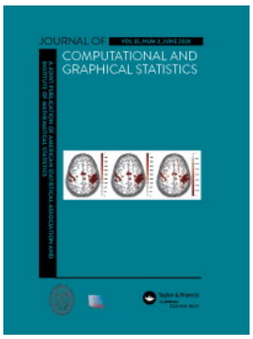

Taylor & Francis
Taylor & Francis Group

Journal of Computational and Graphical Statistics

ISSN: 1061-8600 (Print) 1537-2715 (Online) Journal homepage: www.tandfonline.com/journals/ucgs20

Sparsification Subsampling for Partial Least Squares Regression

Junyi Lin, Mengyu Li, Cheng Meng & Yongdao Zhou

To cite this article: Junyi Lin, Mengyu Li, Cheng Meng & Yongdao Zhou (09 Jun 2026): Sparsification Subsampling for Partial Least Squares Regression, Journal of Computational and Graphical Statistics, DOI: 10.1080/10618600.2026.2686429

To link to this article: https://doi.org/10.1080/10618600.2026.2686429

Sparsification Subsampling for Partial Least Squares Regression

Junyi Lin[a], Mengyu Li[b,*], Cheng Meng[c,†], Yongdao Zhou[d,†]

[a]Institute of Statistics and Big Data, Renmin University of China

[b]Department of Statistics and Data Science, Tsinghua University

[c]Center for Applied Statistics, Institute of Statistics and Big Data, Renmin University of China

[d]NITFID, School of Statistics and Data Science, Nankai University

*Joint first author

[†]Corresponding author, Email: chengmeng@ruc.edu.cn; Email: ydzhou@nankai.edu.cn

Abstract

Partial Least Squares (PLS) regression is a powerful tool for dimensionality reduction, multi-response variable regression, and multi-classification. However, its application to large-scale datasets is often hampered by significant computational burdens. Although variants of PLS algorithms have been proposed to address this challenge, an efficient algorithm that significantly reduces the time complexity remains underexplored. In this paper, we propose a novel PLS regression method based on sparsification subsampling to address this gap. We first construct a sparse sketch matrix using element-wise subsampling to obtain an unbiased approximation of the covariance matrix. Based on this, we propose a core-elements estimator that effectively incorporates the subsampled sparse matrix into the PLS regression. The resulting algorithm employs a kernel-based optimization framework to improve computational efficiency. Theoretically, we demonstrate that under specific probabilities, the approximate matrix obtained by sparsification subsampling achieves minimal variance for any given projection matrix. Finally, a series of Monte Carlo simulations and real-world data analyses validate the superior performance of our approach in terms of estimation accuracy, prediction power, and computational efficiency. Compared to row-wise sampling methods and even full-sample PLS, the proposed algorithm exhibits significant advantages.

Keywords: Multi-response regression, element-wise sampling, matrix sparsification, large-scale data, approximate matrix multiplication

1 Introduction

Partial Least Squares (PLS) regression (Wold, 1966; Höskuldsson, 1988; Cook and Forzani, 2024) is a multivariate statistical method well-suited for analyzing high-dimensional data, particularly in the presence of multiple response variables. Unlike classical linear regression, PLS regression assumes that the observed data can be represented by low-dimensional latent variables, which are iteratively identified as linear combinations of predictors that are most strongly correlated with the responses. By constructing these latent variables over a limited number of iterations, PLS regression achieves dimensionality reduction, effectively mitigating the issue of multicollinearity (Wold et al., 1984) often encountered in high-dimensional data.

Although PLS was originally designed for high-dimensional data with relatively small sample sizes, its computational efficiency has become increasingly important in the era of big data. Given a sample with n observations and p features, early implementations employed the computationally intensive power iteration method (Wold, 1966), and subsequent improvements have brought the time complexity down to $\mathcal{O}(np^2)$ (Lindgren et al., 1993; Dayal and MacGregor, 1997; De Jong, 1993), which is on the same order as ordinary least squares regression. Nevertheless, when the number of observations n becomes very large, the sheer volume of data can still pose a significant computational challenge. Various methods have been proposed to address this issue, including online PLS for streaming data (Zeng and Li, 2014; Arora et al., 2016; Stott et al., 2017; Jordao et al., 2021), acceleration using tree-based discriminant structures (Schwartz et al., 2010), semantic compression (Tabei et al., 2016), and parallelization through matrix partitioning techniques (Lafaye de Micheaux et al., 2019). Despite these advancements, many existing methods lack rigorous time complexity analyses or are only effective for specialized data structures.

Subsampling algorithms have emerged as a powerful tool for reducing the computational costs in large-scale statistical problems, including regression. These methods can be broadly categorized into probabilistic and deterministic approaches. Probabilistic methods often employ leverage scores (Drineas et al., 2006; Ma et al., 2014; Ma and Sun, 2015), influence functions (Ting and Brochu, 2018), or various optimality criteria (Ma et al., 2022) to design sampling probabilities. Deterministic approaches include Information-Based Optimal Subsampling Selection (IBOSS) (Wang et al., 2019; Yu et al., 2023), model-free optimal subsampling based on space-filling designs (Meng et al., 2021; Yi and Zhou, 2023), optimal approximate orthogonal arrays (Wang et al., 2021), predictive inference-based subsampling (Wu et al., 2024), and uniform design-based methods (Zhang et al., 2024; Zhou et al., 2024). The choice between probabilistic and deterministic approaches often depends on specific model characteristics and parameters. Beyond improving computational efficiency, effective subsampling can also address challenges related to measurement constraints (Wang et al., 2017; Meng et al., 2021) and privacy protection (Wang et al., 2019).

Subsampling algorithms have been widely adopted in various statistical learning tasks, including least squares regression (Drineas et al., 2006; Ma et al., 2014; Ma and Sun, 2015; Ting and

Brochu, 2018; Meng et al., 2017; Wang et al., 2019, 2021; Ma et al., 2022), generalized linear models (Ai et al., 2021; Wang et al., 2018; Yu et al., 2022, 2025), nonparametric regression (Ma et al., 2015; Meng et al., 2020, 2022; Zhang et al., 2024), quantile regression (Ai et al., 2021; Wang and Ma, 2021), model-free methods (Meng et al., 2021; Yi and Zhou, 2023; Zhang et al., 2024; Zhou et al., 2024), time series analysis (Xie et al., 2019, 2023), machine learning (Han et al., 2025; Feng and Zhou, 2024; Huang et al., 2026). We also refer readers to Li and Meng (2021) and Yu et al. (2024) for a comprehensive overview. In the context of PLS regression, Xie et al. (2022) and Huang et al. (2024) proposed subsampling methods based on influence functions (Ting and Brochu, 2018) and minimum covariance determinant to remove outliers and improve robustness. However, research on optimizing or improving the computational efficiency of the PLS algorithm from a time complexity perspective remains limited. This gap motivates us to explore new subsampling methods that reduce the computational burden of PLS while preserving its accuracy.

Beyond row sampling, another approach involves sampling individual elements of the observation matrix. As illustrated in Fig. 1, rather than discarding entire rows, sparsification subsampling preserves the most crucial information within each sample. Moreover, Li et al. (2024) demonstrated that element-wise sampling achieves acceleration similar to row-wise sampling when extracting an equal number of elements. Regardless of the specific sampling strategy, the primary concern is the variance properties of the resulting parameter estimates. In PLS regression, the computation and updating of the kernel matrix are central to the algorithm (Lindgren et al., 1993; Dayal and MacGregor, 1997). This motivates us to seek a kernel estimation method that is both faster and exhibits lower variance. Notably, fast algorithms employing sparsification have been successfully applied in diverse fields such as low rank matrix approximations (Achlioptas and Mcsherry, 2007; Xue et al., 2026), principal component analysis (Kundu et al., 2017), linear regression (Li et al., 2024), nonparametric additive models (Li et al., 2024), and optimal transport (Li et al., 2023; Hu et al., 2025; Li et al., 2023; Ouyang et al., 2026). Inspired by the Core-Elements approach (Li et al., 2024), we incorporate a sparse matrix as an auxiliary variable into the original PLS algorithm and propose a novel kernel algorithm based on sparsification to reduce the computational complexity of PLS regression.

Our contributions are summarized as follows. First, we develop a new element-wise sampling method to approximate the covariance matrix, demonstrating its minimal variance under specific sampling probabilities. Moreover, this minimal variance property holds for any projection of the covariance matrix, offering a theoretical advantage over existing methods.

Second, we propose a Sparsification Subsampling for Partial Least Squares (Spar-PLS) algorithm that uses the sparse matrix obtained through our sampling method. We further enhance this algorithm by incorporating established kernel methods proposed by Lindgren et al. (1993) and Dayal and MacGregor (1997).

Finally, through extensive Monte Carlo simulations, we evaluate the performance of our algorithm in terms of parameter estimation, prediction accuracy, and computational time. Our results show that the Spar-PLS method achieves comparable or superior estimation and prediction performance while reducing the running time by 80%-90%. Furthermore, we evaluate

the performance of our algorithm on a wave energy converter dataset, where it consistently outperforms row sampling methods across various scenarios.

The remainder of this paper is organized as follows. Section 2 provides background on the PLS regression algorithms, matrix multiplication approximation algorithms, and element-wise sampling methods for least squares regression. Section 3 details the optimal sampling probability settings and introduces our proposed Spar-PLS method. Sections 4 and 5 present the results of numerical simulations and empirical data analyses, respectively, to evaluate the effectiveness and efficiency of our algorithm. Finally, Section 6 concludes the paper by summarizing our findings, discussing limitations, and suggesting directions for future research. Supplementary materials include notation, technical proofs, additional implementation details, and extended numerical experiments. The implementation code for the proposed method is available at the following link: https://github.com/junyilin559/Spar-PLS.

2 Background

We introduce some notation and definitions. The Euclidean norm (or ℓ_2 norm) of a vector $\boldsymbol{x}$ is defined as $\|\boldsymbol{x}\| = (\sum_i \boldsymbol{x}_i^2)^{1/2}$ (or $\|\boldsymbol{x}\|_2$). For a matrix $\mathbf{X} \in \mathbb{R}^{n\times p}$, let $\boldsymbol{x}_i$ ($i = 1, \ldots, p$) denote the i-th column of $\mathbf{X}$ as a column vector, and let $\boldsymbol{x}_{(j)}$ ($j = 1, \ldots, n$) denote the j-th row of $\mathbf{X}$ as a column vector, its spectral norm (i.e., the largest singular value) and the Frobenius norm are denoted as $\|\mathbf{X}\|_2$ and $\|\mathbf{X}\|_F$, respectively.

2.1 Partial Least Squares

Partial Least Squares (PLS) regression is primarily employed to analyze data with multiple response variables. Consider a set of independently and identically distributed data $\{(\boldsymbol{x}_{(j)}, \boldsymbol{y}_{(j)})\}_{j=1}^n \subseteq \mathbb{R}^p \times \mathbb{R}^l$, where $\{\boldsymbol{x}_{(j)}\}_{j=1}^n \subset \mathbb{R}^p$ represent the predictor variables and $\{\boldsymbol{y}_{(j)}\}_{j=1}^n \subseteq \mathbb{R}^l$ represent the response variables. We denote the predictor matrix as $\mathbf{X} = (\boldsymbol{x}_{(1)}, \ldots, \boldsymbol{x}_{(n)})^\top \in \mathbb{R}^{n\times p}$ and the response matrix as $\mathbf{Y} = (\boldsymbol{y}_{(1)}, \ldots, \boldsymbol{y}_{(n)})^\top \in \mathbb{R}^{n\times l}$. PLS regression is particularly suitable for datasets where the predictors exhibit multicollinearity, making $\mathbf{X}$ be of low rank.

To identify the low-rank structure of $\mathbf{X}$ for predicting the response variables, PLS introduces a latent variable matrix $\mathbf{T} = (\boldsymbol{t}_1, \ldots, \boldsymbol{t}_m) \in \mathbb{R}^{n\times m} (m \le p)$ and considers the following regression model:

$$\begin{cases} \mathbf{X} = \mathbf{T}\mathbf{P}^\top + \mathbf{E}, \\ \mathbf{Y} = \mathbf{T}\mathbf{Q}^\top + \mathbf{F}, \end{cases} \quad (1)$$

where $\mathbf{P} \in \mathbb{R}^{p\times m}$ and $\mathbf{Q} \in \mathbb{R}^{l\times m}$ are the loading matrices, and $\mathbf{E} \in \mathbb{R}^{n\times p}$ and $\mathbf{F} \in \mathbb{R}^{n\times l}$ are the residual matrices. To estimate $\mathbf{T}$ and the associated loading matrices, we initially assume its first

component as $\boldsymbol{t}_1 = \mathbf{X}\boldsymbol{v}_1$ and $\boldsymbol{t}_1 = \mathbf{Y}\boldsymbol{w}_1$. However, if $\mathbf{Y}$ is not an exact linear combination of $\mathbf{X}$, these two equations cannot be simultaneously satisfied. Consequently, we seek an approximate $\boldsymbol{t}_1$ that optimally satisfies these conditions by maximizing the covariance between the latent components derived from $\mathbf{X}$ and $\mathbf{Y}$. For clarity in the subsequent steps, we define $\mathbf{X}_1 = \mathbf{X}$ and $\mathbf{Y}_1 = \mathbf{Y}$.

Considering the two expressions for the first latent variable, $\boldsymbol{t}_1$, namely $\mathbf{X}_1\boldsymbol{v}_1$ and $\mathbf{Y}_1\boldsymbol{w}_1$, the objective is to identify $\boldsymbol{v}_1$ and $\boldsymbol{w}_1$ that maximize the inner product of these two vectors, i.e.,

$$\boldsymbol{v}_1, \boldsymbol{w}_1 = \underset{\boldsymbol{v}_1 \in \mathbb{R}^p, \boldsymbol{w}_1 \in \mathbb{R}^l}{\arg\max} \left\{ (\mathbf{X}_1\boldsymbol{v}_1)^\top \mathbf{Y}_1\boldsymbol{w}_1 \right\} \quad \text{s.t. } \boldsymbol{v}_1^\top \boldsymbol{v}_1 = 1, \boldsymbol{w}_1^\top \boldsymbol{w}_1 = 1. \quad (2)$$

Setting $\boldsymbol{t}_1 = \mathbf{X}_1\boldsymbol{v}_1$, we can then decompose $\mathbf{X}_1$ and $\mathbf{Y}_1$ as follows:

$$\begin{cases} \mathbf{X}_1 = \boldsymbol{t}_1 \boldsymbol{p}_1^\top + \mathbf{X}_2, \\ \mathbf{Y}_1 = \boldsymbol{t}_1 \boldsymbol{q}_1^\top + \mathbf{Y}_2, \end{cases} \quad (3)$$

where the loading vectors $\boldsymbol{p}_1$ and $\boldsymbol{q}_1$ are estimated via least squares:

$$\boldsymbol{p}_1^\top = (\boldsymbol{t}_1^\top \boldsymbol{t}_1)^{-1} \boldsymbol{t}_1^\top \mathbf{X}_1, \quad \boldsymbol{q}_1^\top = (\boldsymbol{t}_1^\top \boldsymbol{t}_1)^{-1} \boldsymbol{t}_1^\top \mathbf{Y}_1. \quad (4)$$

For the residual matrices in (3), defined as $\mathbf{X}_2 = \mathbf{X}_1 - \boldsymbol{t}_1 \boldsymbol{p}_1^\top$ and $\mathbf{Y}_2 = \mathbf{Y}_1 - \boldsymbol{t}_1 \boldsymbol{q}_1^\top$, the same procedure is applied iteratively to obtain $\boldsymbol{t}_2$, $\boldsymbol{p}_2$, $\boldsymbol{q}_2$, $\mathbf{X}_3$, and $\mathbf{Y}_3$, and so forth. This iterative process is repeated *m* times. Consequently, we obtain the decompositions $\mathbf{X} = \boldsymbol{t}_1 \boldsymbol{p}_1^\top + \cdots + \boldsymbol{t}_m \boldsymbol{p}_m^\top + \mathbf{X}_{m+1} = \mathbf{T}\mathbf{P}^\top + \mathbf{E}$ and $\mathbf{Y} = \boldsymbol{t}_1 \boldsymbol{q}_1^\top + \cdots + \boldsymbol{t}_m \boldsymbol{q}_m^\top + \mathbf{Y}_{m+1} = \mathbf{T}\mathbf{Q}^\top + \mathbf{F}$. This entire process is summarized in Algorithm 1.

Algorithm 1: Partial Least Squares

Input: $\mathbf{X}, \mathbf{Y}$

1 Initialize: $\mathbf{X}_1 = \mathbf{X}$ and $\mathbf{Y}_1 = \mathbf{Y}$.

2 **for** $k \leftarrow 1$ *to m* **do**

3 Perform SVD on $\mathbf{X}_k^\top \mathbf{Y}_k$, and extract the left and right singular vectors $\boldsymbol{v}_k$ and $\boldsymbol{w}_k$, respectively, corresponding to the largest singular value. $\mathcal{O}(npl)$

4 Compute the latent variable: $\boldsymbol{t}_k = \mathbf{X}_k \boldsymbol{v}_k$. $\mathcal{O}(np)$

5 Derive the OLS estimators: $\boldsymbol{p}_k^\top = (\boldsymbol{t}_k^\top \boldsymbol{t}_k)^{-1} \boldsymbol{t}_k^\top \mathbf{X}_k, \quad \boldsymbol{q}_k^\top = (\boldsymbol{t}_k^\top \boldsymbol{t}_k)^{-1} \boldsymbol{t}_k^\top \mathbf{Y}_k.$ $\mathcal{O}(np)$

6 Update the residual matrices: $\mathbf{X}_{k+1} = \mathbf{X}_k - \boldsymbol{t}_k \boldsymbol{p}_k^\top$, $\mathbf{Y}_{k+1} = \mathbf{Y}_k - \boldsymbol{t}_k \boldsymbol{q}_k^\top$. $\mathcal{O}(np)$

7 **end**

Output: $\mathbf{T}, \mathbf{P}, \mathbf{Q}$

Efficient kernel-based methods. Partial Least Squares was initially introduced by Wold (1966) via the NIPALS algorithm, which uses power iteration to solve for $\boldsymbol{v}_k$ and $\boldsymbol{w}_k$ in (2). Höskuldsson (1988) further elucidated the properties of this algorithm. The NIPALS algorithm incurs a substantial computational cost. For simplicity, it is assumed that $l < p$. Assuming an average of *a* iterations to solve (2), the overall time complexity of the algorithm is $\mathcal{O}(npma)$. Alternatively, Algorithm 1 performs a SVD on $\mathbf{X}^\top\mathbf{Y}$ to solve (2), which also results in a high complexity of $\mathcal{O}(nplm)$. To address this, Lindgren et al. (1993) proposed a kernel-based approach that computes the covariance matrices $\mathbf{X}^\top\mathbf{X}$ and $\mathbf{X}^\top\mathbf{Y}$ only once, bypassing the high computational burden of the power iteration method and repetitive covariance computation. By substituting the kernel into (2)–(4), the computational cost of these steps is reduced from $\mathcal{O}(npa)$ or $\mathcal{O}(npl)$ to $\mathcal{O}(p^2l)$. For large-scale datasets, the time complexity of the kernel method is dominated by the kernel computation, which is $\mathcal{O}(np^2)$.

Dayal and MacGregor (1997) further enhanced this kernel method by precomputing the covariance matrices and only updating $(\mathbf{X}^\top\mathbf{Y})_k = \mathbf{X}_k^\top\mathbf{Y}_k$ during iterations. The improved kernel method, summarized in Algorithm 2, forms the foundation for the sparsification-based PLS algorithm proposed in this paper.

Algorithm 2: Kernel Partial Least Squares

Input: $\mathbf{X}, \mathbf{Y}$

1 Compute the kernel matrices: $\mathbf{X}^\top\mathbf{X}, \mathbf{X}^\top\mathbf{Y}$ $\mathcal{O}(np^2)$

2 **for** $k \leftarrow 1$ *to m* **do**

3 Perform an SVD on $(\mathbf{X}^\top\mathbf{Y})_k$ and obtain the left and right singular vectors $\boldsymbol{v}_k$ and $\boldsymbol{w}_k$, respectively, corresponding to the largest singular value. $\mathcal{O}(p^2l)$

4 Compute $\boldsymbol{r}_k$: $\begin{cases} \boldsymbol{r}_1 = \boldsymbol{v}_1, \\ \boldsymbol{r}_k = \boldsymbol{v}_k - \sum_{j=1}^{k-1} \boldsymbol{p}_j^\top \boldsymbol{v}_k \boldsymbol{r}_j, \quad k > 1 \end{cases}$ $\mathcal{O}(p^2)$

5 Obtain the OLS estimators: $\boldsymbol{p}_k^\top = \frac{\boldsymbol{r}_k^\top(\mathbf{X}^\top\mathbf{X})_k}{\boldsymbol{r}_k^\top(\mathbf{X}^\top\mathbf{X})\boldsymbol{r}_k}$, $\boldsymbol{q}_k^\top = \frac{\boldsymbol{r}_k^\top(\mathbf{X}^\top\mathbf{Y})_k}{\boldsymbol{r}_k^\top(\mathbf{X}^\top\mathbf{X})\boldsymbol{r}_k}$ $\mathcal{O}(p^2)$

6 Update the kernel matrix: $(\mathbf{X}^\top\mathbf{Y})_{k+1} = (\mathbf{X}^\top\mathbf{Y})_k - \boldsymbol{p}_k\boldsymbol{q}_k^\top(\boldsymbol{t}_k^\top\boldsymbol{t}_k)$ $\mathcal{O}(pl)$

7 **end**

8 Compute the regression coefficient matrix: $\mathbf{B} = \mathbf{R}\mathbf{Q}^{\top}$ $\mathcal{O}(mpl)$

Output: $\mathbf{T}, \mathbf{P}, \mathbf{Q}, \mathbf{B}$

Besides these kernel approaches, other notable PLS algorithms include SIMPLS (De Jong, 1993), OPLS (Trygg and Wold, 2002), Envelope methods (Cook et al., 2013), 3PRF (Kelly and Pruitt, 2015), IDAR (Stocchero, 2019), among others. These methods address various challenges in PLS but remain computationally intensive for large-scale data, highlighting the need for more efficient variants.

2.2 Approximating Matrix Multiplication

As shown in Algorithm 2, the computational bottleneck lies in the initial matrix multiplications. Accelerating these operations can significantly reduce the overall time complexity. For general matrices $\mathbf{A}$ and $\mathbf{B}$, a widely used approach is to subsample rows or columns from the matrices to approximate the matrix multiplication (Drineas et al., 2006; Mahoney, 2016). A standard algorithm for this approximation is presented in Algorithm 3.

Algorithm 3: Basic Matrix Multiplication Approximation Algorithm

Input: Matrices $\mathbf{A} \in \mathbb{R}^{m\times n}, \mathbf{B} \in \mathbb{R}^{n\times l}$; sampling probabilities $\{p_i\}_{i=1}^{n}$.

1 **for** $t \leftarrow 1$ *to* c **do**

2 Sample an index $i_t \in \{1,\ldots,n\}$ with replacement, according to the probability distribution $\Pr[i_t = k] = p_k$.

3 Set $\boldsymbol{c}_t = \boldsymbol{a}_{i_t} / \sqrt{cp_{i_t}}, \boldsymbol{r}_{(t)} = \boldsymbol{b}_{(i_t)} / \sqrt{cp_{i_t}}$

4 **end** $\mathcal{O}(n + cm + cl)$

Output: Matrix multiplication $\mathbf{CR}$ such that $\mathbf{CR} \approx \mathbf{AB}$, where $\mathbf{C} \in \mathbb{R}^{m\times c}$ and $\mathbf{R} \in \mathbb{R}^{c\times l}$ $\mathcal{O}(cml)$

In Algorithm 3, rows (or columns) are sampled based on a predefined probability distribution, and inverse probability weighting is applied to ensure that the resulting matrix product approximates the original product effectively. The quality of this approximation is characterized by the following lemmas.

Lemma 1 .

Given matrices $\mathbf{A}$ *and* $\mathbf{B}$*, the matrices* $\mathbf{C}$ *and* $\mathbf{R}$ *constructed by Algorithm 3 satisfy*

$$\mathbb{E}\left[(\mathbf{CR})_{ij}\right] = (\mathbf{AB})_{ij},$$

$$\mathrm{Var}\left[(\mathbf{CR})_{ij}\right] = \frac{1}{c}\sum_{k=1}^{n}\frac{\mathbf{A}_{ik}^2\mathbf{B}_{kj}^2}{p_k} - \frac{1}{c}(\mathbf{AB})_{ij}^2.$$

Lemma 1 indicates that $\mathbf{CR}$ is an unbiased estimator of $\mathbf{AB}$, with its variance affected by the sampling probabilities in Algorithm 3. Consequently, adjusting these sampling probabilities directly influences the accuracy of the approximation. To further analyze this, Drineas et al. (2006) consider the sum of the element-wise variances, which is equivalent to the expected squared Frobenius norm of the approximation error, denoted by $\mathbb{E}[\|\mathbf{AB}-\mathbf{CR}\|_F^2]$. The following lemma quantifies this relationship.

Lemma 2 .

Given matrices $\mathbf{A}$ *and* $\mathbf{B}$, *the matrices* $\mathbf{C}$ *and* $\mathbf{R}$ *constructed by Algorithm 3 satisfy*

$$\mathbb{E}\left[\|\mathbf{AB}-\mathbf{CR}\|_F^2\right] = \sum_{k=1}^{n}\frac{\|\boldsymbol{a}_k\|_2^2\left\|\boldsymbol{b}_{(k)}\right\|_2^2}{cp_k} - \frac{1}{c}\|\mathbf{AB}\|_F^2.$$

When the sampling probability is set to $p_k = \dfrac{\|\boldsymbol{a}_k\|_2\left\|\boldsymbol{b}_{(k)}\right\|_2}{\sum_{k'=1}^{n}\|\boldsymbol{a}_{k'}\|_2\left\|\boldsymbol{b}_{(k')}\right\|_2}$, $\mathbb{E}[\|\mathbf{AB}-\mathbf{CR}\|_F^2]$ *attains its*

minimum value

$$\mathbb{E}\left[\|\mathbf{AB}-\mathbf{CR}\|_F^2\right] = \frac{1}{c}\left(\sum_{k=1}^{n}\|\boldsymbol{a}_k\|_2\left\|\boldsymbol{b}_{(k)}\right\|_2\right)^2 - \frac{1}{c}\|\mathbf{AB}\|_F^2. \quad (5)$$

Proofs of Lemma 1 and Lemma 2 can be found in Drineas et al. (2006) and Mahoney (2016). In contrast to row (or column) sampling, element-wise sampling provides an alternative approach that focuses on directly approximating the original matrix itself, rather than the resulting product.

Algorithm 4: Element-wise Matrix Approximation Algorithm

Input: Matrix $\mathbf{A}_{m\times n}$, sampling probabilities $\{p_{ij}\}_{i,j=1}^{m,n}$.

1 **for** $i \leftarrow 1$ *to m* **do**

2 **for** $j \leftarrow 1$ *to n* **do**

3 Independently set $\mathbf{A}_{ij}^*$ to $\dfrac{\mathbf{A}_{ij}}{p_{ij}}$ with probability p_{ij}, and to 0 otherwise.

4 **end**

5 **end** $\mathcal{O}(\text{nnz}(\mathbf{A}))$

Output: Matrix $\mathbf{A}^*_{m\times n}$

The operator nnz denotes the number of non-zero elements in a matrix. By carefully selecting the probabilities p_{ij}, Algorithm 4 can ensure that $\|\mathbf{A}^* - \mathbf{A}\|_2$ is effectively controlled (Braverman et al., 2021). Applying Algorithm 4 yields sketches $\mathbf{A}^*$ and $\mathbf{B}^*$ of $\mathbf{A}$ and $\mathbf{B}$, respectively, such that their product $\mathbf{A}^*\mathbf{B}^*$ is an unbiased estimator of $\mathbf{AB}$. As shown in Drineas et al. (2006), specific choices of probability values can further provide an upper bound on the variance of $\mathbf{A}^*\mathbf{B}^*$. However, identifying the optimal probabilities that minimize the variance of this estimator remains an open question.

In Section 3, we extend the fundamental idea of Algorithm 4 to approximate matrix multiplications, aiming to inherit some of the desirable properties exhibited by Algorithm 3 while addressing its limitations.

2.3 Core-Elements Estimator for Linear Models

Following the notation in Model (1), consider a general linear regression model:

$$\mathbf{Y} = \mathbf{XB} + \mathbf{E}, \quad (6)$$

the ordinary least squares estimator for $\mathbf{B}$ is given by:

$$\mathbf{B} = \left(\mathbf{X}^\top\mathbf{X}\right)^{-1}\mathbf{X}^\top\mathbf{Y}, \quad (7)$$

which is analogous to the estimator used in (4). For large-scale datasets, the computational cost of the OLS estimator is dominated by the computation of $\mathbf{X}^\top\mathbf{X}$ and $\mathbf{X}^\top\mathbf{Y}$, with a time complexity of $\mathcal{O}(np^2 + npl)$. To mitigate this, classical row subsampling techniques reduce the time complexity to $\mathcal{O}(rp^2 + rpl)$ by selecting only *r* rows of data, as outlined in Algorithm 3.

Beyond row subsampling, element-wise subsampling offers an alternative for reducing computational complexity. Given the predictor matrix $\mathbf{X}$, we can obtain a sparse sketch $\mathbf{X}^*$ using Algorithm 4. If $\mathbf{X}^*$ contains *rp* nonzero elements (analogous to row subsampling), the time complexity for computing $\mathbf{X}^{*\top}\mathbf{X}$ and $\mathbf{X}^{*\top}\mathbf{Y}$ can also be reduced to $\mathcal{O}(rp^2 + rpl)$.

With this sparse sketch $\mathbf{X}^*$, Li et al. (2024) proposed the Core-Elements estimator, which takes the form:

$$\mathbf{B} = \left(\mathbf{X}^{*\top}\mathbf{X}\right)^{-1}\mathbf{X}^{*\top}\mathbf{Y}. \quad (8)$$

A key advantage of this estimator is its unbiasedness. Under the assumptions in (6), and assuming $\mathbf{X}^{*\top}\mathbf{X}$ is invertible, the expectation of $\mathbf{B}$ is:

$$\mathbb{E}(\mathbf{B}) = \mathbb{E}\left(\left(\mathbf{X}^{*\top}\mathbf{X}\right)^{-1}\mathbf{X}^{*\top}\mathbf{Y}\right) = \left(\mathbf{X}^{*\top}\mathbf{X}\right)^{-1}\mathbf{X}^{*\top}\mathbb{E}(\mathbf{Y}) = \left(\mathbf{X}^{*\top}\mathbf{X}\right)^{-1}\mathbf{X}^{*\top}\mathbf{X}\mathbf{B} = \mathbf{B}.$$

Therefore, $\mathbf{B}$ is an unbiased estimator of $\mathbf{B}$. Notably, this unbiasedness does not impose any restrictions on the choice of $\mathbf{X}^*$. Any matrix $\mathbf{X}^*$, not just those derived from sparsification schemes, will lead to an unbiased estimator $\mathbf{B}$. This flexibility significantly broadens the applicability and versatility of the Core-Elements estimator.

3 Methods

This section introduces the core algorithm of this paper. First, we present a matrix multiplication approximation method based on element-wise subsampling, along with optimal sampling probabilities to minimize estimation variance. Subsequently, we integrate this sparse sketch into the Partial Least Squares regression algorithm, reducing its computational complexity. Finally, we derive the regression coefficients, and propose the improved sparsification algorithm.

3.1 Element-wise Approximation for Matrix Multiplication

To approximate the matrix product $\mathbf{AB}$, where $\mathbf{A} \in \mathbb{R}^{m\times n}$ and $\mathbf{B} \in \mathbb{R}^{n\times l}$, unlike prior work (Drineas et al., 2006), which sparsifies both matrices, we only sparsify $\mathbf{A}$ while retaining the full matrix $\mathbf{B}$ inspired by the Core-Elements estimator in (8). This approach effectively reduces time complexity and has favorable approximation properties.

Specifically, we use Algorithm 4 to construct a sparse matrix $\mathbf{A}^*$ and approximate $\mathbf{AB}$ with $\mathbf{A}^*\mathbf{B}$. Given that $\sum_{i,j} p_{ij} = mc$, where mc denotes the expected number of nonzero elements in $\mathbf{A}^*$, the time complexity of computing $\mathbf{A}^*\mathbf{B}$ is $\mathcal{O}(cml)$, consistent with that of row-wise subsampling in Algorithm 3. Similar to Lemma 1 and Lemma 2, the following theorems detail the approximation properties of $\mathbf{A}^*\mathbf{B}$.

Theorem 1 (Unbiasedness and Variance).

Given matrices $\mathbf{A}$ *and* $\mathbf{B}$, *the matrix* $\mathbf{A}^*$ *obtained from Algorithm 4 satisfies*

$$\mathbb{E}\left[\left(\mathbf{A}^*\mathbf{B}\right)_{ij}\right] = (\mathbf{AB})_{ij},$$

$$\mathrm{Var}\left[\left(\mathbf{A}^*\mathbf{B}\right)_{ij}\right] = \sum_{k=1}^{n}\left(\left(\frac{1}{p_{ik}} - 1\right)\mathbf{A}_{ik}^2\mathbf{B}_{kj}^2\right).$$

Theorem 2 (Expected Approximation Error).

Given matrices $\mathbf{A}$ *and* $\mathbf{B}$*, the expected Frobenius norm of the error in approximating* $\mathbf{AB}$ *using* $\mathbf{A}^*\mathbf{B}$ *is*

$$\mathbb{E}\left[\left\|\mathbf{A}^*\mathbf{B}-\mathbf{AB}\right\|_F^2\right]=\sum_{i=1}^{m}\sum_{k=1}^{n}\left(\left(\frac{1}{p_{ik}}-1\right)\mathbf{A}_{ik}^2\left\|\boldsymbol{b}_{(k)}\right\|_2^2\right).$$

When the sampling probability is set to $p_{ik}\propto|\mathbf{A}_{ik}|\left\|\boldsymbol{b}_{(k)}\right\|_2$ *attains its minimum value*

$$\mathbb{E}\left[\left\|\mathbf{A}^*\mathbf{B}-\mathbf{AB}\right\|_F^2\right]=\frac{1}{mc}\left(\sum_{i=1}^{m}\sum_{k=1}^{n}|\mathbf{A}_{ik}|\left\|\boldsymbol{b}_{(k)}\right\|_2\right)^2-\sum_{i=1}^{m}\sum_{k=1}^{n}\left(\mathbf{A}_{ik}^2\left\|\boldsymbol{b}_{(k)}\right\|_2^2\right),$$

where $\sum_{i=1}^{m}\sum_{k=1}^{n}p_{ik}=mc$.

Remark 1 .

Algorithm 4 can achieve variance reduction comparable to that of Algorithm 3. Specifically, when the element-wise sampling probability p_{ij} *for each row in Algorithm 4 is set to the row sampling probability* p_i *in Algorithm 3, it can be shown that*

$$\mathbb{E}[\left\|\mathbf{A}^*\mathbf{B}-\mathbf{AB}\right\|_F^2]=\sum_{k=1}^{n}\frac{\left\|\boldsymbol{a}_k\right\|_2^2\left\|\boldsymbol{b}_{(k)}\right\|_2^2}{p_k}-\sum_{i=1}^{m}\sum_{k=1}^{n}\left(\mathbf{A}_{ik}^2\left\|\boldsymbol{b}_{(k)}\right\|_2^2\right),$$

and it attains its minimum value

$$\mathbb{E}[\left\|\mathbf{A}^*\mathbf{B}-\mathbf{AB}\right\|_F^2]=\frac{1}{c}\left(\sum_{k=1}^{n}\left\|\boldsymbol{a}_k\right\|_2\left\|\boldsymbol{b}_{(k)}\right\|_2\right)^2-\left\|\mathbf{AB}\right\|_F^2, \quad (9)$$

where $p_k\propto\left\|\boldsymbol{a}_k\right\|_2\left\|\boldsymbol{b}_{(k)}\right\|_2$ *and* $\sum_{k=1}^{n}p_k=c$. *Notably, when* $c=1$, *(9) is equivalent to (5). As c increases, (9) decreases faster than (5) because only the first term in (9) is inversely proportional to c. Furthermore, (9) can be viewed as a special case of Theorem 2 when the constraint* $p_{1k}=p_{2k}=\cdots=p_{mk},\forall k$ *is imposed on the sampling probabilities in Theorem 2. By the principles of constrained optimization, we have*

$$\mathbb{E}\left[\left\|\mathbf{A}^*\mathbf{B}-\mathbf{AB}\right\|_F^2\right]\le\mathbb{E}\left[\left\|\mathbf{AB}-\mathbf{CR}\right\|_F^2\right],$$

where $\mathbf{A}^*$ *is the matrix obtained from Algorithm 4 with the optimal probability in Theorem 2, and* $\mathbf{C}$ *and* $\mathbf{R}$ *are the matrices obtained from Algorithm 3 with the optimal probability in*

Lemma 2. This implies that the approximation $\mathbf{A}^*\mathbf{B}$ *obtained via element-wise sampling in Algorithm 4 and Theorem 2 will generally exhibit a lower variance compared to* $\mathbf{CR}$ *obtained from row sampling in Algorithm 3 and Lemma 2.*

Theorems 1 and 2 demonstrate that element-wise sampling approximation from Algorithm 4 preserves the unbiasedness and generally has lower variance than the row-sampling-based approximation. Furthermore, by introducing projection transformations to the matrices involved in the multiplication, we can derive an even stronger result.

Theorem 3 .

Given matrices $\mathbf{A}$ *and* $\mathbf{B}$ *, and any projection matrices* $\mathbf{S}_1$ *and* $\mathbf{S}_2$ *of appropriate dimensions, consider the approximation of the matrix* $\mathbf{S}_1^\top \mathbf{A}\mathbf{B}\mathbf{S}_2$ *. The* $\mathbf{A}^*$ *obtained by Algorithm 4 satisfies:*

$$\mathbb{E}\left[\left\|\mathbf{S}_1^\top \mathbf{A}^*\mathbf{B}\mathbf{S}_2 - \mathbf{S}_1^\top \mathbf{A}\mathbf{B}\mathbf{S}_2\right\|_F^2\right] = \sum_{i=1}^{n}\sum_{k=1}^{m}\left(\boldsymbol{b}_{(i)}^\top \mathbf{D}\boldsymbol{b}_{(i)}\mathbf{C}_{kk}\mathbf{A}_{ki}^2\left(\frac{1}{p_{ki}}-1\right)\right).$$

Furthermore, when the sampling probability is set to $p_{ki} \propto |\mathbf{A}_{ki}|\sqrt{\mathbf{C}_{kk}\boldsymbol{b}_{(i)}^\top \mathbf{D}\boldsymbol{b}_{(i)}}$ *, the above expression attains its minimum value*

$$\mathbb{E}\left[\left\|\mathbf{S}_1^\top \mathbf{A}^*\mathbf{B}\mathbf{S}_2 - \mathbf{S}_1^\top \mathbf{A}\mathbf{B}\mathbf{S}_2\right\|_F^2\right]$$
$$= \frac{1}{mc}\left(\sum_{i=1}^{m}\sum_{k=1}^{n}|\mathbf{A}_{ki}|\sqrt{\mathbf{C}_{kk}\boldsymbol{b}_{(i)}^\top \mathbf{D}\boldsymbol{b}_{(i)}}\right)^2 - \sum_{i=1}^{n}\sum_{k=1}^{m}\left(\boldsymbol{b}_{(i)}^\top \mathbf{D}\boldsymbol{b}_{(i)}\mathbf{C}_{kk}\mathbf{A}_{ki}^2\right),$$

where $\mathbf{C} = \mathbf{S}_1\mathbf{S}_1^\top, \mathbf{D} = \mathbf{S}_2\mathbf{S}_2^\top$ *and* $\sum_{i=1}^{m}\sum_{k=1}^{n} p_{ik} = mc$.

Remark 2 .

The optimal sampling probabilities derived in Theorem 2 and Theorem 3 have the form $p_{ij} = \frac{|\mathbf{A}_{ij}|}{f_j g_i}$ *, where* $f_j \propto \mathbf{C}_{jj}^{-\frac{1}{2}}$ *depends only on the index j and* $g_i \propto (\boldsymbol{b}_{(i)}^\top \mathbf{D}\boldsymbol{b}_{(i)})^{-\frac{1}{2}}$ *depends only on the index i. Consequently, the elements of* $\mathbf{A}^*$ *obtained via Algorithm 4 can be expressed as:*

$$\mathbf{A}_{ij}^* = \begin{cases} \dfrac{\mathbf{A}_{ij}}{|\mathbf{A}_{ij}|} f_j g_i, & \text{with probability } p_{ij}, \\ 0, & \text{otherwise.} \end{cases}$$

Defining $\boldsymbol{f} = (f_1, \ldots, f_p), \boldsymbol{g} = (g_1, \ldots, g_n), \mathbf{A}^{*'} = (\text{sign}(\mathbf{A}_{ij}))_{n\times p}$ *, we can decompose* $\mathbf{A}^*$ *as:*

$\mathbf{A}^* = (\boldsymbol{f}\boldsymbol{g}^\top) \odot \mathbf{A}^{*'}$.

As shown in Fig. 2, assuming the elements of $\mathbf{A}$ *are positive for simplicity, the resulting* $\mathbf{A}^{*'}$ *is a binary matrix with elements in* $\{0,1\}$. *This decomposition reduces storage requirements and facilitates faster matrix multiplication operations.*

Building upon Theorem 2 and Theorem 3, several specific sampling probability schemes are now introduced.

Corollary 1 (Efficient Sampling Schemes).

For a matrix $\mathbf{X}$, *the matrix* $\mathbf{X}^*$ *obtained via Algorithm 4 satisfies:*

(i) *(Covariance Approximation) When the sampling probability is set to* $p_{ij} = \frac{\beta |x_{ij}| \|x_i\|_2^2}{\sum_{i=1}^{n}\sum_{j=1}^{j} |x_{ij}| \|x_i\|_2^2}$,

the expression $\mathbb{E}[\|\mathbf{X}^{*\top}\mathbf{X} - \mathbf{X}^\top\mathbf{X}\|_F^2]$ *achieves its minimum value.*

(ii) *(Cross-Covariance Approximation) When the sampling probability is set to* $p_{ij} = \frac{\beta |x_{ij}| \|y_i\|_2^2}{\sum_{i=1}^{n}\sum_{j=1}^{j} |x_{ij}| \|y_i\|_2^2}$, *the expression* $\mathbb{E}[\|\mathbf{X}^{*\top}\mathbf{Y} - \mathbf{X}^\top\mathbf{Y}\|_F^2]$ *achieves its minimum value.*

(iii) *(Projection Approximation) Given the SVD decomposition* $\mathbf{X} = \mathbf{U}\mathbf{\Lambda}\mathbf{V}^\top$, *and defining the projection matrix* $\mathbf{S} = \mathbf{\Lambda}^{-\frac{1}{2}}\mathbf{V}$, *when the sampling probability is set to* $p_{ij} = \frac{\beta |x_{ij}| \sqrt{h_{ii}}}{\sum_{i=1}^{n}\sum_{j=1}^{j} |x_{ij}| \sqrt{h_{ii}}}$, *the expression* $\mathbb{E}[\|\mathbf{X}^{*\top}\mathbf{X}\mathbf{S} - \mathbf{X}^\top\mathbf{X}\mathbf{S}\|_F^2]$ *achieves its minimum value, where* $h_{ii} = x_i^\top (\mathbf{X}^\top\mathbf{X})^{-1} x_i$ *is known as the leverage score.*

3.2 Sparsification Subsampling for PLS

Integration of the Sparse Sketch. The sparse sketch $\mathbf{X}^*$ of the observation matrix $\mathbf{X}$ is constructed using Algorithm 4 and Corollary 1. In the context of PLS, we primarily employ the covariance approximation detailed in Corollary 1 to construct the sparse sketch, as $\mathbf{X}^\top\mathbf{X}$ plays a central role in the algorithm. To accelerate the PLS regression, the primary task involves applying this sparse matrix to the most computationally intensive operations, namely $\mathbf{X}^{*\top}\mathbf{X}$ and $\mathbf{X}^{*\top}\mathbf{Y}$. However, given the direct involvement of $\mathbf{X}$ and $\mathbf{Y}$ in Algorithm 1, a coherent rationale is required to justify the integration of $\mathbf{X}^*$ within the PLS algorithm.

A regression assumption analogous to (1) is considered:

$$\begin{cases} \mathbf{X} = \mathbf{T}\mathbf{P}^\top + \mathbf{E}, \\ \mathbf{X}^* = \mathbf{T}^*\mathbf{P}^\top + \mathbf{E}^*, \\ \mathbf{Y} = \mathbf{T}\mathbf{Q}^\top + \mathbf{F}. \end{cases}$$

Here, it is postulated that $\mathbf{X}$ and $\mathbf{X}^*$ share the same parameter matrix $\mathbf{P}$ (this assumption is validated in the proof of Theorem 4). To reduce computational complexity, we replace the computation of $\mathbf{X}^\top\mathbf{Y}$ in (2) with $\mathbf{X}^{*\top}\mathbf{Y}$ for the derivation of parameters $\boldsymbol{u}_1$ and $\boldsymbol{v}_1$. Since $\mathbf{X}^*$ and $\mathbf{X}$ share parameters, it follows that $\boldsymbol{t}_1 = \mathbf{X}\boldsymbol{v}_1$ and $\boldsymbol{t}_1^* = \mathbf{X}^*\boldsymbol{v}_1$. While only $\boldsymbol{t}_1$ is employed in (3) for estimating $\mathbf{X}$ and $\mathbf{Y}$, the introduction of $\boldsymbol{t}_1^*$ motivates the use of the Core-Elements estimation (8) for parameters $\boldsymbol{p}_1$ and $\boldsymbol{q}_1$:

$$\begin{cases} \boldsymbol{p}_1^\top = \left(\boldsymbol{t}_1^{*\top}\boldsymbol{t}_1\right)^{-1}\boldsymbol{t}_1^{*\top}\mathbf{X}, \\ \boldsymbol{q}_1^\top = \left(\boldsymbol{t}_1^{*\top}\boldsymbol{t}_1\right)^{-1}\boldsymbol{t}_1^{*\top}\mathbf{Y}. \end{cases} \quad (10)$$

The residual terms are subsequently derived as follows:

$$\begin{cases} \mathbf{X}_2 = \mathbf{X}_1 - \boldsymbol{t}_1\boldsymbol{p}_1^\top, \\ \mathbf{X}_2^* = \mathbf{X}_1^* - \boldsymbol{t}_1^*\boldsymbol{p}_1^\top, \\ \mathbf{Y}_2 = \mathbf{Y}_1 - \boldsymbol{t}_1\boldsymbol{q}_1^\top. \end{cases} \quad (11)$$

This iterative process computes subsequent terms $\boldsymbol{t}_2, \boldsymbol{t}_2^*, \boldsymbol{p}_2, \boldsymbol{q}_2, \mathbf{X}_3, \mathbf{X}_3^*, \mathbf{Y}_3$ and so on, up to m iterations, resulting in a sparsified PLS algorithm based on $\mathbf{X}^*$.

Reducing time complexity with kernel updates. However, the matrix update operations in (11) still incur a complexity of $\mathcal{O}(np)$ per iteration, which results in a total complexity of $\mathcal{O}(npm)$. To address this, we employ the kernel-based PLS algorithm (Algorithm 2), which avoids direct updates $\mathbf{X}_k$, $\mathbf{X}_k^*$, and $\mathbf{Y}_k$. Instead, we update $\mathbf{X}_k^{*\top}\mathbf{X}_k$ and $\mathbf{X}_k^{*\top}\mathbf{Y}_k$ directly.

During the k-th iteration of the algorithm, the computation of $\boldsymbol{u}_k$ and $\boldsymbol{v}_k$ requires an SVD of $\mathbf{X}_{k-1}^{*\top}\mathbf{Y}_{k-1}$, followed by the derivation of $\boldsymbol{p}_k$ and $\boldsymbol{q}_k$ using $\boldsymbol{t}_k$ and $\boldsymbol{t}_k^*$. Parameters $\boldsymbol{p}_k$ and $\boldsymbol{q}_k$ can be reformulated as

$$\begin{cases} \boldsymbol{p}_k^\top = (\boldsymbol{t}_k^{*\top}\boldsymbol{t}_k)^{-1}\boldsymbol{t}_k^{*\top}\mathbf{X}_k = \dfrac{\boldsymbol{v}_k^\top\mathbf{X}_k^{*\top}\mathbf{X}_k}{\boldsymbol{v}_k^\top\mathbf{X}_k^{*\top}\mathbf{X}_k\boldsymbol{v}_k}, \\ \boldsymbol{q}_k^\top = (\boldsymbol{t}_k^{*\top}\boldsymbol{t}_k)^{-1}\boldsymbol{t}_k^{*\top}\mathbf{Y}_k = \dfrac{\boldsymbol{v}_k^\top\mathbf{X}_k^{*\top}\mathbf{Y}_k}{\boldsymbol{v}_k^\top\mathbf{X}_k^{*\top}\mathbf{X}_k\boldsymbol{v}_k}. \end{cases} \quad (12)$$

We can replace explicit matrix updates in (11) with updates to $\mathbf{X}_k^{*\top}\mathbf{X}_k$ and $\mathbf{X}_k^{*\top}\mathbf{Y}_k$. This is formalized in Theorem 4. Notably, $\mathbf{X}^*$ and $\mathbf{X}$ continue to share the same parameter $\boldsymbol{p}_1$. Indeed, as demonstrated in the proof of Theorem 4, even if the corresponding parameter $\boldsymbol{p}^*$ for $\mathbf{X}^*$ is estimated using alternative formulations, the algorithmic outcomes remain invariant.

Theorem 4 .

Assuming $\mathbf{X}^* = \mathbf{T}^*\mathbf{P}^{*\top} + \mathbf{E}^*$ *, the iterative update formulas for the parameters within the kernel algorithm framework, independent of the specific value assigned to* $\mathbf{P}^*$ *, are expressed as follows:*

$$\begin{cases} \mathbf{X}_{k+1}^{*\top}\mathbf{X}_{k+1} = \mathbf{X}_k^{*\top}\mathbf{X}_k\left(\mathbf{I} - \boldsymbol{v}_k\boldsymbol{p}_k^{\top}\right), \\ \mathbf{X}_{k+1}^{*\top}\mathbf{Y}_{k+1} = \mathbf{X}_k^{*\top}\mathbf{Y}_k - \boldsymbol{p}_k^{'}\boldsymbol{q}_k^{\top}\left(\boldsymbol{t}_k^{*\top}\boldsymbol{t}_k\right), \end{cases}$$

where $\boldsymbol{p}_k^{'} = \dfrac{\mathbf{X}_k^{*\top}\mathbf{X}_k\boldsymbol{v}_k}{\boldsymbol{v}_k^{\top}\mathbf{X}_k^{*\top}\mathbf{X}_k\boldsymbol{v}_k}$.

Theorem 4 establishes that the introduction of the kernel algorithm renders the parameter iteration dependent solely on the data dimensions *p* and *l*, independent of the sample size *n*. This underscores the potential of kernel algorithms to significantly reduce time complexity in large-scale data scenarios. Moreover, the parameter iteration formula is shown to be independent of $\mathbf{P}^*$, thereby justifying the assumption of a shared parameter $\mathbf{P}$ for both $\mathbf{X}^*$ and $\mathbf{X}$ in (10).

In summary, we propose a kernel-based PLS algorithm incorporating sparsification subsampling, ensuring that the primary time complexity is dominated by the initial kernel computation step.

Improved algorithm for regression coefficients. While the sparsification subsampling PLS algorithm based on kernel updating yields the PLS loading parameters $\mathbf{P}$ and $\mathbf{Q}$, it does not directly provide the PLS regression coefficients $\mathbf{B}$ in (6). To circumvent the need for additional computation to obtain $\mathbf{B}$, we have developed an improved kernel method inspired by Algorithm 2 (Dayal and MacGregor, 1997). This method also contributes to further acceleration of the algorithm. For clarity in the subsequent discussion, let $\mathbf{T} = \mathbf{XR}$, which implies $\mathbf{B} = \mathbf{RQ}^{\top}$. Consequently, computing $\mathbf{R}$ during the algorithm suffices to readily obtain $\mathbf{B}$.

As established in Section 2, the *k*-th iteration involves $\boldsymbol{t}_k = \mathbf{X}_k\boldsymbol{v}_k$. Combining this with (11) yields:

$$\boldsymbol{t}_k = \mathbf{X}_k\boldsymbol{v}_k = \left(\mathbf{X}_{k-1} - \boldsymbol{t}_{k-1}\boldsymbol{p}_{k-1}^{\top}\right)\boldsymbol{v}_k = \mathbf{X}_{k-1}\left(\mathbf{I} - \boldsymbol{v}_{k-1}\boldsymbol{p}_{k-1}^{\top}\right)\boldsymbol{v}_k = \mathbf{X}\prod_{i=1}^{k-1}\left(\mathbf{I} - \boldsymbol{v}_i\boldsymbol{p}_i^{\top}\right)\boldsymbol{v}_k.$$

Introducing $\mathbf{H}_{k-1} = \prod_{i=1}^{k-1}(\mathbf{I} - \boldsymbol{v}_i\boldsymbol{p}_i^{\top})$, the following is obtained based on the preceding derivation:

$$\boldsymbol{r}_k = \prod_{i=1}^{k-1}\left(\mathbf{I} - \boldsymbol{v}_i \boldsymbol{p}_i^{\top}\right)\boldsymbol{v}_k = \mathbf{H}_{k-1}\boldsymbol{v}_k.$$

Importantly, the introduction of $\mathbf{H}_k$ enables the derivation of the update formulas for $\mathbf{X}_k^{*\top}\mathbf{X}_k$ and $\mathbf{X}_k^{*\top}\mathbf{Y}_k$.

Theorem 5 .

Given the computed variable $\mathbf{H}_k$ *, the updates for* $\mathbf{X}_k^{*\top}\mathbf{X}_k$ *and* $\mathbf{X}_k^{*\top}\mathbf{Y}_k$ *can be expressed as follows:*

$$\begin{cases} \mathbf{X}_{k+1}^{*\top}\mathbf{X}_{k+1} = \mathbf{X}^{*\top}\mathbf{X}\mathbf{H}_k, \\ \mathbf{X}_{k+1}^{*\top}\mathbf{Y}_{k+1} = \mathbf{X}_k^{*\top}\mathbf{Y}_k - \boldsymbol{p}_k' \boldsymbol{q}_k^{\top}\left(\boldsymbol{t}_k^{*\top}\boldsymbol{t}_k\right). \end{cases}$$

Notably, the introduction of variables $\boldsymbol{r}_k$ and $\mathbf{H}_k$ optimizes the update process for $\mathbf{X}_k^{*\top}\mathbf{X}_k$. The improved algorithm is summarized as follows.

Algorithm 5: Sparsification Subsampling for Improved Kernel Algorithm

Input: $\mathbf{X}, \mathbf{Y}$

1 Compute the sparse projection $\mathbf{X}^*$ of $\mathbf{X}$ using Algorithm 4, the sampling probability is given by Corollary 1 $\mathcal{O}(\mathrm{nnz}(\mathbf{X}))$

2 Compute the approximate covariance matrices $\mathbf{X}^{*\top}\mathbf{X}$ and $\mathbf{X}^{*\top}\mathbf{Y}$ $\mathcal{O}(rp^2)$

3 **for** $k \leftarrow 1$ *to m* **do**

4 Perform an SVD decomposition on $\mathbf{X}_k^{*\top}\mathbf{Y}_k$ and find the vectors $\boldsymbol{v}_k$ and $\boldsymbol{w}_k$ corresponding to the largest singular value. $\mathcal{O}(p^2 l)$

5 $\boldsymbol{r}_k = \mathbf{H}_{k-1}\boldsymbol{v}_k, \mathbf{H}_k = \prod_{i=1}^{k}\left(\mathbf{I} - \boldsymbol{v}_i \boldsymbol{p}_i^{\top}\right)$ $\mathcal{O}(p^2)$

6 $\boldsymbol{p}_k^{\top} = \dfrac{\boldsymbol{v}_k^{\top}\mathbf{X}^{*\top}\mathbf{X}\mathbf{H}_k}{\boldsymbol{v}_k^{\top}\mathbf{X}^{*\top}\mathbf{X}\boldsymbol{r}_k}, \quad \boldsymbol{p}_k'^{\top} = \dfrac{\boldsymbol{r}_k^{\top}\mathbf{X}^{\top}\mathbf{X}^*}{\boldsymbol{v}_k^{\top}\mathbf{X}^{*\top}\mathbf{X}\boldsymbol{r}_k}, \quad \boldsymbol{q}_k^{\top} = \dfrac{\boldsymbol{v}_k^{\top}\mathbf{X}_k^{*\top}\mathbf{Y}_k}{\boldsymbol{v}_k^{\top}\mathbf{X}^{*\top}\mathbf{X}\boldsymbol{r}_k}.$ $\mathcal{O}(p^2)$

7 $\mathbf{X}_{k+1}^{*\mathrm{T}}\mathbf{Y}_{k+1} = \mathbf{X}_k^{*\mathrm{T}}\mathbf{Y}_k - \boldsymbol{p}_k' \boldsymbol{q}_k^{\mathrm{T}}\left(\boldsymbol{t}_k^{*\mathrm{T}}\boldsymbol{t}_k\right)$ $\mathcal{O}(pl)$

8 **end**

9 $\mathbf{B} = \mathbf{R}\mathbf{Q}^{\top}$ $\mathcal{O}(mpl)$

Output: $\mathbf{T}, \mathbf{P}, \mathbf{Q}, \mathbf{B}$

The time complexity of Algorithm 5 is analyzed as follows. For simplicity, it is assumed that $l < p$. The first step, involving sparse subsampling of $\mathbf{X}$, has a time complexity of $\mathcal{O}(\text{nnz}(\mathbf{X}))$. The second step, which performs matrix multiplication, has a time complexity of $\mathcal{O}(rp^2)$. The iterative process of solving the parameters is primarily governed by the SVD computation in each iteration, resulting in a complexity of $\mathcal{O}(mp^2l)$. Consequently, the overall time complexity is $\mathcal{O}(\text{nnz}(\mathbf{X}) + rp^2 + mp^2l)$. In the context of large-scale data, where $n \gg r \gg p$ to mitigate significant performance degradation, the time complexity can be reduced to $\mathcal{O}(rp^2)$. The number of iterations, *m*, is typically determined through cross-validation in preliminary experiments. In most scenarios, only a low-rank latent variable is considered, i.e., $m \ll p$.

4 Simulation Studies

In this section, we evaluate the performance of our proposed algorithm through a Monte Carlo simulation study. We compare Algorithm 5 (SPAR, with the sampling probability defined as in Corollary 1 (i)) with five representative sampling-based competitors: uniform row sampling (UNIF), leverage score-based sampling (LEV), influence function-based sampling (IF) (Xie et al., 2022), minimum covariance determinant-based subset selection (MCD) (Huang et al., 2024), and the full-sample approach (FULL). The main text primarily focuses on sampling-based methods, since they are the most directly comparable to SPAR in terms of reducing the computational burden by using only a subset of the information from the predictor matrix. For completeness, Table 1 summarizes the computational costs, method types, and acceleration strategies of several classical and contemporary PLS algorithms, including online or stochastic PLS variants. Additional experiments providing broader comparisons with these methods are reported in Section C of the Supplementary Material.

To ensure a fair comparison among sampling-based methods, we sample *r* rows for row-sampling methods and select *rp* elements for SPAR, so that the number of nonzero elements used from the predictor matrix is comparable across different sampling strategies. All experiments were conducted using R on a machine with a processor clock speed of 2.20 GHz and 32 GB of memory.

We generate $\mathbf{X}$ and $\mathbf{Y}$ by randomly generating the latent variable $\mathbf{T}$, under the experimental assumptions specified in (1):

$$\begin{cases} \mathbf{X} = \mathbf{T}\mathbf{P}^\top + \mathbf{E}, \\ \mathbf{Y} = \mathbf{T}\mathbf{Q}^\top + \mathbf{F}, \end{cases} \quad (13)$$

where $\mathbf{P}$ and $\mathbf{Q}$ are randomly generated orthogonal matrices, and $\mathbf{T}$ follows the distribution:

$$\boldsymbol{t}_{(i)} \sim N_m\left(0, \boldsymbol{\Sigma}_1\boldsymbol{\Sigma}_2\right), \quad \boldsymbol{\Sigma}_1 = \left(0.5^{|i-j|}\right)_{ij} \quad \boldsymbol{\Sigma}_2 = \text{diag}\left(1, 20^{\frac{1}{m-1}}, 20^{\frac{2}{m-1}}, \ldots, 20\right). \quad (14)$$

Concurrently, $\mathbf{E}$ and $\mathbf{F}$ are residual matrices satisfying:

$$\mathbf{E}_{ij} \sim N\left(0, \sigma_1^2\right), \quad \mathbf{F}_{\mathrm{ij}} \sim N\left(0, \sigma_2^2\right).$$

The variances σ_1^2 and σ_2^2 are determined by the signal-to-noise ratios, defined by $\gamma_1 = \frac{\mathrm{Var}(\mathbf{TP}^\top)}{\sigma_1^2}, \gamma_2 = \frac{\mathrm{Var}(\mathbf{TQ}^\top)}{\sigma_2^2}$.

Note that in (13), we set $\mathbf{P}$ and $\mathbf{Q}$ as orthogonal matrices, which follows the experimental settings of Li et al. (2002); Lafaye de Micheaux et al. (2019). Specifically, Li et al. (2002) explored a low-dimensional scenario and proposed a specific set of orthogonal matrices, while Lafaye de Micheaux et al. (2019) ensured orthogonality by employing a simple sparse matrix. In this paper, we adopt a fixed set of orthogonal matrices for $\mathbf{P}$ and $\mathbf{Q}$ in the subsequent analysis to enhance the model's generalizability. In (14), we use $\mathbf{\Sigma}_1$ to control the covariance structure of $\mathbf{T}$ and $\mathbf{\Sigma}_2$ to control the heteroscedasticity of $\mathbf{T}$, which provides greater generalizability compared to Li et al. (2002); Lafaye de Micheaux et al. (2019).

For the experimental variables, we set the total sample size to $n = 50000$, the dimensionality of the observation matrix to $p = 100$, and the dimensionality of the response matrix to $l = 50$. We designate the latent variable dimensions as $m \in \{5, 10, 15, 20, 25\}$, denoted as (M1)-(M5). For the signal-to-noise ratios γ_1 and γ_2, we consider the following settings:

(R1) $\gamma_1 = \gamma_2 = 0.25$.

(R2) $\gamma_1 = \gamma_2 = 0.5$.

(R3) $\gamma_1 = \gamma_2 = 1$.

Finally, we evaluate subsample sizes of $r \in \{5p, 10p, 15p, 20p, 25p\}$, corresponding to matrix sparsities of $\{1\%, 2\%, 3\%, 4\%, 5\%\}$.

Based on the settings above, we can derive the regression coefficient $\mathbf{B}$:

$$\mathbf{B} = \mathbf{P}\left(\mathbf{P}^\top \mathbf{P}\right)^{-1} \mathbf{Q}^\top. \quad (15)$$

Consequently, we can compute $\mathrm{MSE}_{\mathbf{B}}$ using the estimated $\mathbf{B}$ obtained from different algorithms:

$$\mathrm{MSE}_{\mathbf{B}} = \| \mathbf{B} - \mathbf{B} \|_F .$$

However, under this setting, $\mathbf{X}$ and $\mathbf{Y}$ are generated from the latent variable $\mathbf{T}$, as defined in (13). Consequently, the true $\mathbf{B}$ is given by

$$\mathcal{P}^{\perp}\mathbf{Q}^{\top}, \mathcal{P}^{\perp} = \{\mathbf{R} \in \mathbb{R}^{p\times m} \mid \mathbf{P}^{\top}\mathbf{R} = \mathbf{I}_m\}.$$

This implies that the true $\mathbf{B}$ in this scenario represents the projection of $\mathcal{P}^{\perp}$ onto the space spanned by $\mathbf{Q}^{\top}$. Therefore, directly employing $\mathrm{MSE}_{\mathbf{B}}$ may not furnish a reliable measure of algorithmic performance, as it fails to account for the errors arising from variations within the column space of $\mathcal{P}^{\perp}$.

To address this issue, we consider eliminating the uncertainty of $\mathcal{P}^{\perp}$ by multiplying a $\mathbf{P}^{\top}$ in the left. Given that:

$$\mathbf{P}^{\top}\mathbf{B} = \mathbf{P}^{\top}\mathcal{P}^{\perp}\mathbf{Q}^{\top} = \mathbf{Q}^{\top},$$

we are essentially comparing $\mathbf{P}^{\top}\mathbf{B}$ derived from the algorithm with the true value $\mathbf{Q}^{\top}$ and define the $\mathrm{MSE}_{\mathbf{Q}}$ as:

$$\mathrm{MSE}_{\mathbf{Q}} = \left\| \mathbf{P}^{\top}\mathbf{B} - \mathbf{Q}^{\top} \right\|_F . \quad (16)$$

Furthermore, we also compute the Prediction Mean Squared Error (PMSE) by partitioning the data into training and testing sets:

$$\mathrm{PMSE} = \left\| \mathbf{X}_{\text{test}}\mathbf{B} - \mathbf{Y}_{\text{test}} \right\|_F ,$$

where the parameter $\mathbf{B}$ is derived from the training set $\mathbf{X}_{\text{train}}, \mathbf{Y}_{\text{train}}$ using the various algorithms under consideration. Notably, we can decompose the PMSE as follows:

$$\left\| \mathbf{X}_{\text{test}}\mathbf{B} - \mathbf{Y}_{\text{test}} \right\|_F = \left\| \mathbf{T}_{\text{test}}(\mathbf{P}^{\top}\mathbf{B} - \mathbf{Q}^{\top}) + (\mathbf{E}\mathbf{B} - \mathbf{F}) \right\|_F . \quad (17)$$

As the magnitude of this expression is primarily governed by $\mathbf{P}^{\top}\mathbf{B} - \mathbf{Q}^{\top}$, which is equivalent to the $\mathrm{MSE}_{\mathbf{Q}}$ defined in (16), the PMSE also mitigates some uncertainty by using $\mathbf{P}^{\top}\mathbf{B}$.

By repeating the experiment 10 times, we obtained the results for $\mathrm{MSE}_{\mathbf{B}}$ as depicted in Fig. 3. This figure demonstrates the advantage of the sparsification subsampling method over the row sampling method. Additionally, the subsampling methods demonstrate robustness to variations in SNR and latent variable dimensions, with their performance progressively approaching that of the full-sample method as the sample size increases. However, a certain gap remains between the

$\mathrm{MSE}_{\mathbf{B}}$ achieved by sparsification subsampling and that of the full-sample method, potentially stemming from the inherent uncertainty in the estimation of $\mathbf{B}$.

Meanwhile, Fig. 4 presents the results for $\mathrm{MSE}_{\mathbf{Q}}$, illustrating the significant advantage of the sparsification subsampling method over the row sampling method in this context. Notably, with a sample size of 1000 (corresponding to a 2% sampling ratio), the $\mathrm{MSE}_{\mathbf{Q}}$ achieved by sparsification subsampling closely approximates that of the full-sample method. This finding suggests that the sparsification subsampling method demonstrates superior predictive performance in exploring the direct relationship between latent variables and their corresponding variables.

The PMSE results obtained are presented in Fig. 5, which further corroborates the superiority of the sparsification subsampling method, demonstrating consistently lower PMSE values compared to the row sampling method. It is worth noting that the variability is primarily driven by cross-validation; notably, subsampling methods exhibit variance comparable to the full-sample approach. Interestingly, the PMSE values achieved by sparsification subsampling lie between those of $\mathrm{MSE}_{\mathbf{B}}$ and $\mathrm{MSE}_{\mathbf{Q}}$. This observation aligns with the decomposition of PMSE derived in (17), suggesting that PMSE reflects the model's performance after partially mitigating the uncertainty associated with the estimation of $\mathbf{B}$.

To intuitively demonstrate the efficiency of the sparse algorithm, we compare the running times of various algorithms, including the probability computation, subsampling, and main algorithm execution phases. We set $n = 50000$, $p = 100$, and $l = 50$. The number of principal components, *m*, varies between 5 and 10, while the number of sampled elements, *r*, ranges from 500 to 2500 in increments of 500. We repeat the experiment 10 times and compare the total times of all competing methods. The results are presented in Table 2.

Table 2 compares the runtime for different algorithms. The results indicate that, with sampling ratios ranging from 1% to 5%, the sparsification method achieves a consistent speedup of approximately 10 times compared to the full-sample method. While the runtime of the sparsification method is slightly longer than that of the row sampling method, primarily due to the overhead associated with sparse matrix operations, this difference is expected to diminish as the data scale increases. Furthermore, the runtime of the Spar-PLS method on large-scale datasets exhibits low sensitivity to variations in the number of iterations (m), further validating the effectiveness of the subsampling methods in accelerating computations.

5 Real Data Analysis

Wave energy, a rapidly developing and highly promising renewable energy source, holds substantial potential for addressing the challenges of global warming and climate change. However, optimizing the energy output of large-scale wave farms is a complex task, requiring computationally expensive simulations to account for the hydrodynamic interactions between wave energy converters (WECs). The development of fast and accurate surrogate models is crucial to overcoming these computational challenges. To this end, Neshat et al. (2020) compiled

a comprehensive dataset of WEC arrays, consisting of real-world measurements collected from wave energy converters deployed in offshore scenarios (Perth and Sydney) to detect and calculate energy generation at specific locations. In each experimental setup, either 49 or 100 detectors were positioned at various locations. Consequently, the predictor variables consist of the 98-dimensional or 200-dimensional coordinate information (representing the 2D coordinates of the detectors), while the response variables correspond to the generated power output, having dimensions of 49 or 100, respectively.

As illustrated in Fig. 6, the aggregated data from numerous trials reveals a clear gradient pattern in the energy field relative to the physical positions. This spatial dependence motivates the application of the PLS method to model the relationship between the detector locations and the corresponding power outputs.

For the four scenarios, encompassing 49 and 100 WECs combined with wave conditions from Perth and Sydney (denoted as (C1)-(C4)), we select the number of iterations $m \in \{5,10,15,20\}$ (denoted as (M1)-(M4)). For each scenario, we employ 10-fold cross-validation, setting the subsample size to {2%, 4%, 6%, 8%, 10%} of the total sample size. The mean and variance of the Prediction Mean Squared Error are computed for each fold and presented in Fig. 7.

As illustrated in Fig. 7, the sparsification subsampling method consistently outperforms row subsampling once the sample size stabilizes relative to the number of iterations. This observation aligns with the findings from the simulation study.

Furthermore, we compare the runtime of different algorithms. For this comparison, we utilize the Perth49 dataset, which has the largest sample size, with the subsample size set to $r = 500$ and the number of iterations $m \in \{5,10,15,20\}$. The results are presented below.

As shown in Table 3, the sparse sampling algorithm still achieves a speedup of $80\% - 90\%$, consistent with the conclusions from the previous numerical simulations.

6 Conclusion

Partial Least Squares is an effective supervised dimensionality reduction method. To address the high computational complexity of PLS on large-scale data, we propose a novel Spar-PLS algorithm that applies sparse subsampling techniques to PLS regression. By theoretically setting the subsampling probabilities and applying matrix sparsification approximations, Spar-PLS efficiently handles data with both large-scale and high-dimensional characteristics. Numerical simulations and real data analyses demonstrate that, compared to the conventional PLS algorithm and other sampling strategies, the proposed Spar-PLS method achieves superior performance with enhanced computational efficiency.

Future research directions for the Spar-PLS method are manifold. On the one hand, many studies suggest that the PLS regression coefficient is a biased estimate (Phatak et al., 2002), thus finding a covariance approximation aims to obtain a regression coefficient with less bias, achieving local optimality. While this study derives the minimum variance approximation for matrix multiplication under the Frobenius norm, further investigation for approximations under the

spectral norm and other metrics is warranted. On the other hand, the theoretical properties of PLS are currently only for the single-component and univariate response cases (Nadler and Coifman, 2005; Cook and Forzani, 2019). Future work may derive subsampling probabilities based on existing theoretical properties of the parameters, or by simulating the parameter variance (Odgers et al., 2023) and set probabilities by parameter variance like several classical methods (Wang et al., 2018; Han et al., 2025). In conclusion, further research is needed to explore the asymptotic properties of PLS regression, to determine the optimal sampling probabilities for sparsification algorithms under different norms, and to analyze the convergence behavior during the algorithm's iterative process.

Supplementary Material

Appendix:

provides complete proofs of the theoretical results. (appendix.pdf, a pdf file)

Code:

includes R code for implementing the proposed method and reproducing the numerical results. (code.zip, a zip file)

Acknowledgement

This research was funded by the National Natural Science Foundation of China (No. 12131001), the Fundamental Research Funds for the Central Universities (No. 63263100), and the China Postdoctoral Science Foundation under Grant Number 2025M783111.

Disclosure Statement

The authors report there are no competing interests to declare.

References

Achlioptas, D. and F. Mcsherry (2007). Fast computation of low-rank matrix approximations. *Journal of the Association for Computing Machinery 54*(2), 9–es.

Ai, M., F. Wang, J. Yu, and H. Zhang (2021). Optimal subsampling for large-scale quantile regression. *Journal of Complexity 62*, 101512.

Ai, M., J. Yu, H. Zhang, and H. Wang (2021). Optimal subsampling algorithms for big data regressions. *Statistica Sinica 31*(2), 749–772.

Arora, R., P. Mianjy, and T. Marinov (2016). Stochastic optimization for multiview representation learning using partial least squares. In *International Conference on Machine Learning*, pp. 1786–1794. PMLR.

Braverman, V., R. Krauthgamer, A. R. Krishnan, and S. Sapir (2021). Near-optimal entrywise sampling of numerically sparse matrices. In *Proceedings of Thirty Fourth Conference on Learning Theory*, Volume 134 of *PMLR*, pp. 759–773.

Cook, R. D. and L. Forzani (2019). Partial least squares prediction in high-dimensional regression. *The Annals of Statistics 47*(2), 884–908.

Cook, R. D. and L. Forzani (2024). *Partial Least Squares Regression: and Related Dimension Reduction Methods*. Chapman and Hall/CRC.

Cook, R. D., I. S. Helland, and Z. Su (2013, 07). Envelopes and partial least squares regression. *Journal of the Royal Statistical Society: Series B (Statistical Methodology) 75*(5), 851–877.

Dayal, B. S. and J. F. MacGregor (1997). Improved PLS algorithms. *Journal of Chemometrics 11*(1), 73–85.

De Jong, S. (1993). SIMPLS: an alternative approach to partial least squares regression. *Chemometrics and intelligent laboratory systems 18*(3), 251–263.

Drineas, P., R. Kannan, and M. W. Mahoney (2006). Fast Monte Carlo algorithms for matrices I: Approximating matrix multiplication. *SIAM Journal on Computing 36*(1), 132–157.

Drineas, P., M. W. Mahoney, and S. Muthukrishnan (2006). Sampling algorithms for l2 regression and applications. In *Proceedings of the seventeenth annual ACM-SIAM symposium on Discrete algorithm*, pp. 1127–1136.

Feng, Y. and Y. Zhou (2024). GGD: Grafting gradient descent. *Journal of Machine Learning Research 25*(316), 1–87.

Han, Y., J. Yu, N. Zhang, C. Meng, P. Ma, W. Zhong, and C. Zou (2025). Leverage classifier: Another look at support vector machine. *Statistica Sinica 35*, 1605–1625.

Höskuldsson, A. (1988). PLS regression methods. *Journal of chemometrics 2*(3), 211–228.

Hu, Y., M. Li, X. Liu, and C. Meng (2025). Sampling-based methods for multi-block optimization problems over transport polytopes. *Mathematics of Computation 94*(353), 1281–1322.

Huang, J., X. Kang, Q. Huang, M. Li, C. Meng, and J. Zhang (2026). Efficient approximation of leverage scores in two-dimensional autoregressive models with application to image anomaly detection. *Journal of Computational and Graphical Statistics 35*(1), 89–100.

Huang, X., G. Huang, X. Chen, Z. Xie, S. Ali, X. Chen, L. Yuan, and W. Shi (2024). An adaptive strategy to improve the partial least squares model via minimum covariance determinant. *Chemometrics and Intelligent Laboratory Systems 249*, 105120.

Jordao, A., M. Lie, V. H. C. De Melo, and W. R. Schwartz (2021). Covariance-free partial least squares: An incremental dimensionality reduction method. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pp. 1421–1429.

Kelly, B. and S. Pruitt (2015). The three-pass regression filter: A new approach to forecasting using many predictors. *Journal of Econometrics 186*(2), 294–316.

Kundu, A., P. Drineas, and M. Magdon-Ismail (2017). Recovering PCA and sparse PCA via hybrid-(l1,l2) sparse sampling of data elements. *Journal of Machine Learning Research 18*(75), 1–34.

Lafaye de Micheaux, P., B. Liquet, and M. Sutton (2019). PLS for big data: A unified parallel algorithm for regularised group PLS. *Statistics Surveys 13*, 119 – 149.

Li, B., J. Morris, and E. B. Martin (2002). Model selection for partial least squares regression. *Chemometrics and Intelligent Laboratory Systems 64*(1), 79–89.

Li, M., J. Yu, T. Li, and C. Meng (2023). Importance sparsification for sinkhorn algorithm. *Journal of Machine Learning Research 24*(247), 1–44.

Li, M., J. Yu, T. Li, and C. Meng (2024). Core-elements for large-scale least squares estimation. *Statistics and Computing 34*(6), 1–16.

Li, M., J. Yu, H. Xu, and C. Meng (2023). Efficient approximation of Gromov-Wasserstein distance using importance sparsification. *Journal of Computational and Graphical Statistics 32*(4), 1512–1523.

Li, M., J. Zhang, and C. Meng (2024). Nonparametric additive models for billion observations. *Journal of Computational and Graphical Statistics 33*(4), 1397–1412.

Li, T. and C. Meng (2021). Modern subsampling methods for large-scale least squares regression. *International Journal of Cyber-Physical Systems 2*(2), 1–28.

Lindgren, F., P. Geladi, and S. Wold (1993). The kernel algorithm for PLS. *Journal of Chemometrics 7*(1), 45–59.

Ma, P., Y. Chen, X. Zhang, X. Xing, J. Ma, and M. W. Mahoney (2022). Asymptotic analysis of sampling estimators for randomized numerical linear algebra algorithms. *Journal of Machine Learning Research 23*(177), 1–45.

Ma, P., J. Z. Huang, and N. Zhang (2015). Efficient computation of smoothing splines via adaptive basis sampling. *Biometrika 102*(3), 631–645.

Ma, P., M. Mahoney, and B. Yu (2014). A statistical perspective on algorithmic leveraging. In *Proceedings of the 31st International Conference on Machine Learning*, pp. 91–99. PMLR.

Ma, P. and X. Sun (2015). Leveraging for big data regression. *Wiley Interdisciplinary Reviews: Computational Statistics 7*(1), 70–76.

Mahoney, M. W. (2016). Lecture notes on randomized linear algebra. *arXiv preprint arXiv:1608.04481*.

Meng, C., Y. Wang, X. Zhang, A. Mandal, W. Zhong, and P. Ma (2017). Effective statistical methods for big data analytics. In *Handbook of Research on Applied Cybernetics and Systems Science*, pp. 280–299. IGI Global.

Meng, C., R. Xie, A. Mandal, X. Zhang, W. Zhong, and P. Ma (2021). Lowcon: A design-based subsampling approach in a misspecified linear model. *Journal of Computational and Graphical Statistics 30*(3), 694–708.

Meng, C., J. Yu, Y. Chen, W. Zhong, and P. Ma (2022). Smoothing splines approximation using hilbert curve basis selection. *Journal of Computational and Graphical Statistics 31*(3), 802–812.

Meng, C., X. Zhang, J. Zhang, W. Zhong, and P. Ma (2020). More efficient approximation of smoothing splines via space-filling basis selection. *Biometrika 107*(3), 723–735.

Nadler, B. and R. R. Coifman (2005). Partial least squares, beer's law and the net analyte signal: statistical modeling and analysis. *Journal of Chemometrics 19*(1), 45–54.

Neshat, M., B. Alexander, N. Y. Sergiienko, and M. Wagner (2020). Optimisation of large wave farms using a multi-strategy evolutionary framework. In *Proceedings of the 2020 genetic and evolutionary computation conference*, pp. 1150–1158. Association for Computing Machinery.

Odgers, J., C. Kappatou, R. Misener, S. García Muñoz, and S. Filippi (2023). Probabilistic predictions for partial least squares using bootstrap. *AIChE Journal 69*(7), e18071.

Ouyang, X., H. Zheng, H. Liang, J. Zhang, Y. Qiu, C. Meng, and M. Li (2026). Sparsification techniques for large-scale optimal transport problems. *Wiley Interdisciplinary Reviews: Computational Statistics 18*(1), e70056.

Phatak, A., P. Reilly, and A. Penlidis (2002). The asymptotic variance of the univariate pls estimator. *Linear Algebra and its Applications 354*(1-3), 245–253.

Schwartz, W. R., H. Guo, and L. S. Davis (2010). A robust and scalable approach to face identification. In *Computer Vision–ECCV 2010: 11th European Conference on Computer Vision, Heraklion, Crete, Greece, September 5-11, 2010, Proceedings, Part VI 11*, pp. 476–489. Springer.

Stocchero, M. (2019). Iterative deflation algorithm, eigenvalue equations, and PLS2. *Journal of Chemometrics 33*(10), e3144.

Stott, A. E., S. Kanna, D. P. Mandic, and W. T. Pike (2017). An online NIPALS algorithm for partial least squares. In *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 4177–4181.

Tabei, Y., H. Saigo, Y. Yamanishi, and S. J. Puglisi (2016). Scalable partial least squares regression on grammar-compressed data matrices. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 1875–1884. Association for Computing Machinery.

Ting, D. and E. Brochu (2018). Optimal subsampling with influence functions. In *Advances in Neural Information Processing Systems*, Volume 31. Curran Associates, Inc.

Trygg, J. and S. Wold (2002). Orthogonal projections to latent structures (O-PLS). *Journal of Chemometrics 16*(3), 119–128.

Wang, H. and Y. Ma (2021). Optimal subsampling for quantile regression in big data. *Biometrika 108*(1), 99–112.

Wang, H., M. Yang, and J. Stufken (2019). Information-based optimal subdata selection for big data linear regression. *Journal of the American Statistical Association 114*(525), 393–405.

Wang, H., R. Zhu, and P. Ma (2018). Optimal subsampling for large sample logistic regression. *Journal of the American Statistical Association 113*(522), 829–844.

Wang, L., J. Elmstedt, W. K. Wong, and H. Xu (2021). Orthogonal subsampling for big data linear regression. *The Annals of Applied Statistics 15*(3), 1273–1290.

Wang, Y., A. W. Yu, and A. Singh (2017). On computationally tractable selection of experiments in measurement-constrained regression models. *Journal of Machine Learning Research 18*(143), 1–41.

Wang, Y.-X., B. Balle, and S. P. Kasiviswanathan (2019). Subsampled rényi differential privacy and analytical moments accountant. In *Proceedings of the Twenty-Second International Conference on Artificial Intelligence and Statistics*, pp. 1226–1235. PMLR.

Wold, H. (1966). Estimation of principal components and related models by iterative least squares. *Multivariate analysis*, 391–420.

Wold, S., A. Ruhe, H. Wold, and W. Dunn, Iii (1984). The collinearity problem in linear regression. the partial least squares (PLS) approach to generalized inverses. *SIAM Journal on Scientific and Statistical Computing 5*(3), 735–743.

Wu, X., Y. Huo, H. Ren, and C. Zou (2024). Optimal subsampling via predictive inference. *Journal of the American Statistical Association 119*(548), 2844–2856.

Xiao, H., K. Rasul, and R. Vollgraf (2017). Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*.

Xie, R., S. Bai, and P. Ma (2023). Optimal sampling designs for multidimensional streaming time series with application to power grid sensor data. *The Annals of Applied Statistics 17*(4), 3195–3215.

Xie, R., Z. Wang, S. Bai, P. Ma, and W. Zhong (2019). Online decentralized leverage score sampling for streaming multidimensional time series. In *Proceedings of the Twenty-Second International Conference on Artificial Intelligence and Statistics*, pp. 2301–2311. PMLR.

Xie, Z., X. Chen, et al. (2022). Subsampling for partial least-squares regression via an influence function. *Knowledge-Based Systems 245*, 108661.

Xue, D., M. Li, J. Zhang, and C. Meng (2026). Core-elements subsampling for alternating least squares. *Journal of Computational and Graphical Statistics 0*(ja), 1–21.

Yi, S.-Y. and Y.-D. Zhou (2023). Model-free global likelihood subsampling for massive data. *Statistics and Computing 33*(1), 9.

Yu, J., M. Ai, and Z. Ye (2024). A review on design inspired subsampling for big data. *Statistical Papers 65*(2), 467–510.

Yu, J., J. Liu, and H. Wang (2023). Information-based optimal subdata selection for non-linear models. *Statistical Papers 64*(4), 1069–1093.

Yu, J., H. Wang, and M. Ai (2025). A subsampling strategy for aic-based model averaging with generalized linear models. *Technometrics 67*(1), 122–132.

Yu, J., H. Wang, M. Ai, and H. Zhang (2022). Optimal distributed subsampling for maximum quasi-likelihood estimators with massive data. *Journal of the American Statistical Association 117*(537), 265–276.

Zeng, X.-Q. and G.-Z. Li (2014). Incremental partial least squares analysis of big streaming data. *Pattern recognition 47*(11), 3726–3735.

Zhang, M., Y. Zhou, Z. Zhou, and A. Zhang (2024). Model-free subsampling method based on uniform designs. *IEEE Transactions on Knowledge and Data Engineering 36*(3), 1210–1220.

Zhang, Y., L. Wang, X. Zhang, and H. Wang (2024). Independence-encouraging subsampling for nonparametric additive models. *Journal of Computational and Graphical Statistics 33*(4), 1424–1433.

Zhou, Z., Z. Yang, A. Zhang, and Y. Zhou (2024). Efficient model-free subsampling method for massive data. *Technometrics 66*(2), 240–252.

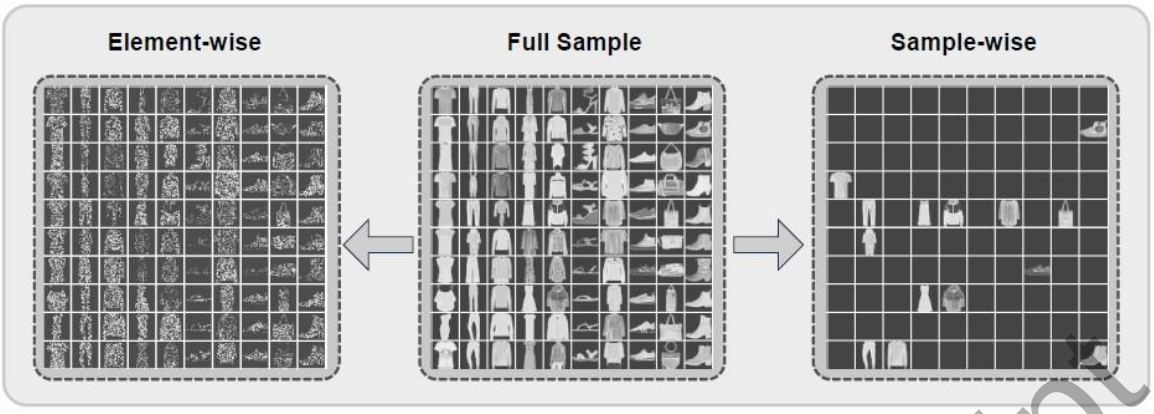

Fig. 1: Illustration of element-wise subsampling versus sample-wise subsampling on the Fashion-MNIST dataset (Xiao et al., 2017). Notably, the element-wise method effectively preserves the key features of each image, whereas row sample-wise method may lead to the complete loss of information for unsampled images.

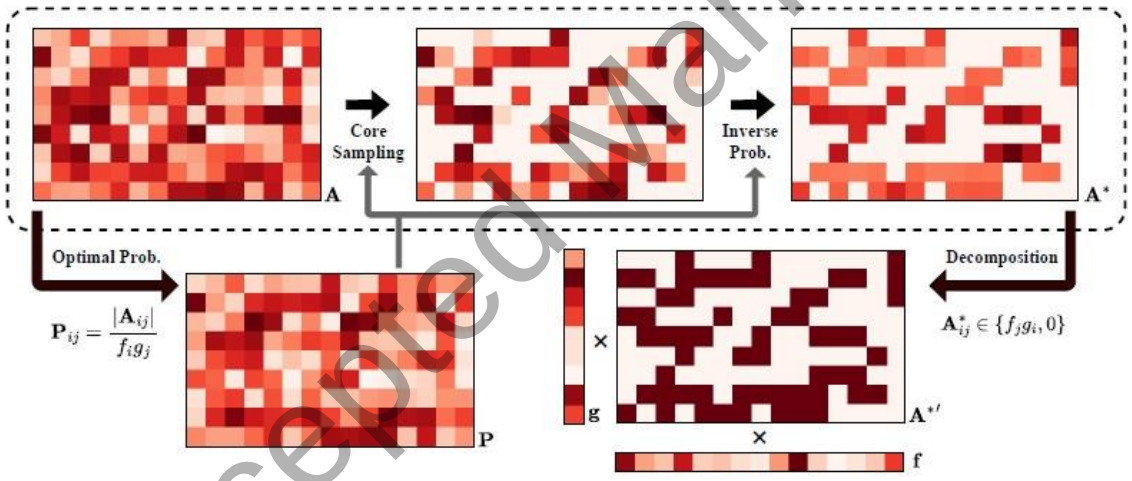

Fig. 2: Illustration of Algorithm 4 and Remark 2, demonstrating how the decomposition $\mathbf{A}^* = (\boldsymbol{f}\boldsymbol{g}^\top) \odot \mathbf{A}^{*\prime}$ leads to reduced storage requirements. Consequently, the resulting $\mathbf{A}^{*\prime}$ and $\boldsymbol{f}, \boldsymbol{g}$ have more compact representations compared to a general $\mathbf{A}^*$.

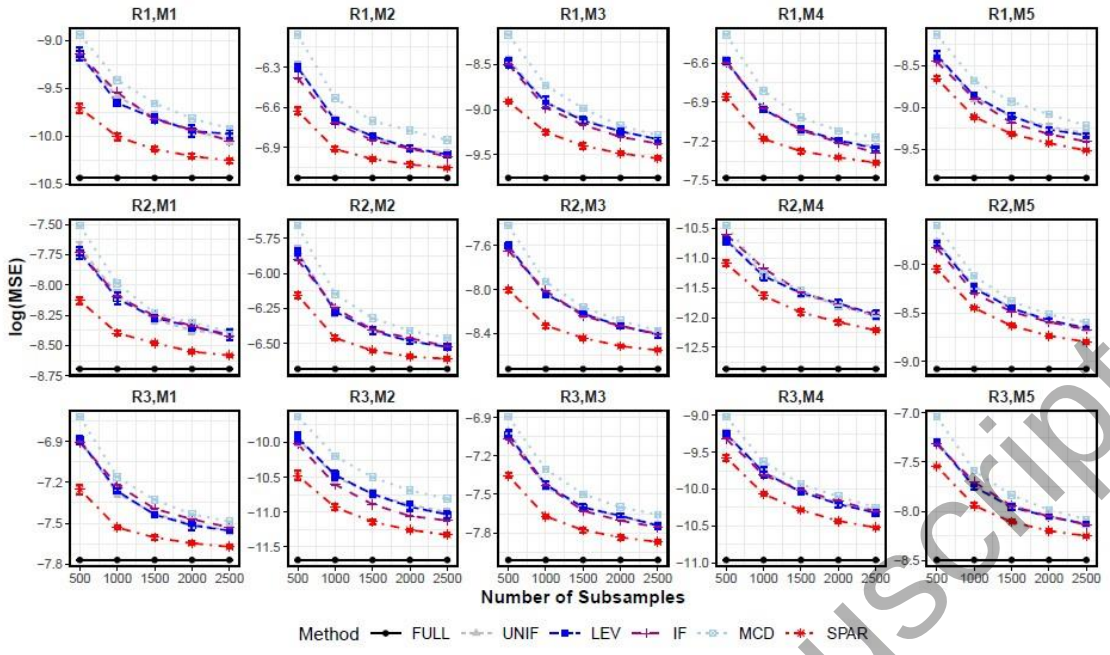

Fig. 3: Comparison of $\mathrm{MSE_B}$ for different algorithms across varying numbers of subsamples *r*. Each row represents the different SNR settings, i.e. (R1-R3), and each column represents the different latent variable dimensions, i.e. (M1-M5).

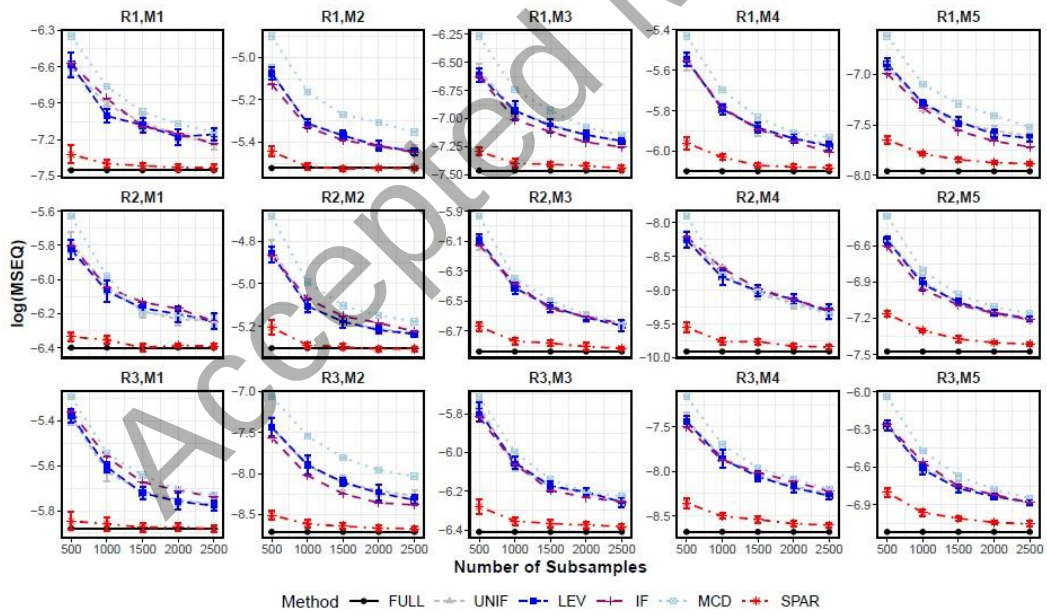

Fig. 4: Comparison of $\mathrm{MSE_Q}$ for different algorithms across varying numbers of subsamples *r*. Each row represents the different SNR settings, i.e. (R1-R3), and each column represents the different latent variable dimensions, i.e. (M1-M5).

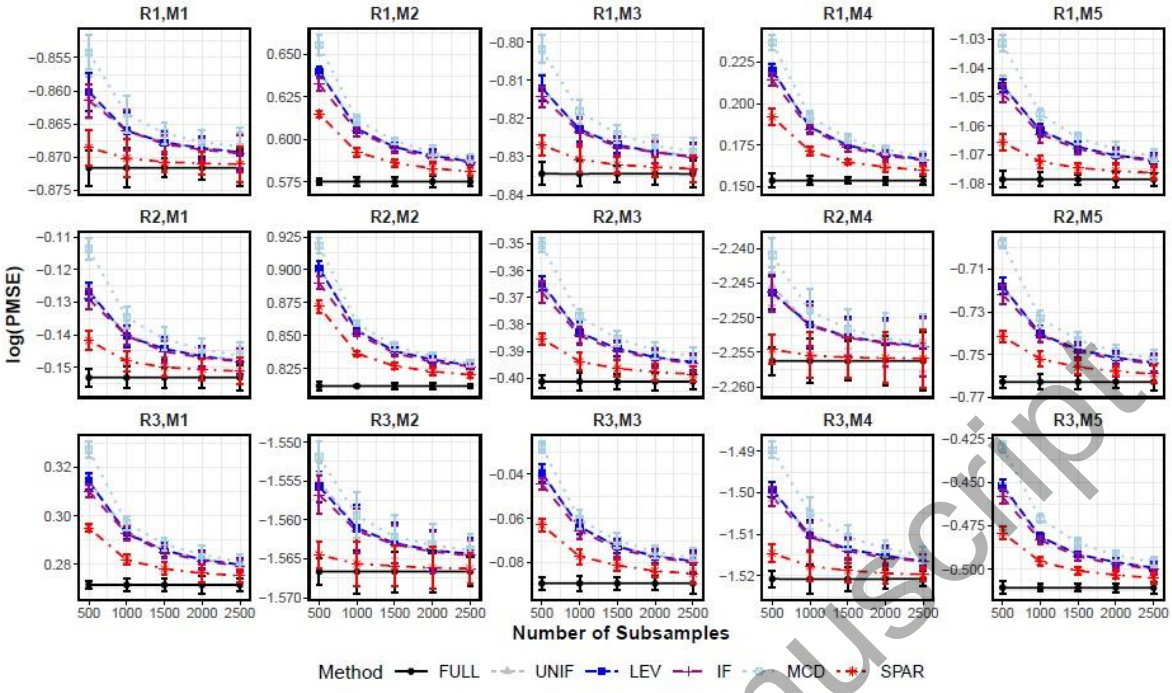

Fig. 5: Comparison of PMSE for different algorithms across varying numbers of subsamples *r*. Each row represents the different SNR settings, i.e. (R1-R3), and each column represents the different latent variable dimensions, i.e. (M1-M5).

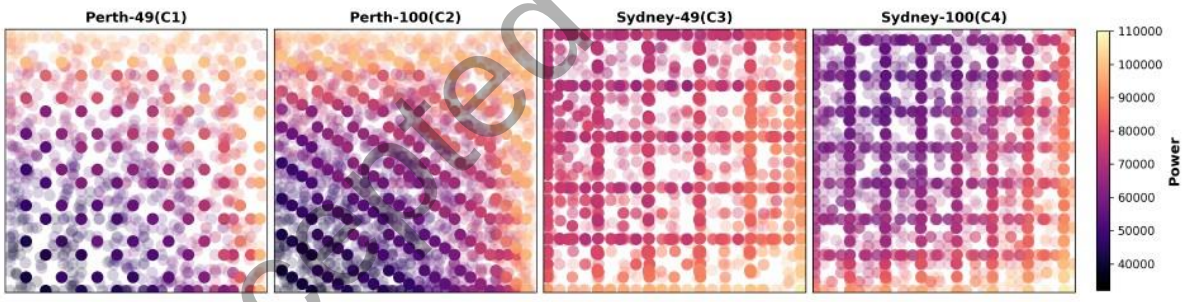

Fig. 6: Illustration of four scenarios from the WEC dataset. Each point represents the physical coordinates of a WEC, with the color intensity indicating the calculated power output at that location.

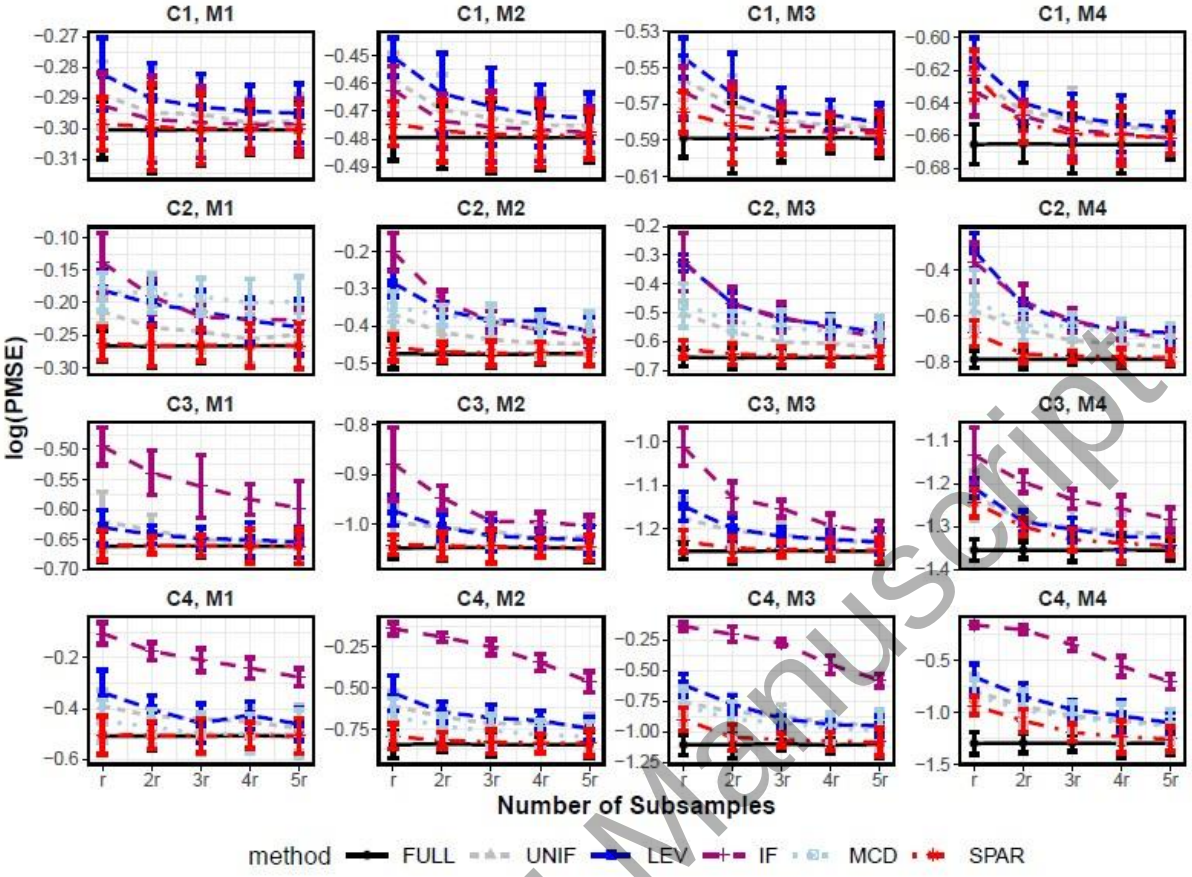

Fig. 7: Comparison of PMSE for different algorithms across varying numbers of subsamples *r*. Each row represents the different scenarios, i.e. (C1-C4), and each column represents the different latent variable dimensions, i.e. (M1-M4).

Table 1: Comparison for different PLS algorithms.

Methods	Pre. Cost*	Main Cost	Method Type	Acceleration Type
PLS (Dayal and MacGregor, 1997)	-	$\mathcal{O}(np^2 + mp^2l)$	-	-
UNIF	-	$\mathcal{O}(rp^2 + mp^2l)$	Randomized	Row Sampling
LEV	$\mathcal{O}(np^2)$	$\mathcal{O}(rp^2 + mp^2l)$	Randomized	Row Sampling
IFPLS (Xie et al., 2022)	$\mathcal{O}(np^2)$	$\mathcal{O}(rp^2 + mp^2l)$	Deterministic	Row Sampling
MCDPLS (Huang et al., 2024)	$\mathcal{O}(np^2L)$	$\mathcal{O}(rp^2 + mp^2l)$	Deterministic	Row Sampling
SGDPLS (Arora et al., 2016)	-	$\mathcal{O}(mpL)$	Randomized	Batch Learning
CIPLS (Jordao et al., 2021)	-	$\mathcal{O}(npl)$	Randomized	Batch Learning
Spar-PLS (proposed)	$\mathcal{O}(\text{nnz}(\mathbf{X}))$	$\mathcal{O}(rp^2 + mp^2l)$	Randomized	Sparse Sampling

[1] Here, each method subsamples *r* rows or *rp* elements from the $n \times p$ predictor matrix.

[2] L denotes the number of iterations required for convergence in the respective methods.

[3] The computational complexities presented above are derived under the assumption that $l < p$.

Table 2: Runtime Comparison of Different PLS Implementations (s)

# Subsamples	500	1000	1500	2000	2500	500	1000	1500	2000	2500
# Components	m=5					m=10				
FULL	0.447	0.465	0.456	0.479	0.460	0.682	0.684	0.643	0.651	0.659
UNIF	0.006	0.009	0.015	0.017	0.022	0.009	0.016	0.027	0.046	0.031
LEV	0.689	0.712	0.706	0.706	0.728	0.701	0.720	0.705	0.707	0.709
IF	1.253	1.252	1.248	1.265	1.278	1.497	1.473	1.511	1.483	1.459
MCD	1.085	1.078	1.084	1.073	1.093	1.067	1.097	1.090	1.089	1.093
SPAR	0.030	0.036	0.049	0.046	0.051	0.035	0.042	0.050	0.056	0.063

Table 3: Comparison of Running Times on Real Data (s)

#Components	$\mathbf{m=5}$	$\mathbf{m=10}$	$\mathbf{m=15}$	$\mathbf{m=20}$
FULL	0.330	0.480	0.622	0.777
UNIF	0.007	0.009	0.012	0.016
LEV	0.377	0.383	0.384	0.384
IF	0.848	0.992	1.138	1.363
MCD	0.491	0.490	0.493	0.544
SPAR	0.042	0.044	0.045	0.066