{-1}(\mathbf{P}^T\boldsymbol{y} + \mu(\boldsymbol{\nu}_k - \boldsymbol{d}_k)) \tag{26}$$

which can be solved exactly in many cases. Because in a number of problems of interest, the term $(\mathbf{P}^T\mathbf{P} + \mu\mathbf{I})$ is invertible. For subproblem (6), it can be exactly solved by introducing the soft thresholding operator. Therefore, SALSA is guaranteed to converge.

Notice that (26) is formally similar to the maximum a posterior (MAP) estimate of $\boldsymbol{\theta}$, from observations $\boldsymbol{y} = \mathbf{P}\boldsymbol{\theta} + \boldsymbol{n}$, where $\boldsymbol{n}$ is white Gaussian noise of variance $1/\mu$ and $\boldsymbol{\theta}$ has a Gaussian prior $\mathcal{N}(\boldsymbol{\nu}_k - \boldsymbol{d}_k, \mathbf{I})$. In LR-SALSA, the local regularization can apply individual penalty parameters on each element of $\boldsymbol{\theta}$. From Bayesian viewpoint, the Gaussian prior becomes $\mathcal{N}(\boldsymbol{\nu}_k - \boldsymbol{d}_k, \mathbf{A}^{-1})$, where $\mathbf{A} = \mathrm{diag}\{\alpha_1, \alpha_2, \ldots, \alpha_M\}$. In this case, the estimate of $\boldsymbol{\theta}$ is $\boldsymbol{m}$ at the convergence of $\boldsymbol{\alpha}$-optimization, essentially taking the similar form to (26). Hence, the subproblem (5) in LR-SALSA can be exactly solved as long as $\boldsymbol{\alpha}$-optimization converges.

As shown in Appendix A, an implicit solution for $\boldsymbol{\alpha}$ is derived by directly maximizing the marginal likelihood function $\ln p(\boldsymbol{y}|\mathbf{P})$. The re-estimation procedure will converge at a point where $\ln p(\boldsymbol{y}|\mathbf{P})$ reaches a stationary point. During the process of re-estimation, many of the α_i tend to infinity, implying that $p(\theta_i|\boldsymbol{y}, \boldsymbol{\alpha}, \beta)$ becomes highly peaked at zero (Tipping, 2001). Then these α_i are discarded from the next iteration and the corresponding terms can be pruned. As illustrated in MacKay (1992), at the optimum value of $\boldsymbol{\alpha}$, the effective total number of well-determined parameters is also ensured. Therefore, the whole LR-SALSA algorithm is guaranteed to converge.

4. Simulation study

In this section, two nonlinear examples represented by Nonlinear AutoRegressive with eXogenous inputs (NARX) models are employed to evaluate the sparsity, accuracy, and computation speed of the feature selection methods. A comprehensive introduction to NARX models can be found in Billings (2013), which will not be repeated here. The utilization of the NARX model for this simulation study is based on the assumption that the accurate selection of NARX model terms and accurate prediction of model parameters can be used as a reliable indicator of the capability of the feature selection method in generating representative features and making accurate predictions.

For this simulation study, except for LR-SALSA, many algorithms including OFR, LASSO, SAL, SBL, and BAL are implemented for the sake of comprehensive comparison. To ensure a fair evaluation, each algorithm has been executed multiple times with different random seeds. Both the best result and statistical analysis of performance indices after repeated executions are presented in this section.

Table 1
The simulation results of Example 1.

Terms	Coefficient	**LR-SALSA**	OFR	LASSO	SAL	SBL	BAL
$z(t-2)$	−0.5	−0.498	−0.5014	−0.4967	−0.5021	−0.4981	−0.5002
$z(t-1)u(t-1)$	0.7	0.6954	0.698	0.6866	0.6901	0.7039	0.7024
$u^2(t-2)$	0.6	0.5992	0.5997	0.5898	0.5967	0.6018	0.5994
$z^3(t-1)$	0.2	0.2008	0.2018	0.1911	0.2005	0.1986	0.1976
$z(t-2)u^2(t-2)$	−0.7	−0.7064	−0.6992	−0.6575	−0.6884	−0.6952	−0.7009
$z(t-4)$	–	–	–	0.0011	–	–	–
$u^2(t-1)$	–	–	–	0.0008	–	–	–
$u(t-1)u^2(t-3)$	–	–	–	0.002	–	–	–
$z(t-4)u^2(t-2)$	–	–	−0.0022	–	–	–	–
MAE_θ	–	**0.0029**	0.0012	0.0157	0.0055	0.0028	0.0013

4.1. Example 1

First, consider the benchmark nonlinear system (Tang et al., 2019):

$$\begin{aligned} z(t) =& -0.5z(t-2)+0.7z(t-1)u(t-1)+0.6u^2(t-2) \\ &+0.2z^3(t-1)-0.7z(t-2)u^2(t-2) \\ y(t) =& z(t)+e(t) \end{aligned} \tag{27}$$

where $u(t)$ and $z(t)$ denote the system input and output at time t, respectively. $u(t)$ is a uniformly distributed white noise within the range of $[-1,1]$. $z(t)$ is the output of the system that is excited by $u(t)$. An external Gaussian noise sequence $e(t)$ is added to $z(t)$. The signal-to-noise ratio (SNR) is set to 15 dB. The maximum lags for input and output are selected as 3 and 4, respectively. The linear candidate terms are $z(t-1), z(t-2), z(t-3), z(t-4), u(t-1), u(t-2)$ and $u(t-3)$. In terms of nonlinearity, the maximum polynomial degree is selected as 3. As a result, based on the dictionary generation equations introduced in Billings (2013), the total number of candidate terms is 119. In this example, 3000 input and output samples are generated for model learning. Thus, the dimension of matrix $\mathbf{P}$ is 3000×119. The mean absolute error (MAE) of the estimated coefficients is used to evaluate the model performance

$$\mathrm{MAE}_\theta = \frac{1}{M}\sum_{i=1}^{M}|\theta_i - \hat{\theta}_i| \tag{28}$$

In the implementation of the proposed LR-SALSA algorithm, some parameters need to be determined in advance. The tolerance tol_α and stop criterion δ_α used in local regularization parameter learning were set to $1e^5$ and $1e^{-5}$, respectively, according to the suggestions in Tipping (2001). The stopping criteria δ_e and δ_θ for variable selection were $1e^{-5}$ and 0.01, which were reasonably small (Tang et al., 2019). The l_1-norm parameter λ was set to 1. In the OFR algorithm, the important terms were selected according to the error reduction ratio criterion, which was carefully tuned and set to 0.05 in this study (Tang et al., 2018). In other algorithms, the l_1-norm parameter λ was critical to model performance. The cross-validation method was adopted to select a value that results in the smallest standard error (Li, Wang and Kruger, 2021). The selected parameters for these methods were: $\lambda_{\mathrm{LASSO}} = 7.2e^{-4}$, $\lambda_{\mathrm{SAL}} = 1$, $\lambda_{\mathrm{SBL}} = 0.4$, and $\lambda_{\mathrm{BAL}} = 0.1$. Additionally, in SAL, the subsampling was repeated 10 times and $\delta_{thr} = 0.7$. In SBL, the CVX solver was used to address the optimization problem and the number of repeated steps was 5 (Pan et al., 2016).

Table 1 presents the best results obtained in this case. It can be seen that all methods can produce sparse models. OFR estimates the model coefficients with the highest accuracy, but a redundant term $z(t-4)u^2(t-2)$ is also selected. LASSO selects more redundant terms and the coefficients of the correct terms are less accurate than others. The other four methods accurately choose the correct terms and estimate coefficients at a similar level of precision. Among them, BAL stands out with a MAE_θ of 0.0013.

To evaluate the stability of these methods, 100 simulation executions were conducted using different random seeds and recorded the resulting performance indices. Fig. 1 shows the box plots for the number of selected terms, MAE_θ, and computation time. As can be seen in Fig. 1(a), the LR-SALSA and SBL algorithms consistently identify the correct terms. The SAL and BAL methods choose redundant terms occasionally. In contrast, the OFR and LASSO algorithms tend to select additional terms. In terms of estimation accuracy, as shown in Fig. 1(b), OFR has the lowest and highest MAE_θ, showing worse stability. The proposed LR-SALSA outperforms LASSO and SAL and achieves similar estimation accuracy to SBL and BAL. However, as shown in Fig. 1(c), LR-SALSA takes the least amount of time to construct a model compared with other methods except for OFR. It takes an extremely long time for feature selection using the SBL algorithm, which indicates the limitation of using the third-party solver. As a result, compared with the other methods, the proposed LR-SALSA method offers a superior accuracy in generating sparse models meanwhile ensuring fast computation.

4.2. Example 2

Consider another nonlinear system (Tang et al., 2019):

$$\begin{aligned} z(t) =& u(t-1)-0.3u(t-2)-0.4u(t-3) \\ &+0.25u(t-1)u(t-2)-0.2u(t-2)u(t-3) \\ &-0.3u^3(t-1)+0.24u^3(t-2)+0.8z(t-1) \\ y(t) =& z(t)+e(t) \end{aligned} \tag{29}$$

where $u(t)$ denotes the system input and $z(t)$ denotes the output at time t. A white noise $u(t) \in [-1,1]$ sampled from the uniform distribution is used as the system input. For the Gaussian noise sequence $e(t)$, the SNR is set to 15 dB. In this example, the maximum lags for input and output are selected as 2 and 3. The maximum polynomial degree is selected as 3. Therefore, these delayed inputs and outputs can form a model term candidate pool with 55 linear and nonlinear terms (Billings, 2013). In total, 3500 samples are generated. The size of $\mathbf{P}$ is 3500×55. The MAE_θ is also used to test the model performance.

The parameter settings for the proposed algorithm were consistent with those in Example 1. Nevertheless, key parameters used in other methods have changed. In the OFR algorithm, the stopping criterion was set to 0.033. In the other four algorithms, the l_1-norm parameters were selected as the following values: $\lambda_{\mathrm{LASSO}} = 3.5e^{-4}$, $\lambda_{\mathrm{SAL}} = 0.01$, $\lambda_{\mathrm{SBL}} = 0.03$, and $\lambda_{\mathrm{BAL}} = 0.05$. The best simulation results are presented in Table 2. Similar to the results in Example 1, the LR-SALSA, SBL, and BAL algorithms can estimate the coefficients of correct terms more accurately. Even though SBL yields the lowest MAE_θ of 0.0044, 6 redundant terms are included in the final model, making the result unreliable. LASSO also produces a redundant model comprising 14 terms and neglects the essential term $u(t-2)$. The OFR and SAL algorithms have selected the true terms but have not estimated the coefficients as accurately as the proposed LR-SALSA algorithm.

The stability test has been conducted in this example as well, and Fig. 2 shows the box plots resulting from 100 simulation executions with different random seeds. As can be seen in Fig. 2(a), the average numbers of selected terms of LR-SALSA and BAL are approximately 8, aligning with the true model term number in Eq. (29). The OFR, LASSO,

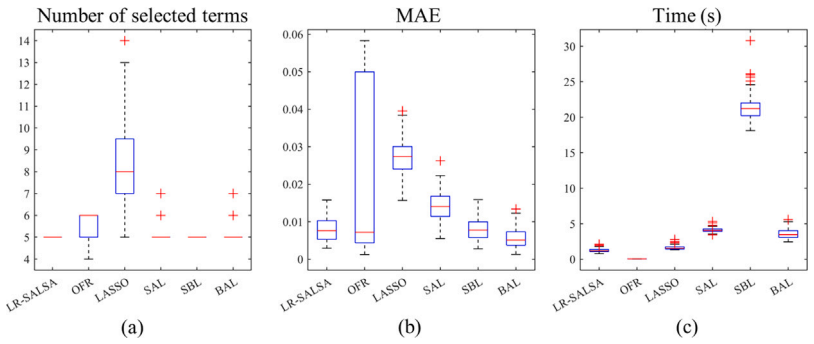

Fig. 1. Box plots of the results produced by LR-SALSA and other common algorithms in example 1: (a) number of selected terms, (b) MAE_θ, and (c) computation time.

Table 2
The simulation results of Example 2.

Terms	Coefficient	**LR-SALSA**	OFR	LASSO	SAL	SBL	BAL
$z(t-1)$	0.8	0.7917	0.8959	0.5157	0.7475	0.8045	0.7847
$u(t-1)$	1	0.9942	0.9942	0.9769	1.009	0.9978	1.0038
$u(t-2)$	−0.3	−0.2808	−0.3871	–	−0.2581	−0.2995	−0.2835
$u(t-3)$	−0.4	−0.3977	−0.4456	−0.2477	−0.3733	−0.459	−0.3983
$u(t-2)u(t-2)$	0.25	0.2484	0.2557	0.2473	0.2401	0.2565	0.249
$u(t-2)u(t-3)$	−0.2	−0.2035	−0.2269	−0.1238	−0.1848	−0.2056	−0.2025
$u^3(t-1)$	−0.3	−0.2918	−0.2917	−0.2699	−0.3101	−0.3002	−0.3155
$u^3(t-2)$	0.24	0.2332	0.2571	0.1301	0.2453	0.2301	0.2316
$z^2(t-1)$	–	–	–	−0.0061	–	–	–
$z(t-1)u(t-1)$	–	–	–	0.0057	–	–	–
$z(t-2)u(t-3)$	–	–	–	0.0122	–	–	–
$z^3(t-2)$	–	–	–	−0.0107	–	–	–
$u^2(t-1)u(t-3)$	–	–	–	−0.0017	–	–	–
$u(t-1)u^2(t-2)$	–	–	–	−0.0091	–	–	–
$u^2(t-2)u(t-3)$	–	–	–	−0.0003	–	–	–
$z(t-1)u(t-3)$	–	–	–	–	–	0.0041	–
$u(t-1)u(t-3)$	–	–	–	–	–	0.0058	–
$z(t-1)u(t-1)u(t-2)$	–	–	–	–	–	−0.0051	–
$z(t-1)u(t-1)u(t-3)$	–	–	–	–	–	−0.0033	–
$z(t-2)u^2(t-2)$	–	–	–	–	–	0.0053	–
$u(t-2)u^2(t-3)$	–	–	–	–	–	−0.102	–
MAE_θ	–	**0.0070**	0.0366	0.1223	0.0213	0.0044	0.0080

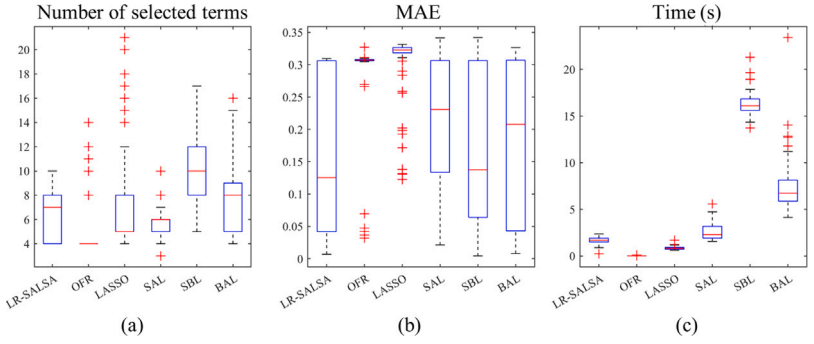

Fig. 2. Box plots of the results produced by LR-SALSA and other common algorithms in example 2: (a) number of selected terms, (b) MAE_θ, and (c) computation time.

and SAL algorithms tend to estimate an over-sparse model, consistently selecting fewer than 6 terms. This indicates that some important terms may be ignored. On the contrary, SBL usually yields a model with redundant terms. Fig. 3 shows the occurrences of true terms in 100 executions. It can be found that terms like $u(t-1)$, $u(t-1)u(t-2)$, and $u^3(t-1)$ are consistently chosen by all six algorithms nearly in every execution. However, other important terms are selected in a higher occurrence only by using LR-SALSA, SBL, and BAL methods. Compared with SBL and BAL, LR-SALSA exhibits superior estimation accuracy and efficiency as can be seen in Fig. 2(b) and (c). Furthermore, in this example, even though OFR and LASSO take less computation time but also produce higher estimation errors. The results in both Examples 1 and 2 demonstrate that LR-SALSA is able to build compact and accurate models, and significantly reduce the computational burden at the same time.

5. Experimental study

To validate the effectiveness of the proposed feature selection method in real-world applications, the datasets obtained from a face milling experiment were employed to develop a model for tool wear prediction. Similar to the simulation study, as a comparison, other common l_1-norm-based feature selection methods were applied to

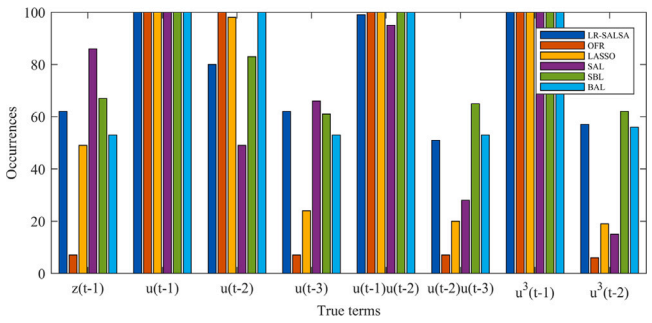

Fig. 3. The occurrences of all true terms in 100 executions with different random seeds in Example 2.

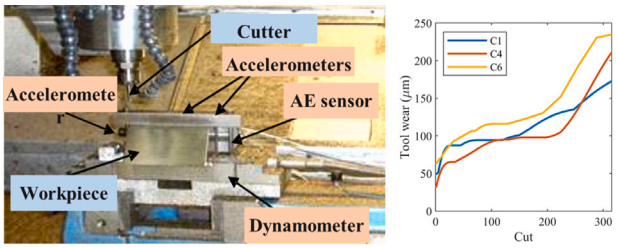

Fig. 4. (a) Experimental setup, (b) the measurements of tool wear.

select important signal features. The model performance was evaluated in terms of sparsity, accuracy, and efficiency.

5.1. Experimental setup

The dataset applied to this case study is sourced from the 2010 Prognostics and Health Management Data Challenge (Li et al., 2009). Fig. 4(a) shows the experimental setup where the milling process was conducted on a high-speed milling machine (Roders Tech RFM 760). The surface of an Inconel 718 workpiece is machined layer by layer using a three-flute ball end mill with a slope of 60°. The machining parameters were set as follows: the spindle speed of the cutter was 10 400 RPM; feed rate was 1555 mm/min; radial cutting depth was 0.125 mm; axial cutting depth was 0.2 mm. During the milling process, tri-axial force, tri-axial vibration, and acoustic emission (AE) signals were collected using dynamometers, accelerometers, and AE sensors, respectively. The sampling frequency was set to 50 kHz. The flank wear of each flute was measured using a microscope after each cut. Fig. 4(b) presents the tool wear measurements for three cutters, namely C1, C4, and C6. The dataset of one cutter included 315 samples corresponding to 315 cuts. It is worth mentioning that the datasets from C2, C3 and C5 were not used in this study because the tool wear measurements were not available for these three cutters.

5.2. Algorithm implementation

In this experiment, six channels of signals, including tri-axial force and vibration, were utilized to construct a tool wear prediction model. The AE signal was not used because the dominant frequency range was too high compared with the tooth passing frequency (Qin et al., 2022). A total of 13 time and frequency domain features, as listed in Table 3, were extracted from each signal. Thus the number of candidate features was 78 (= 13 × 6) for each sample. The datasets from any two cutters among C1, C4, and C6, comprising 630 samples, were allocated for training and validation. Among them, 90% (567 samples) were randomly designated as the training set, and the remaining 10%

Table 3
List of the extracted features.

Features	Time domain	Mean Variance Skewness Kurtosis Root mean square Peak-to-peak value Crest factor
	Frequency domain	Mean Variance Skewness Kurtosis Peak-to-peak value Harmonics

(63 samples) were for validation purposes. The dataset from the other cutter (315 samples) was used for testing. Considering the features were directly used as inputs, the dimensions of the training, validation, and testing feature matrices were 567 × 78, 63 × 78, and 315 × 78, respectively.

After the feature dictionary was generated, each column was normalized using the training set, ensuring that all variables were adjusted to the same scale. Subsequently, the feature selection algorithms were applied to build sparse regression models for tool wear prediction. The prediction accuracy was evaluated on the testing set using two indices: Mean Absolute Error (MAE) and Root Mean Squared Error (RMSE), which are defined below.

$$\begin{aligned} \text{MAE} &= \frac{1}{N}\sum_{i=1}^{N}|y_i-\hat{y}_i| \\ \text{RMSE} &= \sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_i-\hat{y}_i)^2} \end{aligned} \tag{30}$$

where y_i and $\hat{y}_i$ denote the true and predicted tool wear of the ith cutting pass, respectively.

Here the tuning of key parameters is introduced briefly. Most of the parameters in LR-SALSA shared the same values as the simulation study, except that the iteration stopping criterion δ_e was changed to $1e^{-3}$ considering the range of target values and the parameter λ was selected by cross-validation. It should be noted that in the second stage of LR-SALSA, ridge regression with a penalty coefficient of 2 was employed to estimate model coefficients to further avoid overfitting. In the SAL method, the number of subsampling rounds was 10 and δ_{thr} was 0.6. In SBL, the number of total steps was 10. Additionally, cross-validation was utilized to determine the error reduction ratio in OFR and the regularization parameter λ in other methods.

5.3. Experimental results

Fig. 5 shows the tool wear prediction results for all cutters using different feature selection methods. The number of selected features, prediction accuracy, and execution time are summarized in Table 4. It should be noted that all the tests were conducted on a laptop with an Intel(R) Core(TM) i7-7700HQ CPU and 16 GB memory. As can be observed, the proposed LR-SALSA selected 8, 7, and 4 features for C1, C4, and C6. While the number of features selected by LASSO varies from 19 to 49. The other methods produced sparse models with 4 to 14 features. This indicates that LR-SALSA can achieve better model sparsity.

From the prediction results, it can be found LR-SALSA method shows better fitting performance, especially for C1 and C6. The other methods can also make accurate predictions in several cases, for example, the result of C6 using SBL. MAE and RMSE can quantify the prediction accuracy. In the LR-SALSA method, these two indices vary from 8.38 to 10.58 and from 9.97 to 13.92, respectively. However, the

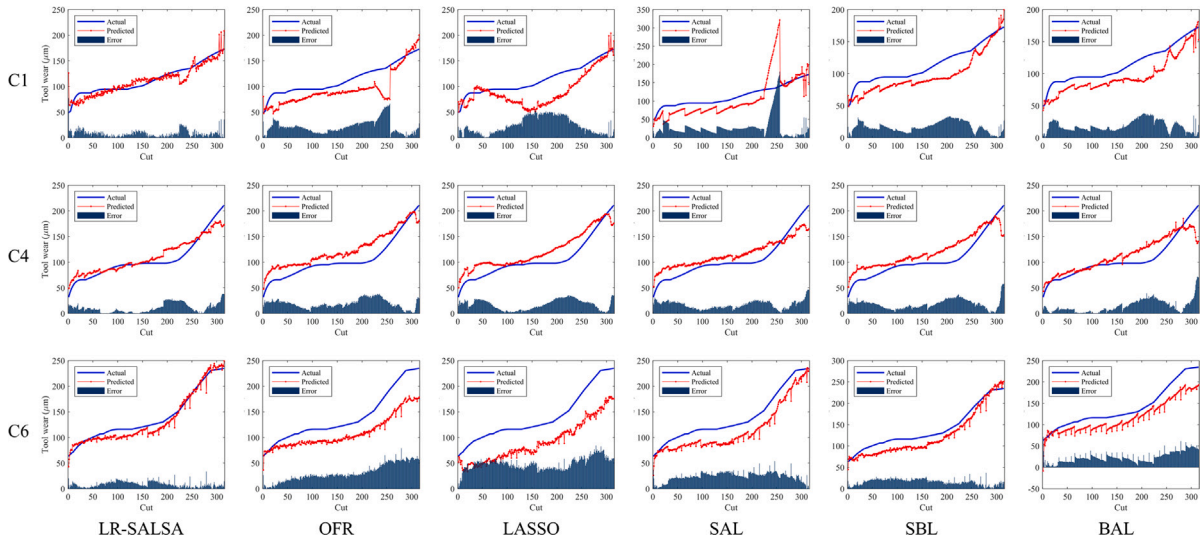

Fig. 5. Tool wear prediction results, from top to bottom: for cutters C1, C4 and C6; from left to right: using LR-SALSA, OFR, LASSO, SAL, SBL, and BAL methods (the maximum polynomial degree: 1).

Table 4
The testing results of C1, C4, and C6 in the milling experiment (the maximum polynomial degree: 1).

Algorithms		**LR-SALSA**	OFR	LASSO	SAL	SBL	BAL
Number of features	C1	8	14	19	8	8	12
	C4	7	8	49	10	14	5
	C6	4	4	45	11	12	8
MAE	C1	8.52	20.95	24.02	27.35	16.77	18.41
	C4	10.58	19.18	17.38	16.92	18.35	14.41
	C6	8.38	31.14	48.82	24.13	16.17	23.5
RMSE	C1	11.27	24.68	28.98	39.15	18.7	20.44
	C4	13.92	21.16	20.05	18.65	20.73	19.34
	C6	9.97	35.26	50.33	25.7	17.57	26.69
Execution time (s)	C1	2.21	0.34	2.52	6.82	6.41	3.4
	C4	1.97	0.21	2.65	6.56	6.07	3.81
	C6	2.33	0.13	2.93	7.11	5.7	3.34

other methods yield larger MAE and RMSE values. Even though the predicted result of SBL fits the actual tool wear well, the indices are twice as large as the proposed method.

In terms of computational efficiency, OFR takes the least time of less than 1 s in all three cases. LR-SALSA takes about 2 s, which is close to LASSO and BAL. While SAL and SBL take the longest time due to the use of subsampling and the third-party solver. Thus the conclusion obtained in the simulation study that LR-SALSA has a computational advantage over SAL, SBL, and BAL still holds. In this experimental study, the results demonstrate the benefits of LR-SALSA in selecting less but representative features and ensuring fast computation with high prediction accuracy.

5.4. The influence of maximum polynomial degree

The construction of dictionary matrix $\mathbf{P}$ plays an important role in the feature selection problem. In the simulation study, the maximum lags and polynomial degrees are selected directly, which are not known in most cases. The experimental results mentioned above are from 78 linear terms of raw signal features. However, the polynomial expansion can enrich the extracted information, which may improve the model accuracy (Billings, 2013). Hence, it is necessary to investigate the performance of the feature selection methods with more complicated inputs. The second-degree terms are generated by combining different features (for instance, X-vibration Mean×Y-vibration Variance), leading to 3159 ($= 78 + 78 \times 79 \div 2$) candidate terms in total. After the data splitting, the dimensions of the training, validation, and testing feature matrices become 567×3159, 63×3159, and 315×3159, respectively.

All the feature selection methods are applied to the new matrices. The parameters are determined in the same way as aforementioned. Table 5 shows the testing performance for all cutters. It can be found that, except for SAL, most methods tend to select more important features when the second-degree terms are included in the dictionary matrix. However, the feature selection still works as the remaining features are significantly less than the candidates. Contrary to expectations, the prediction accuracy decreases compared with Table 4 in most cases. While the proposed LR-SALSA always achieves the lowest fitting errors. Its advantage becomes more obvious given the fact SAL, SBL, and BAL take dozens of times longer in feature selection. However, OFR still completes the task rapidly owing to the characteristics of the forward selection algorithm.

This case study further demonstrates that the proposed LR-SALSA is able to achieve a good trade-off in model sparsity, accuracy and computational efficiency. Its high execution speed is more significant in many scenarios involving complicated datasets. However, it should be acknowledged that sometimes more candidate terms do not necessarily bring about more accurate models. Too many redundant features make it difficult to identify the truly important features. Therefore, how to select an appropriate polynomial degree to construct the feature dictionary deserves more study.

6. Discussion

To better explain the advantages of the proposed LR-SALSA-based feature selection approach in condition monitoring, some discussions are summarized as follows.

- *Scope of application*: The proposed algorithm can be applied to obtain a linear-in-the-parameter model, where the relationship between the response y and the parameters θ is linear. However, a model being linear-in-the-parameter may also represent a nonlinear relationship between the input and output variables. Because the input variables can be transformed or combined in non-linear ways to construct the dictionary matrix $\mathbf{P}$, for example,

Table 5
The testing results of C1, C4, and C6 in the milling experiment (the maximum polynomial degree: 2).

Algorithms		**LR-SALSA**	OFR	LASSO	SAL	SBL	BAL
Number of features	C1	12	16	116	8	34	5
	C4	13	19	63	4	15	15
	C6	9	3	136	7	9	5
MAE	C1	13.71	19.64	17.03	14.5	19.31	16.1
	C4	12.19	20.23	20.49	25.97	15.4	14.48
	C6	13.1	33.51	89.29	28.92	22.77	20.18
RMSE	C1	17.23	24.65	19.9	18.16	27.01	20.2
	C4	14.38	22.07	22.97	34.31	19.01	17.7
	C6	16.36	36.69	97.9	30.56	23.93	21.84
Execution time (s)	C1	10.31	0.92	45.41	472.72	286.07	309.95
	C4	10.58	1.23	44.89	623.94	130.64	216.98
	C6	9.51	0.07	18.2	1267.74	57.58	212.75

the polynomial-expansion models (27) and (29). Linear-in-the-parameter models are widely applied across various fields due to their simplicity, interpretability, and computational efficiency. As the original set of available terms may be very large, feature selection is a crucial step when building a linear-in-the-parameter model. The proposed LR-SALSA algorithm can be applied to both linear and nonlinear models by choosing a subset of important terms and producing a sparse model.

- *Sparsity and accuracy*: From both simulation and experimental studies, it is observed that conventional OFR and LASSO methods often select over-sparse or redundant features, reducing the accuracy of model predictions. SAL and SBL methods have achieved better estimation accuracy but cannot always choose the right terms. Only LR-SALSA and BAL can build sparse models accurately. The unique contribution of this work is that the proposed LR-SALSA algorithm can automatically select the most representative features, which originates from a straightforward local regularization strategy. The regularization parameters are directly related to the contribution of candidate features. As a result, the proposed method can provide a more compact model, addressing the difficulties of over-sparsity and redundancy.
- *Computational speed*: It can be found from the simulation and experimental studies that, the proposed LR-SALSA method exhibits higher computational efficiency when compared with the modern SAL, SBL, and BAL methods. The improved efficiency of LR-SALSA is attributed to its design, which conducts the SALSA algorithm only once compared with SAL, avoids the use of third-party solvers compared with SBL, and requires less computation burden compared with BAL.
- *Limitations*: The number of tuning parameters of LR-SALSA, at this stage, is greater than some of the established methods. Their settings have been introduced in both simulation and experimental studies, which are summarized as follows.

  tol_α: it is used as a threshold to determine which terms should be removed in the learning of local regularization parameters. Increasing tol_α will increase the execution time. It should be reasonably large and is fixed as $1e^5$ in this study according to Tipping (2001).

  δ_α: it is used to determine when the learning of local regularization parameters converges. Increasing δ_α will reduce the execution time. It should be reasonably small and is fixed as $1e^{-5}$ in this study according to Tipping (2001).

  δ_e: it is used to determine when the whole algorithm converges. Increasing δ_e will reduce the execution time and increase the number of selected features. It should be determined according to the range of target values and is set to $1e^{-5}$ and $1e^{-3}$ in simulation and experiment study, respectively.

  δ_θ: it is used as a threshold to determine which terms should be further removed after the iteration process. Increasing δ_θ will reduce the number of selected features. It is set to 0.01 as used in Tang et al. (2019).

  λ: it is the l_1-norm regularization parameter. Increasing λ usually results in more selected features. It is determined automatically through cross-validation.

From the analysis above, it is found that most parameters can be determined according to empirical knowledge. However, in specific applications, fine-tuning of the parameters is unavoidable. A systematic analysis is needed to guide the setting of key parameters and enhance the applicability of LR-SALSA. This will be a focal point of future research endeavors.

7. Conclusion

This article introduces the Local Regularization Assisted Split Augmented Lagrangian Shrinkage Algorithm (LR-SALSA) as a solution to the feature selection problem in condition monitoring. The objective is to solve a l_1-norm optimization problem where the problem is decomposed into three subproblems that can be solved separately. In the first subproblem, individual penalty parameters are introduced to every coefficient such that redundant terms are removed efficiently. An iterative algorithm, based on the Bayesian evidence framework, is derived to optimize the local regularization hyperparameters automatically and guarantee model sparsity. After the important terms are determined, model coefficients are re-estimated by the least squares method. Consequently, the feature selection problem is resolved in a single SALSA iteration, significantly reducing computational expenses. The complexity and convergence of the proposed algorithm are presented. The effectiveness of the proposed approach is demonstrated through two simulation studies and a milling experiment. The results show that only LR-SALSA has a good trade-off among model sparsity, estimation accuracy, and computational efficiency in comparison to five existing feature selection methods. Future research will focus on simplifying the implementation of the algorithm to facilitate its practical applications.

CRediT authorship contribution statement

Yufei Gui: Writing – original draft, Software, Formal analysis. **Xiaoquan Tang:** Project administration, Investigation, Formal analysis. **Zepeng Liu:** Writing – review & editing, Supervision, Methodology, Investigation, Funding acquisition, Formal analysis, Conceptualization.

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.

Appendix A. The evidence framework

In the Bayesian evidence framework, the following prior distributions of $\boldsymbol{y}$ and $\boldsymbol{\theta}$ are first introduced

$$p(\boldsymbol{y}|\mathbf{P},\boldsymbol{\theta},\beta)=\mathcal{N}\left(\boldsymbol{y}|\mathbf{P}\boldsymbol{\theta},\beta^{-1}\mathbf{I}\right) \tag{A.1}$$

$$p(\boldsymbol{\theta}|\boldsymbol{\eta},\boldsymbol{\alpha})=\mathcal{N}(\boldsymbol{\theta}|\boldsymbol{\eta},\mathbf{A}^{-1}) \tag{A.2}$$

where β and α_i govern the precision parameters of noise and θ_i, and $\mathbf{A}=\mathrm{diag}\{\alpha_1,\alpha_2,\ldots,\alpha_M\}$ denotes the precision matrix.

Since $p(\boldsymbol{\theta}|\boldsymbol{y},\mathbf{P},\boldsymbol{\eta},\boldsymbol{\alpha},\beta)\propto p(\boldsymbol{y}|\mathbf{P},\boldsymbol{\theta},\beta)p(\boldsymbol{\theta}|\boldsymbol{\eta},\boldsymbol{\alpha})$, the posterior distribution for $\boldsymbol{\theta}$ is Gaussian and takes the form

$$p(\boldsymbol{\theta}|\boldsymbol{y},\mathbf{P},\boldsymbol{\eta},\boldsymbol{\alpha},\beta)=\mathcal{N}(\boldsymbol{\theta}|\boldsymbol{m},\mathbf{S}) \tag{A.3}$$

where the mean and covariance matrix are given by

$$\boldsymbol{m} = \mathbf{S}(\mathbf{A}\boldsymbol{\eta} + \beta\mathbf{P}^T\boldsymbol{y}) \tag{A.4}$$

$$\mathbf{S} = (\mathbf{A} + \beta\mathbf{P}^T\mathbf{P})^{-1} \tag{A.5}$$

The unknown parameters $\boldsymbol{\alpha}$ and β are determined by evaluating the evidence function (Tipping, 2001). The target is to maximize the marginal likelihood function $p(\boldsymbol{y}|\mathbf{P},\boldsymbol{\eta},\boldsymbol{\alpha},\beta)$ by integrating over the weight vector $\boldsymbol{\theta}$

$$\begin{aligned} p(\boldsymbol{y}|\mathbf{P},\boldsymbol{\eta},\boldsymbol{\alpha},\beta) &= \int p(\boldsymbol{y}|\mathbf{P},\boldsymbol{\theta},\beta)p(\boldsymbol{\theta}|\boldsymbol{\eta},\boldsymbol{\alpha})\,d\boldsymbol{\theta} \\ &= \left(\frac{\beta}{2\pi}\right)^{\frac{N}{2}}\left(\frac{1}{2\pi}\right)^{\frac{M}{2}}\prod_{i=1}^{M}\alpha_i^{\frac{1}{2}}\int \exp\{-E(\boldsymbol{\theta})\}\,d\boldsymbol{\theta} \end{aligned} \tag{A.6}$$

where

$$\begin{aligned} E(\boldsymbol{\theta}) &= \frac{\beta}{2}\boldsymbol{e}^T\boldsymbol{e} + \frac{1}{2}(\boldsymbol{\theta}-\boldsymbol{\eta})^T\mathbf{A}(\boldsymbol{\theta}-\boldsymbol{\eta}) \\ &= E(\boldsymbol{y},\boldsymbol{\eta}) + \frac{1}{2}(\boldsymbol{\theta}-\boldsymbol{m})^T\mathbf{S}^{-1}(\boldsymbol{\theta}-\boldsymbol{m}) \end{aligned} \tag{A.7}$$

$$E(\boldsymbol{y},\boldsymbol{\eta}) = \frac{1}{2}(\beta\boldsymbol{y}^T\boldsymbol{y} + \boldsymbol{\eta}^T\mathbf{A}\boldsymbol{\eta} - \boldsymbol{m}^T\mathbf{S}^{-1}\boldsymbol{m}) \tag{A.8}$$

The derivation of (A.7) is given in Appendix B.

As can be seen, the second term in Eq. (A.7) is the power exponent term of the posterior distribution $\mathcal{N}(\boldsymbol{\theta}|\boldsymbol{m},\mathbf{S})$. The log marginal likelihood is simply written in the form

$$\begin{aligned} \ln p(\boldsymbol{y}|\mathbf{P},\boldsymbol{\eta},\boldsymbol{\alpha},\beta) =& \frac{N}{2}\ln\beta + \frac{1}{2}\sum_{i=1}^{M}\ln\alpha_i - E(\boldsymbol{y},\boldsymbol{\eta}) \\ &+ \frac{1}{2}\ln|\mathbf{S}| - \frac{N}{2}\ln 2\pi \end{aligned} \tag{A.9}$$

To maximize the marginal likelihood function, its derivative with respect to $\boldsymbol{\alpha}$ and β is computed, respectively.

$$\frac{\partial}{\partial\alpha_i}\ln p(\boldsymbol{y}|\mathbf{P}) = \frac{1}{2\alpha_i} - \frac{1}{2}(m_i-\eta_i)^2 - \frac{1}{2}S_{ii} \tag{A.10}$$

$$\frac{\partial}{\partial\beta}\ln p(\boldsymbol{y}|\mathbf{P}) = \frac{1}{2\beta}\left[N - \sum_{i=1}^{M}(1-\alpha_i S_{ii})\right] - \frac{1}{2}\|\boldsymbol{y}-\mathbf{P}\boldsymbol{m}\|_2^2 \tag{A.11}$$

The derivation of (A.10) and (A.11) is given in Appendices C and D, respectively.

The maximum is obtained when Eqs. (A.10) and (A.11) are both equivalent to zero, which gives

$$\alpha_i = \frac{\gamma_i}{(m_i-\eta_i)^2} \tag{A.12}$$

$$\beta^{-1} = \frac{\|\boldsymbol{y}-\mathbf{P}\boldsymbol{m}\|_2^2}{N - \sum_{i=1}^{M}\gamma_i} \tag{A.13}$$

where $\gamma_i = 1-\alpha_i S_{ii}$. Therefore, the regularization parameter in (8) can be given as

$$\mu_i = \frac{\alpha_i}{\beta} = \frac{\gamma_i}{N-\sum_{i=1}^{M}\gamma_i}\frac{\|\boldsymbol{y}-\mathbf{P}\boldsymbol{m}\|_2^2}{(m_i-\eta_i)^2} \tag{A.14}$$

Appendix B. Derivation of (A.7)

$E(\boldsymbol{\theta})$ in (A.7) is re-written as,

$$\begin{aligned} E(\boldsymbol{\theta}) &= \frac{\beta}{2}\boldsymbol{e}^T\boldsymbol{e} + \frac{1}{2}(\boldsymbol{\theta}-\boldsymbol{\eta})^T\mathbf{A}(\boldsymbol{\theta}-\boldsymbol{\eta}) \\ &= \frac{1}{2}\left[\beta\boldsymbol{y}^T\boldsymbol{y} - 2\beta\boldsymbol{\theta}^T\mathbf{P}^T\boldsymbol{y} + \beta\boldsymbol{\theta}^T\mathbf{P}^T\mathbf{P}\boldsymbol{\theta} + \boldsymbol{\theta}^T\mathbf{A}\boldsymbol{\theta} - 2\boldsymbol{\theta}^T\mathbf{A}\boldsymbol{\eta} + \boldsymbol{\eta}^T\mathbf{A}\boldsymbol{\eta}\right] \\ &= \frac{1}{2}\left[\beta\boldsymbol{y}^T\boldsymbol{y} + \boldsymbol{\eta}^T\mathbf{A}\boldsymbol{\eta} + \boldsymbol{\theta}^T(\mathbf{A}+\beta\mathbf{P}^T\mathbf{P})\boldsymbol{\theta} - 2\boldsymbol{\theta}^T(\mathbf{A}\boldsymbol{\eta}+\beta\mathbf{P}^T\boldsymbol{y})\right] \\ &= \frac{1}{2}\left[\beta\boldsymbol{y}^T\boldsymbol{y} + \boldsymbol{\eta}^T\mathbf{A}\boldsymbol{\eta} - \boldsymbol{m}^T\mathbf{S}^{-1}\boldsymbol{m} + (\boldsymbol{\theta}-\boldsymbol{m})^T\mathbf{S}^{-1}(\boldsymbol{\theta}-\boldsymbol{m})\right] \\ &= E(\boldsymbol{y},\boldsymbol{\eta}) + \frac{1}{2}(\boldsymbol{\theta}-\boldsymbol{m})^T\mathbf{S}^{-1}(\boldsymbol{\theta}-\boldsymbol{m}) \end{aligned} \tag{B.1}$$

Appendix C. Derivation of (A.10)

The derivative of $E(\boldsymbol{y},\boldsymbol{\eta})$ with respect to α_i is given by

$$\begin{aligned} \frac{\partial}{\partial\alpha_i}E(\boldsymbol{y},\boldsymbol{\eta}) =& \frac{1}{2}\frac{\partial}{\partial\alpha_i}(\boldsymbol{\eta}^T\mathbf{A}\boldsymbol{\eta}) - \frac{1}{2}\frac{\partial}{\partial\alpha_i}(\boldsymbol{m}^T\mathbf{S}^{-1}\boldsymbol{m}) \\ =& \frac{1}{2}\eta_i^2 - \frac{1}{2}\frac{\partial}{\partial\alpha_i}(\beta^2\boldsymbol{y}^T\mathbf{P}\mathbf{S}\mathbf{P}^T\boldsymbol{y}) \\ &- \frac{1}{2}\frac{\partial}{\partial\alpha_i}(\boldsymbol{\eta}^T\mathbf{A}^T\mathbf{S}\mathbf{A}\boldsymbol{\eta}) - \frac{\partial}{\partial\alpha_i}(\beta\boldsymbol{y}^T\mathbf{P}\mathbf{S}\mathbf{A}\boldsymbol{\eta}) \end{aligned} \tag{C.1}$$

The derivatives of last three terms in (C.1) with respect to α_i are computed respectively as follows,

$$\begin{aligned} \frac{\partial}{\partial\alpha_i}(\beta^2\boldsymbol{y}^T\mathbf{P}\mathbf{S}\mathbf{P}^T\boldsymbol{y}) =& \mathrm{Tr}\left[\frac{\partial}{\partial\mathbf{S}^{-1}}(\beta^2\boldsymbol{y}^T\mathbf{P}\mathbf{S}\mathbf{P}^T\boldsymbol{y})\frac{\partial\mathbf{S}^{-1}}{\partial\alpha_i}\right] \\ =& -\mathrm{Tr}\left[\mathbf{S}(\beta\mathbf{P}^T\boldsymbol{y})(\beta\mathbf{P}^T\boldsymbol{y})^T\mathbf{S}\mathbf{I}_i\right] \end{aligned} \tag{C.2}$$

$$\begin{aligned} \frac{\partial}{\partial\alpha_i}(\boldsymbol{\eta}^T\mathbf{A}^T\mathbf{S}\mathbf{A}\boldsymbol{\eta}) =& 2\boldsymbol{\eta}^T\frac{\partial\mathbf{A}^T}{\partial\alpha_i}\mathbf{S}\mathbf{A}\boldsymbol{\eta} + \boldsymbol{\eta}^T\mathbf{A}^T\frac{\partial\mathbf{S}}{\partial\alpha_i}\mathbf{A}\boldsymbol{\eta} \\ =& 2\eta_i\mathbf{1}_i^T\mathbf{S}\mathbf{A}\boldsymbol{\eta} + \boldsymbol{\eta}^T\mathbf{A}^T(-\mathbf{S}\mathbf{I}_i\mathbf{S})\mathbf{A}\boldsymbol{\eta} \end{aligned} \tag{C.3}$$

where $\mathbf{1}_i = [0\ldots1\ldots0]^T$ denotes a vector whose ith element is 1 and others are 0.

$$\begin{aligned} \frac{\partial}{\partial\alpha_i}(\beta\boldsymbol{y}^T\mathbf{P}\mathbf{S}\mathbf{A}\boldsymbol{\eta}) =& \beta\boldsymbol{y}^T\mathbf{P}\frac{\partial\mathbf{S}}{\partial\alpha_i}\mathbf{A}\boldsymbol{\eta} + \beta\boldsymbol{y}^T\mathbf{P}\mathbf{S}\frac{\partial\mathbf{A}}{\partial\alpha_i}\boldsymbol{\eta} \\ =& \beta\boldsymbol{y}^T\mathbf{P}(-\mathbf{S}\mathbf{I}_i\mathbf{S})\mathbf{A}\boldsymbol{\eta} + \beta\boldsymbol{y}^T\mathbf{P}\mathbf{S}\eta_i\mathbf{1} \end{aligned} \tag{C.4}$$

Let $\boldsymbol{\zeta} = \mathbf{S}\mathbf{A}\boldsymbol{\eta}$, the derivative of $E(\boldsymbol{y},\boldsymbol{\eta})$ with respect to α_i is written as,

$$\begin{aligned} \frac{\partial}{\partial\alpha_i}E(\boldsymbol{y},\boldsymbol{\eta}) =& \frac{1}{2}\eta_i^2 + \frac{1}{2}\mathrm{Tr}\left[(\boldsymbol{m}-\boldsymbol{\zeta})(\boldsymbol{m}-\boldsymbol{\zeta})^T\mathbf{I}_i\right] - \boldsymbol{m}^T\eta_i\mathbf{1} \\ &+ \frac{1}{2}(2\boldsymbol{m}-\boldsymbol{\zeta})^T\mathbf{I}_i\boldsymbol{\zeta} \\ =& \frac{1}{2}\eta_i^2 + \frac{1}{2}(m_i-\zeta_i)^2 - m_i\eta_i + \frac{1}{2}(2m_i-\zeta_i)\zeta_i \\ =& \frac{1}{2}(m_i-\eta_i)^2 \end{aligned} \tag{C.5}$$

Given that

$$\frac{\partial}{\partial\alpha_i}\ln|\mathbf{S}| = \mathrm{Tr}\left[\mathbf{S}^{-1}\frac{\partial\mathbf{S}}{\partial\alpha_i}\right] = \mathrm{Tr}\left[-\frac{\partial\mathbf{S}^{-1}}{\partial\alpha_i}\mathbf{S}\right] = -S_{ii} \tag{C.6}$$

the derivative of $\ln p(\boldsymbol{y}|\mathbf{P})$ with respect to α_i is obtained as follows,

$$\frac{\partial}{\partial\alpha_i}\ln p(\boldsymbol{y}|\mathbf{P}) = \frac{1}{2\alpha_i} - \frac{1}{2}(m_i-\eta_i)^2 - \frac{1}{2}S_{ii} \tag{C.7}$$

Appendix D. Derivation of (A.11)

The derivative of $E(\boldsymbol{y},\boldsymbol{\eta})$ with respect to β is given by

$$\begin{aligned} \frac{\partial}{\partial\beta}E(\boldsymbol{y},\boldsymbol{\eta}) =& \frac{1}{2}\boldsymbol{y}^T\boldsymbol{y} - \frac{1}{2}\frac{\partial}{\partial\beta}(\boldsymbol{m}^T\mathbf{S}^{-1}\boldsymbol{m}) \\ =& \frac{1}{2}\boldsymbol{y}^T\boldsymbol{y} - \frac{1}{2}\frac{\partial}{\partial\beta}(\beta^2\boldsymbol{y}^T\mathbf{P}\mathbf{S}\mathbf{P}^T\boldsymbol{y}) \\ &- \frac{1}{2}\frac{\partial}{\partial\beta}(\boldsymbol{\eta}^T\mathbf{A}^T\mathbf{S}\mathbf{A}\boldsymbol{\eta}) - \frac{\partial}{\partial\beta}(\beta\boldsymbol{y}^T\mathbf{P}\mathbf{S}\mathbf{A}\boldsymbol{\eta}) \end{aligned} \tag{D.1}$$

The derivatives of last three terms in (D.1) with respect to β are computed respectively as follows,

$$\begin{aligned} \frac{\partial}{\partial\beta}(\beta^2\boldsymbol{y}^T\mathbf{P}\mathbf{S}\mathbf{P}^T\boldsymbol{y}) =& 2\beta\boldsymbol{y}^T\mathbf{P}\mathbf{S}\mathbf{P}^T\boldsymbol{y} + \beta^2\frac{\partial}{\partial\beta}(\boldsymbol{y}^T\mathbf{P}\mathbf{S}\mathbf{P}^T\boldsymbol{y}) \\ =& 2\beta\boldsymbol{y}^T\mathbf{P}\mathbf{S}\mathbf{P}^T\boldsymbol{y} - \beta^2\boldsymbol{y}^T\mathbf{P}\frac{\partial\mathbf{S}}{\partial\beta}\mathbf{P}^T\boldsymbol{y} \end{aligned} \tag{D.2}$$

$$\frac{\partial}{\partial\beta}(\boldsymbol{\eta}^T\mathbf{A}^T\mathbf{S}\mathbf{A}\boldsymbol{\eta}) = \boldsymbol{\eta}^T\mathbf{A}^T\frac{\partial\mathbf{S}}{\partial\beta}\mathbf{A}\boldsymbol{\eta} \tag{D.3}$$

$$\frac{\partial}{\partial\beta}(\beta\boldsymbol{y}^T\mathbf{P}\mathbf{S}\mathbf{A}\boldsymbol{\eta}) = \boldsymbol{y}^T\mathbf{P}\mathbf{S}\mathbf{A}\boldsymbol{\eta} + \beta\boldsymbol{y}^T\mathbf{P}\frac{\partial\mathbf{S}}{\partial\beta}\mathbf{A}\boldsymbol{\eta} \tag{D.4}$$

As $\partial\mathbf{S}/\partial\beta = -\mathbf{S}\mathbf{P}^T\mathbf{P}\mathbf{S}$, the derivative of $E(\boldsymbol{y},\boldsymbol{\eta})$ with respect to β is written as

$$\begin{aligned} \frac{\partial}{\partial\beta}E(\boldsymbol{y},\boldsymbol{\eta}) =& \frac{1}{2}\boldsymbol{y}^T\boldsymbol{y} - \boldsymbol{y}^T\mathbf{P}\boldsymbol{m} + \frac{1}{2}\boldsymbol{m}^T\mathbf{P}^T\mathbf{P}\boldsymbol{m} \\ =& \frac{1}{2}\|\boldsymbol{y}-\mathbf{P}\boldsymbol{m}\|_2^2 \end{aligned} \tag{D.5}$$

Given that

$$\frac{\partial}{\partial\beta}\ln|\mathbf{S}| = \mathrm{Tr}\left[\mathbf{S}^{-1}\frac{\partial\mathbf{S}}{\partial\beta}\right] = \mathrm{Tr}\left[-\frac{\partial\mathbf{S}^{-1}}{\partial\beta}\mathbf{S}\right] = -\mathrm{Tr}\left[\mathbf{P}^{T}\mathbf{P}\mathbf{S}\right] = -\beta^{-1}(\mathbf{I}-\mathbf{A}\mathbf{S}) \tag{D.6}$$

the derivative of $\ln p(\boldsymbol{y}|\mathbf{P})$ with respect to β is obtained as follows,

$$\frac{\partial}{\partial\beta}\ln p(\boldsymbol{y}|\mathbf{P}) = \frac{1}{2\beta}\left[N-\sum_{i=1}^{M}(1-\alpha_i S_{ii})\right] - \frac{1}{2}\|\boldsymbol{y}-\mathbf{P}\boldsymbol{m}\|_2^2 \tag{D.7}$$

References

Afonso, M. V., Bioucas-Dias, J. M., & Figueiredo, M. A. T. (2010). Fast image recovery using variable splitting and constrained optimization. *IEEE Transactions on Image Processing*, *19*(9), 2345–2356.

Ardakani, M., Askarian, M., Shokry, A., Escudero, G., Graells, M., & Espuña, A. (2016). Optimal features selection for designing a fault diagnosis system. In *Computer aided chemical engineering*: *Vol. 38*, (pp. 1111–1116). Elsevier.

Billings, S. A. (2013). *Nonlinear system identification: NARMAX methods in the time, frequency, and spatio-temporal domains*. Chichester, West Sussex, United Kingdom: John Wiley & Sons, Inc.

Boyd, S. (2010). Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends® in Machine Learning*, *3*(1), 1–122.

Chen, S., Billings, S. A., & Luo, W. (1989). Orthogonal least squares methods and their application to non-linear system identification. *International Journal of Control*, *50*(5), 1873–1896.

Dai, S. (2023). Variable selection in convex quantile regression: L1-norm or L0-norm regularization. *European Journal of Operational Research*, *305*(1), 338–355.

Dameshghi, A., & Refan, M. H. (2019). Wind turbine gearbox condition monitoring and fault diagnosis based on multi-sensor information fusion of SCADA and DSER-PSO-WRVM method. *International Journal of Modelling and Simulation*, *39*(1), 48–72.

Eckstein, J., & Bertsekas, D. P. (1992). On the Douglas—Rachford splitting method and the proximal point algorithm for maximal monotone operators. *Mathematical Programming*, *55*(1–3), 293–318.

Herawati, N., Nisa, K., Setiawan, E., Nusyirwan, N., & Tiryono, T. (2018). Regularized multiple regression methods to deal with severe multicollinearity. *International Journal of Statistics and Applications*, *8*(4), 167–172.

Kong, K., Dyer, K., Payne, C., Hamerton, I., & Weaver, P. M. (2023). Progress and trends in damage detection methods, maintenance, and data-driven monitoring of wind turbine blades – A review. *Renewable Energy Focus*, *44*, 390–412.

Li, W., Chen, L., Zhao, J., & Wang, W. (2021). Embedded feature selection based on relevance vector machines with an approximated marginal likelihood and its industrial application. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 1–14.

Li, J., Cheng, K., Wang, S., Morstatter, F., Trevino, R. P., Tang, J., et al. (2018). Feature selection: A data perspective. *ACM Computing Surveys*, *50*(6), 1–45.

Li, X., Lim, B. S., Zhou, J. H., Huang, S., Phua, S. J., Shaw, K. C., et al. (2009). Fuzzy neural network modelling for tool wear estimation in dry milling operation. In *1*: *Vol. 1*, *Proceedings of the annual conference of the PHM society 2009* (pp. 1–11).

Li, Z., Nie, F., Bian, J., Wu, D., & Li, X. (2021). Sparse PCA via L2,p-Norm regularization for unsupervised feature selection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1.

Li, Z., Wang, X., & Kruger, U. (2021). Efficient cross-validatory algorithm for identifying dynamic nonlinear process models. *Control Engineering Practice*, *111*, Article 104787.

Lucke, M., Mei, X., Stief, A., Chioua, M., & Thornhill, N. F. (2019). Variable selection for fault detection and identification based on mutual information of alarm series. *IFAC-PapersOnLine*, *52*(1), 673–678.

MacKay, D. J. C. (1992). Bayesian interpolation. *Neural Computation*, *4*(3), 415–447.

Mei, S., Yuan, M., Cui, J., Dong, S., & Zhao, J. (2022). Machinery condition monitoring in the era of industry 4.0: A relative degree of contribution feature selection and deep residual network combined approach. *Computers & Industrial Engineering*, *168*, Article 108129.

Pan, W., Sootla, A., & Stan, G.-B. (2014). Distributed reconstruction of nonlinear networks: An ADMM approach. *IFAC Proceedings Volumes*, *47*(3), 3208–3213.

Pan, W., Yuan, Y., Gonçalves, J., & Stan, G.-B. (2016). A sparse Bayesian approach to the identification of nonlinear state-space systems. *IEEE Transactions on Automatic Control*, *61*(1), 182–187.

Qin, X., Huang, W., Wang, X., Tang, Z., & Liu, Z. (2022). Real-time remaining useful life prediction of cutting tools using sparse augmented Lagrangian analysis and Gaussian process regression. *Sensors*, *23*(1), 413.

Quattoni, A., Carreras, X., Collins, M., & Darrell, T. (2009). An efficient projection for l _1, _∞ regularization. In *Proceedings of the 26th annual international conference on machine learning* (pp. 857–864). Montreal Quebec Canada: ACM.

Tang, T., Bo, L., Liu, X., Sun, B., & Wei, D. (2018). Variable predictive model class discrimination using novel predictive models and adaptive feature selection for bearing fault identification. *Journal of Sound and Vibration*, *425*, 137–148.

Tang, J., Chai, T., Yu, W., & Zhao, L. (2012). Feature extraction and selection based on vibration spectrum with application to estimating the load parameters of ball mill in grinding process. *Control Engineering Practice*, *20*(10), 991–1004.

Tang, X., Zhang, L., & Li, X. (2018). Bayesian augmented Lagrangian algorithm for system identification. *Systems & Control Letters*, *120*, 9–16.

Tang, X., Zhang, L., & Wang, X. (2019). Sparse augmented Lagrangian algorithm for system identification. *Neurocomputing*, *330*, 403–411.

Tipping, M. E. (2001). Sparse Bayesian learning and the relevance vector machine. *Journal of Machine Learning Research*, *1*(Jun), 211–244.

Yang, D., Karimi, H. R., & Pawelczyk, M. (2023). A new intelligent fault diagnosis framework for rotating machinery based on deep transfer reinforcement learning. *Control Engineering Practice*, *134*, Article 105475.

Yang, Y., Ma, Z., Hauptmann, A. G., & Sebe, N. (2013). Feature selection for multimedia analysis by sharing information among multiple tasks. *IEEE Transactions on Multimedia*, *15*(3), 661–669.

Yang, J., Wang, J., Ye, Q., Xiong, Z., Zhang, F., & Liu, H. (2023). A novel fault detection framework integrated with variable importance analysis for quality-related nonlinear process monitoring. *Control Engineering Practice*, *141*, Article 105733.

Zhang, M., Cao, Y., Sun, Y., & Su, S. (2023). Vibration signal-based defect detection method for railway signal relay using parameter-optimized VMD and ensemble feature selection. *Control Engineering Practice*, *139*, Article 105630.

Zhang, R., Nie, F., Li, X., & Wei, X. (2019). Feature selection with multi-view data: A survey. *Information Fusion*, *50*, 158–167.

Zhao, Z., Wang, L., Liu, H., & Ye, J. (2013). On similarity preserving feature selection. *IEEE Transactions on Knowledge and Data Engineering*, *25*(3), 619–632.

Zhu, K. P., Hong, G. S., & Wong, Y. S. (2008). A comparative study of feature selection for hidden Markov model-based micro-milling tool wear monitoring. *Machining Science and Technology*, *12*(3), 348–369.

Zhu, K., Li, G., & Zhang, Y. (2020). Big data oriented smart tool condition monitoring system. *IEEE Transactions on Industrial Informatics*, *16*(6), 4007–4016.