Stepwise Covariance-Free Common Principal Components (CF-CPC) With an Application to Neuroscience

Usama Riaz*, Fuleah A. Razzaq†, Shiang Hu and Pedro A. Valdés-Sosa*1,3

OPEN ACCESS

Edited by:
Adeel Razi,
Monash University, Australia

Reviewed by:
Stavros I. Dimitriadis,
Greek Association of Alzheimer’s Disease and Related Disorders,
Greece
Guido Nolte,
University Medical Center Hamburg-Eppendorf, Germany

*Correspondence:
Pedro A. Valdés-Sosa
pedro.valdes@neuroinformatics-collaboratory.org
Usama Riaz
usama.riazahmad@outlook.com
†These authors have contributed equally to this work and share first authorship

Specialty section:
This article was submitted to Brain Imaging Methods, a section of the journal Frontiers in Neuroscience
Received: 30 July 2021
Accepted: 15 October 2021
Published: 11 November 2021

Citation:
Riaz U, Razzaq FA, Hu S and Valdés-Sosa PA (2021). Stepwise Covariance-Free Common Principal Components (CF-CPC) With an Application to Neuroscience. Front. Neurosci. 15:750290. doi: 10.3389/fnins.2021.750290

Finding the common principal component (CPC) for ultra-high dimensional data is a multivariate technique used to discover the latent structure of covariance matrices of shared variables measured in two or more k conditions. Common eigenvectors are assumed for the covariance matrix of all conditions, only the eigenvalues being specific to each condition. Stepwise CPC computes a limited number of these CPCs, as the name indicates, sequentially and is, therefore, less time-consuming. This method becomes unfeasible when the number of variables p is ultra-high since storing k covariance matrices requires $O(kp^2)$ memory. Many dimensionality reduction algorithms have been improved to avoid explicit covariance calculation and storage (covariance-free). Here we propose a covariance-free stepwise CPC, which only requires $O(kn)$ memory, where n is the total number of examples. Thus for $n << p$, the new algorithm shows apparent advantages. It computes components quickly, with low consumption of machine resources. We validate our method CFCPC with the classical Iris data. We then show that CFCPC allows extracting the shared anatomical structure of EEG and MEG source spectra across a frequency range of 0.01–40 Hz.

Keywords: Ultra-high Dimensional Data, Covariance-free, Neuroimaging, EEG, MEG, common principal component (CPC)

INTRODUCTION

With exceptional advancements in data acquisition capabilities in recent years, there has been a rise in conducting large-scale neuroscience studies. Increased processing power with the availability of High-Power Computing (HPC) setups gives the neuroscience community ability to compute high-resolution spatial and temporal source imaging and source activity localization, especially in EEG and MEG data. These datasets are gathered with lots of different parameters. These parameters can be different due to age, gender, ethnicity, geographical location, capturing modality, and machine parameters.

Analyzing ultra-high-dimensional neuroimaging data has been time-consuming and challenging. There are many solutions, e.g., Principal Component Analysis (PCA), Independent Component Analysis (ICA), Incremental Principal Component Analysis (IPCA), and sparse
incremental Principal Component Analysis (sIPCA) (Yao et al., 2012; Balsubramani et al., 2013; Tang and Allen, 2018; Tabachnick and Fidell, n.d.) developed to overcome high dimensionality. However, only a small number of principal components can explain almost all the variance of a multivariate data and that is where stepwise computation of Principal Components comes. Stepwise PC only a compute a limited number of PCs sequentially. Power method is used to first compute the most dominant PC and then deflation parameter extracts the estimated variance from data (MacKey, 2009; Golub and Van Loan, 1996). Later, the dominant PC is again computed from the remaining data using same procedure, details are in methodology.

However, there are some scenarios when \( k \) different populations or groups, also known as conditions, are being compared or cross analyzed, and a common latent covariance structure is required. This phenomenon is also known as obtaining common principal components (CPCs). The conventional CPC algorithm was first introduced and studied by Flury (1984). CPC is one of many possible generalizations of standard PCA of several covariance matrices (Jolliffe, 2005). The initial motivation for introducing CPC was to study discrimination problems, where the covariance matrices for different conditions are not equal as required by linear discriminant analysis but more generally share a latent joint principal axis (Flury, 1984; Krzanowski, 1984). The CPC model was mainly criticized because it is essentially a method for simultaneous diagonalization of several positive definite matrices. Rather than a dimensionality reduction method, which is usually the main goal in data analysis (Schott, 1989).

To remedy this (Krzanowski, 1984), proposed a simple, intuitive procedure to estimate an approximation of the CPCs based on the PCA of the pooled sample covariance matrix and the total sample covariance matrix, followed by the comparison of their eigenvectors (Schott, 1989, 1999) proposed an improvement where the latent covariance structure spanned by the first \( m \) principal components (PC) and their sum is identical for any \( k \) conditions. This is called Common Subspace Analysis (CSA). These improvements try to achieve the aim of dimensionality reduction. However, CSA still used the inherited concept of finding a common subspace of all groups simultaneously, making it time-consuming and computationally expensive. To remedy this, stepwise CPC was proposed by Trendafilov (2010), which sequentially performs the CPCs functionalities. Stepwise CPC computes fewer latent components, making it computationally less expensive and achieving dimensionality reduction.

The motivation of this study comes from a problem we faced while conducting a previous study where we were trying to decompose the source spectra of EEG and MEG. This decomposition aims to remove pre-identified differences between the two spectra and to develop a transfer function. Estimating common topography between EEG and MEG spectra is one of the elements required for this decomposition process. However, both spectra are ultra-high dimensional, i.e., the number of variables \( p \) is much larger than the number of observations \( n \) (Hu, 2020; Riaz, 2021a). To compute a common latent subspace via stepwise CPC, we need the covariance matrix for all \( k \) conditions, i.e., covariances for EEG and MEG. Since computing covariance requires \( O(p^2) \) memory which can be time-consuming and even impossible when \( p \) is large. Thus, conventional stepwise CPC cannot be applied here, as it will require \( O(kp^2) \) memory space. A covariance-free CPC is required to compute a common latent subspace for ultra-high-dimensional data, which do not compute and save covariance matrix. Some covariance-free methods have been previously proposed to improve other dimensionality reduction methods. For example, IPCA was proposed by Weng et al. (2003) and Yousefi et al. (2017), iterative Kernal PCA proposed by Liao et al. (2010), incremental PCALDA by Dagher (2010), covariance free partial least squares by Jordan et al. (2021). Instead of working with covariance matrices, all these methods achieve dimensionality reduction without computing and saving covariance matrix, which decreases required memory to \( O(n) \).

This article proposes a novel method we call stepwise Covariance-Free CPC (stepwise CFCPC) by merging ”covariance free” and common latent subspace concepts. The following sections of the article are a methodology for CFCP then the description of the datasets we are using to test and validate our method. Later we present the results and compare stepwise CPC and stepwise CFCPC for accuracy, computation time, and memory consumption. Furthermore, we lay down the concluding remarks and suggest applications of this improvement.

**METHODOLOGY AND MATERIALS**

**Principal Components Principal Components**

PCA is a dimensionality reduction technique that computes Principal Components (PCs) to represent data by linear combination of a significantly less number of vectors. These vectors represent the maximum variance of data that a single vector can represent in a given direction. Other PCs that are orthogonal to their previous PC are computed, and every next PC represents a lesser amount of variance from the previous. Principal components are obtained from the covariance matrix of the data, then eigenvalues and eigenvectors of that covariance matrix are computed for dimensionality reduction (Tabachnick and Fidell, n.d.). For a given data generally, the PCs equal to the number of variables is computed. However, only a small number of PCs represent almost all the data variance, which is why only a few PCs are required. This phenomenon gives birth to the idea of stepwise Principal Components.

**Stepwise Principal Component**

Stepwise principal components compute a limited number of PCs and compute them sequentially, which is explicitly required when matrices are of larger sizes. The stepwise PC is obtained by first using the power method to compute the most dominant eigenvalue by normalizing the given matrix, computing the most dominant eigen value, and applying the deflation to extract the variance that has been estimated by \( \lambda \). It works on a diagonalizable square matrix, which in our case is a covariance matrix \( S_i \). The power method iteration algorithm for a given covariance matrix \( S_i \) works as shown below. Here \( \mu \) is the
normalized covariance matrix $S_i$, $\lambda$ is the estimated eigenvector, and $l_{\text{max}}$ are the total number of iteration for convergence. The Power Method Iteration Algorithm defined above gives the largest eigenvalue. The deflation method (MacKey, 2009) omits the estimated covariance to obtain new $S_i$ which is eventually used to compute the next $\lambda$. Stepwise-PC repeats this process iteratively for the given number of eigenvalues.

**Power Method Iteration**

for $i = 1, \ldots, l_{\text{max}}$

- $\mu \leftarrow S_i \mu$
- $\lambda \leftarrow \mu^T \mu$
- $\mu \leftarrow \mu / \sqrt{\lambda}$

end

**Deflation**

$S_i \leftarrow S_i - \lambda \mu \mu^T$

**Conventional Common Principal Component**

Conventional CPC is one of the generalizations of PCA for $k$ conditions and their covariance matrices, as mentioned in Flury (1984). In CPC, it is assumed that all $k$ conditions have the same mean, and their covariance matrices $S_i$ are all positive definite and diagonalizable.

$$H_{\text{CPC}} : Q^T S_i Q = D_i^2, \quad i = 1, 2, \ldots, k$$

(1)

Here $Q$ is the common orthogonal matrix for all conditions and $D_i^2$ is the positive orthogonal matrix for each condition. CPC estimations find the common eigenvectors and eigenvalues for covariance matrices $S_i$ for all $k$ conditions and $n_i (> p)$ degrees of freedom or number of observations such that:

$$S_i \approx QD_i^2Q^T$$

(2)

CPC computes the latent components all at once by computing maximum likelihood components of parameters $Q$ and $D_i^2$, $i = 1, 2, \ldots, k$ using the following optimization problem of minimizing negative log-likelihood as mentioned in Flury (1984).

Minimize $\sum_{i=1}^{k} n_i \log(\det(QD_i^2Q^T)) + \text{trace}(QD_i^2Q^T)^{-1} S_i$]

$$= \sum_{i=1}^{k} n_i \log(\det(\text{textbf{D}_i^2})) + \text{trace}(\text{D}_i^{-2} \odot (Q^T S_i Q))$$

(3)

Subject to $\{Q, D_1, D_2, \ldots, D_k\} \in \varnothing(p) \times D(p)^k$, (4)

Where $\odot$ is the Hadamard product, the Lie group of all $p \times p$ orthogonal matrices is denoted by $\varnothing(p)$ and $D(p)^k = D(p) \times \ldots \times D(p)$. Here $D(p)$ is the linear subspace of all $p \times p$ diagonal matrices. However, to satisfy the first-order optimality condition for a stationary point of CPC objective function, $\sum_{i=1}^{k} n_i Q^T S_i Q_{ij} = D_j^2$ for all $k + 1$ conditions simultaneously. After submitting these values in Equation (3), we can define CPC estimation as the following likelihood problem.

Minimize $\sum_{i=1}^{k} n_i \log(\det(Q^T S_i Q))$ (5)

Subject to $Q \in \varnothing(p)$ (6)

Here $Q$ is the set of all orthogonal matrices that contain all the CPCs, which are computed simultaneously. This is a basically FG diagonalization solution replacing the original problem mentioned in Equations (3) and (4). CPC works efficiently for covariance matrices $S_i$ that are positive definite and positive semi-definite as well.

**Stepwise Common Principal Component**

Stepwise CPC is performed by imitating the standard PCA to achieve dimensionality reduction, i.e., finding the latent components one after another (Trendafilov, 2010). It does not compute all the CPCs at once, and instead, it computes only a desired number of CPCs sequentially. First, it transforms the CPC problem into a vectorize form and then solves the $p$ identical problems expressed in Equations (7) and (8) sequentially.

Minimize $\sum_{j=1}^{k} n_i \log(q_j^T S_i q_j)$

(7)

Subject to $q_j^T q = 1$ (8)

Here $q$ are the common PCs computed sequentially. Stepwise CPC is done by finding the first CPC which will be $q_p$ which will give a minimum of Equation (7) in the unit sphere in $\mathbb{R}^p$, then the next CPC is found that will be $q_{p-1}$ which will give a minimum of Equation (7) in the unit sphere in $\mathbb{R}^p$ and it will be orthogonal to $q_p$. So, each minimum found using this idea will be greater than the previous one as it is found in the orthogonal domain of the previous minimization domain. Since the quantities $q_j^T S_i q_j$ are bounded by the smallest and largest eigenvalues of $S_i$, the objective function in Equation (7) will be bounded on a unit sphere $\mathbb{R}^p$. However, for the purpose of dimensionality reduction, it is more feasible to obtain the CPCs in reverse order, i.e., representing all CPCs by a variational eigenvalues definition represented in Hord and Johnson (2012) as $Q_p = \{q_1, q_2, \ldots, q_p\}$ which $p \times p$ orthogonal matrix containing the CPCs obtained from Equation (7) and (8). So the $jth$ CPC can be shown by the optimization problem shown in Equations (9) and (10).

Minimize $\sum_{j=1}^{k} n_i \log(q_j^T S_i q_j)$

(9)

Subject to $q_j^T q = 1$ and $q_{j}^T Q_{j-1} = 0_{j-1}^T$ (10)
Where $Q_{j-1} = [q_1, q_1, ..., q_{j-1}]$ and the $j$th CPC can be obtained by solving $j$ problems similar to the ones shown in Equations (9) and (10). This eventual solution will be similar to the one computed in the original CPC shown in Equations (5) and (6) but has the feature that it can be stopped at any point that is $1 \leq j \leq p$. To compute the next eigenvector, one can use an already computed $Q_j$ and do not need to compute all the eigenvectors from the start. The first-order optimality conditions for Equations (9) and (10) can be resolved by Theorem 3.1 in Trendafilov (2010) study. The equation resolves to obtain the first-order optimality condition is shown in Equation (11).

$$\Pi_j \left( \sum_{i=1}^{k} n_i S_i \right) q_j = 0_{p \times 1}$$

(11)

Where $\Pi_j$ the deflation parameter or projector $I_p = B_j Q_j^T$ and $\sum_{i=1}^{k} n_i S_i q_j$ is the gradient of CPC objective function from Equation (9). The theorem solves the first-order optimality condition and gives a general case as shown in Equation (13) and for $j = 1$ as shown in Equation (12).

$$\left[ n_1 S_1 q_1 q_1^T + \cdots + n_k S_k q_k q_k^T \right] - n I_p q_1 = 0_{p \times 1},$$

(12)

For $j = 2, ..., p$

$$\left[ (I_p - Q_{j-1} Q_{j-1}^T) \left[ n_1 S_1 q_1 q_1^T + \cdots + n_k S_k q_k q_k^T \right] - n I_p \right] q_j = 0_{p \times 1}.$$  

(13)

Equations (12) and (13) show that the CPCs can be computed by solving $p$ symmetric eigenvalues problems. Trendafilov (2010) solves this problem by using a modified version of the standard power method to solve the eigenvectors and eigenvalues problem to compute the CPCs defined by Equations (12) and (13). The Power Method Iteration algorithm in section “Stepwise Principal Components” is used to compute the CPCs iteratively. The modification is that this algorithm updates the covariance matrix at each step. In particular, this is an algorithm of gradient ascend category—Algorithm 1 in Appendix where the indices of the power iteration are given in the parenthesis. The resultant vector $q_j, j = 1, 2, ..., p$ from Algorithm 1 is the CPCs. A small number of iterations (even less than 3) are required if the eigenvectors are well separated.

### Stepwise Covariance-Free Common Principal Component Implementation

Stepwise, CPC implements (Equations 4–7) by taking the covariance matrices $S_i$ for $k$ conditions, the number of common latent PCs to be computed is $p_{max}$, and $l_{max}$ is the number of iterations required for convergence. Refer to the Algorithm 1 in Appendix. The stepwise CPC algorithm computes only a specific number of CPCs. As output, it gives eigenvectors $Q_{p_{max}}^j$ for common latent sub-space under all $k$ conditions along with $\lambda_{p_{max}}$ the eigenvalues for each particular condition. However, the problem with stepwise CPC arises when the number of variables $p$ becomes too large.

### Stepwise CF-CPC

The idea behind covariance free stepwise CPC is to not compute the covariance at any step of the algorithm. To achieve this purpose, we propose that instead of calculating the covariance, replace it with its mathematical definition of covariance and apply this concept in the basic Algorithm 1 of stepwise CPC. This can be done by expanding the covariance formula, as shown in Equation (14).

$$S_j = \frac{(H_j X_j^T)(H_j X_j)}{n_j}$$

(14)

Here $H$ is the average reference when applied to the will replicate the subtraction of mean from the data that is used conventionally to compute covariance, which is by definition $H_i = I_i - \frac{I_i I_i^T}{n_i}$ and $X_i$ is actual data for which the covariance was computed for stepwise CPC. Equation (14) can also be written as Equation (15).

$$S_j = W_j^T W_j$$

(15)

Here $W_j$ can be defined as $\frac{H_j X_j}{\sqrt{n_j}}$ making Equation (14), as shown in Equation (15).

Using this technique, we can achieve the same results while simplifying and changing the computations and formulations to make it covariance-free, requiring $O(kn)$ memory for $k$ conditions where $n << p$. Stepwise-CFCPC is fast, memory efficient, and will be optimal for ultra-high dimensional data.

### Stepwise CF-CPC Implementation

Algorithm 2 (refer to Appendix) explains the flow of how stepwise CF-CPC works. We are trying to achieve this algorithm to make it covariance-free and optimize it in terms of computations and memory usage. The inputs $p_{max}, l_{max}$ and $n$ are the same as for the original stepwise-CPC, i.e., the following are the steps we have taken for optimization purposes:

- Algorithm 2 takes data $X_i$ as input instead of the covariances $S_i$, which saves memory and computation power.
- We perform singular value decomposition (SVD) (Klema and Laub, 1980) instead of estimating eigenvectors (Andrew, 1973).

$$\Pi_l = I_p - \sum_{a=1}^{j-1} q_a q_a^T$$

(16)

for $j = 1, ..., p_{max}$

- Furthermore, we replace the deflation parameter $\Pi_j$ with its mathematical definition as Equation (9) (MacKey, 2009).
- We substitute the sum of covariances with a matrix formed when the definition of covariance is used from Equation (14).

Initially, the stepwise CPC was implemented for an R-package named “cpcac” (Ziyatdinov et al., 2014). Since our working
environment is MATLAB 2018b (The MathWorks, n.d.), we implemented and verified the stepwise CPC in MATLAB and the same for stepwise CFCPC.

Materials

To test the covariance-free version of stepwise CPC, we used two datasets. The first data we analyzed (for validation purposes) was Fisher's IRIS data (Anderson, 1935; Fisher, 1936), and this is the same data that was analyzed in the original stepwise CPC algorithm (Trendafilov, 2010). Secondly, we have analyzed neuroscience data.

Iris Data

We tested our stepwise CF-CPC algorithm initially on Fisher's IRIS data. The purpose is to validate if the covariance-free stepwise CPC gives the same results as stepwise CPC. Fisher IRIS data has 150 examples/observations and four variables, making it a matrix. To transform the data into , we divided that data into chunks of 50 examples with four variables in each chunk, making three matrices of size . Later we tested both stepwise CPC and stepwise CFCPC on this data for validation purposes.

MEEG Data

The neuroimaging data we are using comprises source spectra of two modalities, electroencephalography (EEG) and magnetoencephalography (MEG), with 45 subjects in each group. EEG subjects were picked from an extensive database of the Cuban Human Brain Mapping project (CHBM) (Valdes-Sosa et al., 2021). MEG data of a similar sample size of 45 were picked fromHuman Connectome Project (HCP) (WU - Minn Consortium Human Connectome Project, 2017). The EEG data we used was source spectra computed from a novel inverse solution BC-VARETA (Gonzalez-Moreira et al., 2018; Paz-Linares et al., 2018). The size of the source spectra is . Here the first dimension representing the number of brain sources/generators. The second dimension is the number of frequency points (in this case, there are frequency points with a step-size of 0.5 Hz, so the total analyzed source spectra was – 40 Hz), and the third dimension represents the number of subjects involved in the study. MEG source spectra were computed using the same inverse solution used of EEG data, and the size of the source spectra matrix was also the same. Before computing the common principal spectral component, we scaled data and log-transformed it for visualization. Since we are analyzing sources at each of the frequency points, we rearranged the source spectra to convert them into a 2D array of size . There is a total of variables, with each group of variables representing one frequency point. So, the inputs go into both algorithms of stepwise CPC are \( p = 8002 \times 80 = 640160 \), \([45 \times 45]\) and \( k = 2 \) where \( p >> n \).

Stepwise Common Principal Component and Stepwise CFCPC Computation

We computed the CPC subspace score along with eigenvalues for each modality using both algorithms. The same dataset and machine specifications are used to compare the results in terms of time consumed to compute the CPC. The specifications are explained below:

- Intel(R) Core(TM) i5-7200U CPU @ 2.50GHz 2.70 GHz
- 16.0 GB DDR3 RAM
- Windows 10–64-bit operating system, x64-based processor
- MATLAB 2018b (The MathWorks, Inc, 1994-2021)

We observed that computing covariance for CPC above a certain number of variables is impossible on this machine because of “out of memory” issues. Additionally, there is a limit for the number of variables on which computing covariance was successful. However, inside the stepwise CPC algorithm, it returns the error of running out of memory. So, we tested both algorithms for a specific number of variables to check two things:

- What is the limit of the number of variables for both algorithms?
- How much time each algorithm takes to compute a certain number of variables.

Once we have verified the number of variables for which the CPCs are successfully computed, we computed the CPCs for the MEEG data discussed above. These CPCs will be the shared space eigenvectors for the sources spectra captured using two different modalities, i.e., EEG and MEG. Once these CPCs are computed, we applied the eigenvalues on the shared common subspace to visualize the common space generated for EEG and MEG, respectively. In the end, we compared the original EEG and MEG spectra with the respective estimated common subspace for accuracy.

RESULTS

As mentioned earlier, we need to compute covariance first and input that to the algorithm for stepwise CPC. For stepwise CF-CPC, pass the iris data as it is. We computed four CPCs \( Q \) and their scale factor \( \lambda \) for both versions of stepwise CPC. The results show that values for all four outcomes of both \( Q \) and \( \lambda \) for Algorithm 1 and Algorithm 2 are the same as shown in Figure 1.

Next, we used neuroscience data as explained in materials sections and applied both algorithms to look for time and memory consumption results. To compute one CPC with both methods, we keep the values for \( p_{\text{max}} = 1 \), \( l_{\text{max}} = 1 \) & \( n = [45 \times 45] \) for both algorithms. We recorded computation time and looked for “out of memory” errors for a series of variables selected from the EEG and MEG datasets and recorded the results in the form of Table 1. It is evident in the table that stepwise CFCPC works smoothly on all different sets of variables from 100 till 640160, not giving any memory errors, and the execution time is very nominal. However, stepwise CPC observes an exponential rise in the execution time when the number of variables is increased. Additionally, stepwise CPC could not compute a single CPC beyond 15,000 variables, and covariance computation was not successful beyond 25,000 variables on the defined system specifications.
TABLE 1 | Comparison of stepwise CPC and stepwise CFCPC (execution time and memory consumption comparison). Successful computation of covariance for a different number of variables. Success or failure in execution of Algorithm 1 and 2 with execution time for the different number of variables.

| Method       | Variables | Covariance computation | Algorithm execution | Time consumed |
|--------------|-----------|------------------------|---------------------|---------------|
| Stepwise CPC | 100       | Success                | Success             | 0.104310s     |
|              | 500       | Success                | Success             | 0.588175s     |
|              | 1,000     | Success                | Success             | 3.158705s     |
|              | 5,000     | Success                | Success             | 441.268313s   |
|              | 8,002     | Success                | Success             | 2000.196520s  |
|              | 15,000    | Success                | Success             | 6607.873001s  |
|              | 25,000    | Success                | Failed              | NA            |
|              | 50,000    | Failed                 | Failed              | NA            |
|              | 100,000   | Failed                 | Failed              | NA            |
|              | 200,000   | Failed                 | Failed              | NA            |
|              | 400,000   | Failed                 | Failed              | NA            |
|              | 640,160   | Failed                 | Failed              | NA            |
| Stepwise CFCPC | 100     | Success                | Success             | 0.011920s     |
|              | 500       | Success                | Success             | 0.035195s     |
|              | 1,000     | Success                | Success             | 0.056952s     |
|              | 5,000     | Success                | Success             | 0.507487s     |
|              | 8,002     | Success                | Success             | 0.746992s     |
|              | 15,000    | Success                | Success             | 1.378456s     |
|              | 25,000    | Success                | Success             | 2.206656s     |
|              | 50,000    | Success                | Success             | 4.378850s     |
|              | 100,000   | Success                | Success             | 8.402318s     |
|              | 200,000   | Success                | Success             | 16.705613s    |
|              | 400,000   | Success                | Success             | 34.469083s    |
|              | 640,160   | Success                | Success             | 65.797402     |

FIGURE 1 | Stepwise CPC vs. covariance free stepwise CPC.

The trend of execution time for both algorithms can be visualized as shown in Figure 2. Stepwise CFCPC consumes less time than stepwise CPC even when the number of variables is low. This execution time remains stable when the number of variables is increased from 100s to 1,000s for stepwise CFCPC. However, stepwise CPC resulted in an exponential increase in computation time as variables are increased. It took almost 5,000-folds more time for the maximum number of variables it could compute CPC successfully. We visualized the computation of CPC for EEG and MEG data for its accuracy and interpreting the source spectra of both modalities. The purpose of computing CPC for these source spectra is to compute a common source topography captured by both modalities in different conditions and find a scale factor or eigenvalue assigned to each modality's source spectra to compute its respective common topography. These common topography and scale factors will be used as an ingredient in another study to identify and remove the differences between the source spectra of EEG and MEG (Riaz, 2020). A visualization of how this common topography is acquired is shown in Figure 3A. The common source topography represents common topographical features of spectra when visualized in the frequency domain. One of the clear components is the alpha peak visible in the alpha band (7-12 Hz). The computed CPC for common topography has the common subspace eigenvectors and eigenvalues for EEG and MEG spectra. These eigenvalues are applied on the computed common subspace estimate the common topography scores for EEG and MEG, as shown in Figure 3B. The solid lines are the original EEG and MEG spectra, whereas the dotted lines show the latent estimated common topography for each modality. This estimated common topography is obtained by applying the scale factor or eigenvalue $\lambda$ to the common topography $Q$ shown in Figure 3A.
The estimated common topographies shown in dotted lines depict that CPC has successfully computed the desired common latent subspace of common topography for each modality. The difference in the original spectra and estimated common topography can be explained because we are only estimating one of the few elements from which the complete source spectra is composed. This common latent subspace computed for common topography between EEG and MEG source spectra is the baseline for the transferal of source spectra between EEG and MEG, as discussed in Riaz (2020) and Riaz (2021b). In the mentioned studies, we are trying to create a transfer function from EEG to MEG source spectra and vice versa. We decompose each spectra into three components, i.e., Common Topography with scale values (computed using the algorithm of this study), Xi-process, and individual differences. These elements are used to synthetic MEG source spectra from EEG and vice versa.

DISCUSSION

In this study, we developed stepwise CFCPC for computing common stepwise subspace in ultra-high dimensional data. First, we validated results stepwise CPC and CFCPC and found that the computation of $Q$ and $\lambda$ with this new method is similar to stepwise CPC. Once our proposed improvement is validated, we tested the neuroimaging dataset (source spectra from two popular databases, i.e., EEG from CHBM and MEG from HCP) for a single CPC to compare the efficiency of both algorithms. The common topographical subspace obtained for EEG and MEG source spectra can be used as a baseline to analyze the phenomenon that the captured source spectra with EEG and MEG pose the same information gathered in different scenarios, conditions, or settings. However, when we try to compute a single CPC with conventional stepwise CPC, it turns out to be highly computationally expensive. The computationally expensive nature of stepwise CPC is evident in Table 1 as well that as the number of variables is increased from 100s to 1,000s, the execution time is increased a lot. However, this is not the case when we used our proposed CFCPC to compute a single CPC. An increase in the number of variables did not affect the computation time too much. In fact, for the maximum number of variables, 640160 CFCPC took just beyond a minute to compute a single CPC, which is not the case for stepwise CPC, which failed to compute CPC beyond 15,000 variables. These results prove our initial assumption that computing covariance for ultra-high dimensional data can be computationally expensive and takes a substantial amount of memory. It is also evident from Table 1 that beyond 15,000 variables, computing covariance took the system to out of
memory error. In stepwise CFCPC, we did not encounter any out-of-memory error, which we are trying to achieve using this algorithm.

There is a rise in the processing and dimensionality reduction in high dimensional datasets with exponential data gathering capabilities. Stepwise CFCPC can be a global solution when computing one or multiple CPCs, and computing covariance can be an issue. The issue of computing covariance for ultra-high dimensional data can occur in other areas of neuroscience (raw EEG, MEG data, neurogenetics Data (Genome-Wide Association Study of Parkinson's Disease: Genes and Environment, n.d.) and many other fields like microarrays in genetics, data from an array of sensors to capture geological or astronomical data, etc. (Bonham-Carter et al., 1990; Pesenson et al., 2010; Alonso-Betanzos et al., 2019). In repetitive resampling like bootstrap, where multiple CPCs are required, we can use stepwise CF-CPC. In this study, we have analyzed 640160 variables which is ultra-high dimensional data. Even most variables in genetics are less than the number of variables handled in this study which tells us that the application of this covariance-free stepwise CPC is valid even in genetics and another ultra-high dimensional datasets.

This version of stepwise CFCPC has its applications in neuroimaging, where CPCs are used to extract common features or principal components of multivariate data. Li (2016) in his study used CPC to perform classification on a multivariate time series EEG data for different clusters obtained from the original time series. Similarly, extracting common spatial patterns (CSP) from multivariate EEG data using CPC is achieved in many studies involving Brain-Computer Interface (BCI) (Wei et al., 2005; Wang and Wu, 2008; Meisher et al., 2018). Similarly, another study conducted interpretable principal components analysis on multilevel multivariate functional data to decompose total variation into subject-level and replicate-within-subject level (i.e., electrode-level) variation. This decomposition provides interpretable components that can be sparse among variates (e.g., frequency bands) and have localized support over time within each frequency band (Zhang et al., 2019). All these studies use CPC to compute their PCs; CSPs from multivariate data with high dimensions and stepwise CFCPC can help resolve these multivariate problems in a lesser amount of time which is especially essential in the case of BCI.

Similarly, in the domain of genetics, where the data is also multivariate and high dimensional, principal components are often required for dimensionality reduction in data analysis. Many studies estimate the time-scale for the evolution of additive genetic variance-covariance matrices (G-matrices), which is a crucial issue in evolutionary biology and genetics (Arnold and Phillips, 1999; Steppan et al., 2002; Conner et al., 2003; Mezey and Houle, 2003; Cheverud and Marroig, 2007). This comparison is also essential to see if different populations have the same genetic structure. This comparison of variances-covariance, i.e., G-matrices, is a high-dimensional problem and is often done using CPCs. Stepwise, CFCPC can help optimize these problems in terms of time and resources.

CONCLUSION

We propose a covariance free improved version of Stepwise CPC initially proposed by Trendafilov (2010). Computing covariance in ultra-high dimensional data causes the failure of the original algorithm as it consumes too much time and goes out of memory error for a large number of variables. The motivation for this study was an analysis of MEEG data where the traditional approaches fail. In contrast, our proposed stepwise CFCPC can work efficiently for ultra-high dimensional data while consuming very minimal memory and taking a small amount of time. Stepwise-CFCPC can be a global solution for computing CPCs for datasets in neuroscience (Areshenkoff et al., 2021), bioinformatics (Foo et al., 2017), gene microarray (Alonso-Betanzos et al., 2019), and multisensory node systems (Bonham-Carter et al., 1990; Pesenson et al., 2010). The improvements in the current algorithm can be finding the optimal number of CPCs to be computed. Currently, we are giving required CPCs to be computed as per data requirements. Further improvements can be made by applying the improvements for CPC analysis suggested in Fernández-Albert et al. (2014), Duras (2020), and Bagnato and Punzo (2021). Another direction for future analysis can be implementing stepwise CFCPC with sparsity conditions.

DATA AVAILABILITY STATEMENT

Publicly available datasets were analyzed in this study. This data can be found here: CHBM, https://www.nature.com/articles/s41597-021-00829-7; HCP, https://www.humanconnectome.org/storage/app/media/documentation/s1200/HCP_S1200_Release_Reference_Manual.pdf.

ETHICS STATEMENT

The studies involving human participants were reviewed and approved by CNEURO, Cuba for CHBM and NIH, United States for HCP data. The patients/participants provided their written informed consent to participate in this study.

AUTHOR CONTRIBUTIONS

PV-S gave the conception and design of the study, supervised the development, experimentation, and writing of the manuscript. UR and FR developed and implemented stepwise-CFCPC algorithm, performed the experiments, wrote the manuscript from first to final draft. SH changed the CPCA R-package to MATLAB script. All authors contributed to manuscript revision, read, and approved the submitted version.

FUNDING

The authors were funded from the University of Electronic Sciences and Technology grant Y03111023901014005 and the National Science Foundation of China (NSFC) with the grants 61871105 81861128001 and 62101003.
REFERENCES

Alonso-Betanzos, A., Bolón-Canedo, V., Morán-Fernández, L., and Sánchez-Maroto, N. (2019). "A review of microarray datasets: where to find them and specific characteristics," in Microarray Bioinformatics, Vol. 1986, eds V. Bolón-Canedo and A. Alonso-Betanzos (New York, NY: Humana), 65–85. doi: 10.1007/978-1-4939-9442-7_4

Andrew, A. L. (1973). Eigenvectors of certain matrices. Linear Algebra Appl. 7, 151–162. doi: 10.1016/0024-377X(73)90049-9

Anderson, E. (1935). The IRISs of the Gaspe peninsula. Bull. Am. Iris Soc. 39, 2–15.

Areshenkoff, C. N., Nashed, J. Y., Hutchison, R. M., Hutchison, M., Levy, R., Cook, D. J., et al. (2021). Muting, not fragmentation, of functional brain networks under general anesthetics. Neuroimage 231:117830. doi: 10.1016/j.neuroimage.2021.117830

Arnold, S. J., and Phillips, P. C. (1999). Hierarchical comparison of genetic variance-covariance matrices. II. coastal-inland divergence in the garter snake, Thamnophis elegans. Evolution 53, 1516–1527. doi: 10.2307/2640897

Bagnato, L., and Punzo, A. (2021). Unconstrained representation of orthogonal matrices with application to common principal components. Comput. Stat. 36, 1177–1195. doi: 10.1007/s00180-020-01041-8

Balubramani, A., Dasgupta, S., and Freund, Y. (2013). "The fast convergence of incremental PCA," in Proceedings of the Advances in Neural Information Processing Systems, (Noida: NIPS). doi: 10.1016/j.compbiomed.2021.104502

Bonham-Carter, G. F., Agterberg, F. P., and Wright, D. F. (1990). "Integration of geological datasets for gold exploration in Nova Scotia," in Introduction to Readings in Geographic Information Systems, eds D. J. Pequett and D. F. Marble. (Boca Raton, FL: CRC Press), 170–182.

Cheverud, J. M., and Marroig, G. (2007). Comparing covariance matrices: random skewers method compared to the common principal components model. Genet. Mol. Biol. 30, 461–469. doi: 10.5900/S4145-47572007000300027

Conner, J. K., Franks, R., and Stewart, C. (2003). Expression of additive genetic variances and covariances for wild radish floral traits: comparison between field and greenhouse environments. Evolution 57, 487–495. doi: 10.1111/j.0014-3820.2003.tb01540.x

Dagher, I. (2010). Incremental PCA-LDA Algorithm. Int. J. Biom. Bioinformatics, 4, 86–99.

Duras, T. (2020). The fixed effects PCA model in a common principal component environment. Commun. Stat. Theory Methods [Epub ahead of print]. doi: 10.1080/03610926.2020.1765255

Fernández-Albert, F., Llorach, R., García-Aloy, M., Ziyatdinov, A., Andres-Lacueva, C., and Perera, A. (2014). Intensity drift removal in LC/MS metabolomics by common variance compensation. Bioinformatics 30, 2899–2905. doi: 10.1093/bioinformatics/btu423

Fisher, R. A. (1936). The use of multiple measurements in taxonomic problems. Ann. Eugen. 7, 179–188. doi: 10.1111/j.1469-1809.1936.tb02137.x

Flury, B. N. (1984). Common principal components in K groups. J. Am. Stat. Assoc. 79, 892–898.

Foo, J. N., Tan, L. C., Iwans, I. D., Au, W. L., Low, H. Q., Krishak, P. M., et al. (2017). Genome-wide association study of Parkinson’s disease in East Asians. Hum. Mol. Genet. 26, 226–232. doi: 10.1093/hmg/ddw379

Golub, G. H., and Van Loan, C. F. (1996). Matrix Computations. Johns Hopkins University Press 3rd Edn.

Gonzalez-Moreira, E., Paz-Linares, D., Martinez-Montes, E., Valdés-Hernández, P. A., Bosch-Bayard, J., brings-Vega, M. L., et al. (2018). Cauking the leakage effect in MEEG source connectivity analysis. Arxiv [Preprint]. Available online at: https://arxiv.org/abs/1802.09046 (accessed July 8, 2021).

Hayney, J. G., and Houle, D. (2003). Comparing G matrices: are common principal components informative? Genetics 165, 411–425. doi: 10.1093/genetics/165.1.411

Paz-Linares, D., Gonzalez-Moreira, E., Martinez-Montes, E., Valdés-Hernández, P. A., Bosch-Bayard, J., Bringas-Vega, M. L., et al. (2018). Cauking the leakage effect in MEEG source connectivity analysis. Arxiv [Preprint]. Available online at: https://arxiv.org/abs/1801.00786v2 (accessed July 8, 2021).

Song, Y., and Van Loan, F. C. (1996). A fast iterative kernel PCA feature extraction for hyperspectral images, in Proceedings of the International Conference on Image Processing, Icip, (Piscataway, NJ: IEEE). 1317–1320

MacKey, L. (2009). "Deflation methods for sparse PCA," in NIPS’08: Proceedings of the 21st International Conference on Neural Information Processing Systems, eds D. Koller, D. Schuurmans, Y. Bengio, L. Bottou. (New York, NY: Curran Associates Inc.), 1017–1024.

Meisher, H., Ramnao, N., and Mitra, S. (2018). Multiclass common spatial pattern for EEG based brain computer interface with adaptive learning classifier. Arxiv [Preprint]. Available online at: https://arxiv.org/abs/1802.09046 (accessed July 8, 2021).

Negre, I. G., and Houle, D. (2003). Comparing G matrices: are common principal components informative? Genetics 165, 411–425. doi: 10.1093/genetics/165.1.411

Paz-Linares, D., Gonzalez-Moreira, E., Martinez-Montes, E., Valdés-Hernández, P. A., Bosch-Bayard, J., Bringas-Vega, M. L., et al. (2018). Cauking the leakage effect in MEEG source connectivity analysis. Arxiv [Preprint]. Available online at: https://arxiv.org/abs/1801.00786v2 (accessed July 8, 2021).

Pesenson, M. Z., Pesenson, I. Z., and McCollum, B. (2010). The data big bang and the expanding digital universe: high-dimensional, complex and massive data sets in an inflationary Epoch. Adv. Astron. 2010:350891. doi: 10.1155/2010/350891

Riaz, U. (2021a). Identifying and eliminating differences between EEG and MEG source spectra. Neuroinform. Assem. [Epub ahead of print].

Riaz, U. (2021b). Identifying and eliminating differences between EEG and MEG source spectra. Neuroinform. Assem. [Epub ahead of print].

Riaz, U. (2021a). Identifying and eliminating differences between EEG and MEG source spectra. Neuroinform. Assem. [Epub ahead of print].

Riaz, U. (2021b). Transferal from EEG to MEG. Int. J. Psychophysiol. 168:810. doi: 10.1016/j.jipysco.2021.07.027

Schott, J. R. (1989). Common principal component subspaces in two groups. Biometrika 76:408. doi: 10.1093/biomet/76.2.408

Schott, J. R. (1999). Partial common principal component subspaces. Biometrika 86, 899–908. doi: 10.1093/biomet/86.4.899

Steppan, S. J., Phillips, P. C., and Houle, D. (2002). Comparative quantitative genetics: evolution of the G matrix. Trends Ecol. Evol. 17, 320–327. doi: 10.1016/S0169-5347(02)02505-3

Tabachnick, B. G., and Fidell, L. S. (n.d.). Using Multivariate Statistics. Manhattan, NY: Harper & Row.1–14.

Tang, T. M., and Allen, G. I. (2018). Integrated principal components analysis. Arxiv [Preprint]. Available online at: http://arxiv.org/abs/1810.00832 (accessed July 8, 2021).

The MathWorks (n.d.). Mathworks. MATLAB. Trendaellite, N. T. (2010). Stepwise estimation of common principal components. Comput. Stat. Data Anal. 54, 3446–3457. doi: 10.1016/j.csda.2010.03.010

Valdés-Sosa, P. A., Galan-Garcia, L., Bosch-Bayard, J., Bringas-Vega, M. L., Aubert-Vazquez, E., Rodriguez-Gil, I., et al. (2021). The cuban human brain mapping project, a young and middle age population-based EEG, MRI, and cognition dataset. Sci. Data 8:45. doi: 10.1038/s41597-021-0087-7

Wang, L., and Wu, X. P. (2008). "Classification of four-class motor imagery EEG data using spatial filtering," in Proceedings of the 2nd International Conference on...
Stepwise Covariance-Free Common Principal Component

Zhang, J., Siegle, G. J., D’Andrea, W., and Krafty, R. T. (2019). Interpretable principal components analysis for multilevel multivariate functional data, with application to EEG experiments. Arxiv [Preprint]. 15261. 1–33. Available online at: http://arxiv.org/abs/1909.08024 (accessed July 8, 2021).

Ziyatdinov, A., Kanaan-Izquierdo, S., Trendafilov, N. T., and Perera-Lluna, A. (2014). Methods to Perform Common Principal Component Analysis (CPCA).

Conflict of Interest: The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

Publisher's Note: All claims expressed in this article are solely those of the authors and do not necessarily represent those of their affiliated organizations, or those of the publisher, the editors and the reviewers. Any product that may be evaluated in this article, or claim that may be made by its manufacturer, is not guaranteed or endorsed by the publisher.

Copyright © 2021 Riaz, Razzaq, Hu and Valdés-Sosa. This is an open-access article distributed under the terms of the Creative Commons Attribution License (CC BY). The use, distribution or reproduction in other forums is permitted, provided the original author(s) and the copyright owner(s) are credited and that the original publication in this journal is cited, in accordance with accepted academic practice. No use, distribution or reproduction is permitted which does not comply with these terms.
APPENDIX

Appendix A1 | Table explaining symbols used in paper and their description.

| Symbol | Description |
|--------|-------------|
| Q      | Set of computed for CPCs |
| k      | Number of conditions or groups |
| S<sub>i</sub> | Covariance matrix |
| D<sup>T</sup> | Diagonizable Orthogonal matrices |
| ρ      | Total number of variables |
| n      | Total number of observations |
| l<sub>0</sub> | Identity matrix for deflation |
| λ      | Number of eigenvalues |
| I<sub>n</sub> | Number of conditions or groups |
| n      | Total number of variables |
| D      | Identity matrix for deflation |
| S      | Covariance matrix |
| p<sub>max</sub> | Limited number of CPCs computed |
| l<sub>max</sub> | Maximum number of iterations for convergence |
| Π      | Deflation parameter |
| q      | Single common CPC |
| X<sub>i</sub> | Dataset captured in k different conditions |

Algorithm 1

**Input**

S<sub>p,p,k</sub>, l<sub>max</sub>, p<sub>max</sub>, n

**Algorithm**:

1. Π ← I<sub>p</sub>
2. S ← n<sub>1</sub>S<sub>1</sub> + n<sub>2</sub>S<sub>2</sub> + ... + n<sub>k</sub>S<sub>k</sub>
3. Q ← eig(S)
4. for j = 1, 2, ..., p<sub>max</sub>
   1. x<sub>(0)</sub> ← q<sub>j</sub> with x<sub>(0)</sub><sup>T</sup>x<sub>(0)</sub> = 1
   2. x<sub>(0)</sub> ← Π x<sub>(0)</sub>
   3. μ<sub>i</sub> ← x<sub>(0)</sub><sup>T</sup> S<sub>i</sub> x<sub>(0)</sub>
   4. for l = 1, ..., l<sub>max</sub>
      1. S ← n<sub>1</sub>μ<sub>1</sub> + ... + n<sub>k</sub>μ<sub>k</sub>
      2. y ← Π S x<sub>(l-1)</sub>
      3. x ← y/√y<sup>T</sup>y
         for i = 1, ..., k
         1. μ<sub>i</sub> ← x<sub>(l)</sub><sup>T</sup> S<sub>i</sub> x<sub>(l)</sub>
      end
   end
   5. q<sub>j</sub> ← x<sub>(l<sub>max</sub>)</sub> (and λ<sub>j</sub> ← μ<sub>i<sub>l<sub>max</sub>)
   6. Π ← Π - q<sub>j</sub><sup>T</sup> q<sub>j</sub>
end
end

for j = 1, ..., p<sub>max</sub>
for i = 1, ..., k

Algorithm 2

**Input**

X<sub>p,k</sub>, p<sub>max</sub>, l<sub>max</sub>, n

**Algorithm**:

1. for i = 1, 2, ..., k
   1. W<sub>i</sub> ← (X<sub>i</sub> - μ<sub>max</sub>)<sup>T</sup>
end
2. W ← <br>\[
\begin{bmatrix}
    n_1 W_1 & \cdots & n_k W_k \\
\end{bmatrix}^T
\]
3. USQ<sub>T</sub> = svd(W) actually econ svd and only Q needed
4. for j = 1, 2, ..., p<sub>max</sub>
   1. x<sub>(0)</sub> ← q<sub>j</sub> with x<sub>(0)</sub><sup>T</sup>x<sub>(0)</sub> = 1
   2. x<sub>(0)</sub> ← x<sub>(0)</sub> - (\sum_{r=1}^{l-1} q<sub>r</sub>q<sub>r</sub><sup>T</sup>) x<sub>(0)</sub>
   3. for i = 1, ..., k
      1. b<sub>(0)</sub> ← W<sub>i</sub> x<sub>(0)</sub>
      2. μ<sub>i</sub><sub>(0)</sub> ← b<sub>(0)</sub><sup>T</sup> b<sub>(0)</sub>
   end
   4. for l = 1, ..., l<sub>max</sub>
      1. a ← W x<sub>(l-1)</sub>
      2. b ← W<sup>T</sup> a
      3. y ← b - (\sum_{r=1}^{l-1} q<sub>r</sub>q<sub>r</sub><sup>T</sup>) b
      4. x ← y/√y<sup>T</sup>y
         for i = 1, ..., k
         1. μ<sub>i</sub><sub>(l)</sub> ← b<sub>(l)</sub><sup>T</sup> b<sub>(l)</sub>
      end
      5. q<sub>j</sub> ← x<sub>(l<sub>max</sub>)</sub> (and λ<sub>j</sub> ← μ<sub>i<sub>l<sub>max</sub>)
      6. Π ← Π - q<sub>j</sub><sup>T</sup> q<sub>j</sub>
end
end
for j = 1, ..., p<sub>max</sub>
for i = 1, ..., k

c<sub>i</sub> ← W<sub>i</sub> q<sub>j</sub>
λ<sub>j</sub> ← c<sub>i</sub><sup>T</sup> c<sub>i</sub>
end
end