Senior Project Report 2102499 Year 2019

Learning Granger causality for multivariate time series using state-space models

Anawat Nartkulpat ID 5930562521

Advisor: Assist. Prof. Jitkomut Songsiri

Department of Electrical Engineering

Faculty of Engineering

Chulalongkorn University

Abstract

This project aims to develop a scheme to learn a Granger causality of time series data. The Granger causality is calculated from state-space parameters estimated using subspace identification method, then permutation test is applied to classify zero and non-zero causality. We explored the parameters that affect the performance of permutation test. It was shown that increasing the number of permutations used in the test improves the performance. Moreover, we found that using all possible permutations can give slightly better performance than randomly sample permutations but only when there is no p-value correction applied. This result encourages us to use a Monte-Carlo permutation test as the correction methods can improve the performance significantly. The order of state-space models in estimation also affects the performance of permutation test. We observed through simulations that underestimation of the model order yields worse performance than overestimation. In the comparison between permutation test and GMM method, the performance of permutation test was found to be generally higher when ground truth models have sparse GC matrices. With large number of data, however, GMM method could give a very close performance to permutation test. When ground truth models have denser GC matrices, the performance of permutation test become worse when the length of time series data is longer and can be worse than GMM method. For the computational cost, permutation test suffers heavily from the excessive use of subspace identification algorithm.

Keywords: Granger causality, Permutation tests, Subspace identification

Contents

List of Figures

List of Tables

1 Introduction

Granger causality (GC) is one of powerful methods for determining causal relationships between variables in time series. For a time series $y(t) = [y_1(t)\ y_2(t)\ \cdots\ y_n(t)]^T$, we say that y_j Granger-causes y_i if the prediction of y_i is improved by using the past information of y_j given all other information of y_k where $k \neq j$. Quantitatively, a GC measure is defined by

$$F_{ij} = \log \frac{\Sigma_{ii}^R}{\Sigma_{ii}}, \qquad i, j = 1, 2, \dots, n$$

where Σ_{ii} is the variance of the prediction error of y_i given all past information and Σ_{ii}^R is the variance of the prediction error of y_i given other past information except y_j. There are many applications that concern about establishing connectivities between time series, so determining zero and non-zero causality is of great interest. In this context, the applications of the Granger causality can be found in many fields especially in neuroscience where the existences of connectivities between regions of a human brain were explored from brain signals such as EEG or fMRI [KFL19].

The Granger causality can be applied to both linear and non-linear dynamical models, but linear model such as vector autoregressive (VAR) models are considered mostly. In VAR models, many properties of the GC measure was studied. It was shown that the GC measures of VAR models has an asymptotic mean-shifted χ^2-distribution [BS14]. This allows many powerful parametric methods to be utilized for further significance tests. Modelling time series with VAR model is simple but lacking of moving average (MA) terms make the model impractical to apply to some of the real systems. MA terms can be induced in many procedures such as data pre-processing, filtering or by observation process. So, a more general model that can be use to describe both VAR and MA models such as state-space models are what we are interested in.

While the generality of the state-space models allows us to estimate models for the data described by both VAR and MA models, and also the combination of them, denoted VARMA models, the statistical distributions of the GC measures of state space models are not well understood. The distribution of the GC measures for the state space models was shown, empirically, to be well-approximated by $\Gamma-$distribution [BS15], but no analytical proof is provided yet. This renders parametric methods less useful. A clustering method was proposed by fitting a Gaussian Mixture Model (GMM) to a vectorized matrices of GC measures [Son19]. In this method, the GC measures were sampled many times to construct a mixture Gaussian distribution and used the first two Gaussian components with the lowest means to determine a threshold for classifying zero entries. While this method required no knowledge about statistical property of the Granger causality, many samples of Granger causality matrix were required to meet the assumption of the central limit theorem and the classification performance of non-zero causality could be deteriorated if the causality is weak.

Therefore, this project aims to develop a scheme for classifying zero and non-zero GC measures by using a non-parametric statistical method, namely, a permutation test, and compare the performance with the GMM method. We start by reviewing the Granger causality and linear models especially the state-space models and its equivalent models, which is the VARMA models. Then we develop a method and code for generating, arbitrarily, ground truth model and time series data for later experiments. Next, we set up experiments to study the classification performance of a permutation test. After that, an experiment for comparing the performance, the computational cost, and the required assumptions of the permutation test and the GMM method is conducted. Lastly, we discuss the results and conclude our method. We expect that this project will provide a scheme for classifying GC patterns based on the permutation test, MATLAB codes for executing the scheme numerically and the comparative results between our method and the GMM method.

2 Project Overview

2.1 Objectives

1. To develop a scheme for classifying the zero patterns of the Granger causality of state-space models using the permutation test.
2. To compare the performance of the permutation test with the GMM method in classification of zero and non-zero entries of Granger causality matrix obtained from state-space model.

2.2 Scope of work

The scope of this project is the following.

1. We perform and verify the estimation of GC pattern on simulation data only.
2. We compare the performance, the computational cost, and required assumptions of the permutation method with the GMM method for learning GC patterns.

2.3 Expected Outcomes

1. A scheme for classifying zero and non-zero causality between variables in time series using state-space models and permutation test.
2. MATLAB codes for executing the developed scheme for classifying zero and non-zero causality.
3. Results from the comparison between the method using permutation test and the GMM method in classifying zero and non-zero causality.

3 Methodology

This section describes the method for learning GC patterns from time series data including the test of significance of GC by permutation test. The scheme for learning GC pattern is proposed in *Figure 1* which will be explained as follows.

From the given n-dimensional time series data, we start with estimating a state-space model in the form

$$x(t+1) = Ax(t) + w(t) \tag{1a}$$
$$y(t) = Cx(t) + v(t) \tag{1b}$$

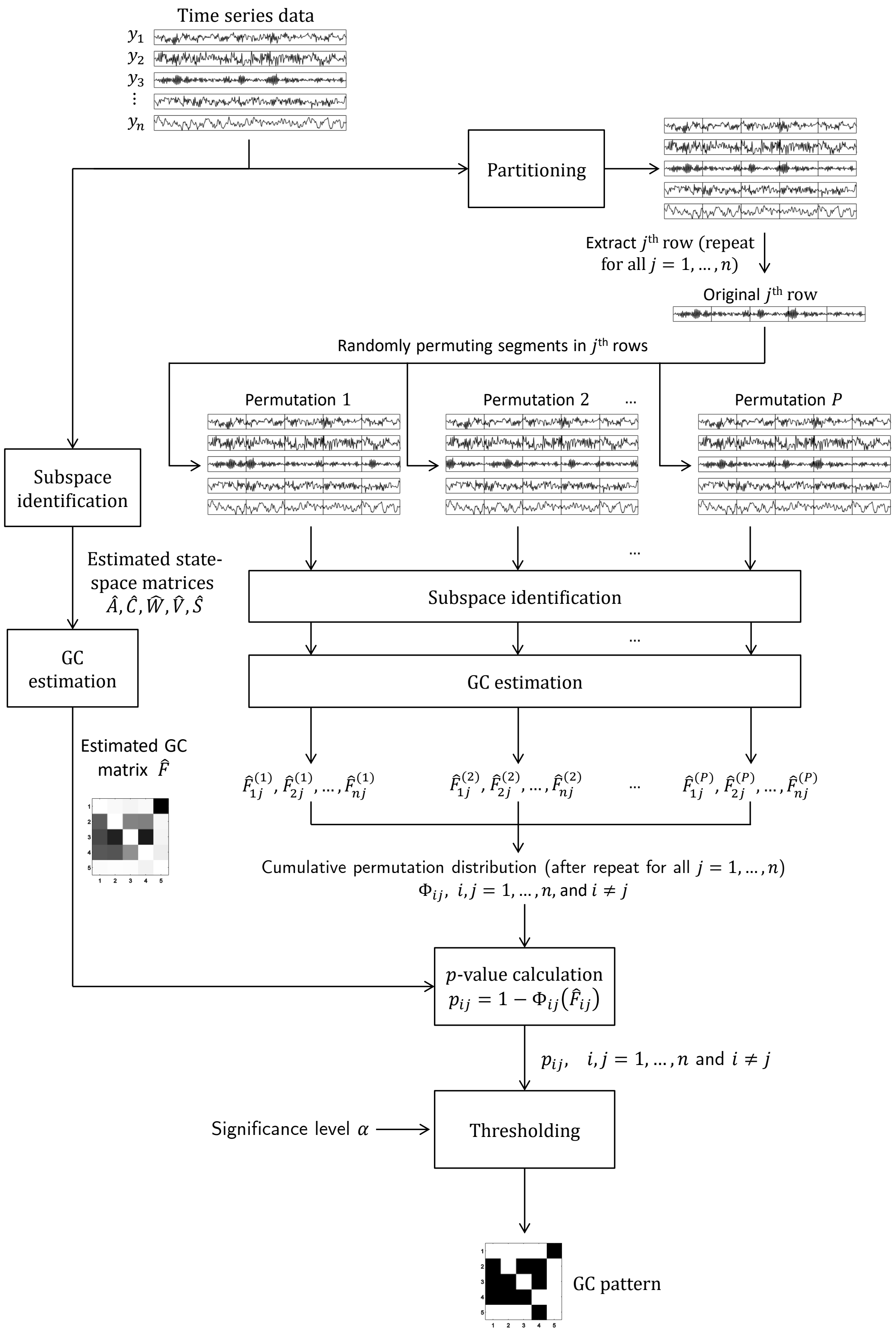
where

$$\mathbf{E} \begin{bmatrix} w(t) \\ v(t) \end{bmatrix} \begin{bmatrix} w(t) \\ v(t) \end{bmatrix}^T = \begin{bmatrix} W & S \\ S^T & V \end{bmatrix}. \tag{2}$$

Then, a matrix of GC measures ($\hat{F} = [\hat{F}_{ij}]$) is calculated from the parameters of the estimated state-space model. By uncertainties in the estimation, F_{ij} are never found to be exactly zero as shown in *Figure 2* (left). So we apply permutation test to test that F_{ij} is zero or non-zero.

To perform permutation test, we return to the time series data. First, we partition the time series into many segments. Next, for each row j^{th}, we permute (rearrange) the segments in this row randomly and insert back into the time series. This permuted time series is then used to estimate state-space parameters by subspace identification algorithm. From these parameters, samples of GC measures $\hat{F}_{ij}^{(k)}$ for $i = 1, \ldots, n$ are obtained. This procedure is repeated from $k = 1, \ldots, P$ for every row $j = 1, \ldots, n$. Then, all GC measures obtained are used to construct a cumulative permutation distribution Φ_{ij} for all $i, j = 1, \ldots, n$ where $i \neq j$.

With the GC matrix $\hat{F}$, we can calculate p-values for each $\hat{F}_{ij}$ using Φ_{ij} and then define a significance level α to test the null hypothesis $H_0 : F_{ij} = 0$ for every entries of F, and, thus, the GC pattern in the form of a binary matrix is acquired as represented in *Figure 2* (right).



Time series data
y_1
y_2
y_3
$\vdots$
y_n
Partitioning
Extract j^{th} row (repeat for all $j = 1, \dots, n$)
Original j^{th} row
Randomly permuting segments in j^{th} rows
Permutation 1
Permutation 2 …
Permutation P
Subspace identification
Estimated state-space matrices $\hat{A}, \hat{C}, \hat{W}, \hat{V}, \hat{S}$
GC estimation
Estimated GC matrix $\hat{F}$
Subspace identification
GC estimation
$\hat{F}_{1j}^{(1)}, \hat{F}_{2j}^{(1)}, \dots, \hat{F}_{nj}^{(1)}$
$\hat{F}_{1j}^{(2)}, \hat{F}_{2j}^{(2)}, \dots, \hat{F}_{nj}^{(2)}$
$\hat{F}_{1j}^{(P)}, \hat{F}_{2j}^{(P)}, \dots, \hat{F}_{nj}^{(P)}$
Cumulative permutation distribution (after repeat for all $j = 1, \dots, n$)
$\Phi_{ij}, \ i, j = 1, \dots, n$, and $i \neq j$
p-value calculation
$p_{ij} = 1 - \Phi_{ij}(\hat{F}_{ij})$
$p_{ij}, \quad i, j = 1, \dots, n$ and $i \neq j$
Significance level α
Thresholding
GC pattern

Figure 1: Our scheme for learning GC pattern using permutation test.

In the following sections, we will describe the necessary procedures used in our scheme for estimating GC patterns including the generation of ground truth models used in all of our simulations, subspace identification algorithm for estimating state-space parameters, calculation of GC matrices, permutation test and performance indices which are considered in our experiments.

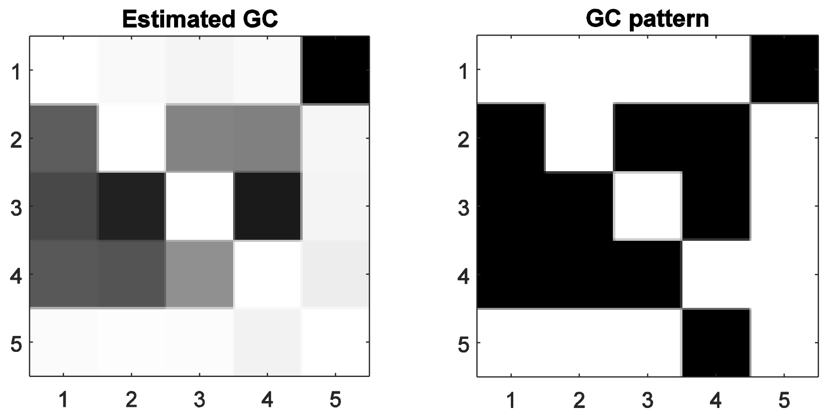


Figure 2: (left) The estimated GC matrix $\hat{F}$ can never be the same as the GC matrix of the ground truth model. So the entries that suppose to be zeros are estimated at best to be close to zero. In this gray-scale representation, the entry at (i,j) is darker as $\hat{F}_{ij}$ is greater than zero and is whiter as $\hat{F}_{ij}$ is closer to zero. (right) The GC pattern in binary form after permutation test. Black entries represent non-zero causalities and white entries represent zero causalities.

3.1 Ground truth model generation

To examine the performance of the permutation method, we require state space models whose parameters and GC matrix are known. We refer to this model as the ground truth model. Since we are interested in the zero pattern of the GC matrix, the ground truth model should be generated arbitrarily with controlled sparsity of zero entries. The scheme for generating ground truth models is shown in *Figure 3*.

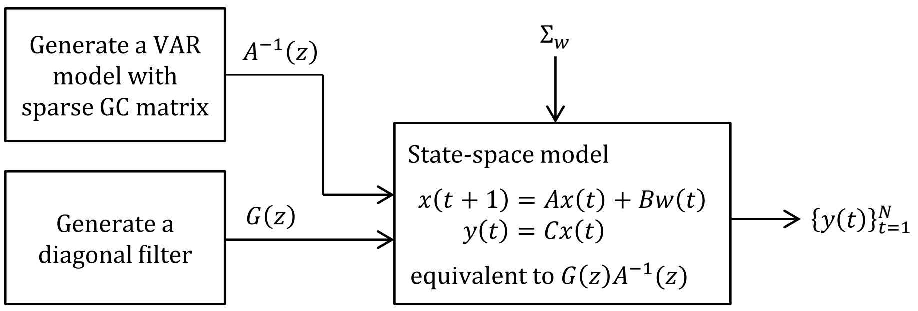


Figure 3: The scheme for generating ground truth state-space models and time series data.

Firstly, we generate a stable VAR model of order p described by

$$y(t) = A_1 y(t-1) + A_2 y(t-2) + \cdots + A_p y(t-p) + w(t),$$

or by a transfer function

$$A^{-1}(z) = (I - A_1 z^{-1} - A_2 z^{-2} - \cdots - A_p z^{-p})^{-1}.$$

It has been shown that, for any VAR models [Lüt05],

$$(A_k)_{ij} = 0, \forall k = 1, \ldots, p \quad \iff \quad F_{ij} = 0.$$

So by generating $A_1, A_2, \ldots, A_p$ with common zero entries, we obtain a model whose GC matrix's sparsity can be chosen arbitrarily.

Next, in order to make the model more general, the VAR model can be filtered to introduce moving average terms. It has been proved that the GC matrix is invariant under any stable invertible diagonal filter that has minimum phase [BS11]. So we consider generating a diagonal filter

$$G(z) = \begin{bmatrix} \frac{p_1(z)}{q_1(z)} & 0 & \cdots & 0 \\ 0 & \frac{p_2(z)}{q_2(z)} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \frac{p_n(z)}{q_n(z)} \end{bmatrix}$$

where $p_i(z), q_i(z) \neq 0$ are polynomials in z. To guarantee the stability and minimum phase, roots of $p_i(z)$ and $q_i(z)$ for all $i = 1, \ldots, n$ must be generated so that they lie inside a unit circle. Since $q_i(z) \neq 0$, the filter is invertible.

From the previous results, passing a VAR model under a stable invertible diagonal filter gives a new system described by a transfer function

$$H(z) = G(z)A^{-1}(z).$$

This transfer function can be realized into a state-space model

$$\begin{aligned} x(t+1) &= Ax(t) + Bw(t), \\ y(t) &= Cx(t). \end{aligned}$$

This state-space model has the same GC pattern with the VAR model and is considered to be our ground truth model. Hence, we may conclude the generation of ground truth model into the following step.

1. Generate a random stable VAR model of order p with common zero entries in its parameters $A_1, A_2, \ldots, A_p$ to obtain the transfer function $A^{-1}(z)$.

2. Generate a random diagonal filter that is stable, invertible and has minimum phase. In this step we obtain the transfer function $G(z)$ of the filter.

3. Realize $G(z)A^{-1}(z)$ using `ss` command with `minimal` option in MATLAB to obtain a minimal state-space form. This state-space model is then used as the ground truth model.

Time series data can be generated directly from the state-space model using `lsim` command in MATLAB.

3.2 Subspace identification

In the calculation of GC matrix, parameters of state-space model are required. So we consider the estimation of a stochastic state-space model shown in (1) with (2) where $x \in \mathbf{R}^n$ and $y \in \mathbf{R}^m$. Suppose that the time series data $\{y(t)\}_{t=1}^N$ is observed. Let define the notation for projection of the row space of the matrix P on the row space of the matrix Q as

$$P/Q \triangleq PQ^T(QQ^T)^\dagger Q$$

where $(QQ^T)^\dagger$ is the pseudo-inverse of (QQ^T). The parameter (A, C, W, V, S) can by obtained by a subspace identification method in the following steps [Son]. Firstly, we project the future output (Y_f) onto the past output (Y_p) space to obtain the orthogonal projection

$$\mathcal{O}_i \triangleq Y_{i|2i-1}/Y_{0|i-1} = Y_f/Y_p$$

where $Y_{0|i-1} = Y_p$ is the observed data from $t = 0$ to $t = i-1$ representing the past output and $Y_{i|2i-1} = Y_f$ is the observed data from $t = i$ to $t = 2i-1$ representing the future output. Let n_o be the rank of $\mathcal{O}_i$. Then, by SVD decomposition,

$$\mathcal{O}_i = \begin{bmatrix} U_1 & U_2 \end{bmatrix} \begin{bmatrix} \Sigma_{n_o} & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} V_1^T \\ V_2^T \end{bmatrix} = U_1 \Sigma_{n_o} V_1^T.$$

Let Γ_i be an extended observability matrix. Then, from [OM12], we have

$$\mathcal{O}_i = \Gamma_i \hat{X}_i$$

where $\hat{X}_i$ is the estimated state. So, for some non-singular matrix T, we can obtain

$$\Gamma_i = U_1 \Sigma_{n_o}^{1/2} T$$

From this result, the estimated states is

$$\hat{X}_i = \Gamma_i^{\dagger} \mathcal{O}_i$$

where $\Gamma_i^{\dagger}$ denotes the pseudo-inverse of Γ_i. Next, the shifted state $\hat{X}_{i+1}$ can be computed by

$$\hat{X}_{i+1} = (\underline{\Gamma_i})^{\dagger} \mathcal{O}_{i-1} = (\underline{\Gamma_i})^{\dagger} (Y_{i+1|2i-1}/Y_{0|i})$$

where $\underline{\Gamma_i}$ denotes Γ_i without the last row. Lastly, we form the equation

$$\begin{bmatrix} \hat{X}_{i+1} \\ Y_{i|i} \end{bmatrix} = \begin{bmatrix} A \\ C \end{bmatrix} \hat{X}_i + \begin{bmatrix} \rho_w \\ \rho_v \end{bmatrix}.$$

We can solve for A and C in least-square sense and obtain

$$\begin{bmatrix} \hat{A} \\ \hat{C} \end{bmatrix} = \begin{bmatrix} \hat{X}_{i+1} \\ Y_{i|i} \end{bmatrix} \hat{X}_i^{\dagger}$$

The covariances of $w(t)$ and $v(t)$ are then obtained by

$$\begin{bmatrix} \hat{W} & \hat{S} \\ \hat{S}^T & \hat{V} \end{bmatrix} = (1/j) \begin{bmatrix} \rho_w \\ \rho_v \end{bmatrix} \begin{bmatrix} \rho_w \\ \rho_v \end{bmatrix}^T.$$

The implementation of this method in MATLAB is available from the author of [OM12] in the function `subid` which can be used to identify deterministic system, stochastic system or the combination of both.

3.3 Granger causality of state-space models

After the stochastic state-space model have been identified, we examine causal relationships between every pair of output y_i and y_j when $i \neq j$. The Granger causality from y_j to y_i is quantified by how prediction of the future of y_i can be improved by the past of y_j given all past information compared to without using the past of y_j. The measure of the Granger causality from y_j to y_i is then defined as [BS15]

$$F_{ij} = \log \frac{\Sigma_{ii}^R}{\Sigma_{ii}} \tag{4}$$

where Σ_{ii} and Σ_{ii}^R are the covariances of the prediction errors of the full model and the reduced model, respectively. Using more information generally gives better prediction, so it follows that $\Sigma_{ii} \leq \Sigma_{ii}^R$. Hence, we always have $F_{ij} \geq 0$.

To obtain Σ_{ii} and Σ_{ii}^R, we note that not using y_j is equivalent to removing y_j from (1). So we consider the full model described by (1) and the reduced model

$$\begin{aligned} x(t+1) &= Ax(t) + w(t), \\ y^R(t) &= C^R x(t) + v^R(t) \end{aligned}$$

where y^R is obtained by removing y_j from y and C^R is obtained by removing the j^{th} row from C. It has been shown in [Son19] that, through Kalman filter, we have

$$\Sigma = \textbf{cov}(y(t) - \hat{y}(t|t-1)) = CPC^T + V$$

where $\hat{y}(t|t-1))$ is the optimal estimator of $y(t)$ in mean squared error sense and P can be solved from the discrete-time algebraic Riccati equation (DARE):

$$P = APA^T - (APC^T + S)(CPC^T + V)^{-1}(CPA^T + S^T) + W. \tag{6}$$

Similarly, Σ^R can be obtained in the same manner by first solving (6) for P^R using C^R and V^R instead of C and V where V^R denotes V with its j^{th} row and column being removed. Then, we have

$$\Sigma^R = C^R P^R (C^R)^T + V^R.$$

By extracting the diagonal entries of Σ and Σ^R, the GC measures can be calculated by (4) for all i. A GC matrix is then defined by

$$F = \begin{bmatrix} F_{11} & F_{12} & \cdots & F_{1n} \\ F_{21} & F_{22} & \cdots & F_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ F_{n1} & F_{n2} & \cdots & F_{nn} \end{bmatrix}.$$

Since we are interest only the causal relationship between different variables, the diagonal entries of F are of no interest and left without computation.

3.4 Permutation test

After obtaining the estimated GC matrix $\hat{F}$, each entries in $\hat{F}$ must be classified into zero or non-zero. Since the estimation is never perfect, $\hat{F}$ is deviated from the true GC matrix F. A statistical test is then required for classification. So we state a null hypothesis as

$$H_0 : F_{ij} = 0. \tag{7}$$

Since $F_{ij} \geq 0$ by definition, our test is a one-tail test. It is not known what the distribution of F_{ij} under the true null hypothesis is. So, our approach is to use a non-parametric test, specifically, a permutation test.

In permutation test, under the true null hypothesis, we construct distributions of the test statistics (F_{ij}), called permutation distributions, from the data [NH02], which is, in our case, the time series data. As the name suggests, the permutation distribution is constructed from the test statistics obtained by every possible permutations (or rearrangements) of the data. The permutation done on the data must be justified under the true null hypothesis, that is, the rearranging **must not affect** the null hypothesis.

Under the null hypothesis (7), y_i is not Granger-caused by y_j. It follows that rearranging the data in y_j has no effect on F_{ij} in this sense. Hence, more samples of F_{ij} can be acquired by estimating GC matrix again from different permutations. For the time series data of length N, there are overwhelming $N!$ possible ways to rearrange the data. To limit the number of permutation, we can permute chunks of the data instead by partitioning the data into many segments with the some length W. For example, see *Figure 4*. This gives $\lfloor \frac{N}{W} \rfloor!$ permutations if we discard the residual data from partitioning. If we only select some of the permutation randomly from all possible permutation, we call the test the **Monte-Carlo** permutation test [Goo05] and we will refer the test using all possible permutations of all segments as **complete** permutation test.

For each permutation on y_j for $j = 1, \ldots, n$, we obtain samples of F_{ij} for $i = 1, \ldots, n$ denoted $\{\hat{F}_{ij}^{(k)}\}_{k=1}^{P}$ where P is the number of permutations. These samples are used to construct (cumulative) permutation distributions Φ_{ij} for all F_{ij}. Note that the total times that subspace identification algorithm is called is, for n-dimensional time series data, nP.

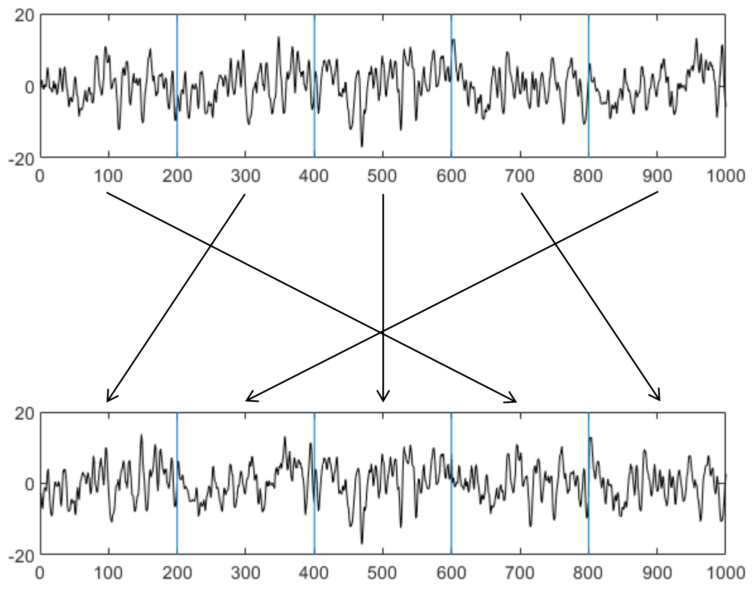


Figure 4: One of the possible permutations of time series data partitioned into 5 segments.

Given a significance level α, we can test the significance of $\hat{F}_{ij}$ to accept or reject the null hypothesis (7). The p-value is then calculated from the probability of F_{ij} being at least as extreme as $\hat{F}_{ij}$. Using the permutation distributions, one-tail p-value for any $\hat{F}_{ij}$ is computed by

$$p_{ij} = 1 - \Phi_{ij}(\hat{F}_{ij}) = \frac{\text{Number of elements in } \left\{k | \hat{F}_{ij}^{(k)} \geq \hat{F}_{ij}\right\}}{P}. \tag{8}$$

If $\hat{F}_{ij} > 0$, we can see that p_{ij} should be small since permuting the data generally neutralized the causality. By choosing α for thresholding, we can classify that

$$F_{ij} \neq 0 \quad \text{if} \quad p_{ij} \leq \alpha, \qquad \text{(reject the null hypothesis)}$$

or classify that

$$F_{ij} = 0 \quad \text{if} \quad p_{ij} > \alpha, \qquad \text{(accept the null hypothesis)}.$$

While this method is simple, testing all $\hat{F}_{ij}$ repeatedly with α as a threshold has some concern about the Type I error (the probability of incorrectly rejected the null hypothesis) since we have to test many hypotheses simultaneously (that is testing for $F_{ij} = 0$ for all i, j when $i \neq j$). Testing multiple hypotheses simultaneously is called multiple comparisons or multiple testing [NH02].

Suppose that $F \in \mathbf{R}^{n \times n}$, it follows that we have a family of $n^2 - n$ hypotheses to be tested (excluding the diagonal). To explore the problem with multiple testing, we consider a family-wise error rate (FWER) which is the probability of having one or more Type I errors [Efr12]. So, if we test each hypothesis using α as a threshold, and the tests are independent, then we have

$$\text{FWER} = 1 - (1 - \alpha)^{n^2 - n} \geq \alpha$$

which is undesirable since the probability of having at least one Type I error will be greater than α. p-values obtained from testing each hypothesis individually with α as a threshold are referred to as the **uncorrected** p-values [NH02]. To cope with this problem, we introduce two p-value correction methods, the Bonferroni correction and the Benjamini-Holchberg procedure [Efr12].

The Bonferroni correction is a method for correcting the p-values by using

$$\tilde{p}_{ij} = \min\{(n^2 - n)p_{ij}, 1\}$$

in testing with α as a threshold or, equivalently, using a modified α:

$$\tilde{\alpha} = \frac{\alpha}{n^2 - n}.$$

as a threshold in testing with uncorrected p-values. By Boole's inequality, it can be shown that [Efr12]

$$\begin{aligned} \text{FWER} &= P\Big\{ \bigcup_{i \neq j} \Big(p_{ij} \leq \tilde{\alpha} \Big) \Big\} \\ &\leq \sum_{i \neq j} P\Big(p_{ij} \leq \frac{\alpha}{n^2 - n} \Big) \\ &= (n^2 - n) \frac{\alpha}{n^2 - n} \\ &= \alpha. \end{aligned}$$

Therefore, by choosing $\tilde{\alpha} = \alpha/(n^2 - n)$ as a threshold in multiple test with uncorrected p-value, our tests become conservative. But, the value of $\tilde{\alpha}$ can be very small such that even large F_{ij} could be classified as zero.

Another method called the Benjamini-Hochberg procedure approaches by controlling false discovery rate (FDR) instead. Let U be the number of false discoveries (incorrectly rejected null hypotheses or false positives) and R be number of discoveries (all rejected null hypotheses or the sum of false positives and true positives). Then, the false discovery rate is defined by [Sto11]

$$\text{FDR} = \begin{cases} \mathbf{E}[U/R] & \text{, if } R \neq 0 \\ 0 & \text{, if } R = 0 \end{cases}$$

The Benjamini-Hochberg procedure starts by sorting uncorrected p-values (p_{ij}) in the ascending order

$$p_{(1)} \leq p_{(2)} \leq \cdots \leq p_{(n^2-n)}.$$

Next, we find the largest index k such that

$$p_{(k)} \leq \frac{k}{n^2 - n} \alpha.$$

Then, we reject the first k hypotheses associated with $p_{(1)}, \ldots, p_{(k)}$. It has been shown in [Efr12] that if the p-values corresponding to the correct null hypotheses are independent, then, by using the Benjamini-Hochberg procedure, we have

$$\text{FDR} \leq \alpha.$$

This method does not guarantee that FWER $\leq \alpha$, so it is not conservative. Instead, the ratio of false discoveries to all discoveries, that is the ratio of false positives to sum of both false positives and true positives, is controlled by α.

In our work, we consider on both correction methods. The test with uncorrected p-values is only considered when the other two methods are unavailable, namely when the complete permutation test is performed with small number of permutations since the achievable minimum p-value can always be greater than the modified significance level $\tilde{\alpha}$.

Now, we can explain our scheme in *Figure 1* in more detail. With a given n-dimensional time series data, we start with estimate state-space parameters and calculate GC matrix from the original data. Then, the length of partition W is chosen (or, equivalently, the number of partitions) and we partition the data into $\lfloor \frac{N}{W} \rfloor$ segments.

For each row, say, j^{th} row, of the time series data, we repeatedly permute the order of segments in j^{th} row for P times. For each permutation, we insert the permuted j^{th} row back into the original time series data and perform subspace identification and GC calculation to obtain samples of GC measure

$\hat{F}_{ij}^{(k)}$ from channel j to channel i when $i = 1, \dots, n$ and $i \neq j$. As this procedure is repeat for all row $j = 1, \dots, n$, we obtain all samples of GC measure $\hat{F}_{ij}^{(k)}$ for $i, j = 1, \dots, n$, $i \neq j$ and $k = 1, \dots, P$. With these samples, we can calculate p-values using (8) or we may construct cumulative permutation distributions Φ_{ij} first and then calculate p-values later.

Now that we have p-values, we can test the hypothesis (7) by choosing a significance level α and thresholding the p-values using α directly or apply p-value correction methods, that is, Bonferroni method and Benjamini-Hochberg procedure. Finally, we obtain the GC pattern in binary matrix form as shown in *Figure 2* (right) where black entries represent non-zero causality and white entries represent zero causality.

3.5 Performance indices

Learning zero and non-zero entries of GC matrix F is basically a binary classification problems of two possible outcomes, $F_{ij} = 0$ and $F_{ij} > 0$. So we adopt the following indices used in classification.

- True positive (TP): correctly classified non-zeros in F
- True negative (NP): correctly classified zeros in F
- False positive (FP): incorrectly classified non-zeros in F
- False negative (FP): incorrectly classified zeros in F

Together with these indices, we can determine the following classification performance.

$$\begin{aligned}
\text{Accuracy (ACC)} &= \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}} \\
\text{True positive rate (TPR)} &= \frac{\text{TP}}{\text{TP} + \text{FN}} \\
\text{True negative rate (TNR)} &= \frac{\text{TN}}{\text{TN} + \text{FP}} \\
\text{False positive rate (FPR)} &= \frac{\text{FP}}{\text{TN} + \text{FP}} = 1 - \text{TNR} \\
\text{False negative rate (FNR)} &= \frac{\text{FN}}{\text{TP} + \text{FN}} = 1 - \text{TPR}
\end{aligned}$$

4 Comparative method: Gaussian Mixture Model (GMM)

In this section, we describe Gaussian Mixture Model (GMM) method for learning GC pattern proposed in [Son19]. This method will be used to compare its performances with our proposed scheme using permutation test. Originally, this method is applied on multi-trials time series data while in this project, we consider long single-trial time series data. So, we adapt this method to our data by splitting a long time series data into many short segments and treat them as separate trials.

A GMM has the form of

$$Y = Z_1 Y_1 + Z_2 Y_2 + \cdots + Z_K Y_K$$

where K is the number of Gaussian components or the number of modes, Y_k is the Gaussian random variable with mean μ_k and variance σ_k for $k = 1, \dots, K$ and $Z = (Z_1, \dots, Z_K)$ is a random vector having a multinomial distribution with the possible values

$$Z = (1, 0, \dots, 0), (0, 1, 0, \dots, 0), \dots, (0, 0, \dots, 0, 1)$$

corresponding to a multinomial pmf $\pi = (\pi_1, \pi_2, \dots, \pi_K)$. The pdf of Y is then given by

$$f(y) = \pi_1 f_1(y; \mu_1, \sigma_1^2) + \cdots + \pi_K f_K(y; \mu_K, \sigma_K^2)$$

where $f_k(y; \mu_k, \sigma_k^2) = (1/\sqrt{2\pi}\sigma_k) \exp(-\frac{(y-\mu_k)^2}{2\sigma_k^2})$ is the Gaussian pdf for $k = 1, \dots, K$.

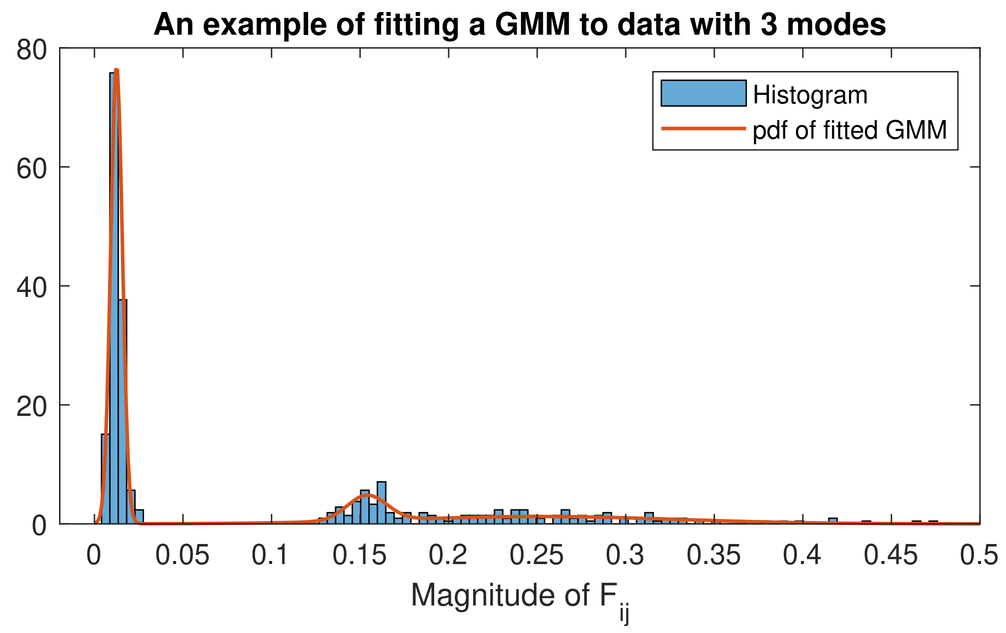


Figure 5: An example of fitting a 3-modes GMM to training data.

As proposed in [Son19], this method suggests fitting a GMM to samples of vectorized averaged GC matrices acquired from GC estimation on each trial, or each segment in our case. The averaging is required by the assumption of the central limit theorem which allows us to treat the averaged samples as having Gaussian distributions instead of working with samples of GC matrices whose distribution is unknown. The obtained samples are then vectorized and used as training data for fitting a GMM. This means all entries except the diagonal of obtained GC matrices are used to train a single GMM. An example of fitting GMM is shown in *Figure 5*. The fitting process employ Expectation-Maximization (EM) algorithm to maximize the log-likelihood function. After fitting, we look for the mode with the smallest mean. This mode is considered to be zero and any entries of the GC matrix that are clustered by the posterior probabilities into this mode are classified as zero. The scheme for this method is shown in *Figure 6*.

In summary, GMM method is performed in the following steps.

1. Partition time series data into N_0 segments. Each segment is then used to estimate a sample of GC matrix $F^{(i)}, i = 1, \dots, N_0$.
2. Find sample means of $F^{(i)}$ over $\bar{n}$ samples (that is averaging $F^{(1)}, F^{(2)}, \dots, F^{(\bar{n})}$ and $F^{(\bar{n}+1)}$, $F^{(\bar{n}+2)}, \dots, F^{(2\bar{n})}$, and so on) to obtain $\bar{F}^{(i)}, i = 1, \dots, N_0/\bar{n} = \bar{N}$.
3. Vectorize all $\bar{F}^{(i)}$ excluding their diagonal entries and then concatenate them together into one single vector. This vector is treated as data for training GMM.
4. Fit a GMM with the previously obtained training data. This can be done using MATLAB function `fitgmdist`.
5. Cluster the entries of a GC matrix $\hat{F}$ obtain from estimation using all time series data. The entries of $\hat{F}$ that are clustered into the mode with the smallest mean are classified as zero and the other are classified as non-zero.

5 Results and Discussion

5.1 Effect of the number of permutation on the performance of permutation test

In this section, an experiment was performed to study the effect of the number of permutations to the performance of the Monte-Carlo permutation test. Our hypothesis was that the performance increased

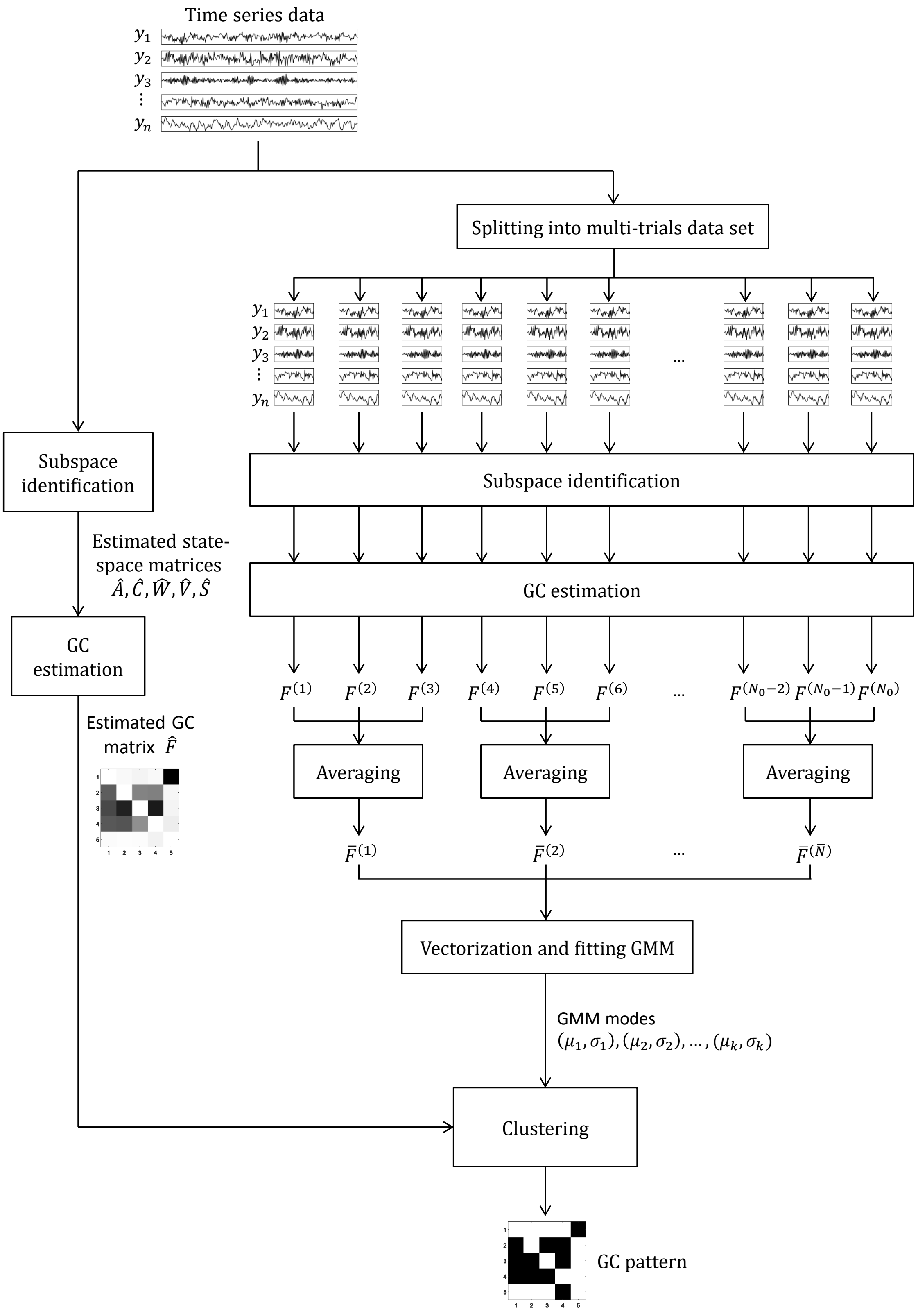

Time series data
y_1
y_2
y_3
$\vdots$
y_n
Splitting into multi-trials data set
Subspace identification
Subspace identification
Estimated state-space matrices $\hat{A}, \hat{C}, \hat{W}, \hat{V}, \hat{S}$
GC estimation
GC estimation
$F^{(1)}$ $F^{(2)}$ $F^{(3)}$ $F^{(4)}$ $F^{(5)}$ $F^{(6)}$ ... $F^{(N_0-2)}$ $F^{(N_0-1)}$ $F^{(N_0)}$
Estimated GC matrix $\hat{F}$
Averaging
Averaging
Averaging
$\bar{F}^{(1)}$ $\bar{F}^{(2)}$ ... $\bar{F}^{(\bar{N})}$
Vectorization and fitting GMM
GMM modes $(\mu_1, \sigma_1), (\mu_2, \sigma_2), \dots, (\mu_k, \sigma_k)$
Clustering
GC pattern

Figure 6: The scheme for learning GC pattern using GMM method.

as the number of permutations increased. This experiment could give us the best number of permutations to be chosen to reduce the computation time.

Experiment setting: We set up the control conditions as follows. We created 15 state-space models of order 20 with output dimension of 5 and the density of non-zero entries is 0.5. Time series data were generated from each models with 1000 time points. The number of states use in subspace method was chosen to be 20 which is the same as the true system. We varied the number of permutation P from 1 to 200 with the same segment length of 100. Then, with the significance level $\alpha = 0.05$, we compared the estimated GC patterns obtained by simple thresholding, Bonferroni correction and Benjamini-Hochberg procedure from each models to the true GC patterns.

Result and discussion: The result in *Figure 7* showed that, when the number of permutations $P < 20$, ACC and TNR increased as P was increased. In contrast, the FPR declined as P increased. For $P > 20$, ACC, TNR and FPR became steady and did not have significant changes. This may imply that the performance stop improving significantly when $P > 1/\alpha$. For simple thresholding, TPR and FNR were constants at 1 and 0, respectively, for all P while Bonferroni correction and Benjamini-Hochberg procedure had slightly decrease in TPR and slightly increase in FNR as P increased. This result agree with our hypothesis that the performance is improved as the number of permutation increase. Since we performed the Monte-Carlo permutation test, the result is expected to be improved as more permutations are used. It is possible that, for simple thresholding case, the TPR and FNR of constants 1 and 0, respectively, were resulted from that the estimated non-zero entries were too extreme. In other word, the true causality was significantly strong so that the chance to have a permutation that gives even higher causality than the estimated causality was very low. Another notable result is that for all performance measures, the Benjamini-Hochberg procedure lay mostly between the simple thresholding and the Bonferroni correction. This shows that the Benjamini-Hochberg procedure can control the FPR better than the simple thresholding by controlling the false discovery rate, which is the ratio of false positives over the sum of false positives and true positives, but slightly worse than the Bonferroni correction which directly controls Type I error. On the other hand, the Benjamini-Hochberg procedure had lower FNR and higher TPR than the Bonferroni correction, demonstrating a trade-off between controlling FPR and FNR.

5.2 Complete permutations and Monte-Carlo permutation test

This experiment was performed to observe the performance of permutations test when using every possible permutation of the time series data compare to randomly select some of them in Monte-Carlo test. Our hypothesis is that, when the given number of permutations used is the same, using every possible permutations gives better performance than using higher number of segments and then randomly sampling permutations.

Experiment setting: Since the number of possible permutations is the factorial of the number of segments for partitioning, we chose this number to be 5 so that the total number of permutations is $5! = 120$ which can be computed in reasonable time. We generated 40 ground truth state-space models with order 40, 10 output dimension and the density of non-zero entries is 0.1. Time series data were generated from these models with 10000 time points. The number of states chosen in subspace identification is 40, the same as the ground truth model. For permutation test, we consider partitioning the time series into 5 and 10 segments and using 120 permutation. So, in 5 segments case, all permutation is used. The significance level is chosen to be $\alpha = 0.05$. We compared the estimated GC patterns obtained by simple thresholding, Bonferroni correction and Benjamini-Hochberg procedure from each models to the true GC patterns to calculate performance indices for each p-value correction procedure.

Result and discussion: From *Table 1*, when using simple thresholding, choosing 5 segments gave slightly better performance as seen in higher ACC, TNR and lower FPR while TPR and FNR were the same. But, in 10 segments case, using Bonferroni correction or Benjamini-Hochberg procedure (both

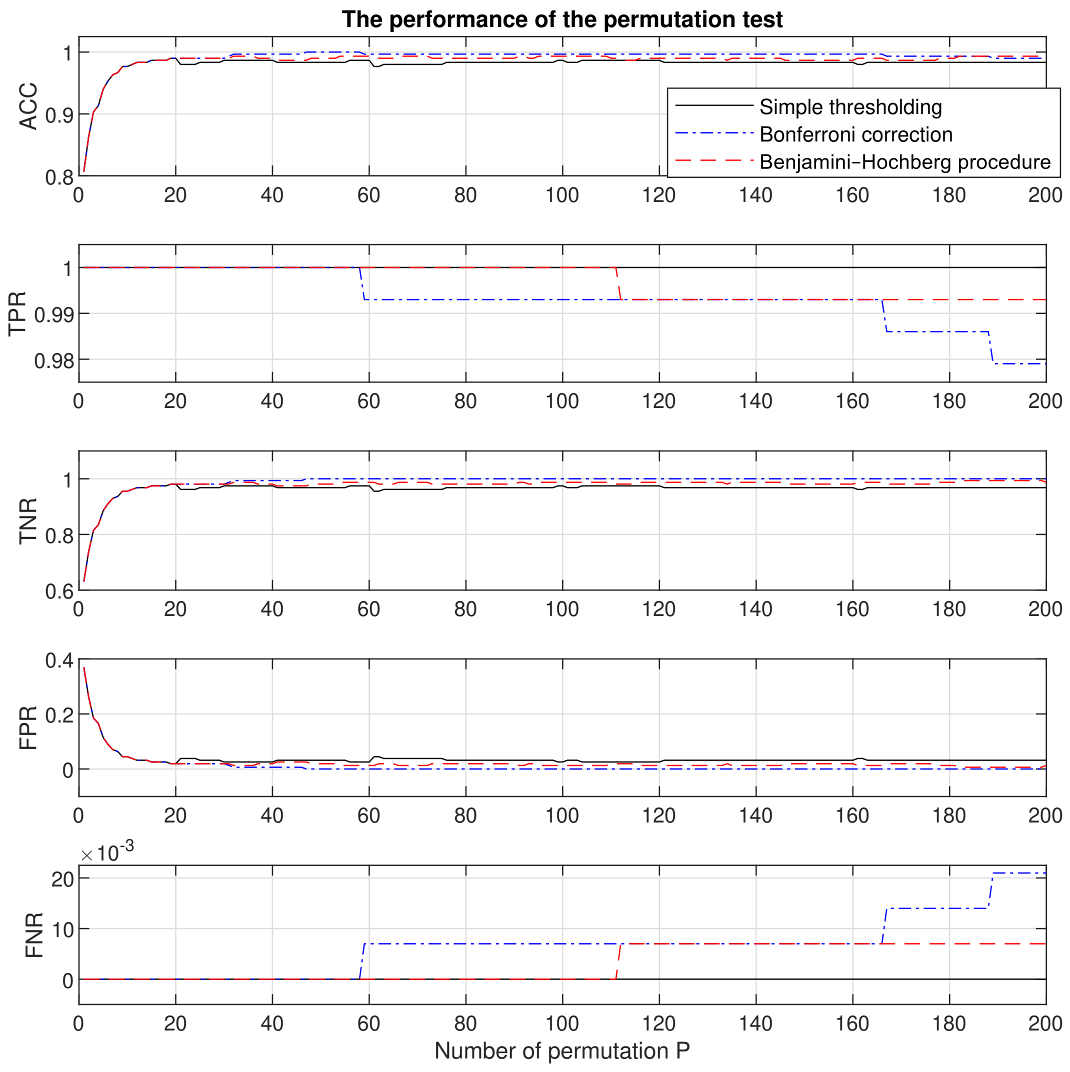


Figure 7: The performances of the Monte-Carlo permutation test for $P = 1, 2, \ldots, 200$ with no correction, with Bonferroni correction and with Benjamini-Hochberg procedure.

showed to have the same performance indices) gave much better performance than any method when using 5 segments. To explain this, we must look at the performance in 5 segments case when using Bonferroni or Benjamini-Hochberg method. We see that in this case, we have TPR $= 0$ which means that non-zero causalities are never detected. The reason is that when using all permutations, the smallest p-value possible is $1/120$ as seen from (8). It follows that the corrected significance level for Bonferroni method is $\alpha/(n^2 - n) = 0.05/90 < 1/120$. Hence, all entries of GC matrix are considered zero by the test which lead to zero TPR. For Benjamini-Hochberg procedure, all p-value less than $1/120$ are rejected, so none is rejected. This shows that, basically, Bonferroni method and Benjamini-Hochberg procedure cannot be used when all permutations are used if the number of all permutations are essentially smaller than $(n^2 - n)/\alpha$.

5.3 Effect of the order of state-space model to the performance of permutation test

In practice, the number of states in a real system is usually unknown. So we carried out this experiment to study the performance of permutation test for testing zero causality when the chosen order of state-

Table 1: The performance of the permutation test when choosing the number of partitioning segments to be 5 and 10 segments respectively. The p-value correction methods used are abbreviated as Simple for thresholding without correction, Bon for Bonferroni method and B-H for Benjamini-Hochberg procedure.

Performance	5 segments			10 segments		
index	Simple	Bon	B-H	Simple	Bon	B-H
ACC	0.9678	0.9047	0.9047	0.9650	0.9944	0.9944
TPR	1	0	0	1	1	1
TNR	0.9644	1	1	0.9613	0.9939	0.9939
FPR	0.0356	0	0	0.0387	0.0061	0.0061
FNR	0	1	1	0	0	0

space models is not equal to the order of ground truth models. We hypothesized that the performance when the order is underestimate is worse than when the order is overestimate.

Experiment setting: We generated 40 ground truth state-space models with order 40, 10 output dimension and the density of non-zero entries is 0.1. Time series data were generated from these models with 10000 time points. We applied subspace identification method with order 20, 40 and 60 respectively. In permutation test, we used 10 segments with length of 1000 time points each and the number of permutations is chosen to be 200. The significance level is $\alpha = 0.05$. We compared the estimated GC patterns obtained by simple thresholding, Bonferroni correction and Benjamini-Hochberg procedure from each models to the true GC patterns to calculate performance indices for each p-value correction procedure.

Result and discussion: From *Table 2*, we see that when the selected order is 20, lower than the ground truth models', ACC, TPR and TNR were the lowest of all selected order when compared with the same correction method. FPR and FNR were also the highest in the same way. When the order is 60, the performance was also worse than the true order, but still was better than 20 states. Hence, underestimating the order of the state-space models worsen the performance of permutation test the most as expected. Another interesting point we can note is that when using Bonferroni correction method, estimation with 60 states give only slightly worse than estimation with 40 states. So even the number of states is overestimated, Bonferroni correction method may still yield a very accurate result.

Table 2: The performance of the permutation test when choosing the order of state-space models to be 20, 40 and 60 respectively. The p-value correction methods used are abbreviated as Simple for thresholding without correction, Bon for Bonferroni method and B-H for Benjamini-Hochberg procedure.

Performance	20 states			40 states			60 states		
index	Simple	Bon	B-H	Simple	Bon	B-H	Simple	Bon	B-H
ACC	0.9369	0.9892	0.9808	0.9678	0.9969	0.9944	0.9406	0.9947	0.9881
TPR	1	0.9883	0.9913	1	1	1	1	1	1
TNR	0.9303	0.9893	0.9797	0.9644	0.9966	0.9939	0.9343	0.9942	0.9868
FPR	0.0697	0.0107	0.0203	0.0356	0.0034	0.0061	0.0657	0.0058	0.0132
FNR	0	0.0117	0.0087	0	0	0	0	0	0

5.4 Comparison of the performance between permutation test and GMM method

In this section, we compared the performance between permutation test which we introduced earlier with GMM method presented in [Son19] when perform on the data generated from ground truth models with sparse GC matrices. We must note that GMM method require large number of data since we have to average GC matrices obtained from partitioning time series into small segments. So the length of the generated time series should be large enough to accommodate the requirement of GMM method. Since permutation test does not require any assumption on time series data (only on the hypothesis we tested), we expected that permutation test would perform better for both 20000 and 50000 time points data.

Experiment setting: We generated 40 ground truth state-space models with order 40, 10 output dimension and the density of non-zero entries is 0.1. Time series data were generated from these models with 20000 and 50000 time points. The number of states chosen in subspace identification is 40, equal to the order of ground truth models.

For permutation test, we chose the number of permutation to be 200 with 10 partitioning segments. This setting was used in both 20000 and 50000 time points cases.

For GMM method, we considered averaging over 4 samples of F in 20000 time points case, each estimated from 1000 time points of data, to obtain $\bar{F}$. So we must have 5 $(= \frac{20000}{4\times 1000})$ samples of $\bar{F}$. For 50000 time points case, we consider averaging over 5 samples of F, each estimated from 2000 time points of data. This also give 5 $(= \frac{50000}{5\times 2000})$ samples of $\bar{F}$. All $\bar{F}$ were then vectorized with their diagonal removed. The number of mode is chosen to be 2. In this form, we can obtain a single GMM model for every entries except the diagonal of $\bar{F}$. The first mode of the GMM with the smallest mean is used to determine if entries of F are zero or non-zero.

Result and discussion: From the result in *Table 3*, when the length of data is 20000 time points, permutation test with both p-values correction methods have better performance than GMM method as shown in higher ACC, TNR and lower FPR. For 50000 time points, the performance of GMM method is only slightly worse than permutation test with Bonferroni correction. Since using more data allows more samples in averaging in GMM method, the assumption of central limit theorem is satisfied even better. So the performance of GMM method is improved significantly as seen in sharply increase in ACC and TNR. For permutation test, the performances are consistent in both cases when using Bonferroni correction. Hence, our hypothesis is true for 20000 time points case but not in 50000 time points case where both methods give about the same performances.

Table 3: The performance of GMM method and permutation test on 20000 and 50000 time points data from ground truth models with sparse GC. The p-value correction methods used are abbreviated as Simple for thresholding without correction, Bon for Bonferroni method and B-H for Benjamini-Hochberg procedure.

Performance indices	20000 time points				50000 time points			
		Permutation test				Permutation test		
	GMM	Simple	Bon	B-H	GMM	Simple	Bon	B-H
ACC	0.9558	0.9297	0.9922	0.9803	0.9933	0.9447	0.9936	0.9869
TPR	1	1	1	1	1	1	1	1
TNR	0.9512	0.9223	0.9914	0.9782	0.9926	0.9389	0.9929	0.9856
FPR	0.0488	0.0777	0.0086	0.0218	0.0074	0.0611	0.0071	0.0144
FNR	0	0	0	0	0	0	0	0

5.5 Performance under different ground truth network densities

In this experiment, we compared the performance of permutation test and GMM method when the GC matrices of ground truth models are sparse and dense. We hypothesize that the performance for both permutation test and GMM method on dense ground truths should be the same as on sparse ground truths.

Experiment setting: The setting of this experiment was the same as section 5.4 except that the density of GC matrices in ground truth models were set to 0.4 instead of 0.1 to represent dense ground

Table 4: The performance of GMM method and permutation test on 20000 and 50000 time points data from ground truth models with dense GC. The p-value correction methods used are abbreviated as Simple for thresholding without correction, Bon for Bonferroni method and B-H for Benjamini-Hochberg procedure.

Performance indices	20000 time points				50000 time points			
		Permutation test				Permutation test		
	GMM	Simple	Bon	B-H	GMM	Simple	Bon	B-H
ACC	0.9472	0.9444	0.9781	0.9614	0.9869	0.9386	0.9672	0.9533
TPR	1	0.9949	0.9667	0.9898	1	0.9898	0.9411	0.9795
TNR	0.9218	0.9201	0.9835	0.9477	0.9807	0.9140	0.9798	0.9407
FPR	0.0782	0.0799	0.0165	0.0523	0.0193	0.0860	0.0202	0.0593
FNR	0	0.0051	0.0333	0.0102	0	0.0102	0.0589	0.0205

truth models. The obtained result is then compared with the result in section 5.4.

Result and discussion: *Figure 8* shows an example of GC pattern obtained from permutation test and GMM method on 20000 time points data from both sparse and dense ground truth models. When comparing the result in *Table 4* with the result in *Table 3* from section 5.4, we see that both permutation test and GMM method yield worse performances when ground truth models have denser GC matrices. The drop in performance is worse for permutation test especially when the length of time series data is 50000 time points as seen in lower ACC, TPR ,TNR and higher FPR and FNR. For GMM method, the performances are close for both dense and sparse ground truth models. As shown in *Figure 9*, when the ground truth has dense GC, non-zero mode spreads wider but it is still clearly distinctable from the zero mode. So the effect of GC density to the performance of GMM should be small. For permutation test, by having dense GC in ground truth models, there are more non-zeros to be tested. So this may result in higher chance of incorrectly classifying the zero causality in permutation test, that is, increasing in FNR and decreasing in TPR. For the heavy decrease in performance seen in 50000 time points case, one possible reason is that when ground truth models having denser GC, there are more causalities between channels in time series data. Permuting the segmented row of time series data with longer segment length (we fixed the number of partitions, so the length is increased when the data is longer) may have a higher chance to preserve or intensify the causality when some segments stay at the same positions, resulting in more GC samples from permutation that are larger than the estimated GC measure. So the obtained p-values can get higher for non-zero causalities and the thresholding might incorrectly classify them as zero causalities.

5.6 Comparison of the computation time between permutation test and GMM method

In this experiment, the computation time required for our scheme using permutation test and GMM method were compared. As permutation test need many samples of GC matrices obtained from each permutation, we expect see higher computation time in using permutation test than in GMM method.

Experiment setting: We set up timers in the previous experiment in section 5.4. In both 20000 and 50000 time points case, we averaged the computation times for GMM method and our method separately over all 40 data sets. Note that for permutation test, parallel computing with 6 threads was employed so that the required times are not too high.

Result and discussion: As expected, for both 20000 and 50000 time points data, computation times for permutation test were much higher than computation times for GMM method as shown in *Table 5*. When increasing the length of time series from 20000 to 50000 time points, GMM method required about 1.5 times higher computation time while permutation test needed more than twice of the computation time.

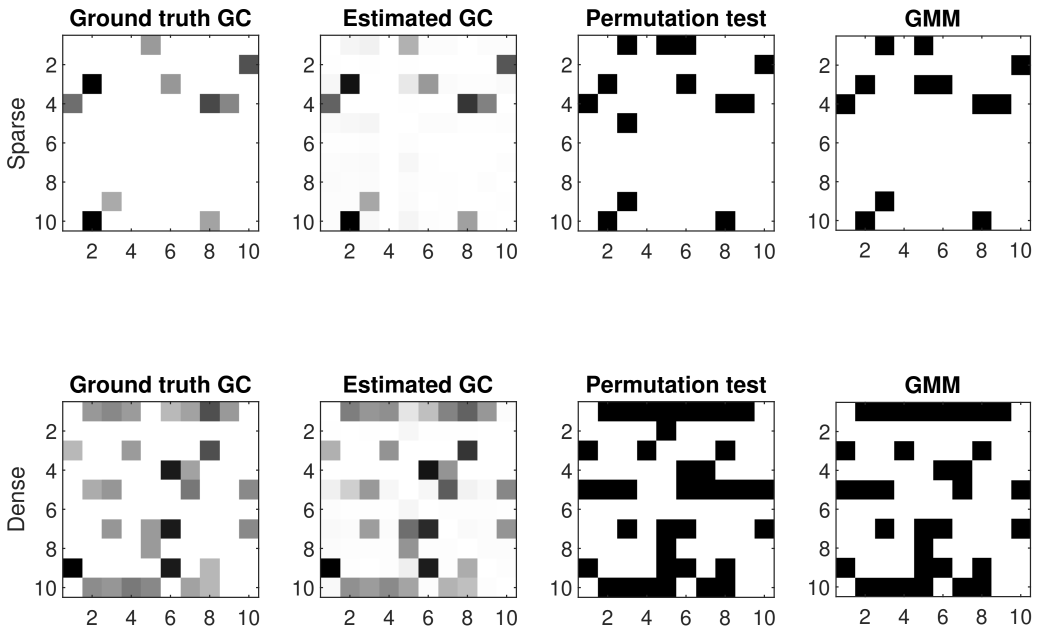


Figure 8: Examples of GC patterns obtained from permutation test and GMM method on 20000 time points time series data generated from the ground truth model with GC densities of 0.1 (sparse) and 0.4 (dense).

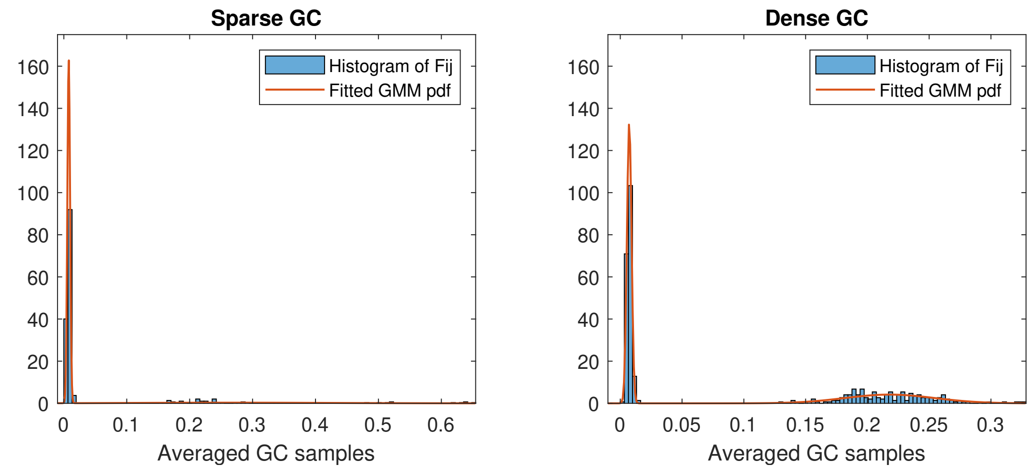


Figure 9: Examples of histograms of averaged GC samples and the fitted GMMs when the GC densities of the ground truth is 0.1 (sparse) and 0.4 (dense).

To explain this result, let consider learning GC pattern from n-dimensional time series data of length N using permutation test with P permutations and GMM method when splitting data into N_0 parts. From *Figure 1*, we can see that the time required to perform permutation test consists mostly of the time required for multiple calls of subspace identification algorithm and GC calculation. The times for the calculation of p-value and thresholding are considered to be negligible. For GMM method, as seen in *Figure 6*, there are also multiple calls of subspace identification algorithm and GC calculation but with GMM fitting and clustering at the end instead. Let $T_{\text{SSID}}(N)$ be the computation time of subspace identification on time series data of length N, T_{GC} be the computation time for computing GC matrix and T_{fitGMM} be the computation time for fitting GMM. Then, we may write the computation time of permutation test as

$$T_{\mathrm{perm}} = (1+nP)T_{\mathrm{SSID}}(N) + (1+nP)T_{\mathrm{GC}}$$

and the computation time of GMM method as

$$T_{\mathrm{GMM}} = (1+N_0)T_{\mathrm{SSID}}(N/N_0) + (1+N_0)T_{\mathrm{GC}} + T_{\mathrm{fitGMM}}$$

In our experiment, $n = 10, P = 200$ and $N_0 = 20$ and 25 for 20000 and 50000 time points data, respectively. So, we have $nP \gg N_0$ which means permutation test requires much more times using subspace identification and computing GC matrix. Other than that, subspace identification is done with shorter length data in GMM method. So, the large difference between computation time of permutation test and GMM method are resulted mainly from the sheer difference between the number of times calling subspace identification algorithm and GC calculation.

Table 5: The computation time (seconds) of GMM method and permutation test on 20000 and 50000 time points data. The computation times shown are averaged from the result of 40 data sets. For permutation test, parallel computing is also employed.

Average computation time (sec)	GMM method	Permutation test
20000 time points	3.1948	278.0139
50000 time points	4.5904	667.2358

6 Conclusions

In this project, we aims to develop a scheme for learning GC pattern of time series data using state-space models and permutation test. The scheme is including estimation of state-space parameters using subspace identification algorithm, calculation of GC matrix and classifying zero and non-zero GC using permutation test.

To explore the factors that affect the performance of our scheme, we carried out three experiments on the generated ground truth models to study the effect of the number of permutations used in the test, to compare between Monte-Carlo and complete permutation test and to study the effect of the order of state-space models used in subspace identification. The results showed that the performance was improved when the number of permutations in the test was increasing especially when p-value correction methods are applied. Next, we found that using complete permutation test gave only slightly better performance only when testing with uncorrected p-values. Moreover, both Bonferroni and Benjamini-Holchberg corrections were invalid for complete permutation test. Since applying p-values correction methods improve the performance significantly, Monte-Carlo test is more preferable. The effect of the order of state-space models was observed to give worse performance when underestimate than when overestimate.

Lastly, the experiment for comparison between our scheme using permutation test and the GMM method was conducted. We considered when ground truth models have sparse and dense GC matrices. The results showed, for sparse ground truths, that when the length of time series data is short, permutation test gives better performance while testing with p-values correction methods. In the other hand, when the length of time series data is long, GMM method can be as good as permutation test with only negligible difference. For dense ground truth models, both permutation test and GMM method perform worse compared to when the ground truth models are sparse. The drop in performance is higher in permutation test especially when perform on long time series data. The comparison of computation time, however, showed that permutation test suffers heavily from the excessive use of subspace identification while GMM method can be perform in a fraction of the computation time required in our scheme.

The scheme for learning GC pattern using permutation test is shown to perform well when the true model has sparse GC. The computational cost, however, is very concerning when apply this scheme to long time series data. So this scheme may be practical when the length of interested time series data is not too long.

7 References

[BS11] L. Barnett and A. K. Seth. Behaviour of Granger causality under filtering: theoretical invariance and practical application. *Journal of neuroscience methods*, 201(2):404–419, 2011.

[BS14] L. Barnett and A. K. Seth. The MVGC multivariate Granger causality toolbox: a new approach to Granger-causal inference. *Journal of neuroscience methods*, 223:50–68, 2014.

[BS15] L. Barnett and A. K. Seth. Granger causality for state-space models. *Physical Review E*, 91(4), 2015.

[Efr12] B. Efron. *Large-scale inference: empirical Bayes methods for estimation, testing, and prediction*, volume 1. Cambridge University Press, 2012.

[Goo05] P. Good. *Permutation, parametric and bootstrap tests of hypotheses: a practical guide to resampling methods for testing hypotheses*, volume 100. 2005.

[KFL19] W. Karwowski, F. Vasheghani Farahani, and N. Lighthall. Application of graph theory for identifying connectivity patterns in human brain networks: A systematic review. *Frontiers in Neuroscience*, 13:585, 2019.

[Lüt05] H. Lütkepohl. *New introduction to multiple time series analysis*. Springer Science & Business Media, 2005.

[NH02] T. E. Nichols and A. P. Holmes. Nonparametric permutation tests for functional neuroimaging: a primer with examples. *Human brain mapping*, 15(1):1–25, 2002.

[OM12] P. Van Overschee and B. De Moor. *Subspace identification for linear systems: Theory—Implementation—Applications*. Springer Science & Business Media, 2012.

[Son] J. Songsiri. EE531 - System Identification, Subspace Method. `http://jitkomut.eng.chula.ac.th/ee531/subspace.pdf`.

[Son19] J. Songsiri. Learning brain connectivity from EEG time series. Technical report, Chulalongkorn University, 2019. `http://jitkomut.eng.chula.ac.th/pdf/eeg_bc_final_jss.pdf`.

[Sto11] J. D. Storey. False discovery rate. *International encyclopedia of statistical science*, pages 504–508, 2011.

8 Appendices

8.1 MATLAB functions

The MATLAB file used in this project is included here. All functions can be found at `https://github.com/anawatnart/GCpermutation`.

Ground truth model generation

- `gen_sparseVAR.m`: Generate VAR parameters with controllable sparsity.
- `gen_diagfilter.m`: Generate diagonal filter with random poles and zeros.

Subspace identification

- `pvo_subspace/subfun/subid.m` [OM12]: Perform subspace identification on time series data.

GC computation

- `calgcss.m`: Calculate GC matrix from state-space parameters.

Permutation test

- `permdist_gc.m`: Calculate all GC matices from each permutation.
- `perm_pval.m`: Calculate p-values from the estimated GC matrix and permutation distribution obtain from `permdist_gc.m`.

GMM method

- `gmm_gc.m` [Son19]: Perform GMM method on averaged samples of GC matrices.