UMinho | 2017

Dominic Noy

PARAMETER ESTIMATION OF THE LINEAR PHASE CORRECTION MODEL BY MIXED-EFFECTS MODELS

Universidade do Minho
Escola de Ciências

Dominic Noy

PARAMETER ESTIMATION OF THE LINEAR PHASE CORRECTION MODEL BY MIXED-EFFECTS MODELS

November, 2017

Universidade do Minho
Escola de Ciências

Dominic Noy

PARAMETER ESTIMATION OF THE LINEAR PHASE CORRECTION MODEL BY MIXED-EFFECTS MODELS

A thesis presented for the degree of
Master of Science in Statistics

Supervised by
Dr. Raquel Menezes

November, 2017

Declaration

Email:

dominic.noy@gmail.com

Title:

Parameter Estimation of the Linear Phase Correction Model by Mixed-Effects Models

Supervisor:

Dr. Raquel Menezes

Year of Conclusion:

2017

University of Minho, 30/11/2017.

The author: [illegible]

Dedication

To Inês.

Acknowledgement

Raquel Menezes, thank you for the abundance of patience, effort, personal and professional support, trust, and most importantly, a lot of fun!

Financial support from the Portuguese Funds through FCT-“Fundação para a Ciência e a Tecnologia”, within the Project UID/MAT/00013/2013.

Abstract

The control of human motor timing is captured by cognitive models that make assumptions about the underlying information processing mechanisms. A paradigm for its inquiry is the *Sensorimotor Synchronization* (SMS) task, in which an individual is required to synchronize the movements of an effector, like the finger, with repetitive appearing onsets of an oscillating external event. The Linear Phase Correction model (LPC) is a cognitive model that captures the asynchrony dynamics between the finger taps and the event onsets. It assumes cognitive processes that are modeled as independent random variables (perceptual delays, motor delays, timer intervals).

There exist methods that estimate the model parameters from the asynchronies recorded in SMS tasks. However, while many natural situations show only very short synchronization periods, the previous methods require long asynchrony sequences to allow for unbiased estimations. Depending on the task, long records may be hard to obtain experimentally. Moreover, in typical SMS tasks, records are repetitively taken to reduce biases. Yet, by averaging parameter estimates from multiple observations, the existing methods do not most appropriately exploit all available information.

Therefore, the present work is a new approach of parameter estimation to integrate multiple asynchrony sequences. Based on simulations from the LPC model, we first demonstrate that existing parameter estimation methods are prone to bias when the synchronization periods become shorter. Second, we present an extended Linear Model (eLM) that integrates multiple sequences within a single model and estimates the model parameters of short sequences with a clear reduction of bias. Finally, by using Mixed-Effects Models (MEM), we show that parameters can also be retrieved robustly when there is between-sequence variability of their expected values.

Since such between-sequence variability is common in experimental and natural settings, we herewith propose a method that increases the applicability of the LPC model. This method is now able to reduce biases due to fatigue or attentional issues, for example, bringing an experimental control that previous methods are unable to perform.

Resumo

O controlo de factores temporais que ocorrem na execução de movimentos é captado por modelos cognitivos. Estes modelos são aproximações do processamento de informação, que ocorre no sistema nervoso. Para investigar este processo é utilizada a "Sensorimotor Synchronization Task" (SMS) que consiste em sincronizar os movimentos, por exemplo, de um dedo com eventos externos repetitivos. O "Linear Phase Correction Model" (LPC) permite prever a evolução da diacronia entre o movimento e o evento externo. Este modelo inclui variáveis aleatórias independentes, tais como atrasos no processamento da informação e execução da resposta.

Para se estimar os parâmetros do LPC são utilizados métodos que incluem as diacronias obtidas na SMS. Estes métodos precisam de sequências longas, no entanto o sincronismo verifica-se durante curtos períodos de tempo. Além disso, registam-se observações múltiplas para diminuir o viés na estimativa. Contudo, recorrendo à média de múltiplas estimativas, nem toda a informação disponível é considerada.

Com vista a colmatar as lacunas identificadas, este trabalho apresenta uma nova abordagem ao nível da estimativa dos parâmetros. Num primeiro momento, com base em simulações do LPC, demonstramos que os métodos existentes são enviesados, quando as sequências são curtas. Num segundo momento, apresentamos o "extended Linear Model" (eLM) que integra diacronias múltiplas no mesmo modelo. Por fim, usando o "Mixed-Effects Model" (MEM), mostramos que os parâmetros podem ser estimados quando os valores esperados variam entre sequências.

Uma vez que tal variabilidade é frequente e observável em contexto real, o método desenvolvido neste trabalho permite maior aplicabilidade do modelo LPC e reduz o viés causado por factores relacionados com problemas de atenção e de fadiga, introduzindo um novo controlo experimental.

Contents

List of Figures

Theoretical Concepts

Abbreviations

CNS	Central Nervous System
SMS	Sensorimotor Synchronization
acvf	autocovariance function
LPC	Linear Phase Correction Model
OLS	Ordinary Least Squares
GLS	Generalized Least Squares
bGLS	bounded Generalized Least Squares
MLE	Maximum Likelihood Estimator
ML	Maximum Likelihood Estimation
REML	Restricted Maximum Likelihood Estimation
BLUE	Best Linear Unbiased Estimator
eLM	extended Linear Model
lMEM	linear Mixed-Effects Model
MEM	Mixed-Effects Model
MA1	Moving Average Model of order 1

Chapter 1

Introduction

Humans coordinate their movements with nearby moving objects in the environment with a remarkable ease. This requires a highly timed communication of the perception-action systems underpinning the movement control. In order to investigate the underlying timing mechanisms, employed by the Central Nervous System (CNS), researchers study participants' attempt to synchronize their movements concurrently with repetitively occurring environmental events. Synchronization can be understood as a simplified type of coordination because it is constrained in space and time. It is particularly important in activities such as music, sports, and manufacturing. Synchronizing movements with a partner was also shown to increase social aspects, such as social attachment and cooperation (Wiltermuth & Heath, 2009; Valdesolo, Ouyang, & DeSteno, 2010; Reddish, Fischer, & Bulbulia, 2013), rapport (Miles, Nind, & Macrae, 2009), and likability (Launay, Dean, & Bailes, 2014), and it was traditionally used as a means to enhance self-esteem and obedience (Valturio, 1921).

The study of motor synchronization is mostly focused on effectors like the fingers (Repp, 2005), the forearms (Mörtl et al., 2012), or the feet (van Ulzen, Lamoth, Daffertshofer, Semin, & Beek, 2008) to be timed with external events like auditory metronomes, light displays, or interacting partner movements (Schmidt & Richardson, 2008; Noy et al., 2017).

Successfull synchronization requires the individual to a) perceive the event onsets; b) perceive one's movement onset; c) compute the asynchrony between both onsets; d) compute the temporal progression of the repeated event series; e) follow all these steps to predict upcoming event onsets.

Based on these perceptual processes, appropriate motor commands can be computed so that the asynchrony—between the movement and the event—becomes reduced to a minimum (Grush, 2004; Van Der Steen & Keller, 2013). When the external event is presented with constant temporal intervals (these may also vary slightly), this paradigm is called *Sensorimotor Synchronization* (SMS) (Repp, 2005).

There are cognitive models accounting for the empirical findings obtained from SMS tasks. Cognitive models usually use a mathematical representation, formalized as a parameterized system of equations that receives input, for example, sensory cues about the onsets and previous asynchronies (Wing & Kristofferson, 1973; Schulze & Vorberg, 2002; Jacoby, Tishby, Repp, Ahissar, & Keller, 2015) and produce output, for example a motor response to reduce the next asynchrony. By solving (or approximating) such systems, its parameters are revealed.

These models can be challenged by comparing their analytical or simulated output—for a given input and set of parameters—with experimental observations. By systematically manipulating the input, it can be validated whether such processes–as postulated by the particular model– in fact underpin the information processing of the CNS.

Because in experiments there are always variables that can neither be manipulated nor controlled–i.e., there is noise within and beyond the CNS–these problems are usually approached in a probabilistic manner. Within the framework of probability theory, a model can be defined as a parametric family of probability distributions. The combination of probability distributions (indexed by parameters) determines the distribution of the input and associates a probability of occurrence to each output. Probabilistic models are used to model cognitive abilities. Usually, the challenge is to determine how the observed quantities relate to the model parameters in question (Myung, 2003).

In cognitive models of motor synchronization, the observed quantities are the asynchrony dynamics between the onsets of oscillating motion of an individual and the onsets of a repetitively appearing stimulus. The subject of inquiry is the relation of these asynchronies to the parameters of the underlying timing model.

Our scope is a) to give a brief overview of such models, b) present their current parameter estimation approaches and limitations, and c) to introduce a novel approach of parameter estimation. In Chapter 2, we present the synchronization models of interest. In Chapter 3, we first, present the most recent parameter estimation method and illustrate that it is biased when certain experimental conditions are not met (i.e., when the asynchrony sequences become shorter). We, then, present an extended Linear Model revealing superior estimation performance in such conditions. Finally, we present a Mixed-Effects Model that also accounts for additional intergroup-specific factors, and therefore most robustly estimates the model parameters. The main contribution of this work is the finding of robust parameter estimation methods that allow validating the LPC model on more complex empirical observations from movement synchronization experiments.

Throughout this manuscript, there are presented gray-shaded text fields referred to as "Notes". These introduce mathematical theorems and general models. They can be utilized to gain further insight into theoretical concepts but their considerations are not necessary for following the study.

Chapter 2

Event-Based Timing Models

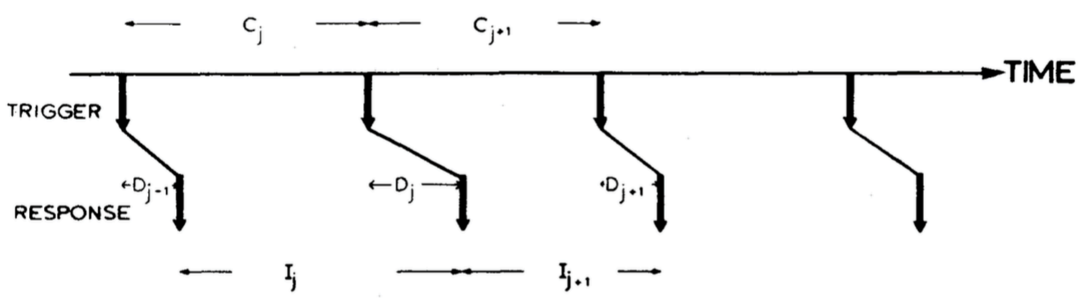
2.1 Timing of Discrete Motor Responses

In order to account for human timing processes, Wing and Kristofferson (1973) developed a probabilistic cognitive model, which describes the timing behavior of individuals who have to execute repetitive movements at constant temporal intervals. When the intervals are determined by an external metronome that suddenly stops and the individual is required to continue executing the constant movement intervals, this method is called the synchronization-continuation paradigm. Based on the variability of the movement intervals (i.e., the time between two successive taps), Wing and Kristofferson (1973) proposed the following model (see Figure 2.1) [1]:

$$I_j = C_j - D_{j-1} + D_j, \tag{2.1}$$

where I_j is the movement interval j, C_j is the internal representation of the interval I_j (*Time Keeper*), and D_j comprises the perceptual and motor delays. I_j is the temporal response interval bounded by two successive taps, which are determined by $C_{j-1} - D_{j-2} + D_{j-1}$ and $C_j - D_{j-1} + D_j$. In follow-up studies, this was changed to $C_j - D_{j-1} + D_j$ and $C_{j+1} - D_j + D_{j+1}$ (Schulze & Vorberg, 2002). C_j and D_j are defined as independent random variables with $C_j \sim NV(\mu_C, \sigma_C^2)$ and $D_j \sim NV(\mu_D, \sigma_D^2)$.

[1] For the introduction of the existing models and techniques, we used the notation of the original articles. For this reason, notations of the same variables and parameters can vary throughout this work.



Reprinted from Wing and Kristofferson (1973).

Figure 2.1: Model of continuation tapping: "Schematic of a two-process mechanism for the timing of repetitive discrete motor responses" (Wing & Kristofferson, 1973, p.6).

By assuming C_j and D_j $(i \neq j)$ as independent (consult Note 1 and Note 2 for a brief introduction of the concept of independent random variables), the parameters of the model can be analytically estimated based on the serial dependence of the successive movement intervals:

$$\rho_I(1) = \frac{\gamma_I(1)}{\gamma_I(0)},$$

$$\begin{aligned}\gamma_I(1) &= E[(I_j - \mu_I)(I_{j-1} - \mu_I)],\ j = \ldots, -1, 0, 1, \ldots, \\ &= E[(C_j - D_{j-1} + D_j - \mu_I)(C_{j-1} - D_{j-2} + D_{j-1} - \mu_I)], \\ &= -E[(D_{j-1} - \mu_D)(D_{j-1} - \mu_D)], \\ &= -\sigma_D^2, \end{aligned} \tag{2.2}$$

$$\begin{aligned}\gamma_I(0) &= E[(I_j - \mu_I)(I_j - \mu_I)], \\ &= \sigma_I^2, \\ &= E[(C_j - D_{j-1} + D_j - \mu_I)(C_j - D_{j-1} + D_j - \mu_I)], \\ &= \sigma_C^2 + 2\sigma_D^2,\end{aligned}$$

where $\rho_I(1)$ is the lag-1 autocorrelation of successive movement intervals, $\gamma_I(1)$ is the lag-1 autocovariance, $\gamma_I(0)$ is the lag-0 autocovariance, i.e. the variance of the movement intervals (I_j), μ_D is the mean of the perceptual and motor delays, and μ_C is the mean of the constant stimulus interval. The former can be set to zero without loss of generality and the latter is determined by the experimenter. See Note 3 for the introduction of an estimator.

Note 1: *Random Variable*

If Ω is the sample space of all possible outcomes of an experiment, and to all elements of Ω is associated a probability measure and a real valued function X, then X is referred to as random variable.

Note 2: *Independence*

The random variables X and Y are independent if their realizations do not affect the distributions of the other random variables.
For (cumulative) distribution functions:

$$F_{X,Y}(x,y) = F_X(x)F_Y(y).$$

For probability density functions:

$$f_{X,Y}(x,y) = f_X(x)f_Y(y).$$

Assuming independent random variables X_j, X_i, Y_j, and Y_i $(i \neq j)$ implies:

$$\begin{aligned} Cov(X_i, X_j) = 0,\ Cov(Y_i, Y_j) = 0, \\ Cov(X_i, Y_i) = 0,\ Cov(X_i, Y_j) = 0, \\ \forall i \neq j \end{aligned}$$

The model in Equation 2.1 suggests that $\gamma_I(1)$ is different of zero due to the simultaneous presence of D_{j-1} and D_j at the j^{th} iteration. Assuming independent random variables, γ_I is supposed to be zero at larger lags (> 1). Taking into account that D_j comprises perceptual and motor delays, the serial dependence of I_j may reflect the degree of noise (variability) within their respective information processing pathways (Wing & Kristofferson, 1973).

This was supported by systematic manipulation of the experimental conditions. It was possible to decompose the overall variability of σ_I^2 into the variabilities of the two independent subprocesses σ_C^2 and σ_D^2 by experimentally increasing the time between two successive stimulus onsets. While C_j increased σ_T^2, σ_D^2 was unaffected. This makes it possible to dissociate both processes (see Wing & Kristofferson, 1973). Such dissociation was consistent with theoretical concepts and highlights the applicability of quantitative cognitive modeling to activities like music or sports. It allows to attribute performance variability to its causes so that training/treatments can be developed to address such causes and therefore reduce the variability of the identified process.

Note 3: *Estimator*

Random variables are distributed according to a parametrized model (e.g., the normal distribution). The parameters of the model (e.g., μ and σ) are unknown and have to be estimated from an observed sample of realizations of the random variable.

A "statistic" is a function of these random variables (e.g., the sample mean $\frac{1}{N}\sum_i^N x_i,\ i = 1, \ldots, N$). Using the statistic, the parameter of the model can be estimated. If θ is the model parameter, then a statistic is called the estimator $\hat{\theta}$ of θ.

If $\hat{\theta}$ estimates θ "on the long run", $\hat{\theta}$ is considered an unbiased estimator of θ. The "bias" of θ is defined as

$$B(\hat{\theta}) = E[\hat{\theta} - \theta],$$

where E[.] is the expected value. The bias is used to reflect the "accuracy" of the $\hat{\theta}$. When comparing two estimators $\hat{\theta}_1$ and $\hat{\theta}_2$, the estimator with lower variance is considered as more "efficient".

For evaluating the goodness of fit of $\hat{\theta}$, it is often used the "Mean Squared Error", which is the sum of the efficiency and the squared bias:

$$MSE(\hat{\theta}) = Var(\hat{\theta}) + B(\hat{\theta})^2.$$

2.2 The Linear Phase Correction model (LPC)

Based on Wing and Kristofferson (1973)'s model, Schulze and Vorberg (2002) developed the *Linear Phase Correction model* (LPC) (see Figure 2.2 and Equation 2.3). Figure 2.3 presents an example of an asynchrony sequence that can be observed when an individual synchronizes ones' finger taps with a metronome in a SMS task for 100 metronome onsets. The asynchrony sequence is a set of random variables captured by the LPC model:

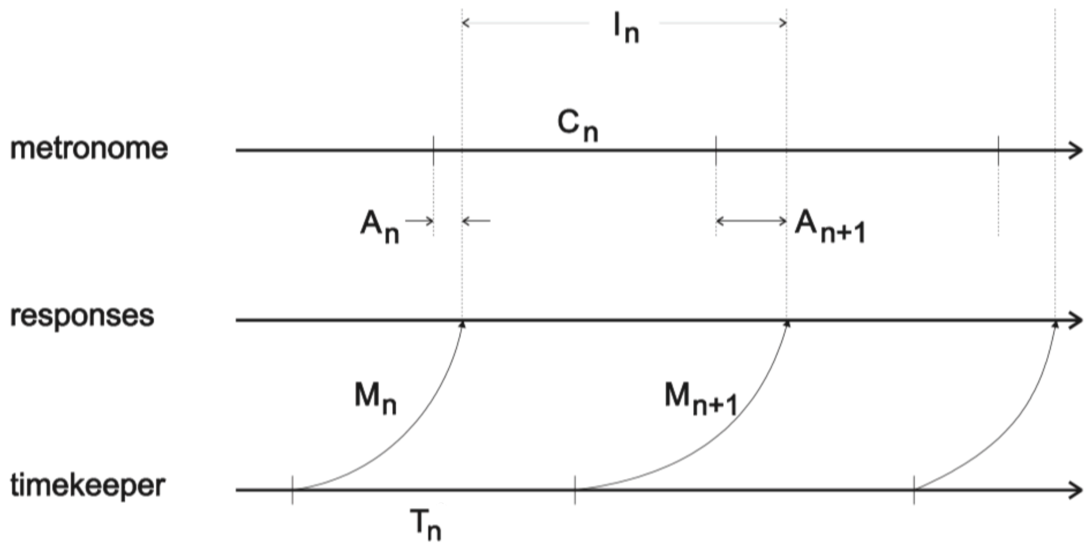
$$A_{n+1} = (1-\alpha)A_n + T_n + M_{n+1} - M_n - C, \tag{2.3}$$

where A_n is the asynchrony at iteration n, C is a constant metronome interval, M_n is the motor delay, and T_n is the Time Keeper interval. M_n and T_n are random variables with $M_n \sim NV(\mu_M, \sigma_M^2)$ and $T_n \sim NV(\mu_T, \sigma_T^2)$

Thus, the LPC describes the temporal behavior of the observed asynchronies A_{n+1} as a linear combination of the preceding asynchronies A_n, a cognitive representation of the external event structure T_n, and the information processing delays within the CNS, M_n and M_{n+1}. Although there have been attempts to treat them separately, similar to the previous model in Equation 2.1, Schulze and Vorberg (2002) summarized the information processing delays (perceptual delays & motor delay) in a single variable, referred to as motor delays M_n. This is so because with the existing approaches it is not possible to find a unique solution for both, the motor and perceptual delay parameters. This is referred to as "parameter identifiability problem". Parameter identifiability implies that different combinations of parameters cannot lead to the same results (Schulze & Vorberg, 2002).

The parameters of the model in Equation 2.3 can be estimated by the empirical autocovariance function (acvf) (see Schulze & Vorberg, 2002). The LPC received empirical support for its validity (see e.g., Repp, 2005; Torre & Balasubramaniam, 2009; Zelaznik, Spencer, & Ivry, 2002) and was extended to circumstances in which the base tempo of the metronome changed (i.e., C_n as a function of n) and therefore μ_T had to be adjusted (i.e., period correction) (Repp & Keller, 2004; Repp, 2001) or when the metronome adjusted its intervals as a function of the individuals' movement dynamics (Repp & Keller, 2008).

Yet, there were published two articles demonstrating shortcomings of the estimation method of Schulze and Vorberg (2002) (Jacoby, Tishby, et al., 2015; Jacoby, Keller, Repp, Ahissar, & Tishby, 2015). The authors argued that it is biased when the asynchrony sequences were obtained from SMS tasks with variable metronome inter-



Reprinted from Schulze and Vorberg (2002).

Figure 2.2: Linear Phase Correction Model. “Timing diagram for the two-level model of synchronization” (Schulze & Vorberg, 2002, p. 83). This is the most prominent Linear Phase Correction Model.

vals. When the temporal intervals changed or phase perturbations occurred—what is common in natural settings—the parameters had to be estimated by fitting the empirical acvf to computer simulations. This is slow and often no unique solution exists. Therefore, Jacoby, Tishby, et al. (2015) suggested an alternative method of parameters estimation, called the “bounded Generalized Least Squares method” (bGLS). See Note 4 for an introduction of Least Squares Estimators. The bGLS method formalizes the serial dependence of asynchronies as a simple regression problem in which succeeding asynchronies linearly depend on previous asynchronies.

This method could master observations from experiments with variable metronomes and was generally less biased, more efficient, and faster than the traditional estimation techniques (Schulze & Vorberg, 2002) for a wide range of settings (Jacoby, Tishby, et al., 2015). In addition, the bGLS method could capture the synchronization dynamics of two or more interacting individuals that coordinate in a group, as for instance when musical orchestra elements had to be coordinated (see also Wing, Endo, Bradbury, & Vorberg, 2014). A detailed description of the bGLS method follows in Section 3.1.

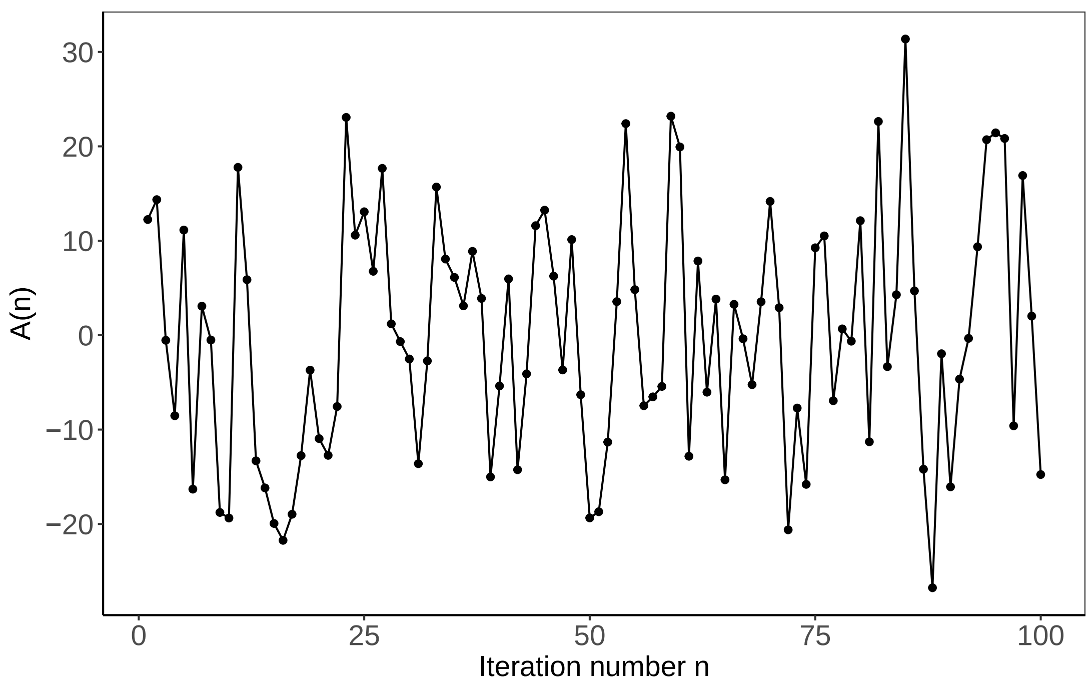


Figure 2.3: Output of the LPC model simulation with $\alpha = 0.5$, $\sigma_T = 5$, and $\sigma_M = 10$. It shows a sequence of asynchronies $A_n,\ n = 1, \ldots, 100$. The goal of the individual is to minimize A_n. Due to variability within the CNS, A_n fluctuates around μ_A, which is here set to zero.

Note 4: *Least Squares Estimators*

Ordinary Least Squares (OLS). In order to estimate parameters of the linear model, it can be used the OLS method:

$$\hat{\beta} = (X^T I X)^{-1} X^T y,$$
$$Var(\hat{\beta}) = \sigma^2 (X^T I X)^{-1},$$

where X is a N x N design matrix, I is a N x N identity matrix, and y is a N x 1 column vector of observations. Thus, the OLS assumes independent and identically distributed observations. It is biased when these assumptions are not met.

Generalized Least Squares (GLS). The GLS estimator relaxes those assumptions by modeling heteroscedasticity and correlations between observations through the N x N variance-covariance matrix W that can have varying diagonal and off-diagonals entries:

$$\hat{\beta}^{GLS} = (X^T W^{-1} X)^{-1} X^T W^{-1} y,$$
$$Var(\hat{\beta}^{GLS}) = (X^T W^{-1} X)^{-1}.$$

While the OLS is then different than the Maximum Likelihood Estimator (MLE) (see Note 6, the OLS coincides with the MLE, when W is diagonal and homoscedastic), the GLS coincides with the MLE in such circumstances. It is also known as the best linear unbiased estimator (BLUE) of the population parameters.

Feasible Generalized Least Squares (fGLS). In order to use the GLS, the entries of W must be specified. The problem is that W is unknown and therefore the GLS cannot be used as such. For this reason, W is estimated from the empirical observations and then β is estimated by $\hat{\beta}^{GLS}$.

When alternately estimating W and β with the GLS, $\hat{\beta}^{GLS}$ is called a fGLS estimator

Concluding, we presented briefly the framework of linear phase correction models.

The bGLS method appears to be currently the “state-of-art method” of LPC model parameter estimation. The scope of this work is to present two alternatives to the bGLS method. In order to evaluate and compare the performance of different methods, we simulated asynchrony sequences that could be the output of an experiment using SMS tasks. This was done by running the LPC model in Equation 2.3, see Appendix B. It received as input an initial asynchrony A_1, constant metronome onsets with iteration number, and a set of parameters (σ_T, σ_M & α). Parameter settings were held close to those of previous studies (Jacoby, Tishby, et al., 2015). The output of a single simulation is a sequence of asynchronies. Such a simulation was performed multiple times and the parameter estimation methods were evaluated by considering its bias and efficiency with which they recovered the set of parameters of the LPC model that had generated the data.

The main chapter of this thesis, Chapter 3, is structured as follows: First, we present and scrutinize the above-mentioned bGLS method. Therefore, we replicate the code of Jacoby, Tishby, et al. (2015) (from Matlab code to R code) and compare it with their results. Subsequently, we present and evaluate two alternative methods on the simulated output. The reader can consult the Appendices for reproducible R codes of all performed simulations, estimations, and method validations.

Chapter 3

Parameter Estimation Methods

3.1 The bounded Generalized Least Square (bGLS) Method

The goal of this section is to present the bGLS method and replicate the results of Jacoby, Tishby, et al. (2015). Section 3.1.1 describes the formal method, section 3.1.2 describes the computations, and section 3.1.3 shows the results of the perfomed computations on the simulated data (see Appendices B, C, & D for the here developed R programs).

3.1.1 Method

In order to introduce the bGLS method, it is convenient to write the LPC model in matrix form:

$$y = Bx + Z, \tag{3.1}$$

where

$$y = \begin{vmatrix} A_1 - E[A_k] \\ \vdots \\ A_N - E[A_k] \end{vmatrix}, B = \begin{vmatrix} A_0 - E[A_k] \\ \vdots \\ A_{N-1} - E[A_k] \end{vmatrix}, \ x = (1-\alpha), \ Z = \begin{vmatrix} H_0 \\ \vdots \\ H_{N-1} \end{vmatrix},$$

and where A_k is the asynchrony at iteration $k = 1, \ldots, N$, N is the length of the sequence, $E[A_k]$ is the expected value of A_k, α is the correction coefficient. For this approach, N should be the same for all sequences.

Note 5: *Expected Value*

The expected value of a random variable is the average of realizations of X referred to as x obtained from a theoretically infinite number of repetitions of an experiment. Because X is here a continous random variable, it admits a probability density funcion $f(x)$ (pdf). The expected value of X is:

$$E[X] = \int_{-\infty}^{\infty} xf(x)dx$$

Z follows a multivariate normal distribution with zero mean and N x N variance-covariance matrix Σ. Considering that $Z = [Z_0, Z_1, \dots, Z_{N-1}]^T$, where $Z_k = T_k + M_{k+1} - M_k - E[T_k]$, it can be specified by $\gamma_Z(j) = Cov[Z_k, Z_{k+j}]$ according to

$$\begin{aligned}
\gamma_Z(1) &= Cov[(T_k + M_{k+1} - M_k), (T_{k+1} + M_{k+2} - M_{k+1})] \\
&= Cov[M_{k+1}, -M_{k+1}] \\
&= -\sigma_M^2, \\
\\
\gamma_Z(0) &= Var[T_k + M_{k+1} - M_k] \\
&= \sigma_T^2 + 2\sigma_M^2, \\
\\
\gamma_Z(j) &= 0,\ j > 1,
\end{aligned} \tag{3.2}$$

so that

$$\begin{aligned}
&Z \sim MVN(0, \Sigma),\ \Sigma = \gamma_Z(0)I + \gamma_Z(1)\Delta, \\
&\gamma_Z(0) = 2\sigma_M^2 + \sigma_T^2,\ \gamma_Z(1) = -\sigma_M^2,\ \gamma_Z(j) = 0,\ j > 1,
\end{aligned}$$

$$\Delta = \begin{bmatrix} 0 & 1 & 0 & \cdots & 0 & 0 \\ 1 & 0 & 1 & \ddots & \vdots & 0 \\ 0 & 1 & 0 & \ddots & 0 & \vdots \\ \vdots & 0 & \ddots & 0 & 1 & 0 \\ 0 & \vdots & \ddots & 1 & 0 & 1 \\ 0 & 0 & \cdots & 0 & 1 & 0 \end{bmatrix}.$$

I is a N x N identity matrix and Δ is a N x N matrix determining non-zero correlations.

In short, the asynchrony in the next iteration A_{k+1} is linearly related to the asynchrony in the previous iteration A_k captured by Bx. This is conceptually related to the fact that the individual attempts to correct the asynchrony by a correction parameter α. However, at each iteration, there is also variability within the CNS arising from noise within the time keeper, motor, and perceptual processes. This variability is captured by Z_k, which is not independent from Z_{k-1} and Z_{k+1} and therefore should lead to autocorrelated asynchrony sequences of A_k.

Next, it is presented how the parameters of the model in Equation 3.1 can be estimated by the Maximum Likelihood Estimator. For its introduction, consult Note 6.

3.1.2 Computation

The likelihood of x and $\Sigma(\sigma_T, \sigma_M)$ given Z ($Z = y - Bx$) is

$$L(x, \Sigma(\sigma_T, \sigma_M) \mid Z) = \frac{1}{\sqrt{(2\pi)^N |\Sigma|}} e^{-\frac{1}{2}(y-Bx)^T \Sigma^{-1}(y-Bx)}, \tag{3.3}$$

where Σ can be determined by the acvfs $\gamma_Z(0)$ and $\gamma_Z(1)$ (see Equation 3.2).

The log-likelhood function is

$$l(x, \Sigma(\sigma_T, \sigma_M) \mid Z) = -\frac{N}{2} log(2\pi) - \frac{1}{2} log(|\Sigma|) - \frac{1}{2}(y - Bx)^T \Sigma^{-1}(y - Bx). \tag{3.4}$$

Since x and Σ (i.e., α, σ_T and σ_M) are unknown, their estimation requires to iteratively estimate x and Σ referred to as *Feasible Generalized Least Squares* (see Jung, 1987 in Repp, Keller, & Jacoby, 2012). This means that x was estimated when Σ was fixed at a particular (estimated) value and Σ was estimated when x was fixed at a particular (estimated) value. Because, the MLE estimator coincides with the GLS estimator, as noted by Jacoby, Tishby, et al. (2015), we performed the estimations by maximizing $l(x, \Sigma|Z)$ of Equation 3.4. The function $l(x, \Sigma|Z)$ is usually maximized rather than $L(x, \Sigma|Z)$ because it is computationally easier. Both functions, have a 3-dimensional shape spanned by the parameters x, σ_M and σ_T and the best estimator of all three parameters is the position within the parameter space where their joint function has its maximum, which is the same for both $L(x, \Sigma|Z)$ and $l(x, \Sigma|Z)$.

Note 6: *Maximum Likelihood Estimator (MLE)*

The MLE is another parameter estimator. While OLS, GLS, and fGLS estimators search for the parameters that minimize the sum of squared residuals of the model, the MLE searches for parameters that make the observations most likely. $y = [y_1, \ldots, y_m]$ are observations, randomly sampled from a population of interest. w is a vector of parameters. $f(y \mid w)$ is the probability to observe y given w, called conditional probability densitiy function.
The MLE makes explicit assumptions about the distribution that had generated the observations:
Assuming the elements to be independent, that is y_i and y_j, $\forall i \neq$j, the pdf of y can then be written as

$$f(y \mid w) = f_1(y_1 \mid w) f_2(y_2 \mid w) \ldots f_m(y_m \mid w).$$

Usually the experimenter observed a sample y and wants to find w. This is approached by the *likelihood function*:

$$L(w \mid y) = f(y \mid w).$$

the likelihood function denotes the likelihood of the parameter w given the observations y.
By maxizing $L(w \mid y)$—therefore called MLE—we obtain the parameters w that most likely produced y.

$$\underset{w_1, w_2, \ldots, w_r}{max L(w)} \Rightarrow \frac{\partial L}{\partial w_i}(i = 1, \ldots, r) = 0.$$

In general, the MLE is a consistent, unbiased, and efficient estimator of the parameters.

The bGLS algorithm

In order to obtain the LPC model parameters, it is used the bGLS algorithm of Jacoby, Tishby, et al. (2015). Input are $nseq$ sequences of N asynchronies ($A_{ik},\ i = 1,\ldots,nseq,\ k = 1,\ldots,N$). Output are $\hat{\alpha}$, $\hat{\sigma}_M$, and $\hat{\sigma}_T$. For the first iteration, matrix $\Sigma_{niter=1}$ is assumed/set as the identity matrix I. Then, the following equations are iterated $Niter$ times ($niter = 1,\ldots,Niter$) for each sequence i, with $Niter = 20$.

Iteration:

1. x_{niter} is estimated by OLS with $\Sigma_{niter} = I$: $x_{niter} = (B^T\Sigma_{niter}^{-1}B)^{-1}(B^T\Sigma_{niter}^{-1})y$.
2. The residuals d of the model fit are obtained by computing $d_{niter} = y - Bx_{niter}$.
3. $\gamma_{niter}(0)$ is obtained by computing the $acvf_d(0)$ of the residuals d at lag 0.
4. $\gamma_{niter}(1)$ is obtained by computing the $acvf_d(1)$ of the residuals d at lag 1.
5. $\gamma_{niter}(1)$ is adjusted by decreasing/increasing so that:
$$0 < -\gamma_{niter}(1) < \gamma_{niter}(0) + 2\gamma_{niter}(1)^1.$$
6. $\Sigma_{niter+1}$ is specified by $\gamma_{niter}(0)$ and $\gamma_{niter}(1)$.

At the last iteration ($Niter = 20$), the parameter estimates of each sequence i are obtained by

$$\hat{\alpha}_i = 1 - x_{20},\ \hat{\sigma}_{M_i} = \sqrt{-\gamma_{20}(1)},\ \hat{\sigma}_{T_i} = \sqrt{\gamma_{20}(0) - 2\hat{\sigma}_{M_i}^2},\ i = 1,\ldots,nseq.$$

This procedure is then repeated for each sequence i ($i = 1,\ldots,nseq$), with $nseq = 15$, and the average of the $nseq$ parameter estimates is taken as final estimate:

$$\frac{1}{nseq}\sum_{i=1}^{nseq}\hat{\alpha}_i = \hat{\alpha},\ \frac{1}{nseq}\sum_{i=1}^{nseq}\hat{\sigma}_{M_i} = \hat{\sigma}_M,\ \frac{1}{nseq}\sum_{i=1}^{nseq}\hat{\sigma}_{T_i} = \hat{\sigma}_T.$$

[1]Step 5 of the iteration is an additional constraint justified in Jacoby, Keller, et al. (2015), referred to as "bounded", giving their method the name **b**GLS.

See Appendix C for the implementation of these computations in R.

3.1.3 Results & Discussion

Figure 3.1 shows the means and the 95% confidence intervals of the parameter estimates with the bGLS method ($\hat{\alpha}$, $\hat{\sigma}_T$, and $\hat{\sigma}_M$) for different α. For the range of α from 0.1 to 1.2, the bGLS method estimates α and σ_T almost without bias and σ_M with a slight negative bias of approximately 0.5. At this α range, the estimations of α and σ_T are very similar to the results of Jacoby, Tishby, et al. (2015). Yet, for σ_M, Jacoby, Tishby, et al. (2015) presented less biased but also less efficient estimations compared to the present results. However, Jacoby, Tishby, et al. (2015) conducted extensive simulations, with more than 1000 repetitions of the method, which might by parts explain such differences. Furthermore, while Jacoby, Tishby, et al. (2015) did not present estimates at $\alpha > 1.2$, larger values are theoretically possible up to $\alpha = 2$. At this range, our results revealed that the bias for α and σ_T is increased. On the other hand, σ_M seems to be estimated with slightly more accuracy. Nevertheless, those biases are very small and estimations might be sufficiently satisfying for most applications.

Overall, we are the opinion that the results are quite similar to the results of Jacoby, Tishby, et al. (2015)'s study and, therefore, we propose that their method was here successfully replicated.

Limitations

Sequence length. A limitation with the above-described approach, and former methods, is that they require long and constant asynchrony sequences. The more traditional methods used the acvf of the asynchrony sequence (Schulze & Vorberg, 2002). For a meaningful acvf, it was suggested that the length should be at least $min(N) \geq 50$ (Murteira, Muller, & Turkman, 1993). Similarly, the bGLS method searches for an approximated MLE. This is only reliable and unbiased if the sequence is relatively large (Ljung, 1998 in Jacoby, Keller, et al., 2015). As stated by Jacoby, Tishby, et al. (2015), this should be at least $min(N) \geq 30$. Considering the slightly biased estimates in Figure 3.1, our results point towards that this may be even larger than 30.

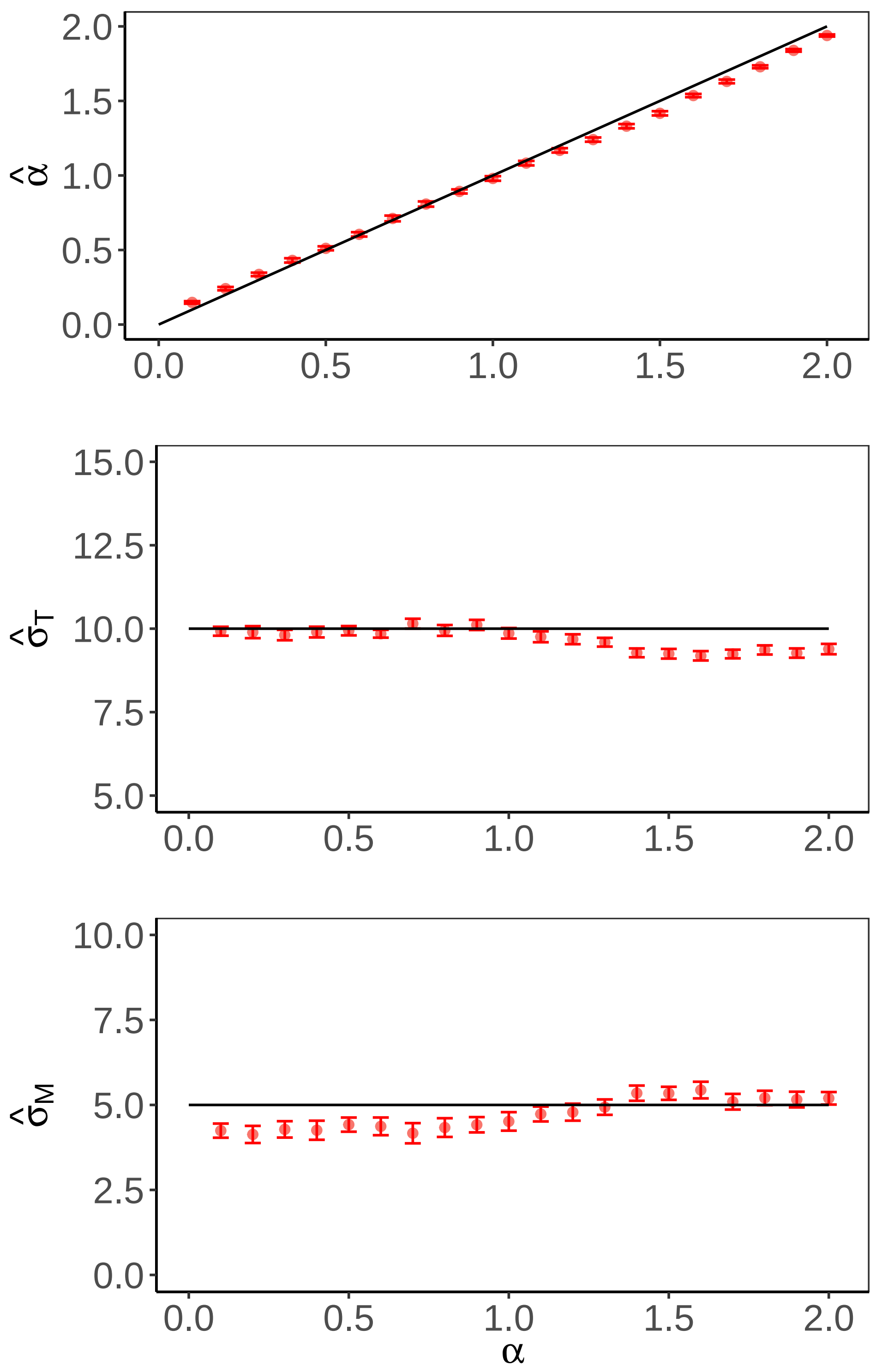


Figure 3.1: Parameter estimation by the bGLS method. The plots show the sample mean and the 95% confidence intervals of 50 estimates for different α values. Red dots are the means. When lying on the black line, the estimates coincide with the true values. The sequences were obtained by simulating the LPC model with $\sigma_M = 5$, $\sigma_T = 10$ and α according to the ordinate. Each estimate was obtained from $nseq = 15$ asynchrony sequences of length $N = 30$.

However, in many natural situations that an experimenter might want to simulate, synchronization can be observed for only very short time periods. In dance, partners alternately synchronize and eventually desynchronize their movements; in manufacturing work, the demand to coordinate with machines and other workers may be repetitive but short lasting; in a symphony orchestra, instruments such as cello, violin, piano, and celesta stand alone or together, and sometimes start and stop for very short time periods. A typical strategy in gait rehabilitation is that the patient synchronizes the stepping pattern during walking with external cues (see e.g., Lim et al., 2005), but only for a few steps, probably in order to avoid fatigue. Short lasting interactions that involve movement synchronization also exist in sports and in everyday coordination. These activities have in common that the movements become synchronized very quickly and last only short periods of time. Up to now, there do not seem to exist appropriate estimation approaches within the framework of event-based timing models, presented above, that can deal with short-lasting synchronization phenomena.

Experimental Design. Another important limitation of the bGLS method and former methods is that it disregards information due to averaging. In a typical experimental paradigm, one makes inferences about parameters of the model that is supposed to have generated the behavior. To achieve this, behavioral records are usually obtained repetitively, called trials or runs, and an average of these records, or their parameter estimates, across trials is taken.

When the trials were performed by the same individual, averaging may eliminate noise caused by factors such as tiredness, training effects, and distraction. When the trials were performed by different individuals (or groups), averaging may eliminate irrelevant between-subject variability not related to the study goals. Yet, by averaging, one may lose essential information, outliers can bias the results, and if trials with different lengths are included, they are weighted equally inducing further bias. It should be desirable to estimate the parameters without applying such “mean-function”. This is particularly important when there is little information on each trial, that is, when trials are short.

Solutions

For the present problem, an asynchrony sequence can be regarded as the output of the performance of a participant in one of multiple trials of an experiment (see Figure 3.2 for an illustration). In order to implement such so called repeated-measurement paradigm, appropriate methods must be developed that can estimate the parameters from multiple and short asynchrony sequences. Models that may account for these patterns of results are known as longitudinal models, mixed effect models, multilevel regression models, extended linear models, panel data models, growth curve models, etc. The repeated measurements of asynchronies can be viewed as multilevel, where the lowest level are the asynchronies nested within the sequence. At this lowest level, according to the LPC, the asynchronies are not independent, and the model parameters could then be estimated based on the within-sequence correlation structure. In the remainder of this manuscript, we adopted the terminology of Pinheiro and Bates (2000). Models that capture within-sequence correlations are referred to as extended Linear Models (see Note 8). Those models allow for the inclusion of all sequences within a single model rather than computing an average.

Furthermore, asynchrony sequences may vary owing to other factors not controlled here, which might have a unique contribution on each sequence. This could affect different parameters causing variability between the expected asynchronies of the sequences. Such between-sequence variability can be captured by random effects, which are associated with each individual sequence, sampled randomly from the population of sequences. When the model incorporates random effects, it is referred to as linear Mixed-Effects Models (see Note 9). Since this approach focused on stationary asynchronies, only the intercept of each sequence varied. Therefore, the model included a random intercept term. Finally, when incorporating both, a particular within-sequence correlation structure and a random intercept, the models referred to as Mixed-Effects Models (see Note 10 and Pinheiro & Bates, 2000).

The remainder of this manuscript is structured as followed: First, we present the extended Linear Model (eLM) and illustrate its superior estimation performance to the bGLS method, particularly when the sequences are short. Second, we show that the bGLS method and the eLM fail in estimating parameters from sequences with varying intercepts. Finally, we demonstrate that the Mixed-Effects Model (MEM) can robustly estimate the parameters of the LPC model by accounting for within-sequence correlations and between-sequence variability.

Note 7: *Stationarity*

Strict stationarity:

If X_k is stochastic process that models the observable discrete time series $x_{k_1}, x_{k_2}, \ldots, x_{k_N}$, at time points $k_1, \ldots, k_N$, a strictly stationary process is defined by the equality of the joint distribution of the individuals cumulative distribution functions:

$$F_X(x_{k_1}, \ldots, x_{k_N}) = F_X(x_{k_1+\delta}, \ldots, x_{k_N+\delta}). \quad (A1)$$

This means that the joint cumulative distribution function is invariant to arbitrary time-shifts (δ) and implies

$$E[x_k] = \mu,\ Var[x_k] = \sigma^2,\ Cov[x_{k_1}, x_{k_2}] = \gamma(\mid k_1 - k_2 \mid). \quad (A2)$$

Weak stationarity:

A weak stationary process relaxes the assumption $A1$ and assumes the covariance to depend only on the distance between k_1 to k_2, $\gamma(\mid k_2 - k_1 \mid)$ (Brockwell & Davis, 2016).

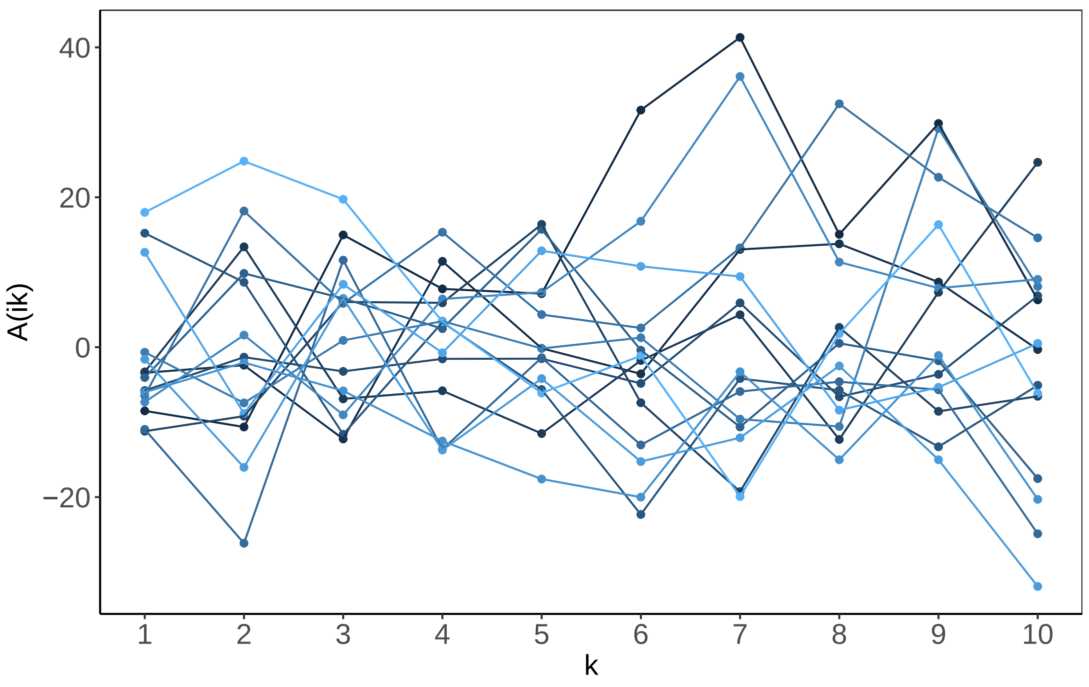


Figure 3.2: Illustration of multiple "short" asynchrony sequences A_{ik}, $i = 1, \ldots, 15$, $k = 1, \ldots, 10$. The asynchronies were produced by simulating the LPC model $nseq = 15$ times, with the parameters set to $\alpha = 0.5$, $\sigma_T = 10$, $\sigma_M = 5$. Each line segment represents one independent sequence. A_{1i} was randomly sampled from a uniform distribution $\mathcal{U}(-20, 20)$.

Note 8: *The Extended Linear Model (eLM)*

In the linear model $y = X\beta + \epsilon$, the error ϵ is identically distributed (iid). This means that each element in the vector ϵ follows the same pdf, here $NV(0, \sigma^2)$, and is independent.

In the extended Linear Model (eLM), this assumption is relaxed and ϵ_{ij} can be correlated with ϵ_{ik}, $j \neq k$, within group i. Then, the model is formalized as:

$$y_i = X_i\beta + \epsilon_i, \ i = 1, \ldots, m,$$

$$\epsilon_i = \begin{bmatrix} \epsilon_{i1} \\ \epsilon_{i2} \\ \epsilon_{i3} \\ \vdots \\ \epsilon_{ij} \\ \vdots \\ \epsilon_{in_i} \end{bmatrix} \sim MVN(0, \sigma^2\Lambda_i), \ j = 1, \ldots, n_i,$$

where

· y_i is a n_i x 1 response vector

· X_i is an n_i x p fixed-effects design matrix

· β is p x 1 fixed-effects vector

· ϵ_i is a n_i x 1 within-group error vector

· Λ_i is a n_i x n_i positive-definite matrix parametrized by a vector of parameters λ.

All elements of the error vector ϵ_i are independent of the error vector ϵ_j, $i \neq j$ but ϵ_{ih} can be correlated with ϵ_{ik}, $h \neq k$, within group i, specified by the non-diagonal elements of Λ_i. The errors ϵ_{ih} and ϵ_{ik} are related to the position vectors p_{ih} and p_{ik} only through the distance $abs(j - k)$ and their particular values are irrelevant. The dependence among the within-group errors is called correlation structure.

Note 9: *Linear Mixed-Effects Model (lMEM)*

The linear Mixed-Effects Model is an extention of the linear model. The model consists of fixed effects and random effects. Fixed effects are the part of the conventional linear model. Random effects are related to particular units (e.g., participant or trial). Both are called “effects” because they reflect the deviation from the overall expected value of the response variable. “Fixed” referres to the effects that are constant across the units. “Random” referres to the effects that are randomly sampled from a population of units, according to some model, here $NV(0, \sigma^2_{Random})$. Since the observations within the same unit share the same random effects, observations within a unit can be correlated.
The standard form of the linear Mixed-Effects Model is

$$y_i = X_i\beta + Z_i b_i + \epsilon_i,\ i = 1, \ldots, m,$$
$$b_i \sim MVN(0, \Psi),\ \epsilon_i \sim MVN(0, \sigma^2 I),$$

where

· m is the total number of units

· y_i is a n_i x 1 response vector

· X_i is an n_i x p fixed-effects design matrix

· β is p x 1 fixed-effects vector

· Z_i is a n_i x q random-effects design matrix

· b_i is a q x 1 random-effects vector

· ϵ_i is a n_i x 1 within-group error vector

· Ψ is a q x q symmetric and positive semi-definite matrix, with q being the total number of random effects

· I is a n_i x n_i identity matrix

· σ^2 is the error variance.

Note that the variability accounted for by random effects could also by modeld by additional fixed effects. Fixed effects should be used when one wants to make inferences about the exact levels of the classification factor. Random effects should be used when making interences about the factors' population rather than about the particular levels.

Note 10: *Mixed-Effects Model (MEM)*

The lMEM can be extended by relaxing the assumption of independently and identically distributed within-group errors with mean zero and constant variance. This more general Mixed-Effects Model is a combination of the eLM and the lMEM. It accounts for within-group variability and allows within-group errors to be correlated (and to be heteroscedastic) .

$$
\begin{aligned}
&y_i = X_i\beta + Z_i b_i + \epsilon_i,\ i = 1, \ldots, m,\\
&b_i \sim MVN(0, \Psi),\ \epsilon_i \sim MVN(0, \sigma^2\Lambda_i),
\end{aligned}
$$

where the variables are defined as in Note 8 and Note 9.

3.2 The extended Linear Model (eLM)

In this section, we present the eLM method. It is used to estimate the parameters of the LPC model. For this reason, as in the previous section, we simulated asynchrony sequences from the LPC model in Equation 2.3 (see Appendix F), estimated the LPC parameters by the eLM method from these simulations (see Appendix G), and repeated this estimation multiple times in order to validate the performance of the eLM method (see Appendix H).

3.2.1 Method

We developed the eLM method based on Pinheiro and Bates (2000). It is able to capture multiple sequences of asynchronies within a single model. Each asynchrony is denoted by $a_{ij},\ i = 1, \ldots, m,\ j = 1, \ldots, n_i$, where i indexes the sequence and j indexes the j^{th} asynchrony within sequence i. The length of a_i is denoted by n_i and N denotes here the length of all sequences together $N = \sum_{i=1}^{m} n_i$.

The model can be written for each sequence i as:

$$\begin{aligned} y_i &= x_i\beta + s_i, \\ s_i &\sim MVN(0, \Sigma_i), \end{aligned} \tag{3.5}$$

where y_i is a $(n_i - 1)$ x 1 column vector of asynchronies of sequence i, x_i is a $(n_i - 1)$ x 1 column vector of asynchronies of sequence i one iteration earlier than the asynchronies in vector y_i, s_i is a $(n_i - 1)$ x 1 column vector of the errors of sequence i, and Σ_i is a $(n_i - 1)$ x $(n_i - 1)$ variance-covariance matrix:

$$\Sigma_i = \begin{bmatrix} \sigma_T^2 + 2\sigma_M^2 & -\sigma_M^2 & 0 & \cdots & 0 & 0 \\ -\sigma_M^2 & \sigma_T^2 + 2\sigma_M^2 & -\sigma_M^2 & \ddots & \vdots & 0 \\ 0 & -\sigma_M^2 & \sigma_T^2 + 2\sigma_M^2 & \ddots & 0 & \vdots \\ \vdots & 0 & \ddots & \sigma_T^2 + 2\sigma_M^2 & -\sigma_M^2 & 0 \\ 0 & \vdots & \ddots & -\sigma_M^2 & \sigma_T^2 + 2\sigma_M^2 & -\sigma_M^2 \\ 0 & 0 & \cdots & 0 & -\sigma_M^2 & \sigma_T^2 + 2\sigma_M^2 \end{bmatrix}.$$

s_i corresponds to Z of the bGLS method: $s_i = [s_{i1}, s_{i2}, \ldots, s_{in_i-1}]^T$, $s_{ij} = T_{ij} + M_{ij+1} - M_{ij} - E[T_{ij}]$. We changed its notation to prevent confusion with the random effects column vector z_i, which is introduced in Section 3.3. See Note 9 for the formalization of the linear Mixed-Effects Model.

The model including all sequences is then

$$Y = MVN(X\beta, \Sigma), \tag{3.6}$$

where Y and X are column vectors with dimension $(N - m)$ x 1, and Σ is a variance-covariance matrix with dimension $(N - m)$ x $(N - m)$:

$$\begin{array}{ccccccc}
Y & & X & \beta & & S & \\
\begin{bmatrix} a_{12}-E[a_{1j}] \\ a_{13}-E[a_{1j}] \\ a_{14}-E[a_{1j}] \\ \vdots \\ a_{i2}-E[a_{ij}] \\ a_{i3}-E[a_{ij}] \\ a_{i4}-E[a_{ij}] \\ \vdots \\ a_{m2}-E[a_{mj}] \\ a_{m3}-E[a_{mj}] \\ a_{m4}-E[a_{mj}] \end{bmatrix} & = & \begin{bmatrix} a_{11}-E[a_{1j}] \\ a_{12}-E[a_{1j}] \\ a_{13}-E[a_{1j}] \\ \vdots \\ a_{i1}-E[a_{ij}] \\ a_{i2}-E[a_{ij}] \\ a_{i3}-E[a_{ij}] \\ \vdots \\ a_{m1}-E[a_{mj}] \\ a_{m2}-E[a_{mj}] \\ a_{m3}-E[a_{mj}] \end{bmatrix} & (1-\alpha) & + & \begin{bmatrix} s_{11} \\ s_{12} \\ s_{13} \\ \vdots \\ s_{i1} \\ s_{i2} \\ s_{i3} \\ \vdots \\ s_{m1} \\ s_{m2} \\ s_{m3} \end{bmatrix} & ,
\end{array}$$

where $n_i = 4,\ \forall i,\ i = 1, \dots, m$, for illustration purpose only.

3.2.2 Computation

The LPC model parameters α, σ_T, and σ_M can be obtained from β and Σ. Based on the approach of Pinheiro and Bates (2000), a single β and Σ can be estimated by a model including all sequences.

For computational reasons, σ^2 was factored out of Σ_i:

$$\frac{\Sigma_i}{\sigma^2} = \Lambda_i. \tag{3.7}$$

Λ_i is parametrized by λ:

$$\Lambda_i = \begin{bmatrix} 1 & \lambda & 0 & \cdots & 0 & 0 \\ \lambda & 1 & \lambda & \ddots & \vdots & 0 \\ 0 & \lambda & 1 & \ddots & 0 & \vdots \\ \vdots & 0 & \ddots & 1 & \lambda & 0 \\ 0 & \vdots & \ddots & \lambda & 1 & \lambda \\ 0 & 0 & \cdots & 0 & \lambda & 1 \end{bmatrix}.$$

Because it is a positive-definite matrix, it has an invertible square root $\Lambda_i^{\frac{1}{2}}$ so that $\Lambda_i = (\Lambda_i^{\frac{1}{2}})^T \Lambda_i^{\frac{1}{2}}$. Then, $\Lambda_i^{-1} = \Lambda_i^{-\frac{1}{2}} (\Lambda_i^{-\frac{1}{2}})^T$, where $\Lambda_i^{-\frac{1}{2}}$ is the inverse of $\Lambda_i^{\frac{1}{2}}$. The transformation to a linear model is then achieved by:

$$y_i^* = (\Lambda_i^{-\frac{1}{2}})^T y_i, \; s_i^* = (\Lambda_i^{-\frac{1}{2}})^T s_i, \; x_i^* = (\Lambda_i^{-\frac{1}{2}})^T x_i, \tag{3.8}$$

which provides the linear model:

$$y_i^* = x_i^* \beta + s_i^*, \tag{3.9}$$

where $s_i^* \sim MVN((\Lambda_i^{-\frac{1}{2}})^T 0, \; \sigma^2 (\Lambda_i^{-\frac{1}{2}})^T \Lambda_i \Lambda_i^{-\frac{1}{2}}) = MVN(0, \; \sigma^2 I_i)$.

For a fixed λ, the conditional MLEs are:

$$\begin{aligned} \hat{\beta}(\lambda) &= ((X^*)^T X^*)^{-1} (X^*)^T Y^*, \\ \hat{\sigma^2}(\lambda) &= \frac{(Y^* - X^* \hat{\beta})^T (Y^* - X^* \hat{\beta})}{(N-m)}, \end{aligned} \tag{3.10}$$

where $X = [X_1, \ldots, X_m]^T, \; Y = [Y_1, \ldots, Y_m]^T, \; \beta = (1-\alpha)$, and $N = \sum_i^m n_i$.

In the so called "profiled log-likelihood", β can then be replaced by its conditional MLE so that β is expressed as a function of λ, $\beta(\lambda)$. Therefore, the profiled log-likelihood is solely a function of λ:

$$l(\lambda|y)^{profiled} = const - (N-m) log \sqrt{(Y^* - X^* \hat{\beta})^T (Y^* - X^* \hat{\beta})} - \frac{1}{2} \sum_{i=1}^{m} log |\Lambda_i|. \tag{3.11}$$

By optimizing Equation 3.11 and using $\hat{\lambda}$ in Equation 3.10, the MLEs for $\hat{\beta}$ and $\hat{\sigma}^2$ can be computed. Subsequently, by using $\hat{\lambda}$, $\hat{\beta}$, and $\hat{\sigma}^2$, the final LPC model parameters are obtained by:

$$\begin{aligned} \hat{\alpha} &= 1 - \hat{\beta}, \\ \hat{\sigma}_M &= \sqrt{-\hat{\sigma}^2 \hat{\lambda}}, \\ \hat{\sigma}_T &= \sqrt{\hat{\sigma}^2 - 2\hat{\sigma}_M^2}. \end{aligned} \tag{3.12}$$

As an alternative to implementing the above-described algorithm, one could also utilize the "gls" function of the nlme R-package and define the within-sequence correlation structure as a Moving Average Model of order 1, referred to as MA(1). $\hat{\alpha}$ could then be obtained by computing $1 - \hat{\beta}$ and $\hat{\sigma}_M$ and $\hat{\sigma}_T$ could be obtained by reparameterizing the variance-covariance matrix according to Equation 3.13. This is possible because, similar to the LPC model, the MA(1) assumes non-zero autocorrelations only at lag $h = 1$.

$$LPC : Z_{ij} = T_{ij} + M_{i(j+1)} - M_{ij}.$$

$$\gamma_Z(h) = \begin{cases} 2\sigma_M^2 + \sigma_T^2, & \text{if } \; h = 0 \\ -\sigma_M^2, & \text{if } |h| = 1 \\ 0, & \text{if } |h| \geq 2, \end{cases}$$

$$MA(1) : X_t = \epsilon_t - \theta\epsilon_{t-1}. \tag{3.13}$$

$$\gamma_X(h) = \begin{cases} \sigma_\epsilon^2(1 - \theta^2), & \text{if } h = 0 \\ -\theta\sigma_\epsilon^2, & \text{if } |h| = 1 \\ 0, & \text{if } |h| \geq 2. \end{cases}$$

$$\Rightarrow \sigma_M^2 = \sigma_\epsilon^2\theta, \; \sigma_T^2 = \sigma^2(1 - \theta^2) - 2\sigma_M^2$$

3.2.3 Results & Discussion

For method validation, we estimated 50 times the LPC parameters with the above-presented eLM method and the bGLS method from $50 \; x \; 15 \; = \; 450$ simulated asynchrony sequences. Results revealed that the eLM method is less biased for different sequences lengths compared to the bGLS method (see Figure 3.3). While the bias of the bGLS method increased with decreasing sequence length and the size of the 95% confidence intervals remained very similar, the eLM method seems unbiased at any length but increased the confidence intervals.

This was expectable, considering that the bGLS method averaged estimates from single sequences. When estimating the parameters by approximating a MLE from a short sequence, estimations can fail easily, which results in estimates that may consistently deviate from the theoretical mean. In addition, mean estimates are very susceptible to outliers. Thus, when there are few asynchronies, bGLS estimates can be biased.

In contrast, in the eLM method, a single parameter $\hat{\lambda}$ is estimated by maximizing a profiled MLE (see Equation 3.11) involving all indexed sequences. Afterwards, two single parameters $\hat{\beta}(\lambda)$ and $\hat{\sigma}(\lambda)$ can be estimated by the conditional MLEs (see Equation 3.10) and a simple transformation reveals then the final parameter estimates of the LPC model $\hat{\alpha}$, $\hat{\sigma}_M$, and $\hat{\sigma}_T$ (see Equation 3.12). This method employs each single sequence for parameter estimation while other sequences provide additional information about "what is going on" in the particular sequence. This makes the eLM method more resistant to estimation biases (see Figure 3.4). However, short sequences should still lead to a less efficient estimation. This is here reflected by an increase of the standard deviation (σ) (see Figure 3.5).

Notwithstanding, the SD in the eLM method is particularly high at α around 1. Similar results were reported for the unbounded GLS method (see Jacoby, Keller, et al., 2015). In future studies, additional bounding conditions could be included and evaluated for the eLM method.

A common practice is to evaluate the Goodness of models by considering the Mean Squared Error (MSE) (see Note 3). We, however, considered the bias and SD separately because they are more transparent and intuitive, and for the present study purpose, the MSE might weight the SD too strong. For example, we find it more relevant to consider that the confidence intervals of the eLM estimates did not exceed the biased estimates of the bGLS method. This means that even largely deviating estimates were still as good as the biased estimates of the bGLS method for retrieving the correct parameters.

Concluding, the eLM method seems to exploit a trade-off between accuracy (mean deviation from the parameter) and efficiency (variability of estimations) in favor of the former. When uncertainty increases due to less information within the simulated observations, its efficiency decreased in order to hold a high accuracy. 95% estimates of the eLM method seem at least as accurate or more accurate as the estimates from the bGLS method. For this reason, we suggest that the eLM method should be preferred when multiple and stationary sequences are available. Although the choice of one over the other method depends on the particular study goals and should be carefully considered by the user, the advantage of our modeling approach becomes very obvious in section 3.3.

Limitations

A shortcoming of the eLM method and the bGLS method is that they presume the same parameter settings among all sequences. Apparently, proposing a general model, like the LPC, only has value if the process in question is stable in its parameter settings, as long as the environment is constant.

However, when the asynchronies were obtained from experiments, there should be variability between the measurements that are not related to the LPC model. For instances, in an repeated-measurement design, identical experimental conditions among trials are impossible to achieve. When there are noise factors that are independent among all asynchronies, the variability is captured by the error term of the model. Yet, there may be factors that have a unique contribution on each trial and are hard to control. This leads to variability between trials that is neither accounted for by the LPC model nor by the introduced parameter estimation methods.

Concretely, we have shown that the eLM is appropriate if the mean asynchrony is expected to be constant among trials. Jacoby, Tishby, et al. (2015) normalized the asynchronies by the mean asynchrony obtained from so many asynchrony records as possible. However, if each sequence is exposed to variability factors whose magnitude is unique to each sequence, such a normalization is inappropriate. Another possibility is to normalize each sequence by the mean asynchrony of the respective sequence. This can be done when using the eLM method. Yet, for short sequences, such mean asynchrony might not be very representative.

Solutions

Figure 3.6 shows asynchrony sequences whose expected asynchronies vary between the sequences by different magnitudes. One possibility to account for this variability could

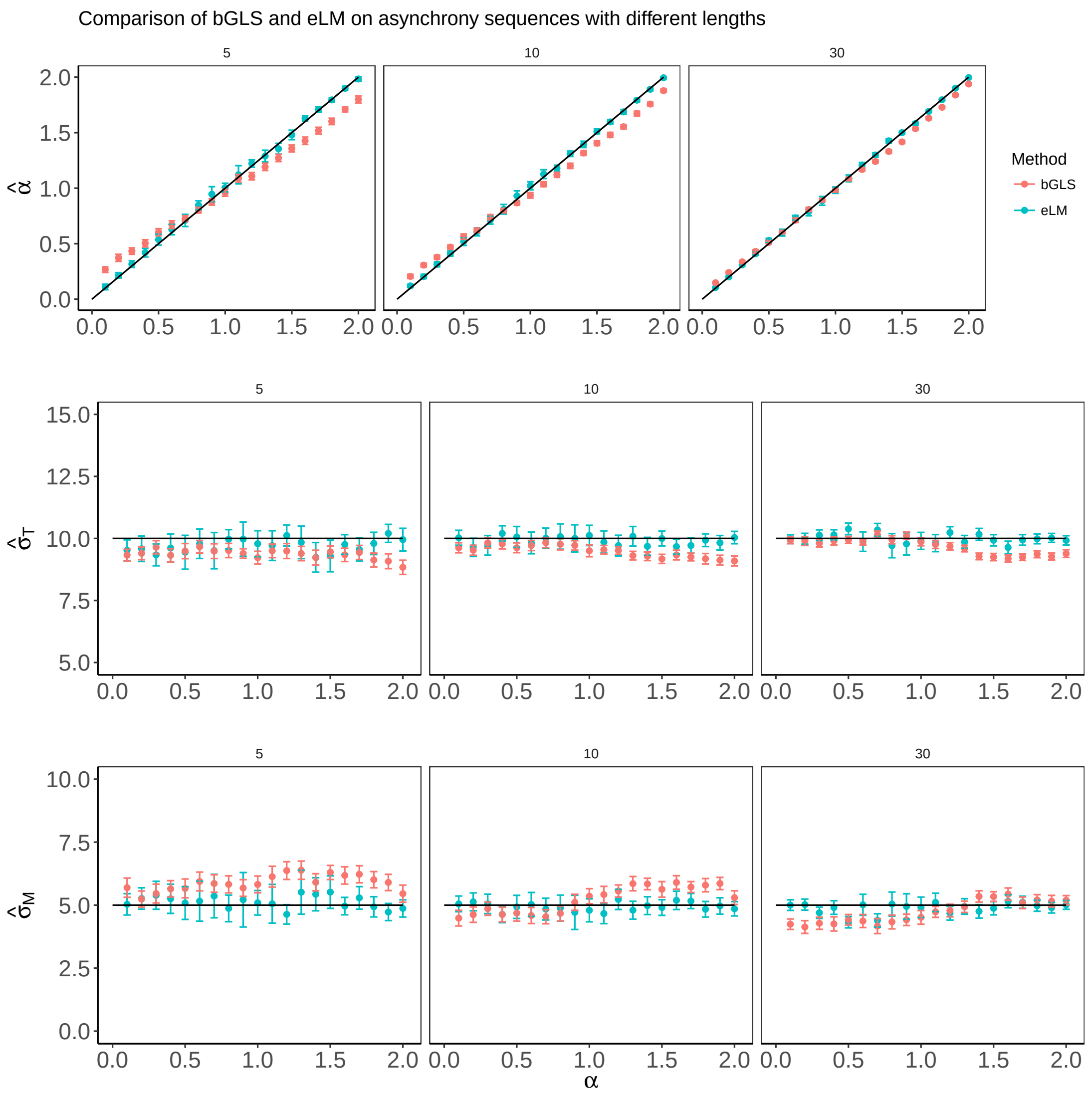


Figure 3.3: Comparison of the LPC model parameter estimation methods as a function of true α for different sequence lengths. The plots show the mean and the 95% confidence intervals of 50 estimates for each α. The estimates of the bGLS method are displayed in red and the estimates of the eLM method are displayed in magenta. Dots are the mean estimates. When lying on the black line, the estimates coincide with the true values. The sequences were obtained by simulating the LPC model with $\sigma_M = 5$, $\sigma_T = 10$ and α according to the ordinate. Each estimate was obtained from $m = 15$ asynchronies sequences with varying length ($n_i = 30,\ n_i = 10,\ n_i = 5,\ \forall i,\ i = 1, \ldots, 15$).

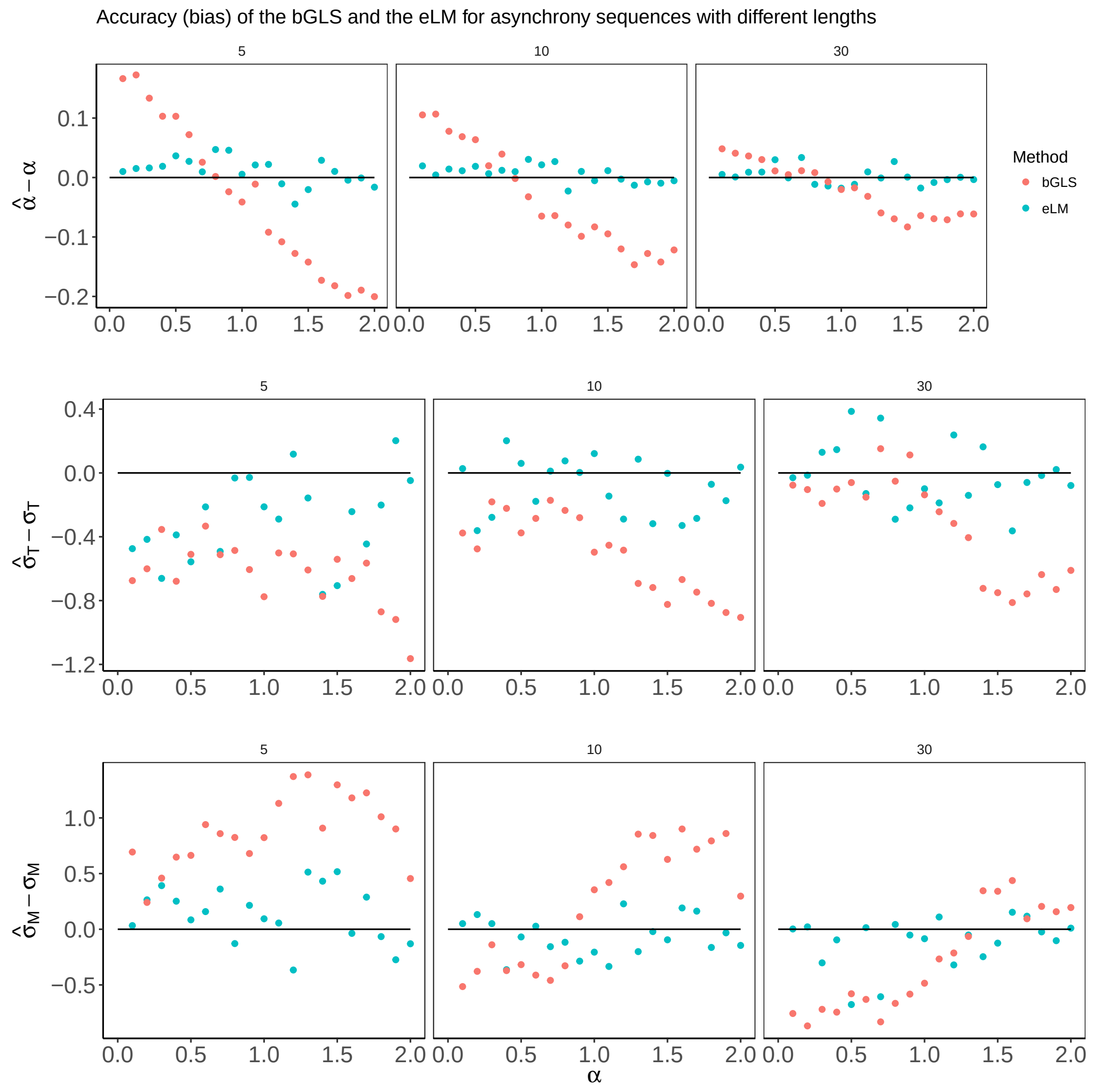


Figure 3.4: LPC model parameter estimation bias as a function of true α for different sequence lengths. Estimation biases of the bGLS method are displayed in red and the estimation biases of the eLM method are displayed in magenta. When lying on the black line, the bias is minimal.

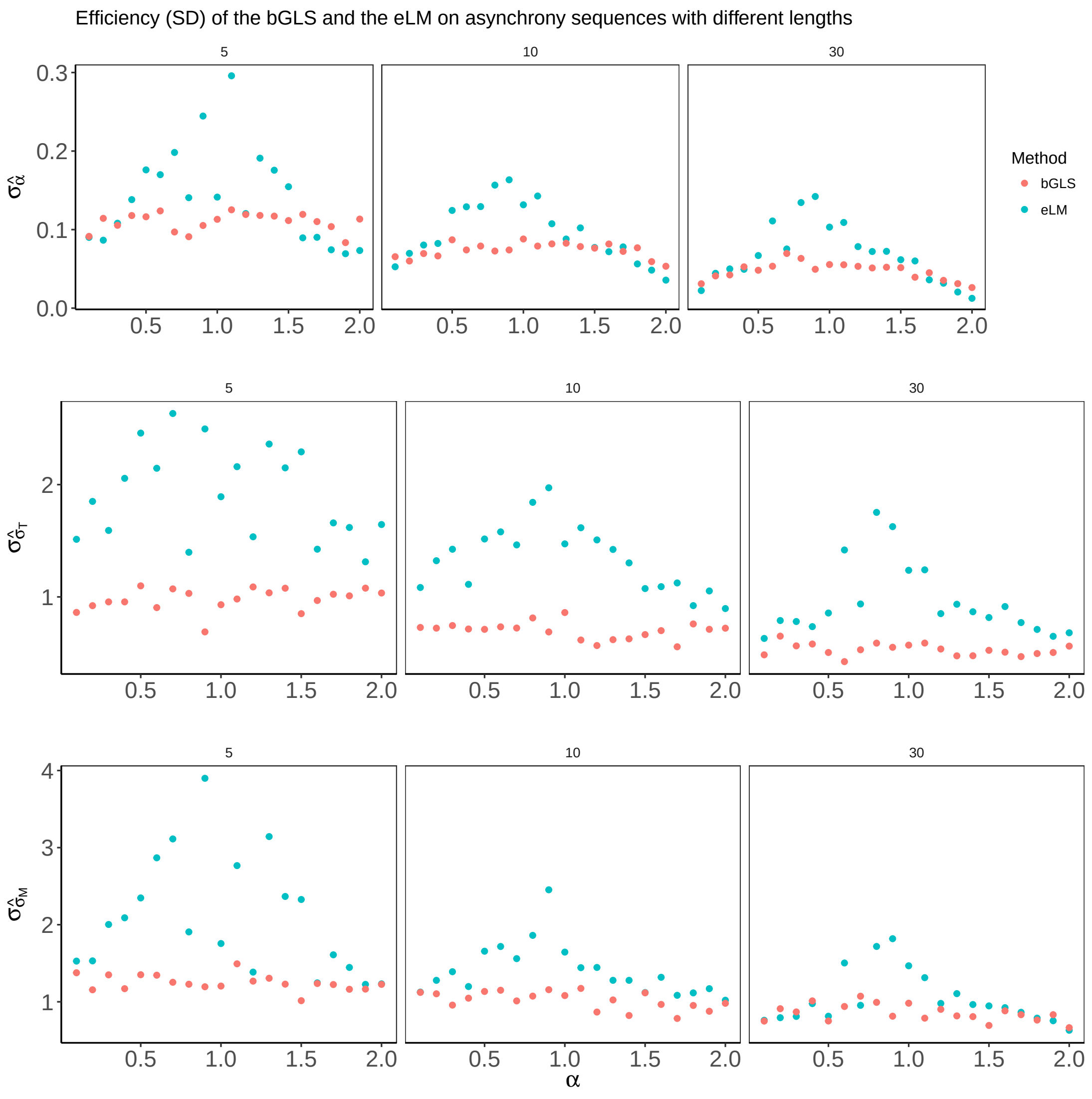


Figure 3.5: The standard deviations (σ) of the LPC model parameter estimations as a function of true α for different sequence lengths. The σs of the bGLS method are displayed in red and the σs of the eLM method are displayed in magenta.

be to incorporate another fixed effect in the eLM, with as many parameters as there are sample means. Considering that this value was sampled from a continuous distribution, the number of parameters would equal the number of sequences $i = 1, \ldots, m$. Yet, we were not interested in making inferences about the specific effects of these "noise factors" but must control them for an unbiased estimation of the LPC model parameters. For this reason, a solution could be to account for varying μ_{a_i} among sequences i by incorporating random-effects (random intercepts) in the eLM model, making it a single-level Mixed-Effects model (MEM) (see also Note 10). This requires the estimation of far fewer parameters than when using fixed effects and it seems theoretically more plausible. In the next section, we introduce the MEM method and subsequently compare its performance with the performance of the bGLS and the eLM methods.

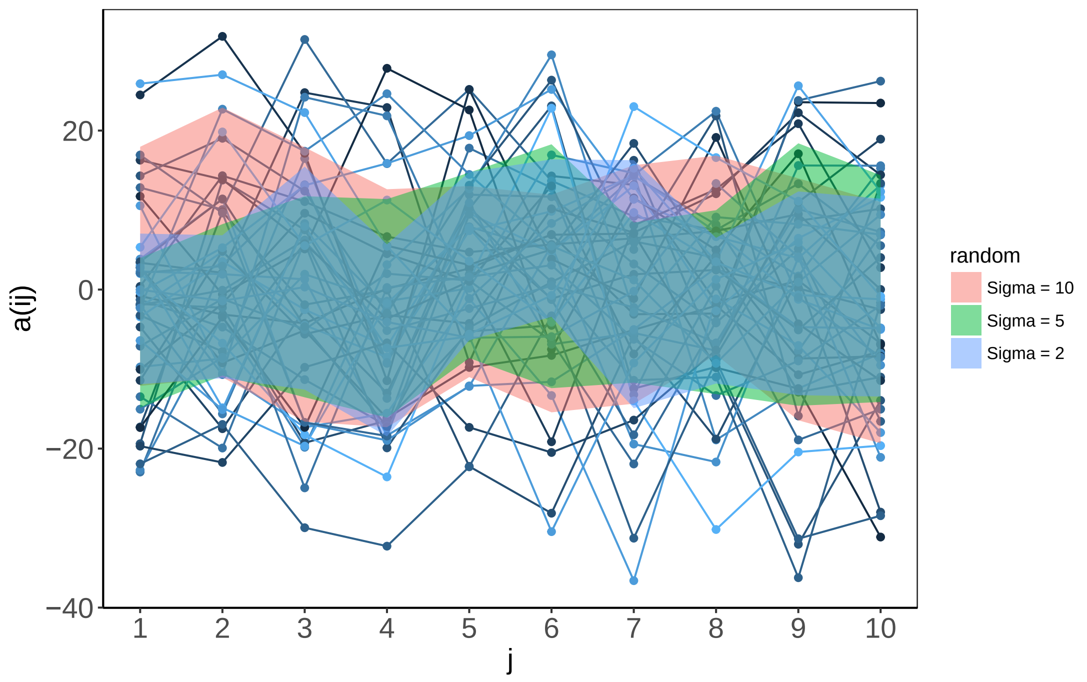


Figure 3.6: Illustration of multiple ($m = 45$) short ($n_i = 10$) asynchrony sequences (a_{ij}) with between-sequence variability of $E[a_{ij}]$. The asynchronies were produced by simulating the LPC model with $\alpha = 0.5$, $\sigma_T = 10$, $\sigma_M = 5$. Each line segment represents one independent sequence. The a_{i1} was randomly sampled from a uniform distribution $\mathcal{U}(-20, 20)$. A random intercept was added to each sequence that shifted the expected value $E[a_{ij}]$ away from zero. The intercept was sampled from $NV(0, \sigma_b^2)$, $\sigma_b = 2,\ 5,\ 10$. 15 sequences were exposed to one of the three additional σ_b conditions. The colored areas are the empirical standard deviations. The area increases slightly with σ_b.

3.3 The Mixed-Effects Model (MEM)

In this section, we present the MEM method. It is used to estimate the parameters of the LPC model when μ_i varies among sequences. For this reason, we simulated asynchrony sequences from the LPC model in Equation 2.3 and added after each simulation a constant b_i to each a_{ij} sampled from $NV(0, \sigma^2)$, $\sigma = 2, 15, 10$ (see Appendix I for the R code). From these simulations, we estimated the LPC parameters by the MEM method (Appendix J), the bGLS method (Appendix C), and the eLM method (Appendix G). These estimations were repeated multiple times in order to validate and compare the performance of the methods (see Appendix K, D, & K).

3.3.1 Method

The MEM is denoted as

$$\begin{aligned} &y_i = x_i\beta + z_i b_i + s_i, \\ &b_i \sim NV(0, \sigma_b^2),\ s_i \sim MVN(0, \sigma^2\Lambda_i),\ i = 1, \ldots, m, \end{aligned} \tag{3.14}$$

where y_i, x_i, and s_i are defined as in Equation 3.5: y_i is a $(n_i - 1)$ x 1 column vector of asynchronies of sequence i, x_i is a $(n_i - 1)$ x 1 column vector of asynchronies of sequence i one iteration earlier than the asynchronies in vector y_i, s_i is a $(n_i - 1)$ x 1 column vector of the errors of sequence i, and Λ_i is a $(n_i - 1)$ x $(n_i - 1)$ covariance matrix.

The b_i is a m x 1 column vector of random effects for sequence i and z_i is a $(n_i - 1)$ x m design matrix, indexing b_i. The b_i is normally distributed with zero mean and (the scalar) standard deviation σ_b. It represents the variability between the expected asynchrony values $E[a_{ij}]$ among the sequences. The b_i and s_i are independent within and between sequences.

$$
\begin{array}{cccccccccccc}
Y & & X & \beta & & Z & & b & & S & \\
\begin{bmatrix} a_{12} \\ a_{13} \\ a_{14} \\ \vdots \\ a_{i2} \\ a_{i3} \\ a_{i4} \\ \vdots \\ a_{m2} \\ a_{m3} \\ a_{m4} \end{bmatrix}
& = &
\begin{bmatrix} a_{11} \\ a_{12} \\ a_{13} \\ \vdots \\ a_{i1} \\ a_{i2} \\ a_{i3} \\ \vdots \\ a_{m1} \\ a_{m2} \\ a_{m3} \end{bmatrix}
& (1-\alpha) & + &
\begin{bmatrix} 1 & \dots & 0 & \dots & 0 \\ 1 & \dots & 0 & \dots & 0 \\ 1 & \dots & 0 & \dots & 0 \\ \vdots & & \vdots & & \vdots \\ 0 & \dots & 1 & \dots & 0 \\ 0 & \dots & 1 & \dots & 0 \\ 0 & \dots & 1 & \dots & 0 \\ \vdots & & \vdots & & \vdots \\ 0 & \dots & 0 & \dots & 1 \\ 0 & \dots & 0 & \dots & 1 \\ 0 & \dots & 0 & \dots & 1 \end{bmatrix}
& &
\begin{bmatrix} b_1 \\ \vdots \\ b_i \\ \vdots \\ b_m \end{bmatrix}
& + &
\begin{bmatrix} s_{11} \\ s_{12} \\ s_{13} \\ \vdots \\ s_{i1} \\ s_{i2} \\ s_{i3} \\ \vdots \\ s_{m1} \\ s_{m2} \\ s_{m3} \end{bmatrix}
& ,
\end{array}
$$

where $n_i = 4,\ \forall i,\ i = 1, \dots, m$, for illustration purpose only.

3.3.2 Computation

In order to obtain the parameters β, σ_b, and σ^2, the following likelihood function can be maximized:

$$
L(\beta, \delta, \sigma^2 \mid y) = \frac{1}{(2\pi\sigma^2)^{(N-m)/2}} exp\left(\frac{-\sum_{i=1}^{m} \| \tilde{y}_i - \tilde{x}_i \beta - \tilde{z}_i \hat{b}_i \|^2}{2\sigma^2} \right) \prod_{i=1}^{m} \frac{\mid \delta \mid}{\sqrt{\mid \tilde{z}_i^T \tilde{z}_i \mid}}, \tag{3.15}
$$

where δ parametrizes the variance-covariance matrix of the random effects b_i (which is here a scalar σ_b), σ is the residual standard error of s_i, $\tilde{y}_i$, $\tilde{x}_i$, and $\tilde{z}_i$ are the augmented data vectors

$$
\tilde{y}_i = \begin{bmatrix} y_i \\ 0 \end{bmatrix}, \ \tilde{x}_i = \begin{bmatrix} x_i \\ 0 \end{bmatrix}, \ \tilde{z}_i = \begin{bmatrix} z_i \\ \delta \end{bmatrix}, \ \delta = \sqrt{\frac{\sigma^2}{\sigma_b^2}}, \tag{3.16}
$$

and $\hat{b}_i$ is estimated by:

$$
\hat{b}_i = (\tilde{z}_i^T \tilde{z}_i)^{-1} \tilde{z}_i^T (\tilde{y}_i - \tilde{x}_i \beta), \ i = 1 \dots, m. \tag{3.17}
$$

Since the OLS for $\hat{b}_i$ depends on β, and the OLS for $\hat{\beta}$ depends on b_i, they must be estimated jointly (iteratively).

However, because the within-sequence errors are correlated, it was performed a linear transformation of the variables, as previously (see Equation 3.8):

$$y_i^* = (\Lambda_i^{-\frac{1}{2}})^T y_i,\ s_i^* = (\Lambda_i^{-\frac{1}{2}})^T s_i,\ x_i^* = (\Lambda_i^{-\frac{1}{2}})^T x_i,\ z_i^* = (\Lambda_i^{-\frac{1}{2}})^T z_i, \tag{3.18}$$

which provided the linear Mixed-Effects Model

$$\begin{aligned} &y_i^* = x_i^* \beta + z_i^* b_i + s_i^*, \\ &b_i \sim NV(0, \sigma_b^2),\ s_i^* \sim MVN(0, \sigma^2 I),\ i = 1, \ldots, m. \end{aligned} \tag{3.19}$$

Its profiled likelihood function can be expressed as

$$L(\beta, \delta, \sigma^2, \lambda \mid y)^{profiled} = L(\beta, \delta, \sigma^2, \lambda \mid y^*) \prod_{i=1}^{m} |\Lambda_i^{-1/2}|, \tag{3.20}$$

where λ parametrizes Λ_i, as in Equation 3.11. By optimizing Equation 3.20, its best fitting parameters can be obtained from which α, σ_T, and σ_M were computed. For a detailed description of the proof and most efficient computation of $L(\beta, \delta, \sigma^2, \lambda \mid y)^{profiled}$, see Pinheiro and Bates (2000).

The above-described computational method was implemented in the “lme” function of the “nlme” R-package. The within-sequence correlation structure was defined as a Moving Average Model of order 1. The $\hat{\alpha}$ could be obtained by computing $1 - \hat{\beta}$ and $\hat{\sigma}_M$ and $\hat{\sigma}_T$ could be obtained by reparameterizing the variance-covariance matrix according to Equation 3.13. See Appendix J for the R code.

3.3.3 Results & Discussion

The results revealed that the bGLS method and the eLM method deteriorated with increasing between-sequence variability. The eLM method normalized the sequences by the sample mean of each sequence and the bGLS method computed the sample mean by all asynchronies from all sequences The biases increased with between-sequence variability and was smaller for lower α. The patterns of these biases is quite complex and we do yet not know how to interpret them.

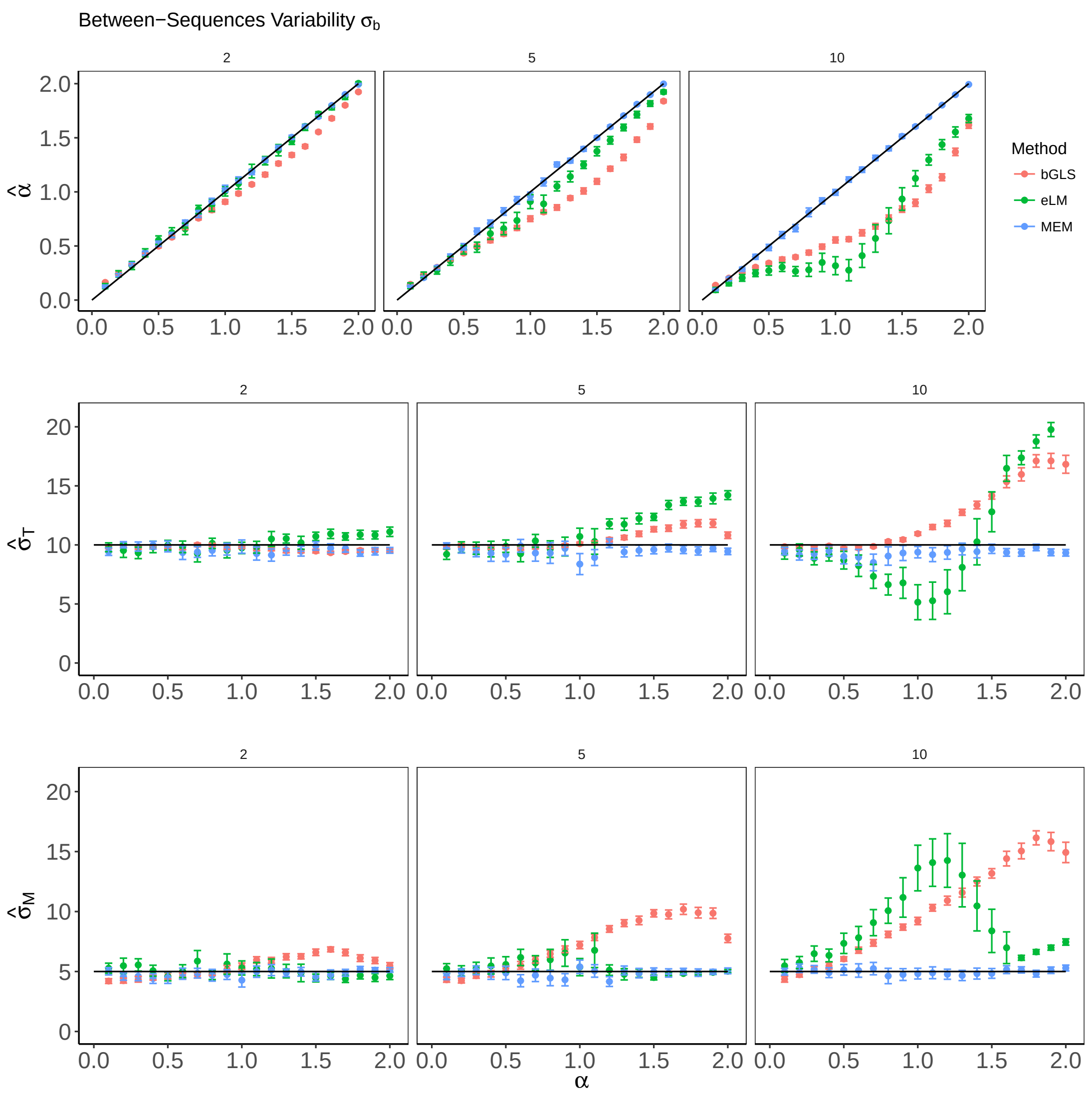


Figure 3.7: Comparison of the bGLS, eLM, and MEM when there is additional between-sequence variability of $E[a_{ij}]$ that is not incorporated in the LPC model. This between-sequence variability was sampled from $NV(0, \sigma_b^2)$, $\sigma_b = 2, 5, 10$, as indicated at the top of the plots. The parameters were estimated for different α from sequences of length $n_i = 30$. The sequences were obtained by simulating the LPC model with $\sigma_T = 10$ and $\sigma_M = 5$.

In contrast, the MEM method, which modeled the between-sequence variability by random effects, led to unbiased estimates for different magnitudes of variability (see Figure 3.7). We did not compare the efficiency of the models, as previously, because Figure 3.7 clearly illustrates its superior performance when there is between-sequence variability.

We further benchmarked the functions showing that the most complex MEM method is slightly faster than the bGLS method (MEM=15ns, bGLS=20ns) (see Appendix M). We, therefore, suggest that the MEM method is an appropriate alternative that can be used for single sequences when they are sufficiently long and stationary, and for short and multiple sequences when they are stationary.

Chapter 4

General Discussion

The main goal of this work was to introduce unbiased methods of parameter estimation of the LPC model. Therefore, we simulated asynchrony sequences from the LPC model, replicated the currently "state of art" estimation method (bGLS), and compared this with two here-developed methods in conditions that often occur in experimental setups.

4.1 Contributions of the extended Linear Model (eLM)

We demonstrated that the bGLS is prone to bias when the asynchrony sequences become shorter. We suggest that this owes to the inefficient technique of averaging parameter estimates, particularly when there is little information on each sequence. For this reason, we presented eLM, which integrates multiple sequences into a single model. Our results revealed that eLM estimates the model parameters of longer and shorter sequences with less bias than the bGLS.

Besides the simultaneous consideration of multiple sequences, another advantage of such an approach is that it can deal with balanced and unbalanced data. This implies that sequences of different lengths and/or missing values can be included. In contrast to the bGLS method, the eLM weights stronger the sequences with greater lengths, what is appropriate because longer sequences include more information and lead to less biased estimates. As far as we know, the authors of the bGLS method did not address this issue. We assume that shorter sequences or sequences with missing values were disregarded.

4.2 Contributions of the Mixed-Effects Model (MEM)

Subsequently, we applied a MEM. This model considers the between-sequence variability of the expected asynchrony within each sequence. By relating random effects to the asynchronies sharing the same sequence, the MEM could flexibly account for this variability. It provided unbiased estimates where the bGLS and the eLM methods largely deteriorated.

For the simulation, we produced between-sequence variability by adding a value (sampled from $NV(0, \sigma_b^2)$, with different σ_b^2) that was constant within a sequence but variable between sequences. How could one justify the validity of this manipulation?

When the different sequences result from repeated measurements on the same individual, such between-sequence variability might reside from an interplay of physiological factors—the properties of the individual sensory system—and psychological factors—for instances attentional focus and distraction. In order to achieve synchronization, the asynchronies have to be perceived as such. The perception of asynchrony depends on a complex interaction of a multitude of sensory cues from various modalities (e.g., tactile, auditory, & visual). Different sensory systems vary in propagation, transduction, transmission, and processing times, leading to different magnitudes of the physical (actual) asynchronies when an environmental event is represented by multiple modalities (see e.g., Noy, 2017).

The attentional focus of the individual on a particular sensory cue determines the size of the asynchrony that is required to be perceived as synchronous. Thus, attention might affect the information processing delays represented by the parameter μ_M in the LPC model. Consequently, the individual might attempt to achieve and stabilize different asynchronies, resulting in different mean asynchronies among sequences.

Yet, it is not clear why the attentional focus should vary between sequences and not within a sequence. In a typical synchronization experiment, event sequences are presented visually on a computer screen or aurally through headphones and suddenly appear and disappear. Before a sequence is presented, the participant's attention is purposefully caught by the presentation of, for instance, a visual fixation cross or a beep sound. Individual sequences are usually separated by short time periods. During stimulus presentation within a trial (e.g., a sequence of 10 to 30 metronome clicks), an individual should be able to stay focused and remains within a similar cognitive state. However, completing the task may require 15min to 120min; time enough for the individual's mind to wander and to focus different stimulus attributes. Although these issues should be approached in further studies, we believe the presence of attentional shifts during such paradigms can produce between-sequence variability by affecting information processing delays.

When sequences are the performance of different individuals, then the between-sequence variability could owe to factors specific to each individual. This is, for example, the individual's focus of attention. But also, the parameter settings of the underlying LPC model (e.g., perceptional delays and motor delays) should be different among individuals. Such individual differences produce very large between-sequence variations.

While the former parameter estimation methods cannot be used when each sequence is the performance of a different participant, the MEM approach can be implemented. A possible application would be to assess the effects of particular experimental conditions—for example, some stimulus properties—on general timing parameters. Then, one is not interested in making inferences about the differences of the LPC parameters among individuals, but still, has to control them in order to achieve unbiased estimates. This can be done by incorporating random effects on the individual level, as illustrated by the present work.

Finally, another variability factor that could be controlled by the MEM is methodological. Variability between sequences could result from the stimulus-presenting or the performance-capturing systems.

Concluding, we presented several examples that emphasize that it is highly relevant to include random intercept parameters into methods that estimate the parameters of the LPC model from experimental data. Here, we simulated sequences with between-sequence variability and assessed a model with a single random intercept but such a model could also incorporate multiple and nested random-effects.

4.3 Limitations

It must be mentioned that the eLM and the MEM approach estimated the parameters by maximizing likelihood functions (ML). Alternatively, one could have used restricted maximum likelihood functions (REML), which are generally more robust since they consider the number of degrees of freedom (Pinheiro & Bates, 2000). In order to test this, we compared the ML with the REML on the same observations and did not observe any significant differences (see Appendix N). It is known that both functions estimate the same fixed effects and that the ML estimate is unbiased for large overall sample sizes (Pinheiro & Bates, 2000). This is usually the case in experimental setups. Nevertheless, future studies should approach this question by estimating the parameters with both functions while systematically manipulating the size of the sample.

A limitation might be that the presented methods require stationary asynchronies. This is difficult to assure for short synchronization periods taking into account that synchronization might be a highly transient process (see e.g., van Ulzen et al., 2008). Nevertheless, stationary asynchronies are an important requirement of the LPC model and we suggest that non-stationarity should be prevented by cautiously designing the experiments and preparing the data set for analysis, rather than being modeled explicitly.

Another limitation of this work might be that between-sequence variability values were chosen without being externally validated. For the LPC model parameter settings, we could use settings similar to previous studies (Jacoby, Tishby, et al., 2015; Schulze & Vorberg, 2002). For the between-sequence variability, however, we have chosen values based on several tests and theoretical plausibility. Future studies should address this question and actually quantify the between-sequence variability that occurs in SMS tasks.

Related to the previous limitations, the here developed methods were validated on simulated asynchrony sequences. The next step should be to validate the methods on observations obtained from experiments.

Finally, our work was strictly concerned with the LPC model. The principal assumption of the LPC model is that corrections are performed on the perceived deviations from the participants' taps from the corresponding stimulus event onset. Surely, this is a quite simplistic model of reality since it presumes that even highly small asynchronies are registered by the individual. There are plenty of studies showing that asynchronies falling into a temporal integration window are actually not perceived as such and, consequently, might not be corrected (see Vroomen & Keetels, 2010). It would be interesting to evaluate this model regarding the inquiry of asynchrony thresholds for awareness, phase and period correction, etc. (see Repp, 2005).

Nevertheless, this work does not address the plausibility of the LPC model but instead proposes a more flexible approach to parameter estimations, likely to increase the applicability of the model to more complex settings. Moreover, motivated by parsimony as a fundamental principle for developing models, the LPC still finds great use in basic and applied research (see Jacoby, Tishby, et al., 2015).

4.4 Further Contributions

One advantage of the approaches here developed is the existence of validated software for fitting the eLM and the MEM, namely the "nlme" and the "lmer" R-packages. Their use requires a reparametrization of the variance-covariance matrix, but, besides being more robust, they are also quicker than the bGLS method.

Moreover, in order to examine the different parameter estimation methods, we translated the Matlab code provided by Jacoby, Tishby, et al. (2015) into R code and adjusted it for the particular question. We also implemented computational methods presented in Pinheiro and Bates (2000), in order to flexibly modify the Mixed-Effects model structure for the purpose of our study. This complemented the above-mentioned R-packages and will be provided in CRAN (2017). In addition, all programs (R codes) developed for this study will be available on GitHub (2017).

4.5 Conclusions

In sum, we provided a general framework of Mixed-Effects Models to estimate the parameters of the LPC model. We do not claim for the overall validity of the LPC. A more profound exploration of the LPC applicability to a large scope of natural settings is outside the scope of this work. Nevertheless, we demonstrated that Mixed-Effects Models are highly useful for achieving unbiased and efficient parameter estimations of the LPC from synchronization performances in SMS tasks. It remains to explore the extention of these methods, to more complex and realistic models, incorporating period correction, phase transition, and non-stationary asynchronies.

References

Brockwell, P. J., & Davis, R. A. (2016). *Introduction to time series and forecasting.* springer.

CRAN. (2017). *The comprehensive r archive network.* https://cran.r-project.org. (Accessed: 2017-09-30)

GitHub. (2017). *Github.* https://github.com.

Grush, R. (2004). The emulation theory of representation: Motor control, imagery, and perception. *Behavioral and brain sciences*, *27*(03), 377–396.

Jacoby, N., Keller, P. E., Repp, B. H., Ahissar, M., & Tishby, N. (2015). Lower bound on the accuracy of parameter estimation methods for linear sensorimotor synchronization models. *Timing & Time Perception*, *3*(1-2), 32–51.

Jacoby, N., Tishby, N., Repp, B. H., Ahissar, M., & Keller, P. E. (2015). Parameter estimation of linear sensorimotor synchronization models: phase correction, period correction, and ensemble synchronization. *Timing & Time Perception*, *3*(1-2), 52–87.

Launay, J., Dean, R. T., & Bailes, F. (2014). Synchronising movements with the sounds of a virtual partner enhances partner likeability. *Cognitive processing*, *15*(4), 491–501.

Lim, I., van Wegen, E., de Goede, C., Deutekom, M., Nieuwboer, A., Willems, A., ... Kwakkel, G. (2005). Effects of external rhythmical cueing on gait in patients with parkinson's disease: a systematic review. *Clinical rehabilitation*, *19*(7), 695–713.

Miles, L. K., Nind, L. K., & Macrae, C. N. (2009). The rhythm of rapport: Interpersonal synchrony and social perception. *Journal of experimental social psychology*, *45*(3), 585–589.

Mörtl, A., Lorenz, T., Vlaskamp, B. N., Gusrialdi, A., Schubö, A., & Hirche, S. (2012). Modeling inter-human movement coordination: synchronization governs joint task dynamics. *Biological cybernetics*, *106*(4-5), 241–259.

Murteira, B. F., Muller, D. A., & Turkman, K. F. (1993). *Analise de sucessoes cronologicas.* McGraw-Hill, Lisboa.

Myung, I. J. (2003). Tutorial on maximum likelihood estimation. *Journal of mathematical Psychology*, *47*(1), 90–100.

Noy, D. (2017). *A multimodal approach to interpersonal gait synchronization.* (Unpublished doctoral dissertation). Uminho. (unpublished thesis)

Noy, D., Mouta, S., Lamas, J., Basso, D., Silva, C., & Santos, J. A. (2017). Audiovisual integration increases the step synchronization of side-by-side walkers. *Human Movement Science*.

Pinheiro, J., & Bates, D. (2000). *Mixed-effects models in s and s-plus.* Springer, New York.

Reddish, P., Fischer, R., & Bulbulia, J. (2013). Let‚Äôs dance together: synchrony, shared intentionality and cooperation. *PloS one*, *8*(8), e71182.

Repp, B. H. (2001). Processes underlying adaptation to tempo changes in sensorimotor synchronization. *Human movement science*, *20*(3), 277–312.

Repp, B. H. (2005). Sensorimotor synchronization: a review of the tapping literature. *Psychonomic bulletin & review*, *12*(6), 969–992.

Repp, B. H., & Keller, P. E. (2004). Adaptation to tempo changes in sensorimotor synchronization: Effects of intention, attention, and awareness. *Quarterly Journal of Experimental Psychology Section A*, *57*(3), 499–521.

Repp, B. H., & Keller, P. E. (2008). Sensorimotor synchronization with adaptively timed sequences. *Human movement science*, *27*(3), 423–456.

Repp, B. H., Keller, P. E., & Jacoby, N. (2012). Quantifying phase correction in sensorimotor synchronization: empirical comparison of three paradigms. *Acta psychologica*, *139*(2), 281–290.

Schmidt, R. C., & Richardson, M. J. (2008). Dynamics of interpersonal coordination. In *Coordination: Neural, behavioral and social dynamics* (p. 281-308). Springer.

Schulze, H.-H., & Vorberg, D. (2002). Linear phase correction models for synchronization: Parameter identification and estimation of parameters. *Brain and Cognition*, *48*(1), 80–97.

Torre, K., & Balasubramaniam, R. (2009). Two different processes for sensorimotor synchronization in continuous and discontinuous rhythmic movements. *Experimental Brain Research*, *199*(2), 157–166.

Valdesolo, P., Ouyang, J., & DeSteno, D. (2010). The rhythm of joint action: Synchrony promotes cooperative ability. *Journal of Experimental Social Psychology*, *46*(4), 693–695.

Valturio, R. (1921). *De re militari.* Wechel.

Van Der Steen, M. C., & Keller, P. E. (2013). The adaptation and anticipation model (adam) of sensorimotor synchronization. *Frontiers in human neuroscience*, *7*, 253.

van Ulzen, N. R., Lamoth, C. J., Daffertshofer, A., Semin, G. R., & Beek, P. J. (2008). Characteristics of instructed and uninstructed interpersonal coordination while walking side-by-side. *Neuroscience Letters*, *432*(2), 88–93.

Vroomen, J., & Keetels, M. (2010). Perception of intersensory synchrony: a tutorial review. *Attention, Perception, & Psychophysics*, *72*(4), 871–884.

Wiltermuth, S. S., & Heath, C. (2009). Synchrony and cooperation. *Psychological science*, *20*(1), 1–5.

Wing, A. M., Endo, S., Bradbury, A., & Vorberg, D. (2014). Optimal feedback correction in string quartet synchronization. *Journal of The Royal Society Interface*, *11*(93), 20131125.

Wing, A. M., & Kristofferson, A. B. (1973). Response delays and the timing of discrete motor responses. *Perception & Psychophysics*, *14*(1), 5–12.

Zelaznik, H. N., Spencer, R., & Ivry, R. B. (2002). Dissociation of explicit and implicit timing in repetitive tapping and drawing movements. *Journal of Experimental Psychology: Human Perception and Performance*, *28*(3), 575.

Appendices

A Auxiliary functions

```r
#Matrix inverse
#compute inverse of invertible square-root of Lambda, called Y
matrix_inverse<-function(LAMBDA)
  {
    E <- eigen(LAMBDA)
    V <- E$values
    Q <- E$vectors
    Y <- Q%*%diag(1/sqrt(V))%*%t(Q)
    Y
  }


#Off diagonal function
#Allows filling the offdiagonal entries of a matrix
off_diag<-function(N)
  {
    mat<-matrix(1, nrow=N, ncol=N)
    # A companion matrix that indicates how "off" a diagonal is
    delta <- row(mat) - col(mat)
    # Set these to select on the "delta" matrix
    low <- -1
    high <- 1
    # Operate on the "mat" matrix
    mat[delta < low | delta > high] <-0
    diag(mat)<-0
    mat
  }



#Multiplot was retrieved from http://www.cookbook-r.com/Graphs/Multiple_graphs_on_one_page_(ggplot2)/
#Allows plotting multiple independent plots together
{
  multiplot <- function(..., plotlist=NULL, file, cols=1, rows=1, layout=NULL,
                        labs=list(), labpos=list(c(0.525,0.35), c(0.023,0.67))) {
    require(grid)

    # Make a list from the ... arguments and plotlist
    plots <- c(list(...), plotlist)

    numPlots = length(plots)

    # If layout is NULL, then use 'cols' to determine layout
    if (is.null(layout)) {
      # Make the panel
      # ncol: Number of columns of plots
      # nrow: Number of rows needed, calculated from # of cols
      layout <- matrix(seq(1, cols * rows),
                       ncol = cols, nrow = rows)
    }

    if (numPlots==1) {
      print(plots[[1]])

    } else {
      # Set up the page
      grid.newpage()
      pushViewport(viewport(layout = grid.layout(nrow(layout), ncol(layout))))

      # Make each plot, in the correct location
      for (i in 1:numPlots) {
        # Get the i,j matrix positions of the regions that contain this subplot
        matchidx <- as.data.frame(which(layout == i, arr.ind = TRUE))

        print(plots[[i]], vp = viewport(layout.pos.row = matchidx$row,
                                        layout.pos.col = matchidx$col))
      }

      if(!length(labs) == 0){
        grid.text(labs[1], x=labpos[[1]][1], y=labpos[[1]][2], gp=gpar(fontsize=16))
        grid.text(labs[2], x=labpos[[2]][1], y=labpos[[2]][2], rot=90, gp=gpar(fontsize=16))
      }
    }
  }
}
```

B Simulation function (bGLS)

```
  Simulate_bGLS<-function(N, nseq, alpha, st, sm)
  {
    As<-matrix(NA, nrow=N, ncol=nseq)
    for(o in 1:nseq)
    {
      M=rnorm(N+2)
      T=rnorm(N+2)

      Z=st*T[1:(N+1)]-sm*M[2:(N+2)]+sm*M[1:(N+1)]
      AA=rep(0,N+2)
      for(I in 1:(N+1))
      {
        AA[I+1]<-(1-alpha)*AA[I]+Z[I]
      }
      As[,o]<-AA[3:(N+2)]
    }
    As
  }


#Simulation function with random intercept
Simulate_bGLS_RANDOM<-function(N, nseq, alpha, st, sm, random)
{
  As<-matrix(NA, nrow=N, ncol=nseq)
  for(o in 1:nseq)
  {
    M=rnorm(N+2)
    T=rnorm(N+2)

    Z=st*T[1:(N+1)]-sm*M[2:(N+2)]+sm*M[1:(N+1)]
    AA=rep(0,N+2)
    for(I in 1:(N+1))
    {
      AA[I+1]<-(1-alpha)*AA[I]+Z[I]
    }
    As[,o]<-AA[3:(N+2)]+rnorm(1,0,random) #add random intercept
  }
  As
}
```

C Parameter estimation function (bGLS)

```
  Estimate_bGLS<-function(As, MEAN_A)
  {
    ITER=20
    TRESH=1e-3

    N=dim(As)[1]
    nseq=dim(As)[2]

    #MEAN_A<-mean(As)
    esm=As-MEAN_A

    b=esm[2:N,]
    B=esm[1:(N-1),]


    alpha_s<-rep(NA, nseq)
    st_s<-rep(NA, nseq)
    sm_s<-rep(NA, nseq)

    #forloop for each nseq
    for (KK in 1:nseq)
    {
      #KK<-1
      b1<-b[,KK]
      B1<-B[,KK]

      z<-solve(t(B1)%*%B1)%*%t(B1)%*%b1
      zold<-z

      for (II in 1:ITER)
      {
        d=b1-B1*z
        K11<-var(d)
        K12<-cov(d[1:(length(d)-1)], d[2:(length(d))] )

        #apply bound of the bGLS
        if(K12>0){K12=0}
        if(K11<3*(-K12)){K11=-3*K12}


        #off diagonal matrix
        mat<-matrix(1, nrow=N-1, ncol=N-1)
        delta <- row(mat) - col(mat)
        low <- -1
        high <- 1
        mat[delta < low | delta > high] <-0
        diag(mat)<-0
        mat
        CC<-K11*diag(1, nrow=N-1, ncol=N-1)+K12*mat
        iC<-solve(CC)

        z<-solve(t(B1)%*%iC%*%B1)%*%t(B1)%*%iC%*%b1

        if(max(abs(z-zold))<TRESH) {break}
        zold=z
      }

      alpha_s[KK]<-1-z
      sm_s[KK]<-sqrt(-K12+2.2204e-16)
      st_s[KK]<-sqrt(K11-2*(sm_s[KK]^2))
    }

    alpha<-mean(alpha_s)
    st<-mean(st_s)
    sm<-mean(sm_s)
    out<-data.frame(alpha, st, sm)
    names(out)<-c("alpha","st","sm")
    out
  }


Estimate_bGLS_RANDOM<-function(As, MEAN_A)
{
  ITER=20
  TRESH=1e-3

  N=dim(As)[1]
  nseq=dim(As)[2]

  MEAN_A<-mean(As)
  esm=As-MEAN_A

  b=esm[2:N,]
  B=esm[1:(N-1),]



  alpha_s<-rep(NA, nseq)
  st_s<-rep(NA, nseq)
  sm_s<-rep(NA, nseq)

  #forloop for each nseq
  for (KK in 1:nseq)
  {
    b1<-b[,KK]
    B1<-B[,KK]

    z<-solve(t(B1)%*%B1)%*%t(B1)%*%b1
    zold<-z

    for (II in 1:ITER)
    {
      d=b1-B1*z
      K11<-var(d)
      K12<-cov(d[1:(length(d)-1)], d[2:(length(d))] )

      #apply bound of the bGLS
      if(K12>0){K12=0}
      if(K11<3*(-K12)){K11=-3*K12}
      #off diagonal matrix
      mat<-matrix(1, nrow=N-1, ncol=N-1)
      delta <- row(mat) - col(mat)
      low <- -1
      high <- 1
      mat[delta < low | delta > high] <-0
      diag(mat)<-0
      mat

      CC<-K11*diag(1, nrow=N-1, ncol=N-1)+K12*mat
      iC<-solve(CC)

      z<-solve(t(B1)%*%iC%*%B1)%*%t(B1)%*%iC%*%b1

      if(max(abs(z-zold))<TRESH) {break}
      zold=z
    }

    alpha_s[KK]<-1-z
    sm_s[KK]<-sqrt(-K12+2.2204e-16)
    st_s[KK]<-sqrt(K11-2*(sm_s[KK]^2))
  }

  alpha<-mean(alpha_s)
  st<-mean(st_s)
  sm<-mean(sm_s)
  out<-data.frame(alpha, st, sm)
  names(out)<-c("alpha","st","sm")
  out
}
```

D Method validation (bGLS)

```
###---Set parameters
#Parameters chosen according to Vorberg and Schulze (2002) Table 1
alphaS=seq(from=0.1, to=2, by=0.1)
st=sqrt(100)
sm=sqrt(25)
N=31
nseq=15
MEAN_e=0
###---

###---Simulate Mode
SIMULATION_REPEATS=50 #
#summary variables for estiamtes.
estimates_alpha=matrix(NA, nrow=SIMULATION_REPEATS, ncol=length(alphaS))
estimates_st=matrix(NA, nrow=SIMULATION_REPEATS, ncol=length(alphaS))
estimates_sm=matrix(NA, nrow=SIMULATION_REPEATS, ncol=length(alphaS))

for(KK in 1:SIMULATION_REPEATS)
{
  for(I in 1:length(alphaS))
  {
    #I<-1
    alpha=alphaS[I]
    es=Simulate_bGLS(N, nseq, alpha, st, sm) #simulate data
    results=Estimate_bGLS(es,MEAN_e) #estimate parameters
    estimates_alpha[KK,I]<-results$alpha
    estimates_st[KK,I]<-results$st
    estimates_sm[KK,I]<-results$sm
  }
}

#join data
ALPHA$type<-"ALPHA"
SM$type<-"SM"
ST$type<-"ST"
data<-rbind(ALPHA,SM,ST)
###---

#---Plotting
require(ggplot2)
fs<-15
plt_alpha<-ggplot(data[data$type=="ALPHA",], aes(x=alphaS, y=mean)) +
  geom_point()+
  theme_bw() +
  geom_errorbar(aes(ymin=cl, ymax=cu), colour="black", width=.05)+
  annotate("segment", x=0, xend=2,y=0, yend=2, col="green",linetype=2)+
  ylab("hat{Alpha}")+
  xlab("")

plt_st<-ggplot(data[data$type=="ST",], aes(x=alphaS, y=mean)) +
  geom_point()+
  theme_bw() +
  geom_errorbar(aes(ymin=cl, ymax=cu), colour="black", width=.05)+
  annotate("segment", x=0, xend=2,y=10, yend=10, col="green",linetype=2)+
  ylim(0, 20)+
  ylab("hat{Sigma T}")+
  xlab("")

plt_sm<-ggplot(data[data$type=="SM",], aes(x=alphaS, y=mean)) +
  geom_point()+
  theme_bw() +
  geom_errorbar(aes(ymin=cl, ymax=cu), colour="black", width=.05)+
  annotate("segment", x=0, xend=2,y=5, yend=5, col="green",linetype=2)+
  ylim(0, 20)+
  ylab("hat{Sigma M}")+
  xlab("Alpha")
```

E Method validation for asynchrony sequences that vary among sequences (bGLS)

```
###---Set Parameters
#Parameters chosen according to Vorberg and Schulze (2002) table 1.
alphaS=seq(from=0.1, to=2, by=0.1)
st=sqrt(100)
sm=sqrt(25)
N=31
nseq=15
random=10
MEAN_e=0
###---
```

```
###---Simulate Model
SIMULATION_REPEATS=50
# summary variables for estiamtes.
estimates_alpha=matrix(NA, nrow=SIMULATION_REPEATS, ncol=length(alphaS))
estimates_st=matrix(NA, nrow=SIMULATION_REPEATS, ncol=length(alphaS))
estimates_sm=matrix(NA, nrow=SIMULATION_REPEATS, ncol=length(alphaS))

for(KK in 1:SIMULATION_REPEATS)
{
  for(I in 1:length(alphaS))
  {
    #I<-1
    alpha=alphaS[I]
    es=Simulate_bGLS_RANDOM(N, nseq, alpha, st, sm, random) #simulate data
    results=Estimate_bGLS_RANDOM(es,mean(As)) #or 0
    estimates_alpha[KK,I]<-results$alpha
    estimates_st[KK,I]<-results$st
    estimates_sm[KK,I]<-results$sm
  }
}
###---

###---DRAW results
mean_alpha<-apply(estimates_alpha, 2, function(x) mean(x)) #mean
sdt_alpha<-apply(estimates_alpha, 2, function(x) sd(x)) #standard deviation
#confidence interval
ALPHA<-as.data.frame(cbind(mean_alpha, mean_alpha+qnorm(0.975)*sdt_alpha/sqrt(SIMULATION_REPEATS),
                           mean_alpha-qnorm(0.975)*sdt_alpha/sqrt(SIMULATION_REPEATS), alphaS))
names(ALPHA)<-c("mean", "cl", "cu", "alphaS")
##st
mean_st<-apply(estimates_st, 2, function(x) mean(x))
sdt_st<-apply(estimates_st, 2, function(x) sd(x))
ST<-as.data.frame(cbind(mean_st, mean_st+qnorm(0.975)*sdt_st/sqrt(SIMULATION_REPEATS),
                        mean_st-qnorm(0.975)*sdt_st/sqrt(SIMULATION_REPEATS), alphaS))
names(ST)<-c("mean", "cl", "cu", "alphaS")
##sm
mean_sm<-apply(estimates_sm, 2, function(x) mean(x))
sdt_sm<-apply(estimates_sm, 2, function(x) sd(x))
SM<-as.data.frame(cbind(mean_sm, mean_sm+qnorm(0.975)*sdt_sm/sqrt(SIMULATION_REPEATS),
                        mean_sm-qnorm(0.975)*sdt_sm/sqrt(SIMULATION_REPEATS), alphaS))
names(SM)<-c("mean", "cl", "cu", "alphaS")

ALPHA$type<-"ALPHA"
SM$type<-"SM"
ST$type<-"ST"
data_RANDOM<-rbind(ALPHA,SM,ST)
###---

###---Plotting
require(ggplot2)
fs<-15
plt_alpha<-ggplot(data_RANDOM[data_RANDOM$type=="ALPHA",], aes(x=alphaS, y=mean)) +
  geom_point()+
  theme_bw() +
  geom_errorbar(aes(ymin=cl, ymax=cu), colour="black", width=.05)+
  annotate("segment", x=0, xend=2,y=0, yend=2, col="green",linetype=2)+
  theme(axis.line = element_line(colour = "black"),panel.grid.major = element_blank(),
        panel.grid.minor = element_blank(),panel.background = element_blank(),
        strip.background = element_blank(),
      axis.text=element_text(size=fs),axis.title=element_text(size=fs))+
  ylab("hat{Alpha}")+
  xlab("")

plt_st<-ggplot(data_RANDOM[data_RANDOM$type=="ST",], aes(x=alphaS, y=mean)) +
  geom_point()+
  theme_bw() +
  geom_errorbar(aes(ymin=cl, ymax=cu), colour="black", width=.05)+
  annotate("segment", x=0, xend=2,y=10, yend=10, col="green",linetype=2)+
  ylim(0, 20)+
  theme(axis.line = element_line(colour = "black"),panel.grid.major = element_blank(),
      panel.grid.minor = element_blank(),panel.background = element_blank(),
      strip.background = element_blank(),
      axis.text=element_text(size=fs),axis.title=element_text(size=fs))+
  ylab("hat{Sigma T}")+
  xlab("")

plt_sm<-ggplot(data_RANDOM[data_RANDOM$type=="SM",], aes(x=alphaS, y=mean)) +
  geom_point()+
  theme_bw() +
  geom_errorbar(aes(ymin=cl, ymax=cu), colour="black", width=.05)+
  annotate("segment", x=0, xend=2,y=5, yend=5, col="green",linetype=2)+
  ylim(0, 20)+
  theme(axis.line = element_line(colour = "black"),panel.grid.major = element_blank(),
      panel.grid.minor = element_blank(),panel.background = element_blank(),
      strip.background = element_blank(),
      axis.text=element_text(size=fs),axis.title=element_text(size=fs))+
  ylab("hat{Sigma M}")+
  xlab("Alpha")

multiplot(plt_alpha, plt_st, plt_sm, rows=3)
```

F Simulation function (eLM)

```
  Simulate_eLM<-function(N, nseq, alpha, st, sm)
  {
    As<-matrix(NA, nrow=N, ncol=nseq)
    for(o in 1:nseq)
    {
      M=rnorm(N+2)
      T=rnorm(N+2)

      Z=st*T[1:(N+1)]-sm*M[2:(N+2)]+sm*M[1:(N+1)]
      AA=rep(0,N+2)
      for(I in 1:(N+1))
      {
        AA[I+1]<-(1-alpha)*AA[I]+Z[I]
      }
      As[,o]<-AA[3:(N+2)]
    }
    As
  }
```

G Parameter estimation function (eLM)

```
eLM<-function(As, MEAN_A)
{
  MEAN_A=0
  esm=As-MEAN_A

  b=esm[2:dim(As)[1],]
  B=esm[1:(dim(As)[1]-1),]

  #construct dataframe depending on nseq=1 or >1
  if(dim(data.frame(B))[2]>1){
    require(reshape2)
    data<-melt(b); names(data)<-c("n", "nseq", "b")
    data$B<-melt(B)$value
  } else {data<-data.frame(c(1:length(b)), rep(1,length(b)), b, B);
  names(data)<-c("n", "nseq", "b", "B")}


  ##--Alpha_hat as function of lambda
  fun_lambda<-function(lambda)
  {
    #Construct subset LAMBDA
    out<-lapply(split(data, data$nseq), function(x)
    {
      M<-dim(x)[1]
      #contruct LAMBDA
      diagonal=diag(M)
      offdiagonal=off_diag(M)*lambda
      LAMBDA=diagonal+offdiagonal
      #transformation
      LAMBDA_inv=matrix_inverse(LAMBDA)

      y_star<-LAMBDA_inv%*%x$b
      x_star<-LAMBDA_inv%*%x$B
      list(y_star, x_star, LAMBDA)
    })

    ##OLS
    #x and y are stacked for OLS
    x_star<-unlist(lapply(out, `[[`, 1))
    y_star<-unlist(lapply(out, `[[`, 2))

    #get each individual LAMBDA
    ind_LAMBDA<-lapply(out, `[[`, 3)

    #OLS
    alpha_lambda_hat=solve(t(x_star)%*%x_star)%*%t(x_star)%*%y_star

    #output
    return(list(x_star, y_star, alpha_lambda_hat, ind_LAMBDA))
  }
  ##--

  ##--PROFILED MAXIMUM LIKELIHOOD
  MLE<-function(lambda)
  {
    #lambda<--0.2
    out<-fun_lambda(lambda)


    x_star<-out[[1]]
```

```
      y_star<-out[[2]]
      M<-length(x_star)
      alpha<-out[[3]]
      ind_LAMBDA<-out[[4]]

      LA<-sum(unlist(lapply(ind_LAMBDA, function(x) log(det(x)))))
      return(-(-M*log(sqrt(t(y_star-x_star%*%alpha)%*%(y_star-x_star%*%alpha)))-0.5*LA))
    }
    ##--

    lambda_hat<-optim(par=0.1, MLE,
                      method = c("L-BFGS-B"),
                      lower = -0.5, upper = -0.1)$par

    M<-dim(data)[1]
    #contruct LAMBDA
    diagonal=diag(M)
    offdiagonal=off_diag(M)*lambda_hat
    LAMBDA=diagonal+offdiagonal

    ##--transformation
    LAMBDA_inv=matrix_inverse(LAMBDA)
    y_star<-LAMBDA_inv%*%data$b
    x_star<-LAMBDA_inv%*%data$B

    ##--OLS
    alpha_hat=1-solve(t(x_star)%*%x_star)%*%t(x_star)%*%y_star
    sig_e<-summary(lm(y_star~x_star))$sigm^2

    sm<-sqrt(-(sig_e*lambda_hat))+2.2204e-16)
    st<-sqrt(sig_e-2*(sm^2))

    data.frame(alpha_hat, st, sm)
  }
```

H Method validation (eLM)

```
###---Set Parameters
alphaS=seq(from=0.1, to=2, by=0.1); st=10; sm=5; N=6
nseq=15
MEAN_e=0
###---

###---Simulation Model
BOOT<-50 #number of simulation
estimate_BOOT<-list()
for(KK in 1:BOOT)
{
  #KK<-1
  estimates=list()
  for(I in 1:length(alphaS))
  {
    #I<-1
    alpha=alphaS[I]
    es=Simulate_eLM(N, nseq, alpha, st, sm) #simulate data
    results=eLM(es,MEAN_e)
    results<-cbind(results, alphaS[I], nseq, N);
    names(results)[4:6]<-c("alphas", "N", "nseq")
    estimates[[I]]<-results
  }
  estimates<-do.call(rbind,estimates)
  estimates$boot<-KK
  estimate_BOOT[[KK]]<-estimates
}
estimates_BOOT<-do.call(rbind, estimate_BOOT)
estimates_BOOT<-estimates_BOOT[complete.cases(estimates_BOOT),]
###---

###---DRAW results
mean_est<-aggregate(.~ alphas, estimates_BOOT, FUN = function(x) mean(x))
sd_est<-aggregate(.~ alphas, estimates_BOOT, FUN = function(x) sd(x))

#mean
mean_alphas<-mean_est[,1:2]; names(mean_alphas)<-c("alphas", "mean"); mean_alphas$type<-c("ALPHA")
mean_st<-mean_est[,c(1,3)]; names(mean_st)<-c("alphas", "mean"); mean_st$type<-c("ST")
mean_sm<-mean_est[,c(1,4)]; names(mean_sm)<-c("alphas", "mean"); mean_sm$type<-c("SM")
mean_eLM<-rbind(mean_alphas, mean_st, mean_sm)

sd_alphas<-sd_est[,1:2]; names(sd_alphas)<-c("alphas", "sd"); mean_alphas$type<-c("ALPHA")
sd_st<-sd_est[,c(1,3)]; names(sd_st)<-c("alphas", "sd"); mean_st$type<-c("ST")
sd_sm<-sd_est[,c(1,4)]; names(sd_sm)<-c("alphas", "sd"); mean_sm$type<-c("SM")
sd_eLM<-rbind(sd_alphas, sd_st, sd_sm)

mean_eLM$cl<-mean_eLM$mean-qnorm(0.975)*sd_eLM$sd/sqrt(BOOT)
mean_eLM$cu<-mean_eLM$mean+qnorm(0.975)*sd_eLM$sd/sqrt(BOOT)

eLM_data<-data.frame(mean_eLM$mean, mean_eLM$cl, mean_eLM$cu, mean_eLM$alphas, mean_eLM$type)
names(eLM_data)<-c("mean", "cl", "cu", "alphaS", "type")
###---

###---Plotting
eLM_alpha_standard<-ggplot(eLM_data[eLM_data$type=="ALPHA",], aes(x=alphaS, y=mean)) +
  geom_point()+
  theme_bw() +
  geom_errorbar(aes(ymin=cl, ymax=cu), colour="black", width=.05)+
  ggtitle("bGLS Normal")+
  annotate("segment", x=0, xend=2,y=0, yend=2, col="green",linetype=2)+
  theme(axis.line = element_line(colour = "black"),
        panel.grid.major = element_blank(),
        panel.grid.minor = element_blank(),
        panel.background = element_blank(),
        strip.background = element_blank(),
        axis.text=element_text(size=fs),
        axis.title=element_text(size=fs))+
  ylab("hat{Alpha}")+
  xlab("")

eLM_st_standard<-ggplot(eLM_data[eLM_data$type=="ST",], aes(x=alphaS, y=mean)) +
  geom_point()+
  theme_bw() +
  geom_errorbar(aes(ymin=cl, ymax=cu), colour="black", width=.05)+
  annotate("segment", x=0, xend=2,y=10, yend=10, col="green",linetype=2)+
  ylim(0, 20)+
  theme(axis.line = element_line(colour = "black"),
        panel.grid.major = element_blank(),
        panel.grid.minor = element_blank(),
        panel.background = element_blank(),
        strip.background = element_blank(),
        axis.text=element_text(size=fs),
        axis.title=element_text(size=fs))+
  ylab("hat{Sigma T}")+
  xlab("")

eLM_sm_standard<-ggplot(eLM_data[eLM_data$type=="SM",], aes(x=alphaS, y=mean)) +
  geom_point()+
  theme_bw() +
  geom_errorbar(aes(ymin=cl, ymax=cu), colour="black", width=.05)+
  annotate("segment", x=0, xend=2,y=5, yend=5, col="green",linetype=2)+
  ylim(0, 20)+
  theme(axis.line = element_line(colour = "black"),
        panel.grid.major = element_blank(),
        panel.grid.minor = element_blank(),
        panel.background = element_blank(),
        strip.background = element_blank(),
        axis.text=element_text(size=fs),
        axis.title=element_text(size=fs))+
  ylab("hat{Sigma M}")+
  xlab("Alpha")

multiplot(eLM_alpha_standard, eLM_st_standard, eLM_sm_standard, rows=3, cols=1, labs=list("", ""))
```

I Simulation function (MEM)

```
Simulate_MEM<-function(N, nseq, alpha, st, sm, random)
    {
      As<-matrix(NA, nrow=N, ncol=nseq)
      for(o in 1:nseq)
      {
        M=rnorm(N+2)
        T=rnorm(N+2)

        Z=st*T[1:(N+1)]-sm*M[2:(N+2)]+sm*M[1:(N+1)]
        AA=rep(0,N+2)
        for(I in 1:(N+1))
        {
          AA[I+1]<-(1-alpha)*AA[I]+Z[I]
        }
        As[,o]<-AA[3:(N+2)]+rnorm(1,0,random)
      }
      As
    }
```

J Parameter estimation function (MEM)

```
Estimate_MEM<-function(As, MEAN_A) #MEAN_A from data #output alpha, st, sm)
    {
      esm=As
      b=esm[2:dim(As)[1],]
      B=esm[1:(dim(As)[1]-1),]

      #wide to long
      require(reshape2)
      data<-melt(b);
      names(data)<-c("n", "nseq", "b")
      data$B<-melt(B)$value

      require(eLMe)
      MA1.lme<-lme(b~B, data=data, random=~1|nseq, correlation=corARMA(q=1, form=~1|nseq), method="ML")

      #obtain parameters
      beta_hat<-summary(MA1.lme)$coefficients$fixed[2]
      theta<-coef(MA1.lme$modelStruct$corStruct,unconstrained=FALSE)

      alpha_hat<-1-beta_hat
      sig_e<-summary(MA1.lme)$sigma


      sigM2<--theta*sig_e^2
      sigT2<-sig_e^2*(1-theta^2)-2*sigM2

      sm_hat<-sqrt(sigM2)
      st_hat<-sqrt(sigT2)

      data.frame(alpha_hat, st_hat, sm_hat)
    }
```

K Method validation (MEM)

```
###---Set Parameters
alphaS=seq(from=0.1, to=2, by=0.1)
st=10; sm=5; N=31; nseq=15; random=20; MEAN_e=0

###---Simulate the Model
BOOT<-50 #number of repetitions
estimate_BOOT<-list()
  for(KK in 1:BOOT)
  {
    #KK<-1
    estimates=list()
    for(I in 1:length(alphaS))
    {
      #I<-1
      alpha=alphaS[I]
      es=Simulate_MEM(N, nseq, alpha, st, sm, random) #simulate data
      results=Estimate_MEM(es,MEAN_e)
      results<-cbind(results, alphaS[I], nseq, N);
      names(results)[4:6]<-c("alphas", "N", "nseq")
      estimates[[I]]<-results
    }
    estimates<-do.call(rbind,estimates)
    estimates$boot<-KK
    estimate_BOOT[[KK]]<-estimates
  }

estimates_BOOT<-do.call(rbind, estimate_BOOT)
estimates_BOOT<-estimates_BOOT[complete.cases(estimates_BOOT),]
###---
```

```r
###---DRAW results
mean_est<-aggregate(.~ alphas, estimates_BOOT, FUN = function(x) mean(x))
sd_est<-aggregate(.~ alphas, estimates_BOOT, FUN = function(x) sd(x))

#mean
mean_alphas<-mean_est[,1:2]; names(mean_alphas)<-c("alphas", "mean"); mean_alphas$type<-c("ALPHA")
mean_st<-mean_est[,c(1,3)]; names(mean_st)<-c("alphas", "mean"); mean_st$type<-c("ST")
mean_sm<-mean_est[,c(1,4)]; names(mean_sm)<-c("alphas", "mean"); mean_sm$type<-c("SM")
mean_MEM<-rbind(mean_alphas, mean_st, mean_sm)

sd_alphas<-sd_est[,1:2]; names(sd_alphas)<-c("alphas", "sd"); mean_alphas$type<-c("ALPHA")
sd_st<-sd_est[,c(1,3)]; names(sd_st)<-c("alphas", "sd"); mean_st$type<-c("ST")
sd_sm<-sd_est[,c(1,4)]; names(sd_sm)<-c("alphas", "sd"); mean_sm$type<-c("SM")
sd_MEM<-rbind(sd_alphas, sd_st, sd_sm)

mean_MEM$cl<-mean_MEM$mean-qnorm(0.975)*sd_MEM$sd/sqrt(BOOT)
mean_MEM$cu<-mean_MEM$mean+qnorm(0.975)*sd_MEM$sd/sqrt(BOOT)

MEM_data<-data.frame(mean_MEM$mean, mean_MEM$cl, mean_MEM$cu, mean_MEM$alphas, mean_MEM$type)
names(MEM_data)<-c("mean", "cl", "cu", "alphaS", "type")
###---


###---Plotting
MEM_alpha_standard<-ggplot(MEM_data[MEM_data$type=="ALPHA",], aes(x=alphaS, y=mean)) +
    geom_point()+
    theme_bw() +
    geom_errorbar(aes(ymin=cl, ymax=cu), colour="black", width=.05)+
    ggtitle("bGLS Normal")+
    annotate("segment", x=0, xend=2,y=0, yend=2, col="green",linetype=2)+
    theme(axis.line = element_line(colour = "black"),
          panel.grid.major = element_blank(),
          panel.grid.minor = element_blank(),
          panel.background = element_blank(),
          strip.background = element_blank(),
          axis.text=element_text(size=fs),
          axis.title=element_text(size=fs))+
    ylab("hat{Alpha}")+
    xlab("")

MEM_st_standard<-ggplot(MEM_data[MEM_data$type=="ST",], aes(x=alphaS, y=mean)) +
    geom_point()+
    theme_bw() +
    geom_errorbar(aes(ymin=cl, ymax=cu), colour="black", width=.05)+
    annotate("segment", x=0, xend=2,y=10, yend=10, col="green",linetype=2)+
    ylim(0, 20)+
    theme(axis.line = element_line(colour = "black"),
          panel.grid.major = element_blank(),
          panel.grid.minor = element_blank(),
          panel.background = element_blank(),
          strip.background = element_blank(),
          axis.text=element_text(size=fs),
          axis.title=element_text(size=fs))+
    ylab("hat{Sigma T}")+
    xlab("")

MEM_sm_standard<-ggplot(MEM_data[MEM_data$type=="SM",], aes(x=alphaS, y=mean)) +
    geom_point()+
    theme_bw() +
    geom_errorbar(aes(ymin=cl, ymax=cu), colour="black", width=.05)+
    annotate("segment", x=0, xend=2,y=5, yend=5, col="green",linetype=2)+
    ylim(0, 20)+
    theme(axis.line = element_line(colour = "black"),
          panel.grid.major = element_blank(),
          panel.grid.minor = element_blank(),
          panel.background = element_blank(),
          strip.background = element_blank(),
          axis.text=element_text(size=fs),
          axis.title=element_text(size=fs))+
    ylab("hat{Sigma M}")+
    xlab("Alpha")
```

L Comparison of the bGLS method and the eLM by Mean Squared Error (MSE)

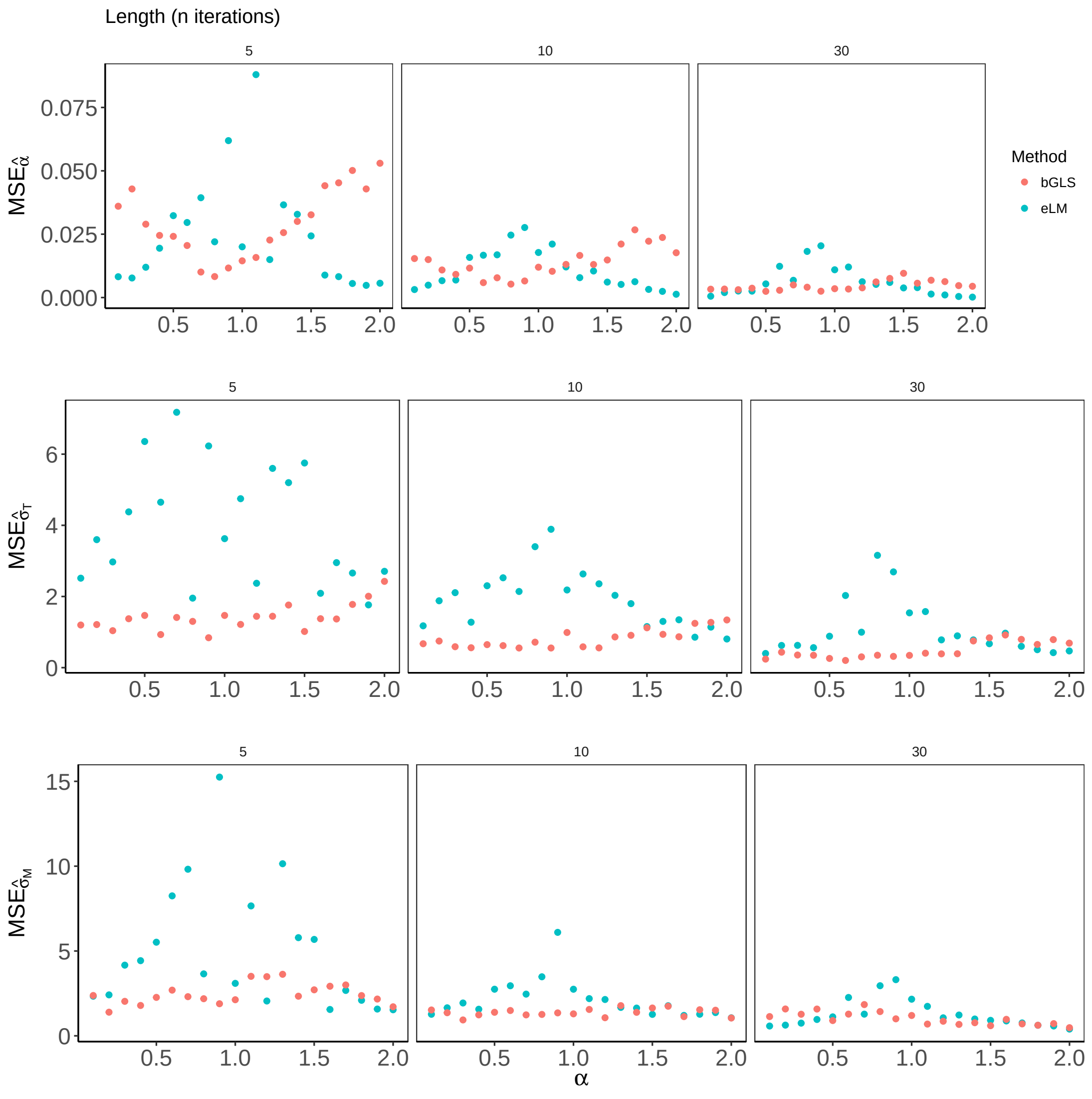


Figure 1: Mean Squared Error (MSE) of LPC model parameter estimation as a function of true α for different sequence lengths. MSE of the bGLS method are displayed in red and the MSE of the eLM method are displayed in magenta.

M Benchmark function

```
N<-10; nseq=15; a=0.5; st=10; sm=5; random=5
  As_MEM<-Simulate_MEM(N=N, nseq=nseq,  a=a, st=st, sm=sm, random=random)
  As_bGLS<-Simulate_bGLS_RANDOM(N=N, nseq=nseq,  a=a, st=st, sm=sm, random=random)
  mbm = microbenchmark::microbenchmark(Estimate_bGLS_RANDOM(As,mean(As)),
                                       Estimate_bGLS_RANDOM(As_MEM,mean(As_MEM)))
```

N Maximum Likelihood vs. Restricted Maximum Likelihood

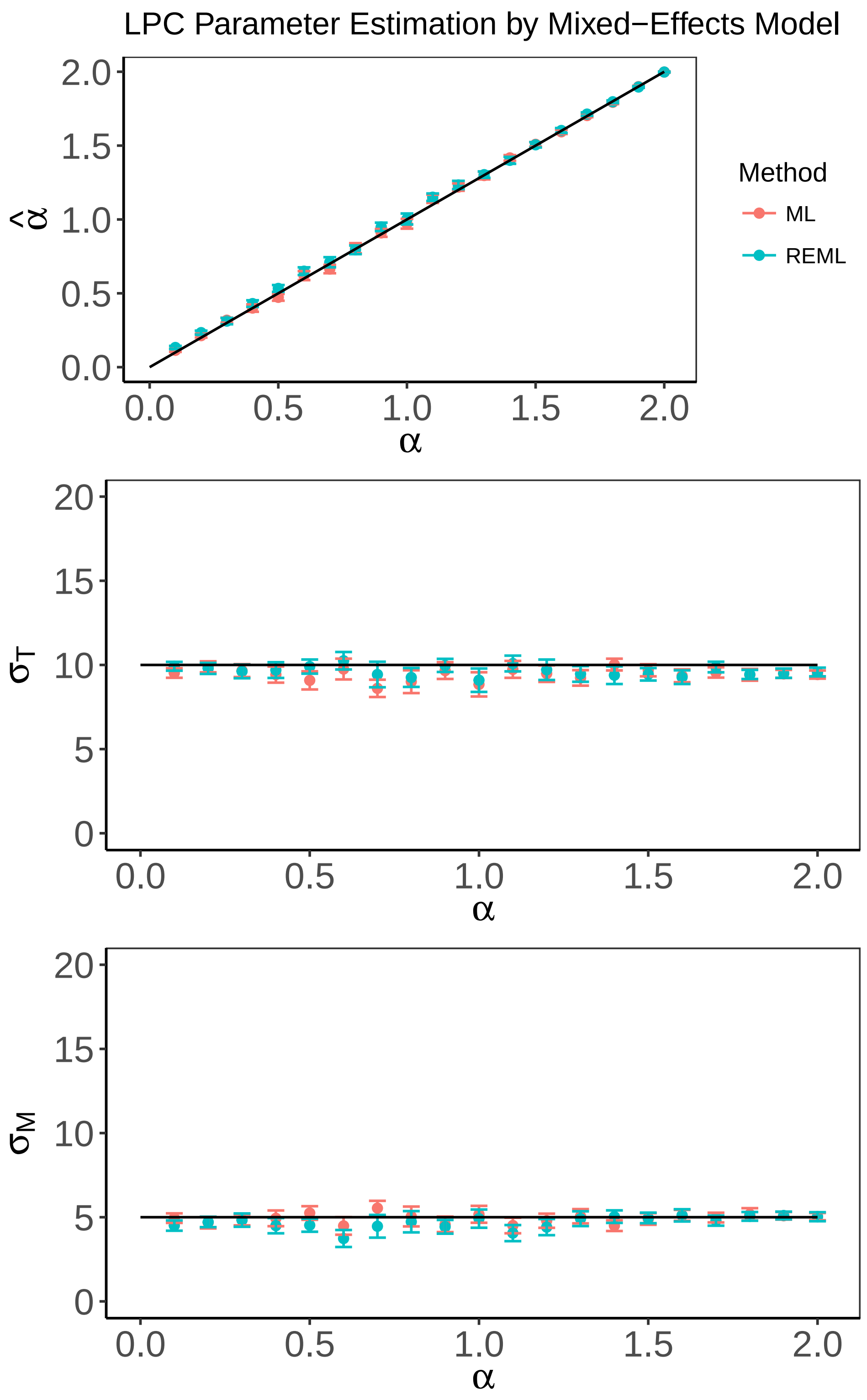


Figure 2: Comparison of Maximum Likelihood Estimation (ML) and Restricted Maximum Likelihood Estimation (REML). The between-sequence variability was sampled from $NV(0, 5^2)$. The parameters were estimated for different αs from sequences of length $n_i = 30$, simulated $m = 15$ times from the LPC model with $\sigma_T = 10$ and $\sigma_M = 5$.