



UNIVERSITY OF OSNABRÜCK

DDIFTOOL: A Matlab Toolbox for Analysis of Causality in neural data

by
Edoardo Pinzuti

Neuroinformatics Department
Institute of Cognitive Science

December 18, 2016

Contents

Description of the work	vi
List of Figures	vii
List of Tables	viii
1 Introduction	1
1.1 What is DDIFTOOL ?	1
1.2 Implementation	1
1.3 Organization of the document	1
2 Installation and Configuration	3
2.1 Download	3
2.2 Installation	3
3 Background	4
3.1 Overview	4
3.2 Embedding	5
3.3 Gaussian Processes	8
3.4 Statistical model and Estimation of delays	10
3.4.1 Statistical model	10
3.4.2 Estimation of interaction delays	11
3.4.3 Inference by Leave-One-Out (LOO)	11
3.4.4 Confidence intervals	12
3.4.5 Estimating hyperparameters	12
3.4.6 Reconstruction Error Graph (REG)	12
3.4.7 Model selection	13
3.4.8 Why does the time-series need to be sampled down? . .	14
4 Analysis Routine	15
4.1 Overview	15
4.2 Input data and configuration parameters	15

4.3	Workflow	18
4.4	Core Functions	21
4.4.1	<i>DDIFT_prepare.m</i>	21
4.4.2	<i>DDIFT_delays_analysis.m</i>	23
4.5	Plotting results	28
4.5.1	Reconstruction Error Graph (<i>REG_plot.m</i>)	28
4.5.2	<i>plot_connectivity.m</i>	29
5	Simulation and Analysis examples	32
5.1	Overview	32
5.2	Simulated data: Linear mixing and Noise	32
5.2.1	Results	35
5.2.2	Graph Algorithm	37
5.3	Connectivity analysis from simulated EEG	39
5.3.1	Result	41
5.4	Discussion	42
	Appendix A	44
A.1	Table of result	44
A.2	Matlab script	45
	Bibliography	48

Declaration of Authorship

I hereby certify that the work presented here is, to the best of my knowledge and belief, original and the result of my own investigations, except as acknowledged, and has not been submitted, either in part or whole, for a degree at this or any other university.

signature

city, date

Description of the work

The objective of this thesis is to transfer the work "A statistical Framework to Infer Delay and Direction of Information Flow from Measurements of Complex Systems" (Schumacher et al., 2015) into a FieldTrip related toolbox, in a Matlab language. The thesis is composed of two main parts: coding of the toolbox and documentation of its functionalities.

In the first part, I transferred the statistical method proposed in Schumacher et al. (2015) from Python language to Matlab language. I implemented a series of different functions that allow to work with the statistical method. Then, I designed and coded the toolbox to specifically relate to the FieldTrip toolbox .

The second part consists of the present documentation. Here, I provide a description of the toolbox (functions and its use) from which the user can start. Furthermore, I present different analysis simulations relevant in neuroscience to demonstrate the applicability of the toolbox in analysis of neural data.

List of Figures

3.1	Embedding	5
3.2	Functional reconstruction mapping	8
3.3	Example of prior and posterior distribution	9
4.1	Workflow of the toolbox	19
4.2	Auto-mutual information	22
4.3	Overview of <i>DDIFT_delays_analysis.m</i>	25
4.4	A representation of the matrix <i>dlResult</i>	26
4.5	Example of a Reconstruction error graph (REG)	28
4.6	Output plot of <i>plot_connectivity.m</i>	29
4.7	Short caption	31
5.1	Cascade effect and common drive scenarios	34
5.2	Result linear mixing	35
5.3	Result cascaded effect	36
5.4	Result common drive	37
5.5	Graph analysis	39
5.6	Simulation of sources time-series	40
5.7	Source Localization	41
5.8	Result of simulated sources	42

List of Tables

4.1	Input data in DDIFTool	16
4.2	Configuration parameter	17
4.3	Prepared data in the structure <i>DDIFT_prepare</i>	23
4.4	Results provided in the structure <i>Result.est_param</i>	26
4.5	Results provided in the structure <i>Result.cfg</i>	27
4.6	Parameter for the configuration structure in order to use <i>plot_connectivity.m</i>	30
A.1	Results of linear mixing simulations.	44

Chapter 1

Introduction

1.1 What is DDIFTOOL ?

DDIFTool (**D**elay & **D**irection of **I**nformation **F**low **T**oolbox) is an open-source Matlab toolbox that allows to infer delay and direction of information flow of complex systems from time series. It integrates with FieldTrip toolbox (Oostenveld et al., 2010) to analyze neural data. The theoretical work and the validity of the method implemented on this toolbox refers to Schumacher et al. (2015).

1.2 Implementation

DDIFTool is implemented in MATLAB (2016). The user interacts with DDIFTool through MATLAB scripts (.m-files). DDIFTool does not offer a graphical user interface.

1.3 Organization of the document

This document provides a user documentation of DDIFTool's functionalities and analysis strategies. First, we will give some information about the installation of DDIFTool in Chapter 2, next we will provide a short description of the proposed framework, to better understand the input parameter required, as well as the key concepts of the proposed statistical method in Chapter 3. Then, we will describe the core functions and additional functionalities in Chapter 4. Last, we will present some simulations and example scripts to perform analysis of neural data in Chapter 5. Useful description of the functions can be also found in the individual function's help text using

MATLAB's **help** function.

Chapter 2

Installation and Configuration

2.1 Download

It's possible to download the last DDIFTool version from <https://github.com/EPinzuti>.

DDIFTool uses some TSTOOL (<http://www.dpi.physik.uni-goettingen.de/tstool/>) functionalities and so it is included in the DDIFTool distribution. In addition, FieldTrip have to be downloaded from: <http://www.fieldtriptoolbox.org/download>.

2.2 Installation

After downloading and unpack the DDIFTool and Fieldtrip archives, add them to you MATLAB path using addpath command. TSTOOL does not need to be added to your path.

Example Matlab paths:

```
1 restoredefaultpath;  
2 addpath(' /.../.../ Fieldtrip_toolbox/fieldtrip ');  
3 ft_defaults;  
4 addpath(' /.../.../ DDIFT_toolbox/DIFTOOL ');
```

Note: DDFTOOL requires FieldTrip toolbox and TSTOOL to work properly.

Chapter 3

Background

3.1 Overview

The main goal of this section is to provide an introduction of the proposed statistical method implemented in DDIFTool toolbox. After the following paragraphs, the reader should have a sufficient understanding of the main aspects of the method, enabling him to start using the toolbox and perform related analysis. Nevertheless, we leave a more detailed and exhaustive mathematical description to Schumacher et al. (2015). First, we will present a general description of two key concepts, namely: Embedding theory and Gaussian Processes. Embedding theory within the framework of differential topology received much attention from its first formulation (Whitney, 1936), these concepts and the related theorems are often complicated, especially for non-experts of the field. To this end, we use more an intuitive approach here. Gaussian processes (GP) are well known in the context of supervised learning (Williams and Rasmussen, 2006). We will briefly present the use of Gaussian process to formulate a Bayesian framework for regression. Finally, we will describe how the statistical model is implemented in DDIFTool and how this statistical framework allows to estimate the direction of information flow and delay from complex measurements (time series).

3.2 Embedding

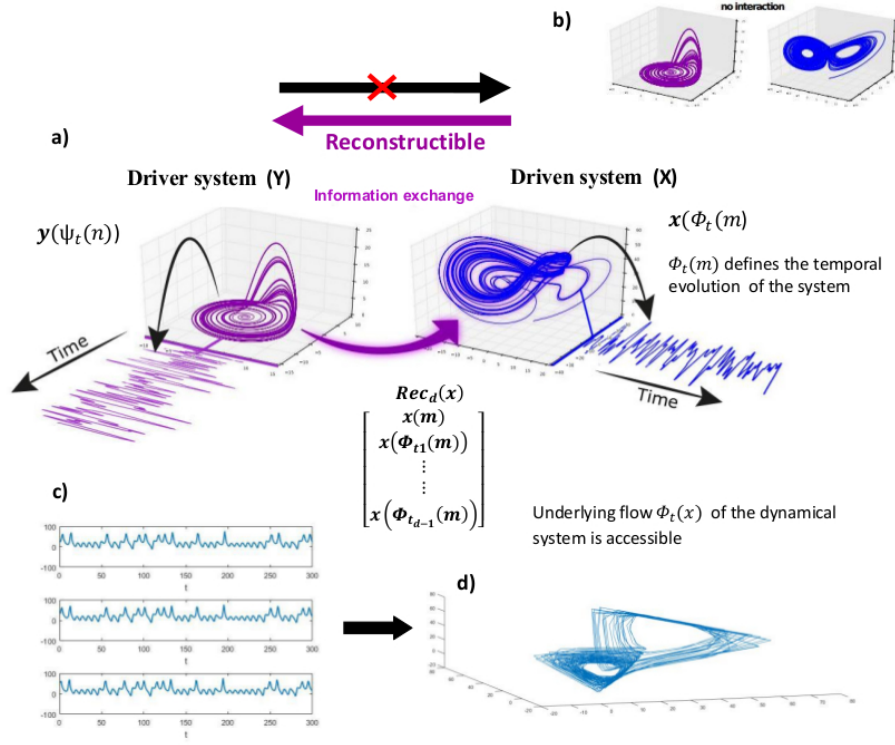


Figure 3.1: Embedding. (a) A Rossler system Y is delay-coupled with a Lorenz system X , where the information of the former flow into the latter, leading to a smooth warping on the manifold to incorporate such information. The two arrows represent the device that measures the system and maps each system's state to an observation (time series). (b) Rossler system and Lorenz system uncoupled. (c) Shifted reconstruction vectors (Rec_d). (d) Example of a reconstructed shadow manifold.(Figure (a-b) adapted from Schumacher (2015)).

In dynamical system theory, causality can be conceived as the direction of interaction between coupled systems (Schumacher, 2015). To illustrate the scenario of directed interaction in coupled system, we used an example taken from Schumacher (2015). Figure 3.1a shows a system Y (Rössler system), delay-coupled to a system X (Lorenz system), injecting its temporal state information over time into X 's state. The driver Y causes alteration of the driven system's state space, leading to a smooth deformation of the Lorenz attractor manifold (compare Figure 3.1a and 3.1b). The goal is to infer

this interaction from the observed time series, whose dependencies might be reflections of the geometrical encoded information (Schumacher, 2015), even we do not have a direct access to the underlying structure of the systems. A measurement can be thought as a (smooth) function that maps the phase state space of the dynamical system (manifold M with f dimensions) to the real number, $x: M \rightarrow \mathbb{R}$ (Aeyels, 1981). So, by means of a measure device, (in our case Local Field Potential (LFP) or EEG electrodes on the scalp), the information of the dynamical process is measured indirectly. As it shown in Figure 3.1a, the function $x(\phi_t(m))$ realizes this mapping for the driven system, giving the observed time series of real-value number (Aeyels, 1981). The measurements are taken at a set of points $t_i \in [0, T]$ for an interval of length T . This set is called sample program $P = (t - d, \dots, t - 1, t)$ and defines the reconstruction vector:

$$Rec_d = [x(m), x(\phi_{t_1}(m)), \dots, x(\phi_{t_{d-1}}(m))] \quad (3.1)$$

Where d is the reconstruction dimension and reflects the intrinsic dimensionality of the reconstructed system (Schumacher, 2015). The reconstruction vectors (obtained from a sliding window of size d) can be used, for example, as new coordinates to plot the trajectories in d -dimensional state space (see Figure 3.1c and Figure 3.1c). Takens (1981) showed that if d is large enough ($d > 2f$) the reconstructed shallow manifold maintains all the main mathematical properties as well as the topology of the original manifold. So, there exist a one to one mapping between the two manifolds (original and reconstructed one) and the temporal evolution ϕ on M becomes accessible via the reconstruction vectors (see Figure 3.1d and Sugihara et al. (2012)¹). Interestingly, embedding theory provides a framework in which the information, incorporated in the driven system, is accessible via delay maps of real-values measurement functions (Schumacher, 2015). Since the geometry of the driven system's manifold contains information of the driver due to the coupling, the chances to predict or reconstruct the driver's system state are significant (see skew product embedding theorem in Stark et al. (2003)). Furthermore, the opposite is not possible (to reconstruct system's state of X from Y), as it is shown in Figure 3.1a the driver Y does not contain any information of the system X , allowing us to exploit this asymmetries to infer causal interaction.²

As brief introduction of the method, others two aspects are worth taking

1 Series of movies for introduction to Takens's Theorem, and relationship between time series and dynamic attractors, can be found at Sugihara's Lab (2012).

2 We do not consider, here, situations of generalized synchronization (see Schumacher et al. (2015)).

into consideration. Firstly, as stated in Schumacher et al. (2015), delay-coupled systems, such as spatially distributed neural networks, are not time-invertible (the temporal evolution is described by a semi-flow not invertible in time, see concept of endomorphism in Takens (2002)). Takens (2002) showed that the system state that corresponds to the end of the sample program P , $x(\phi_{t_d-1}(m))$, it is always reconstructible, even in the presence of not time invertible systems³, and can be used for inference. Secondly, there is one additional complication if we only assume data derived from a single autonomous (skew-product) system. An autonomous system is a dynamical system that has no external input (Manjunath et al., 2012). It will evolve based on the same deterministic law and the observation function x will depend only on m , $\phi_t(m)$. Considering the context of LFP or EEG data, this does not seem to be the case. These measurements capture information from sub-populations of neurons that interact with each other and will receive others influences from other brain areas (Schumacher, 2015). To this end, this intrinsic noise clearly alters the dynamic of the system and has to be considered in the statistical model. Stark et al. (2003) provided a framework to consider stochastic dynamical system in situations where the measurements are taken from a lower-dimensional local subsystem (neural sub-population) weakly coupled to the high-dimensional global system (like the brain). To account for the extrinsic forcing, for both subpopulations (driver and driven), two random variables w, w' are introduced (see Figure 3.2). These stochastic terms are unknown, so the uncertainty have to be dealt with at the level of a statistical inference and indeed are included in the statistical model (Schumacher et al., 2015).

³ Dynamical systems, whose temporal evolution is not invertible, might not admit embedding (see Takens (2002) and Schumacher et al. (2015)).

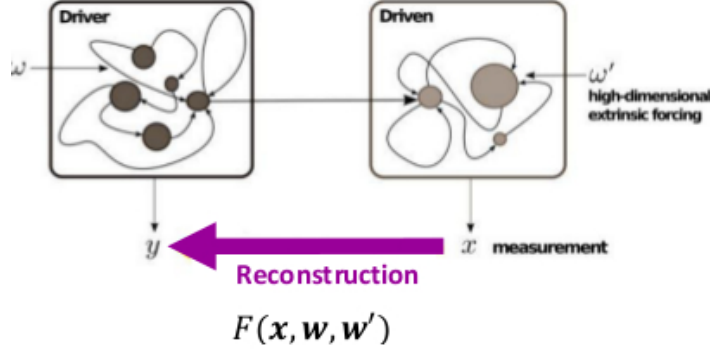


Figure 3.2: Functional reconstruction mapping. A function F that realizes the reconstruction from the dynamical system and it is subject to high-dimensional forcing w . (Figure taken from Schumacher et al. (2015)).

Due to embedding theory we can assume that exists a function F that maps measurements of the driven system to measurements y of the driver, but not the other way (Schumacher et al., 2015). Below (Subsection 3.4.1), the function $F(x, w, w')$ will be formalized in the statistical framework of Gaussian process regression.

3.3 Gaussian Processes

A Gaussian process (GP) is a collection of random variables, any finite number of which has joint Gaussian distribution (Williams and Rasmussen, 2006). The function f (a generic function) is distributed as a GP (can be thought as a series of infinite long vectors), and is a generalization of a Gaussian distribution whose mean and covariance is a vector and a matrix, respectively. A GP is defined by its mean function and covariance function. The mean function of a real process $f(z)$, is:

$$m = \mathbf{E}[f(z)], \quad (3.2)$$

and a covariance function:

$$Cov = [f(z_i), f(z_j)] = \mathbf{E}[f(z_i), f(z_j)] =: [K(z, z)]_{ij}. \quad (3.3)$$

Let be the mean function $m = 0$, then the Gaussian process can be written as:

$$f(z) \sim GP(0, [K(z, z)]). \quad (3.4)$$

In the statistical model employed here, the covariance function $K(z, z)$, or kernel, corresponds to the Volterra series model (Eq. 3.11). See Schumacher et al. (2015) for the specification of $K(z, z)$ as Volterra series model. Here, we only indicate the result of such derivation; the covariance function is:

$$k(z_i^T, z_j) = \exp(z_i^T, z_j), \quad (3.5)$$

and specifies the covariance between pairs of random variables. Furthermore, Franz and Schölkopf (2006) showed that also Volterra series can be represented implicitly as element of Reproducing Kernel Hilbert Space (RKHS)⁴.

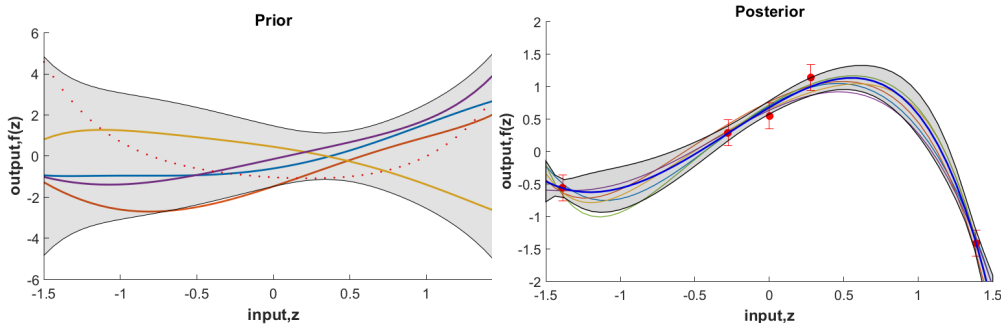


Figure 3.3: Example of prior and posterior distribution. Left, five functions drawn at random Gaussian process prior with Volterra model as covariance function. Right, five random functions drawn from the posterior (prior conditioned on five noisy observations, indicated with red dots) and the estimated mean function in blue. Additionally, the grey shaded area represents the pointwise mean plus and minus two time the standard deviation for each input value.

To illustrate distribution over functions using GP, we can draw samples from the function f and shows the value of f at a finite number of locations z_* (Figure 3.3, Prior). The prior in Figure 3.3 can be used for Bayesian inference. In case of modelling real situations, we have only access to noisy observation, thereof $y = f(z) + \epsilon$. Then, the prior becomes:

$$\text{Cov}(y) = K(z, z) + \sigma_\epsilon^2 I. \quad (3.6)$$

The joint distribution for unseen data f_* , and the training output y , according to the prior is:

$$\begin{pmatrix} y \\ f_* \end{pmatrix} \sim N \left(0, \begin{bmatrix} K(z, z) + \sigma_\epsilon^2 I & K(z, z_*) \\ K(z_*, z) & K(z_*, z_*) \end{bmatrix} \right). \quad (3.7)$$

⁴ An introduction to RKHS can be found at the course: Statistical Learning Theory and Applications at MIT University (USA) (Rosasco, 2015)

The uncertainty in the estimate of unseen target f_* , is summarized by the predictive distribution:

$$P(f_+|z_*, z, y) = N(m_*, s_*), \quad (3.8)$$

where m_* is the expectation of the distribution:

$$m_* := \mathbf{E}[f_*] = K(z_*, z)K_y^{-1}y, \quad (3.9)$$

and s_* , the variance, captures the uncertainty of the model:

$$s_* := \text{cov}[f_*] = K(z_*, z_*) - K(z_*, z)K_y^{-1}K(z, z_*), \quad (3.10)$$

Thus, we can update the prior to the posterior in the light of the training data and draw function from the posterior, as in Figure 3.3, and compute the predictive distribution for noisy test data y_* . Usually in a Bayesian paradigm the goal would be to assess how likely is a certain prediction of unseen future data, given the data we actually observed. Differently here, we want to estimate how well a target time series can be reconstructed in term of the covariate of another time series (Schumacher et al., 2015).

3.4 Statistical model and Estimation of delays

3.4.1 Statistical model

DDIFTool allows to model the function F (see Figure 3.2) via both nonparametric and parametric regression, using a Gaussian processes formalization. To this end the model used is a Volterra series, with the polynomial form:

$$F(x) = h_0 + \sum_{k_1=0}^{d-1} h_1(k_1)x_{k_1} + \sum_{k_1=0}^{d-1} \sum_{k_2=0}^{d-1} h_2(k_1, k_2)x_{k_1}x_{k_2} + \dots, \quad (3.11)$$

Where $x = (x_0, \dots, x_{d-1})^T \in X \subset \mathbb{R}^d$. With this formalization the user can decided to truncate the series at particular order n ($h_i = 0$ for $i > n$) or to use the full expansion n . In case of the full Volterra series expansion, regression is performed in RKHS, implemented with the function *empirical_GPR.m*, whereas with the parametric model, regression is achieved by the *empirical_map.m* function. In this first case h_i are coefficients, which are subject to uncertainty. In principle the nonparametric model should be employed but, as pointed in Schumacher et al. (2015), the possibility to switch to the parametric one allows to control the complexity of the model. This procedure can help avoiding serious overfitting, in case relevant information is not

optimally represented in the data. Thus, the statistical model is formalized as:

$$y_j = F(x_i, \dots, x_{i+d-1}, w_0, \dots, w_h) + \epsilon_j, \quad (3.12)$$

$$y_j = F(z_j + \epsilon_j). \quad (3.13)$$

Where $z_j \in \mathbb{R}^{d+h}$, $\epsilon_j \sim \mathcal{N}(0, \sigma_\epsilon^2)$, and $w_k \sim \mathcal{N}(0, \sigma_w^2)$. ϵ represents the measurement noise of the target time series, while w represents the uncertainty from the hidden stochastic drivers and covariate measurements noise.

3.4.2 Estimation of interaction delays

DDIFTool estimates the interaction delay τ via the function *DDIFT_delays_analysis.m* (see section 4.4. Core functions). Observing once more Figure 3.1a, if the information flow is delayed by τ from the driver to the driven system, the latter will incorporate at time t the state information of the driver from time $t - \tau$. Obviously we do not know the true τ , so the model is fitted multiple times with a set of candidate delays. DDIFTool will loop over each candidate and try to reconstruct the past state of the driver y at a *reconstructible area* (see Subsection 4.5.1). The target sample y_{j^*} , where $j^* = \operatorname{argmax}_j j \leq t - \tau$, is the earliest sample reconstructible, since it is the last value in the sample program P (see Section 3.2, Takens (2002) and Schumacher et al. (2015)). The j^* defines the onset of the *reconstructible area*.

3.4.3 Inference by Leave-One-Out (LOO)

The goal is to determine how well a target time-series (driver system) can be reconstructed from a second covariate time series (driven system). The criterion to evaluate the goodness of the reconstruction is the variance; where the lowest variance should be observed for the delay candidate within the *reconstructible area*. Thus, the inference is in two steps:

- 1 LOO point estimator $\hat{y}_j = m_{\setminus(j)}$ conditional on the data set $D \setminus \{x_j, y_j\}$ where j th is removed. Where m is the expected value of the predictive distribution (see Eq. 3.9):

$$m_* := \mathbf{E}[y_*] = K(z_*, z) K_y^{-1} y, \quad (3.14)$$

It minimizes the expected squared-error loss to target time series y_* (Berger, 1985).

- 2 As measure of performance: Normalized root-mean-squared error (NRMSE), as LOO cross-validation (LOO_CV) estimator of the variance $\hat{y} \in \mathbb{R}^D$ of the target time series y :

$$NRMSE(x, y) = \sqrt{\frac{\sum_{i=1}^D (x_i - y_i)^2}{D\sigma_y^2}}. \quad (3.15)$$

Where σ_y is the standard deviation of the target time series. This normalization assumes stationarity. In case of non-stationarity of the time-series, the NRMSE is distorted. Thus, might be necessary to perform the causal analysis in a restricted window of interest (see Schumacher et al. (2015)). The upper bound is 1.0 (see Subsection 3.4.6) and it correspond to a reconstruction indicating no interesting functional dependencies (Schumacher et al., 2015).

DDIFTool realizes the inference procedure within the functions *empirical_GPR.m* and *empirical_map.m*. In the Subsection 4.4.2, it is possible to see the output of these functions.

3.4.4 Confidence intervals

Confidence intervals for the NRMSE indicate the uncertainty of the statistical model. DDIFTool quantifies the variability of the NRMSE, resampling the predictive process as $\hat{y}^{(b)} \in \mathbb{R}^D$. To yield a distribution NRMSE ($\hat{y}, \hat{y}^{(b)}$). DDIFTool employs bootstrapping of residuals (*compute_bootstrap.m*), generating $B = 50,000$ sample sets $\hat{y}^{(b)}$. Confidence intervals are computed from 99 percentiles.

3.4.5 Estimating hyperparameters

The hyperparameters $\sigma_\epsilon^2, \hat{\sigma}'$ estimation is achieved by the approximation of the predictive distribution in a leave-one-out-cross-validation (Sundararajan and Keerthi, 2001). This predictive distribution is used as the objective function to find the hyperparameter (*estHyperparam_loo.m*). Schumacher et al. (2015) provides a comprehensive description of the derivation of $\hat{\sigma}'$ from the noisy covariates w and why it can be treated as a single parameter and estimated accordingly.

3.4.6 Reconstruction Error Graph (REG)

The Reconstruction error graph (REG) is obtained plotting NRMSE against candidate delays and provides the base to estimate the true delay τ .

DDIFTool implement an analytic method to have an automatic detection of the true delay τ detecting the onset of reconstructibility, y_{j*} , in the REG. DDIFTool interpolates the NRMSE by cubic spline function (*spineFit.m*), the resulting smooth function is $I(t)$ with $t \geq 0$. The onset of reconstructibility is determined with the derivative of the function $I(t)$ and its global minimum (left boundary of the area of confidence) $a_0 = \operatorname{argmin}_t \frac{\partial I}{\partial t}$. The right corner a_1 (right boundary of area of confidence) detection is achieved by the Wang and Brady algorithm (Wang and Brady, 1995), finding a point where the curvature relaxed $\frac{\partial I}{\partial t} = 0$ (*corner_detection.m*). The true delay τ should reside inside this interval $[a_0, a_1]$ and is obtained as: $\tau = \frac{a_0 + a_1}{2}$. The description of the REG plot can be found in Subsection 4.5.1 or in Schumacher et al. (2015).

3.4.7 Model selection

In the model selection process, the model parameters: the reconstruction dimension d and the model order n , have to be determined by the user. Takens (1981) showed that it is possible to reconstruct the shallow version of any f -dimensional manifold if $d > 2f$. So, mathematically it does not matter whether one uses the minimum embedding dimension or greater value of d , since once $d > 2f$ holds, the geometry of the attractor is unfolded (Kennel et al., 1992). For the purpose of the method here, setting d to a large value reduces the amount of points for inference (the reconstruction vector Rec_d would span a larger area). Furthermore, increasing the parameter d does not always improve the reconstructability and might lead to a decreasing of global minimum in the REG (Schumacher et al., 2015). Possible explanation is either enough geometrical information are captured or additional information is not accessible even with coarser reconstruction.

Finally with regard to d , DDIFTool has an additional functionality (*embed_cv2.m*) for the estimation of the embedding dimension d , implemented in *DDIFT_prepare.m*. During the pre-processing step the user can perform an explorative analysis between a pair of time-series to determine what could be a meaningful value d (see Subsection 4.4.1). Nevertheless, the final selection should be made accordingly to the resulting REG and the pre-processing step for embedding dimension is considered explorative. Regarding the model order n , models of order $n = 5$ behave similarly to the nonparametric models and usually overfitting is already evident at $n = 3$ Schumacher et al. (2015). It is important to keep in mind that models with order $n > 3$ require extensive RAM. As standard routine we advise to test models with order $n = 1, 2, 3, \infty$ and select the best model based onto the resulting REG. Where the main criteria to consider should be: the steepness and the depth of the sink, and

that the model does not suffer from overfitting.

3.4.8 Why does the time-series need to be sampled down?

The time-series analysis might need different pre-processing steps to be optimal. To this end, *DDIFT_prepare.m* incorporates a downsampling procedure to estimate a relevant time scale of the paired time-series, adequate to the statistical model. A common procedure to estimate the relevant time scale is to compute the auto-mutual information. This pre-processing step is implemented in *DDIFT_prepare.m* with the subfunction *mutual_subsampling.m* (see Subsection 4.4.1). In case of high sample resolution the difference between neighbouring sample is small and subsequent samples are highly correlated. Having an oversample time-series can lead to overfitting and reconstructing the target time series y on a short time scale might not reflect functional dependencies (Schumacher et al., 2015). In general the Rec_d needs to span a relevant part of the time-series otherwise the geometry of the attractor is not unfolded. When dealing with two time-series that evolve on different time-scale introducing additional lags in the reconstruction vector can be beneficial (see *cfg.es* in Section 4.2) or another solution is to increase d .

Chapter 4

Analysis Routine

4.1 Overview

This section provides a description of the structure of main analysis strategies. First, we will give a characterization of: the inputs parameter required to use DDIFTool, the algorithms needed to estimate the parameters and a general overview of the workflow to perform causal analysis on time series (pre-processing, delay analysis and plotting). Second, the main functions and additional functionalities will be described in detail, allowing the user to achieve a complete understanding of the method.

4.2 Input data and configuration parameters

The input data format has to meet with the Fieldtrip raw data format, therefore is a MATLAB structure that contains the fields *trial*, *time*, *label* and *fsample*. The fields *trial*, *time* and *label* have to be a cell arrays. *Trial* and *time* have the length of the number of trials where each cell of the field *trial* is composed of a matrix: (number of channel x number of samples), for each trial. Each cell of the field *time* contains a vector (1 x number of samples), time indices in seconds, for each trial. The field *label* includes the channel names as a string (e.g. 'channel001') and *fsample* is a scalar value of the sampling rate in Hertz (e.g. 1000). As mentioned above, this structure is the same as the Fieldtrip raw data format, no changes are required in case the data structure comes already from it. The DDIFTool uses the Fieldtrip function *ft_checkdata.m* to ensure correctness of the input data. Input data in a FieldTrip raw data structure:

Table 4.1: Input data in DDIFTool: Fields in a FieldTrip raw data structure.

Field name	Dimension	Input type	Unit	Description
<i>trial</i>	{n.trial} (n.chan x n.sample)	Cell array of double array		Data for each trial
<i>time</i>	{n.trial} (1xn.sample)	Cell array of double array	Second	Time indices for each trial
<i>label</i>	{n.chan x1}	Cell of string		Labels of channels
<i>fsample</i>	Scalar	Integer number	Herzt	Sampling rate

The configuration parameters are essential to work with the DDIFTool functions. These parameters have to be defined in a MATLAB structure called *cfg*. Within this structure the user can define the set of input parameters required to specify the field name, whereas configuration field left empty by the user can have a default value. The table below (Table 4.2) provides a brief description of parameters needed to work with the function *DDIFT_prepare.m*.

Table 4.2: Parameter for the configuration structure of the function *DDIFT_prepare.m*.

Fields of cfg	Default value	Input type	Description
<i>delay</i>		vector with integer number	[start end step size] e.g. [0 60 5]. It defines the delay grid of REG. The spacing should agree with the system time scale.
<i>order</i>		vector with integer number or string	Model order and model type. Integer numbers are required to work with the parametric model (1,2,3,...), ' <i>inf</i> ' for the non-parametric model.
<i>es</i>		vector with integer number or string	[dimension, delay]. Specify dimension and delay of embedding. If ' <i>compute</i> ' <i>cfg.param</i> and <i>cfg.tau</i> have to be provided by the user.
<i>param</i>		vector with integer number	[min max]. Range of embedding dimensions to test.
<i>tau</i>	1	vector with integer number	[min max] Range of embedding delays to test.
<i>subsampling</i>	1	vector with integer number	[dimension, delay]. Value to downsample the time series. If ' <i>compute</i> ' the user can provide the input from the console once mutual information is performed.
<i>padding</i>	' <i>False</i> '	string	If ' <i>True</i> ' a zero-padding of length $d - 1$ is added at the beginning of each trial.
<i>numvalidate</i>	1/5 length of data	integer	Sample points taken from the end of the time series for validation.
<i>channel</i>		string	Names of channels to analyse .
<i>combination</i>		string	' <i>manual</i> ' to use a specific channel pairs in the analysis or ' <i>All</i> ' for all possible channels combinations. If <i>cfg.combination</i> is set to ' <i>manual</i> ', <i>cfg.channel</i> have to be a n. channels x 2 cell array.

<i>time</i>		vector with integer number	[start end]. First and last point of the time vector of interest (in seconds) for analysis.
<i>sampling</i>	<code>fsample</code>	integer	Sampling rate of time series 1/fsample.
<i>testing</i>	<code>'True'</code>	string	<code>'True'</code> to test pairs channel time series in both directions or <code>'False'</code> to test only direction as entered in 'channel' by the user.
<i>display</i>	<code>'True'</code>	string	<code>'True'</code> or <code>'False'</code> for plotting results.
<i>verbosity</i>	<code>'info_m'</code>	string	Output of console.
<i>parallel</i>	<code>'False'</code>	string	Set to <code>'True'</code> for parallel computing. It takes the default number of workers.
<i>fold_name</i>		string	Specify a folder name to save data.

Remark on `cfg.es` and `cfg.tau`: The configuration parameter `cfg.es` takes two inputs: dimension of embedding d and delay of embedding. A clarification is needed for the latter since the term delay, here, do not indicate the candidate delay to estimate the true τ but the lags in the reconstruction vector Rec_d (see Section 3.2), same holds true for the `cfg.tau`, in case the user wants to use the explorative embedding estimation routine. The delay parameter in the embedding space is related to the underlying timescale or measurement noise in the system (see Stark et al. (2003)). In case of an optimal subsampling this can be set to 1. In situations when the timescale of the covariate time-series and the target time-series diverge significantly, it might be useful to consider the delay parameter for embedding (see Schumacher et al. (2015)).

4.3 Workflow

Below, a schematic representation of DDIFTool and example scripts of how to use the functions in MATLAB. In the next paragraphs (4.4 Core functions, 4.5 Plotting Result) a more detailed description of the major functions is offered as well as of additional functionalities, that might be useful to optimize the analysis.

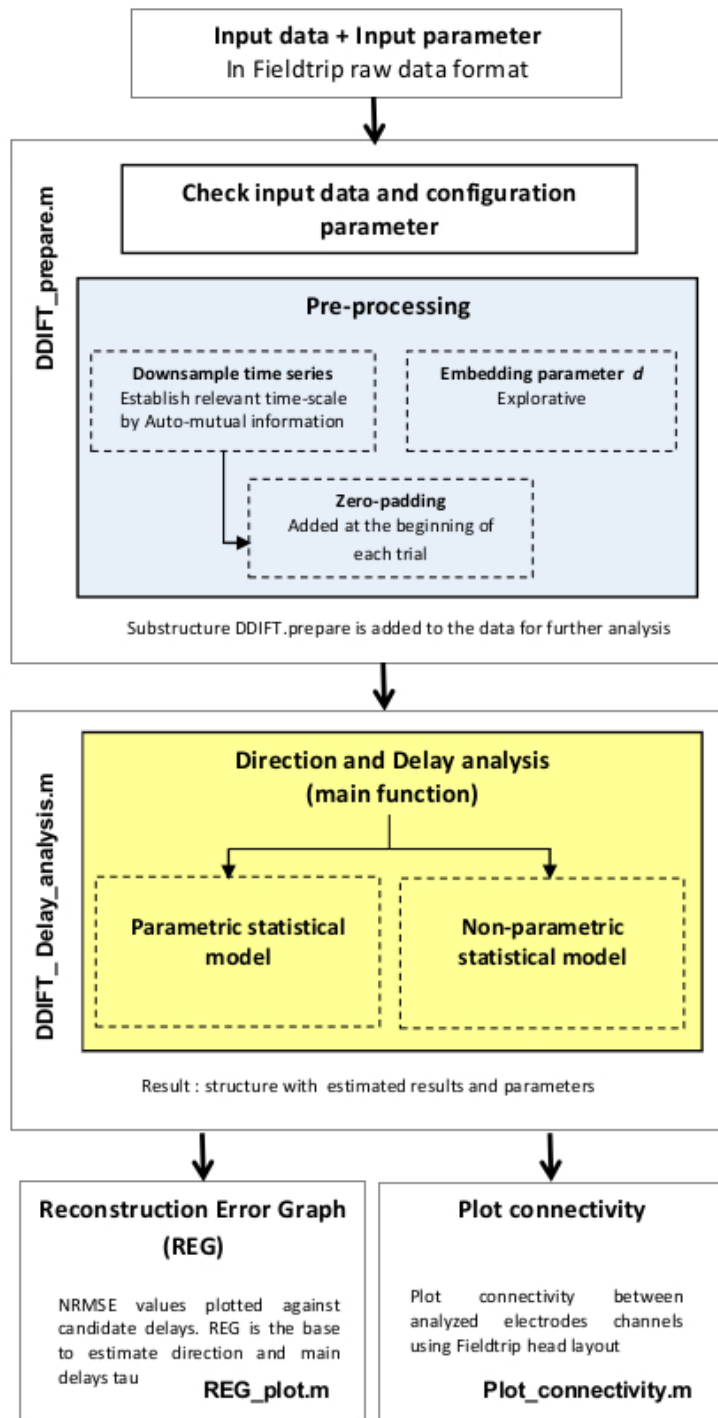


Figure 4.1: Workflow of the toolbox. Overview of DDIFTTool

The first step is to prepare the *data* and *cfg* structures for the DDIFTTool analysis. This is done using the function *DDIFT_prepare.m*.

Listing 4.1: *DDIFT_prepare.m*

```

1  cfg=struct();
2  cfg.delay=[0 40 2];
3  cfg.order=2;
4  cfg.es=[20 1];
5  cfg.subsampling=5;
6  cfg.Zero-padding='True';
7  cfg.numvalidate=30;
8  cfg.channel={'VirtualChannel_pc3','VirtualChannel_pc2'};
9  cfg.combination='manual';
10 cfg.display='True';
11 cfg.testing='True';
12 cfg.time=[0 0.97];
13 cfg.sampling=0.001; % 1000 Hz
14 cfg.verbosity='info_m';
15
16 % Prepare data for analysis
17 data_prepare= DDIFT_prepare(cfg,data);

```

Secondly, the output of *DDIFT_prepare.m* have to be entered to the main analysis function *DDIFT_delays_analys.m*.

Listing 4.2: *DDIFT_delays_analys.m*

```

1 % Data analysis
2 [Result]= DDIFT_delays_analysis(data_prepare);

```

Finally, the user can plot the results. REG is automatically showed if the *cfg.display* is set to 'True'. Nevertheless, REG can be successively plotted using *REG_plot.m*. In addition, the user can show interaction and delay between channels using *plot_connectivity.m*.

Listing 4.3: *plot_connectivity.m*, *REG_plot.m*

```

1 %Plotting Result
2 REG_plot(data_prepare,Result);
3 plot_connectivity(Result,data);

```

4.4 Core Functions

4.4.1 *DDIFT_prepare.m*

Description: This function checks the correctness of the data structure (fields names and matrix form) and inputs parameters. It selects channels indicated by the user and the time of interest. Furthermore, three pre-processing step might be needed to optimize the analysis, namely: embedding parameters, downsampled time-series estimation and zero-padding. The relevance of these pre-processing steps are discussed in Chapter 3 and below.

- 1 Estimation of embedding parameter d and τ .
 - 2 DDIFTTool checks if the time-series needs to be down-sampled by mutual information.
 - 3 Zero-padding time series. The user can decide to add a zero-padding of length $d - 1$ (where d is the embedding dimension) to the beginning of each trial time series.
- 1) The first pre-processing step is concerned with the embedding parameter estimation d . The proposed method does not need this routine and the REG already provides enough information to select the appropriate d . As stated in 3.1 Embedding, mathematically there is no upper bound for the size of d (Takens, 1981). Once the size of d is $d > 2f$, Takens's theorem is satisfied. Here, increasing the size of d might give a more pronounced minimum in the REG (Schumacher et al., 2015) and in principle the method works also with high dimensions. On the other hand, there will always be a trade off between size of d and the amount of data available for inference (see Schumacher et al. (2015)). Besides this clarification, the aforesaid routine can be used as first insight on the parameter d and help in the selection process, giving a coarse idea of a meaningful d size. The embedding estimation uses a predictive modelling approach for computing embedding space, from a range of different embedding dimension and delays embedding (τ). It returns the best d and τ for each time series pairs.

```
1 cfg.es='compute'  
2 cfg.param=[4 30]; % range of testing dimension  
3 cfg.tau=[1 2]; % range of testing delays
```

- 2) The second pre-processing step is to check if the time-series need to be downsample at relevant time scale. To this end,

mutual information is computed if the `cfg.subsampling` is set to `'compute'` (`cfg.subsampling='compute'`). The implementation of mutual information follows the Non-Parametric Entropy Estimation Toolbox (Ver Steeg, 2000) and it uses OpenTSTOOL functions to estimate k-nearest neighbors distances efficiently. For each channel selected the DDIFTool gives back a plot with the mutual information. Then, the user can insert the downsample parameter by an integer number. In general a good heuristic is to jointly downsampling the time series until the mutual information between neighbouring samples is less than 1 (Schumacher et al., 2015). DDIFTool reports the first local minimum of the mutual information (Fraser and Swinney, 1986) together with the estimated mutual information plot (see Figure 4.2). This step is highly recommended, especially with neural data, due to the high sampling resolution.

```
1  cfg.subsampling='compute';
```

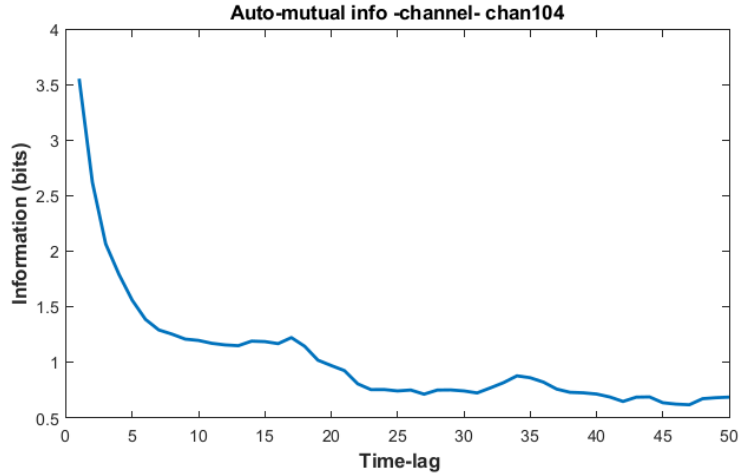


Figure 4.2: Auto-mutual information. Example of a mutual information plot for the selected channel.

3) The final step is zero-padding. A zero-padding of length $d - 1$ is attached at the beginning of each trial time series, if `cfg.padding` is set to `'yes'` (`cfg.padding='yes'`). To explain the importance of this pre-processing step in a certain context, we will describe an example taken from a real LFP data analysis in Schumacher et al. (2015). The LFP time series was recorded at a rate of 1024 Hz and it needed to be sampled down by a step size of 15 to achieve a sufficient level for the statistical model, as expected.

Now, if the Rec_d vector is set to $d = 20$, Rec_d spans an area of more than 300 ms and the first 300 ms after stimulus onset will be lost for the analysis (Schumacher et al., 2015). To improve upon this situation, Schumacher et al. (2015) used zero-padding so that the first i samples (where $i > d$) are now included in the analysis. We recommended this procedure in similar situations, when losing information after stimulus onset is unacceptable.

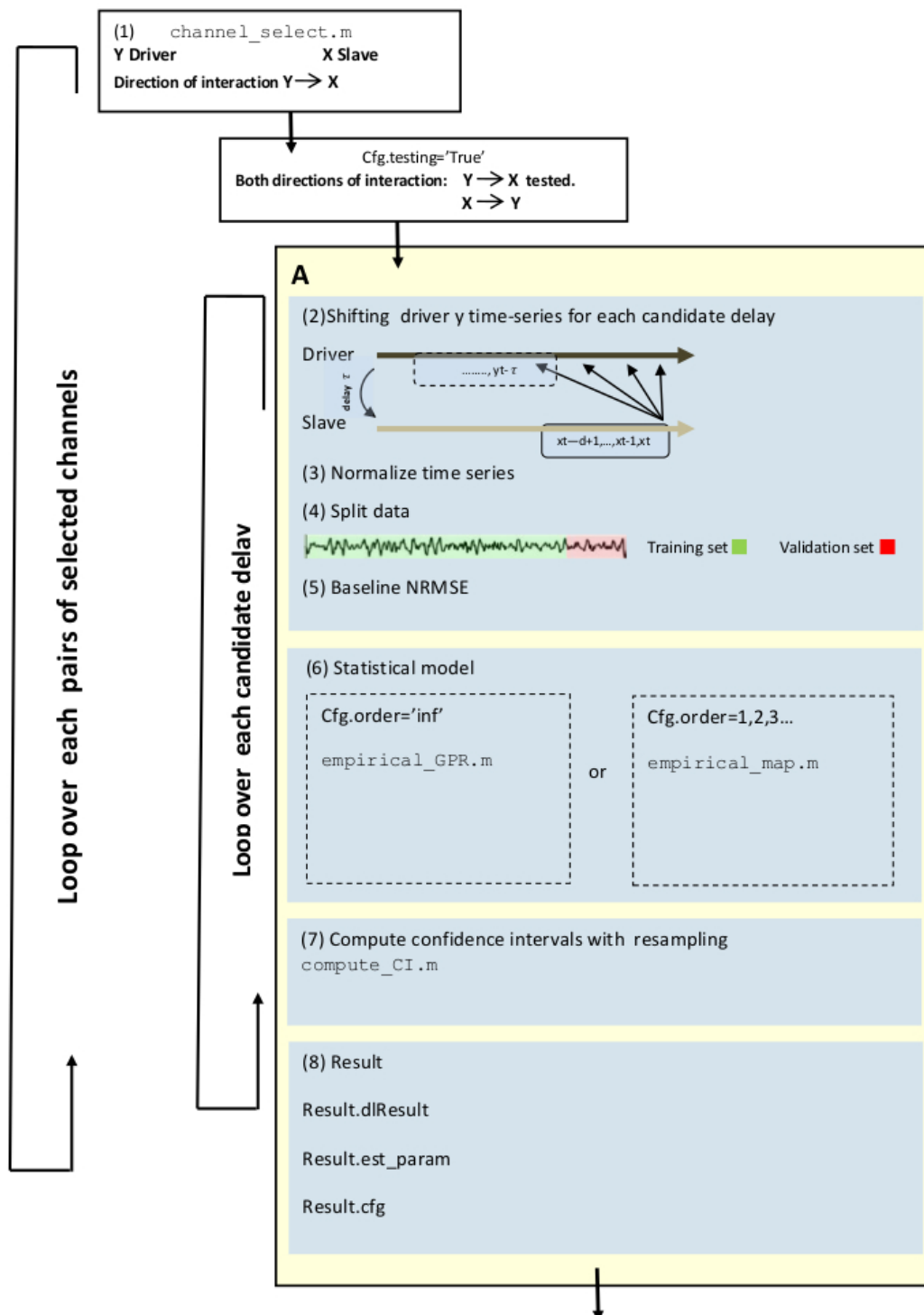
Output Description: The function *DDIFT_prepare.m* returns a substructure *DDIFT_prepare* that is attached to the input data structure and will be used for the analysis.

Table 4.3: Prepared data in the structure *DDIFT_prepare*.

Field name	Dimension	Data type	Unit	Description
<i>pre_ch</i>	1 x n.chan	struct		Selected channels from the user are kept for subsequent analysis
<i>timeindices</i>	[1x2]	integer	samples	Indices in samples of the time of interest selected by the user .
<i>es</i>	n.chan x 4	string		3 rd 4 th columns, estimated d and τ by embedding routine in both directions. 1 st 2 nd columns channel names.
<i>subsampling</i>	n.chan x 2	string/integer		1 st column channel name, 2 nd column estimated subsampling values by mutual information.

4.4.2 *DDIFT_delays_analysis.m*

Description: This function performs the major analysis routine. For any defined pairs of channels, it loops through each candidate delay to fit the statistical model and at each iteration the target times-series is reconstructed (Figure 4.3, Panel A). Finally, it generates the REG plot with the estimated delay τ and the direction of interaction (Figure 4.3, Panel B).



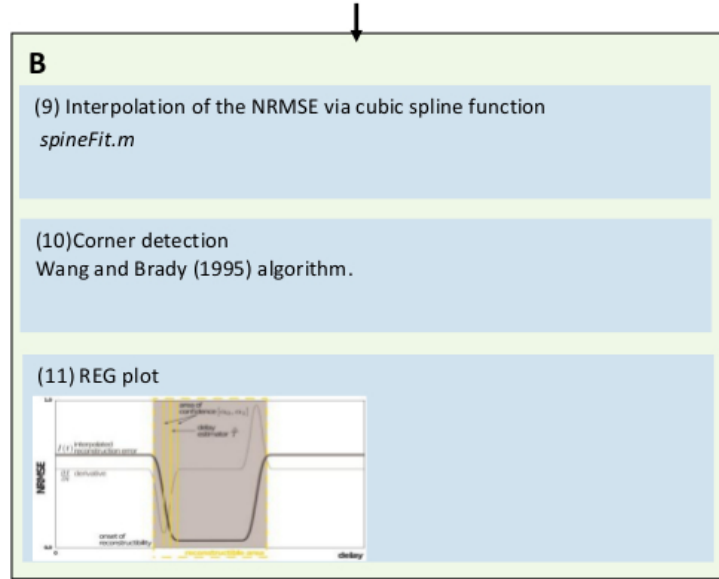


Figure 4.3: Overview of the main steps of the function *DDIFT_delays_analysis.m*. Panel A includes all steps related to the main analysis and Panel B includes steps for the generation of the REG.

Following a description of the different steps :

- 1 It checks the correctness of the input data and extracts the selected channels by the user. The channel pairs are analyzed as provided by the user (`cfg.channel='manual'`) or all possible combination (`cfg.channel='All'`) are tested.
- 2 If `cfg.testing='True'` the function computes both possible direction of interaction either $X \rightarrow Y$ or $Y \rightarrow X$.
- 3 Target time-series y is shifted for a candidate set of negative delays.
- 4 Baseline NRMSE on training data set (minimum shifted NRMSE between time series).
- 5 Time-series are normalized by *normalize.m* or *ztrasf.m* in case of kernel regression to facilitate positive-definite kernel matrices.
- 6 The reconstruction and inference procedure are carried with one of the selected statistical model.
- 7 Estimation of confidence intervals and parametric bootstrapping of residuals. It returns confidence intervals from 99 percentiles.

- 8 Results are added in three substructures: *.dlResult*, *.est_param* and *.cfg*.
- 9 Interpolation of NRMSE via cubic spline.
- 10 Automatic detection of the confidence area and true delay τ .
- 11 Plot of REG as final output of the delay analysis.

Output Description:*Result.dlResult*

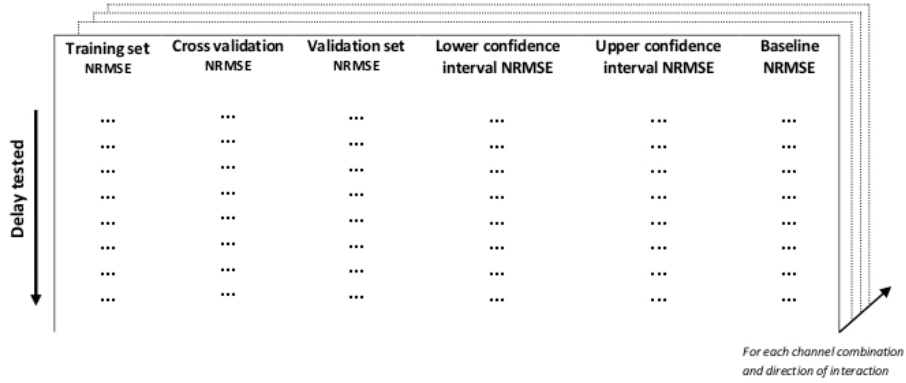


Figure 4.4: A representation of the matrix *dlResult*. It is composed of n rows (number of candidates delay tested) and 6 columns (NRMSE values estimated). The matrix *dlResult* is generated for each channel combination tested and for both direction of interactions (*cfg.testing='True'*).

Output Description:*Result.est_param*

Table 4.4: Results provided in the structure *Result.est_param*.

Field name	Dimension	Data type	Unit	Description
<i>delay</i>				Delay is a structure. It contains the parameters estimated for each candidate delay τ .
<i>param_ci</i>	[1x2]	integer	ms	Upper and lower confidence intervals for the estimated delay τ .
<i>param_final_delay</i>	single value	integer	ms	The estimated delay τ .

<i>param_pNrmse_t</i>	Interpolation of NRMSE training.
<i>param_pNrmse_cv</i>	Interpolation of NRMSE cross-validation.
<i>param_pNrmse_lower</i>	Interpolation of NRMSE lower confidence interval.
<i>param_pNrmse_upper</i>	Interpolation of NRMSE upper confidence interval.

Output Description: *Result.cfg*

Table 4.5: Results provided in the structure *Result.cfg*.

Field name	Dimension	Data type	Unit	Description
<i>channelcombination</i>	{no.channel combination x 2}	strings		Cell array with channel labels that specifies the channel combinations analyzed.
<i>delay</i>	[1x2]	integer	ms	Range of tested delays in ms.
<i>iteration</i>	single value	integer		It indicates how many channels pairs were analysed.
<i>test</i>	single value	integer		It indicates if one or both directions of interaction were tested.

Result.cfg keeps basic information of the analysis and it is used in the functions: *REG_plot.m* (see Subsection 4.5.1) and *plot_connectivity.m* (see Subsection 4.5.2).

4.5 Plotting results

4.5.1 Reconstruction Error Graph (*REG_plot.m*)

```
1 % Plotting Result
2 REG_plot(data_prepare, Result)
```

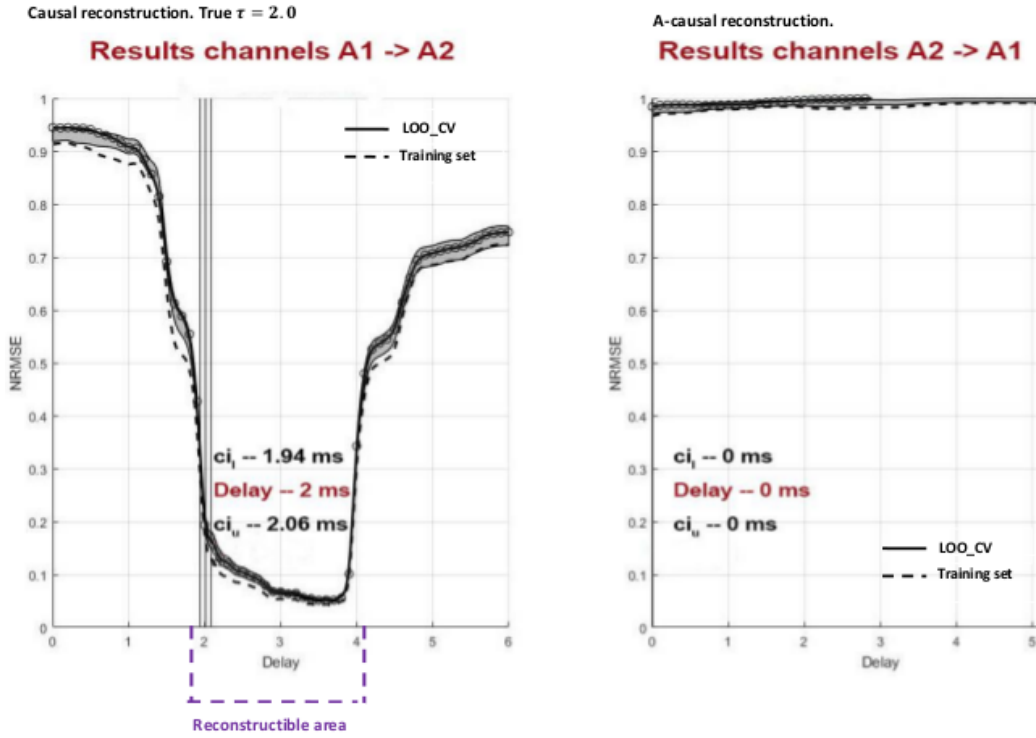


Figure 4.5: Example of a Reconstruction error graph (REG), with both direction of interaction, between time series A1 A2, tested.

The figure 4.5 shows an example of the REG plot. As described in Background (Chapter 3), the interpolated NRMSE is plotted against the candidates delays. The REG is highly informative about the direction of interaction and the true delay (Delay - 2.0, in red), with the confidence area that incorporates the uncertainty of the delay estimation $[ci_l \ ci_u]$. The sink of the NRMSE is within the reconstructible area $[2-4]$, where the indicator of the true τ is the onset of the reconstructible area. In case of a-causal analysis $A2 \rightarrow A1$ no reconstruction is possible at any candidate delays, and NRMSE does not deviate from 1.0. The function *REG_plot.m* allows to plot the

REG of all channels analysis performed, at later stage, or select a specific channel pairs of interest. Thus, the user can always access to the REG plot with the *Result* structure, output of the *DDIFT_delays_analysis.m* function (see: Listing 4.2 and Listing 4.3 in Section 4.3).

4.5.2 *plot_connectivity.m*

```
1 % Plotting Result
2 plot_connectivity(Result,data)
```

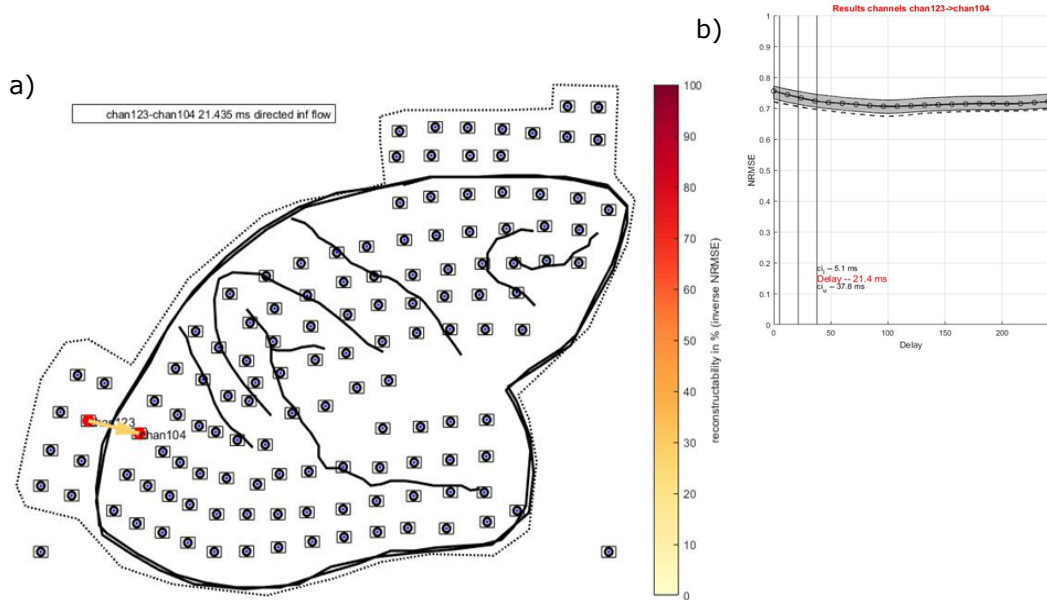


Figure 4.6: Output plot of *plot_connectivity.m*. (a) Example of an head layout of a macaque monkey with the estimated interaction between the analyzed channels (output of the function). (b) REG plot of channels analysis. The detected interaction is visualized as an arrow in the head plot (a).


The Figure 4.6 illustrates an example of a connectivity plot between channels. The image shows the interaction between channels pairs detected during the causal analysis (4.6b), where the colour of the arrow indicates the percentage

of reconstructability (inverse NRMSE). Furthermore, the legend reports the channels names (highlight with red circle in the image), the estimated delay in ms and the type of interaction ('**directed information flow**'). The function *plot_connectivity.m* takes two inputs: *Result* and *data*. The structure *Result* is the output of *DDIFT_delays_analysis.m* (see) and *data* is the initial structure in the Fieldtrip raw format. The user needs to add additional fields to the structure *Result.cfg* to perform the plotting, as it is shown in table 4.6.

Table 4.6: Parameter for the configuration structure in order to use *plot_connectivity.m*.

Field name	Default value	Input type	Description
<i>layout</i>			Structure from Fieldtrip routine (see FieldTrip reference for <i>prepare_layout</i>) or name of an ascii file *.lay.
<i>plothead</i>	0	integer	0 to use FieldTrip function with related layout, 1 to draw head layout on-line.
<i>export</i>	0	integer	1 to export a table with summary of results, 0 no export.
<i>showlabels</i>	'False'	integer	'False' to plot only analyzed channel labels. If 'True' all channel labels are displayed.
<i>emarker</i>	'o'	string	Marker symbol.
<i>ecolor</i>	[0 0 0]	integer	Marker color in rgb MATLAB format.
<i>emarkersize</i>	2	integer	Marker size.
<i>hlmarker</i>	'o'	string	Highlight marker.
<i>hlcolor</i>	[1 0 0]	integer	Highlight marker color in rgb MATLAB format.
<i>hlmarkersize</i>	4	integer	Highlight marker size.
<i>hlinewidth</i>	2	integer	Linewidth of the head plot.

The function can handle every head layout coming from Fieldtrip and automatically plot arrows for different types of interaction such as: **'directed information flow'** and **'bidirectional flow'** (in case a delay estimate is obtained in both analysis direction). Linear or putative synchronization interactions (Schumacher et al., 2015) are not automatically reported in the plot. Thus, we recommend to use *plot_connectivity.m* as visualization of the pairwise channels analysis, while the REG plots should be considered as diagnostic and always employed to critically assess the performed analysis.

 1x6 [table](#)

	1	2	3	4	5	6
	Channel_A	Channel_B	delay	Type_Interaction_A_B	Type_Interaction_B_A	reconstructibility
1	'chan123'	'chan104'	21.4350	'directed inf flow'	'no interaction'	'28%'

Figure 4.7: Example of table with summary of results.

Chapter 5

Simulation and Analysis examples

5.1 Overview

This section will provide different simulations to present DDIFTool and example scripts to analyze neural data in combination with FieldTrip. The goal of this section is therefore twofold.

Firstly, we will simulate data to mimic electrophysiological recordings, not to illustrate the validity of the implemented method, already confirmed elsewhere (see Schumacher et al. (2015)), but its robustness under different cases relevant in neuroscience. Secondly, we will demonstrate the applicability of DDIFTool to reconstruct source activity of two interacting brain regions with simulated EEG data.

5.2 Simulated data: Linear mixing and Noise

To establish effective connectivity and information flow between areas in the brain the proposed method has to be robust against noise, at both source level and measurements recordings, as well as to linear cross-talk between signals, common issues with EEG or MEG due to volume conduction problem. Furthermore, methods that want to recover causal link in network need to overcome spurious interactions that might arise in bivariate analysis.

As showed by Nolte et al. (2008) and stated in Bastos and Schoffelen (2015), instantaneous linear mixing of the signals is always present in EEG and MEG data. Among various consequences it can deteriorate signals asymmetries, making the estimation of effective connectivity harder, or it can lead to false interpretation of causal interactions, if the same signal is observed in two measurements with different amount noise (Nolte et al., 2008). In addition, reconstructed neural network obtained from bivariate analysis might be erro-

neous in cases of cascade effect or common drive input with different delays of coupling (Wollstadt et al., 2015). The first may arise in situation of cascades of information "chain" of sources as pointed in Wollstadt et al. (2015), (see Figure 5.1a). The second case is showed in Figure 5.1b, where a spurious interaction can be present between node Y and node K due to the common input driver X, with different coupling delays.

To illustrate the scenario of reduction of signal asymmetries, we firstly generated signals from AR (Autoregressive Processes) with linear or threshold coupling, with different delay of interaction and then we linearly mixed them. This simulation follows the approach adopted in Vicente et al. (2011) and Nolte et al. (2008) for other causality methods: Transfer Entropy (TE) and Phase-slope index (PSI), respectively. The signals X and Y were generated by AR(10) with the following equation:

$$x(t+1) = \sum_{i=0}^9 a_i x(t-i) + \sigma \eta(t). \quad (5.1)$$

Where: the coefficients of a_i are drawn from a normalized Gaussian distribution, the innovation term η account for a Gaussian white noise source and σ regulates the strength of the noise. To simulate a causal relationship between X and Y we added to one process a term related to the past dynamic of the other, giving rise to two unidirectional coupled autoregressive processes (Vicente et al., 2011).

$$y(t) = (y_-) + \begin{cases} \gamma_{lin}(t+\delta) \\ \gamma_{thresh} \frac{1}{1+\exp(b_1+b_2x(t-\delta))} \end{cases} \quad (5.2)$$

Where $D(.)$ represents the past dynamic of y and y_- represents past values of y Vicente et al. (2011). Two types of interaction were tested: linear (lin) and threshold (thresh). The setting of the parameters in the threshold function b_1 and b_2 (0 and 50 respectively) as well as the update of $y(t)$ are the same as in Vicente et al. (2011) to allow comparison with the Transfer Entropy (TE) method. Furthermore, δ represents the delay of interaction. In our simulations δ takes the value of 20 or 50 samples.

Finally the signal pairs X and Y were transformed in two linear instantaneous mixtures X_ϵ Y_ϵ by:

$$X_\epsilon(t) = (1 - \epsilon)X(t) + \epsilon Y(t) \quad (5.3)$$

$$Y_\epsilon(t) = \epsilon X(t) + (1 - \epsilon)Y(t) \quad (5.4)$$

Where ϵ represents the amount of linear mixing. Following Vicente et al. (2011) we investigated different value of $\epsilon = (0.1, 0.2, 0.4)$. Note, that in the

case of $\epsilon = 0.5$ the signals are identical and no interaction can be detected (Vicente et al., 2011). We expected to see the effect of linear mixing in the NRMSE reconstruction since the signals will start to be similar at increasing ϵ , but still a visible sink for values $\epsilon = 0.1, 0.25, 0.4$ should be detected; allowing to correctly identify the direction of interaction and the true delay. Below (Subsection 5.2.1), we report results of the AR(10), threshold coupled, simulations for different ϵ , as an example. Since the results are consistent throughout the diverse generation of the AR processes (linear with delay of 20 and 50 samples and threshold with delay of 20 samples), we report these results in Section A.1.

Finally, we simulated the reconstruction of a simple network in the two cases of cascade effects and common driver input. The networks were generated with interacting signals as AR(10) processes, threshold coupled, with different delays of interaction (Figure 5.1), as before. In the cascade effect situation, the indirect pathway from node X to node K (dot arrow in Figure 5.1a), should show a consistent decreasing in term of NRMSE reconstructability, in line with Ye et al. (2015) (where the resulting Cross-Convergent Map decreases substantially with indirect links), compared to direct links (w_{xy} , w_{yz} , Figure 5.1a). Whereas the expected delay should be detected as a sum of $w_{xy} + w_{yz}$, (see Figure 5.1a). In the common input driver scenario, Wollstadt et al. (2015) pointed the possible detection of a spurious interaction between node K and node Y , with delay estimated as: $w_{xy} - w_{xk}$. We report the results of both test cases in Subsection 5.2.1.

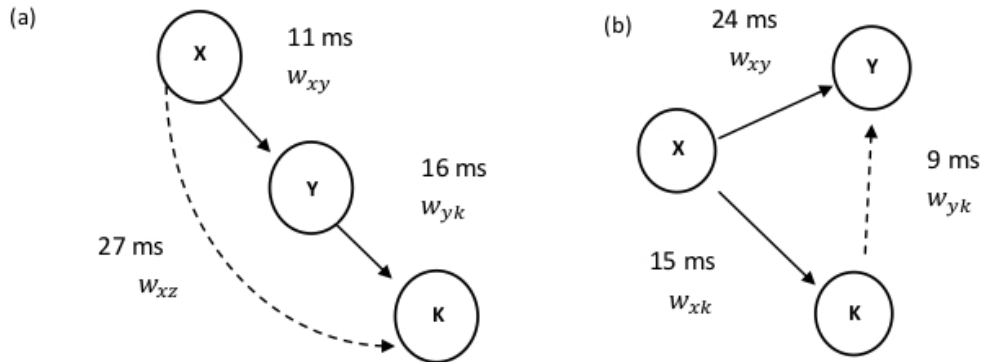


Figure 5.1: Cascade effect and common drive scenarios. (a) A system X causes Y causes K such that indirect causation from X to K occurs (dot arrow). (b) A system X causes Y and K with different coupling delays 24 ms and 15 ms, respectively. A spurious interaction from K to Y might be detected due to the common drive (Wollstadt et al., 2015).

5.2.1 Results

```

1 %Generate AR unidirectional processes
2 [data]= AR_process;
3 %Structure with cfg parameters
4 cfg=struct();
5 cfg.delay=[0 80 2];
6 cfg.order=2;
7 cfg.verbosity='info_m';
8 cfg.es=[9 1];
9 cfg.subsampling=1;
10 cfg.numvalidate=150;
11 cfg.display='True';
12 cfg.testing='True';
13 cfg.channel={'X','Y'};
14 cfg.combination='manual';
15 cfg.time=[0 1.5];
16 cfg.sampling=0.001;
17 %Analysis
18 data_o=DDIFT_prepare(cfg,data);
19 [Result]=DDIFT_delays_analysis(data_o);

```

Instantaneous linear mixing:

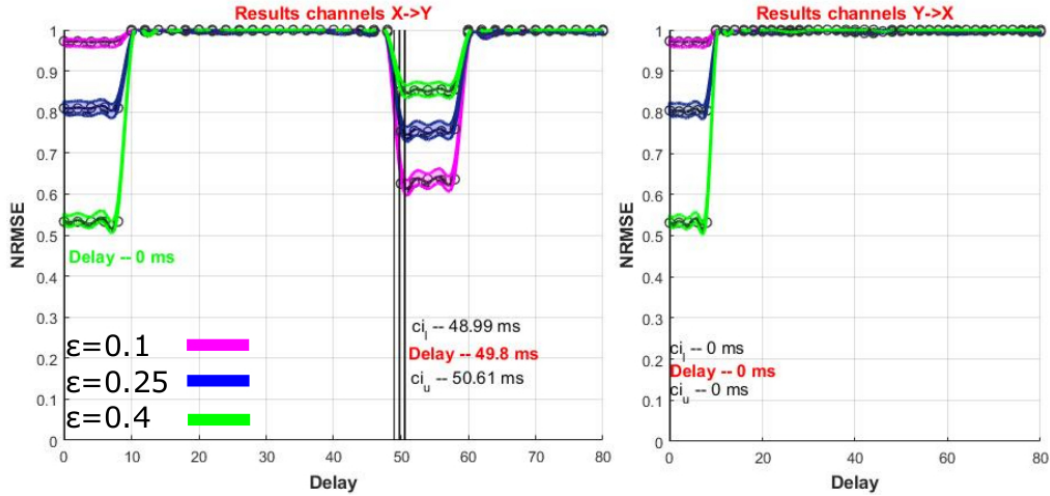


Figure 5.2: Result linear mixing. REG plots for two unidirectionally threshold coupled signals ($X \rightarrow Y$), linearly mixed with different ϵ values. The interaction delay is 50 samples

As expected the effect of the linear mixed is clearly visible in the NMRSE, with signals that become more similar at increasing ϵ values. Interestingly, even if the NRMSE baseline deviates from 1.0 in the acausal reconstruction ($Y \rightarrow X$), the sink provides information about the direction and delay of interaction correctly. Furthermore, the reduction of the signals asymmetries

due to the linear mixing is reflected in the NRMSE reconstruction at increasing ϵ values.

Cascade effect:

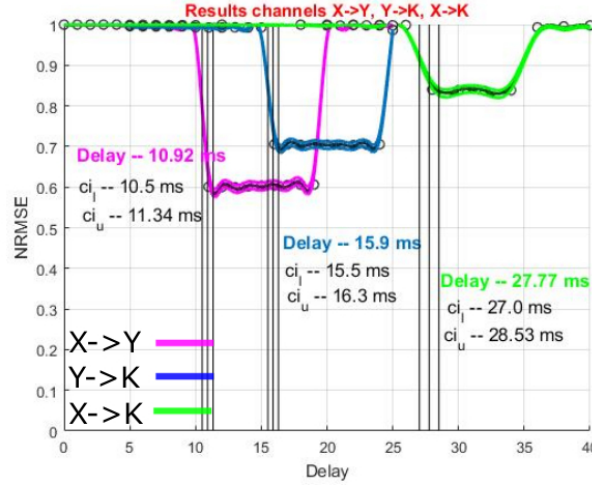


Figure 5.3: Result cascaded effect. REG plots for the cascade effect scenario. A system X causes Y (in purple) and Y causes K (in blue) such that indirect causation from X to K occurs (in green).

In the "cascade effect" simulation the method showed is sensitiveness to direct and indirect causal pathways. As in Ye et al. (2015) (with convergent cross-map analysis), the NRMSE is lower for indirected link of interaction compared to the direct ones. Furthermore, the delays estimation of the indirect link as a sum of the delays of each direct link, is a reliable signature of the "cascade effect" and indeed used also in Wollstadt et al. (2015) to recover such scenario.

Common drive:

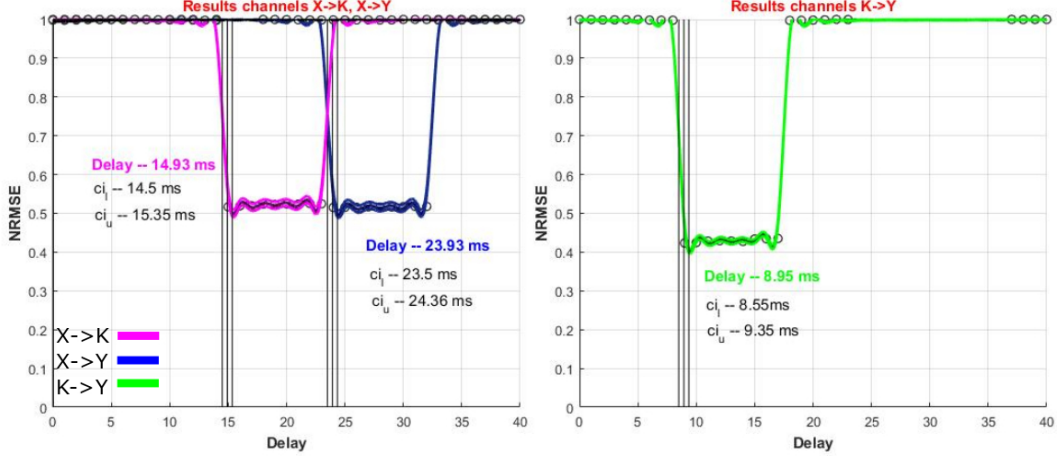


Figure 5.4: Result common drive. REG plots for the common drive scenario. A system X causes K (in purple) and causes Y (in blue) with different coupling delays. A spurious interaction from K to Y is detected due to the common drive (in green).

For the common input scenario, Figure 5.4 shows the spurious interaction detected between nodes K and Y due to the common driver X , that drives both nodes (Y and K) with different coupling delays. As expected the delay $w_{xy} - w_{xk}$ (9 ms) is within the confidence intervals. We advise to use a graph algorithm such as the one proposed in Wollstadt et al. (2015) to detect false causal links in large network of multiple neural sources and we suggest a possible improvement of it (Subsection 5.2.2).

5.2.2 Graph Algorithm

Wollstadt et al. (2015) proposed a graph algorithm to find potentially spurious interactions in analysis of multi-nodes networks based on time-signatures, to improve the validity of the network representation. The algorithm works in combination with TE but it can be used with any analysis method that gives as output a delay estimate. The algorithm is implemented in Matlab and it is part of the open source toolbox TRENTOL (Lindner et al., 2011) with the function *TEgraphanalysis.m*.

We leave a detailed description of the graph algorithm to Wollstadt et al. (2015), here, we present the main idea behind it and the parameter θ that is relevant to incorporate the uncertainty of the delay estimate. Firstly, the algorithm reconstruct a graph of interactions (Figure 5.5b) from a connec-

tivity matrix where every entry represents the estimated delay of interaction between a pair of channels (or sources) from i th rows to j th columns (Figure 5.5a). Connections are weighted with the estimated interaction delay (Figure 5.1 and Figure 5.5b). In addition, the user has to provide a parameter θ to account for measurements noise and imprecision of the TE analysis. Then, iteratively the algorithm searches for alternative paths (assuming a cascade effect, see Figure 5.1a) and in case it finds it, the edge is tagged as spurious. Based on the result of the cascade effect, the algorithm looks for possible common drives scenario identifying graph motif that form acyclic triangles (see Figure 5.1b and Figure 5.5b). As we showed in section 5.2.1 Results spurious interactions have time-signature that can be exploit for identification. Finally, tagged links are removed to yield a more conservative network. As it shown in Figure 5.5b each link is defined by the estimated delay of interaction \pm a paramter θ . Indeed, the algorithm assumes a possible imprecision in the estimated delay, so each link is considered with an additional threshold θ , the same for all links. However, the user does not have any information about the imprecision of the estimated delay a priori. Moreover, high or low values of the threshold θ (in ms) can lead to significant different results in the reconstruction of the analyzed network and analysis of interaction of different nodes might require a particular threshold that differ from others. To this end, we suggest to assign to the parameter θ the value of the confidence interval for each link of interaction, since it is the output of the statistical method implemented in DDIFTool. This step will allow to use the algorithm without any specification of the threshold by the user and additionally it might give a more reliable reconstruction, since each link in the network comes with its own confidence interval (in ms). Note, the REG plot provides already enough information, as it is shown in Figures 5.3 and 5.4, of the underlying interactions, so we suggest the use of such algorithm, for example, in reconstruction of large networks.

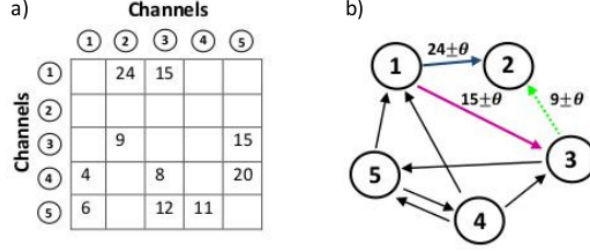


Figure 5.5: Graph analysis. (a) Connectivity matrix between channels, where each entry is the delay estimate from the causality analysis. (b) A simple reconstructed network that exhibits a common drive scenario as presented in section 5.2.1 Results.

5.3 Connectivity analysis from simulated EEG

The brain can be thought as a high dimensional dynamical system, composed of many interacting subsystems (Chicharro and Ledberg, 2012), main with non-linear interactions and in the presence of noise. Thus, understanding information flow from neural time series, is a difficult task (see Chicharro and Ledberg (2012) and Haufe et al. (2013), for a critical assessment of connectivity measures and causal-effect investigations).

New methods that profess of being able of causal analysis should be tested with synthetic data and in scenarios where the ground-truth is known. To this end, Haufe and Ewald (2016) provide a useful benchmark, publicly available online, to validate and assess the performance of the above-mentioned method, with pseudo-EEG data.

The generated pseudo-EEG data have a good percentage of realism and a series of interesting features (a complete and detailed list in Haufe and Ewald (2016)). To cite only few properties: a realistic finite element volume conductor with six tissue types, source location confined to the cortical manifold and projecting electrical current perpendicular to the local surface, brain like background noise with $(1/f)$ spectra, measurements noise and interacting source with time-delayed influences. The reader can find an extensive description of the different simulation and modelling steps (forward model, head model, region of interest ect.) in Haufe and Ewald (2016). Here, the two interacting sources were simulated with: an AR(10), threshold coupling and delay of 100 samples (see Eq. 5.1 and 5.2). Each of these sources were placed randomly to one of the eight regions of interest (ROIs) identical to the eight defined octants of the brain. Furthermore, a base line signal (no-interaction) was simulated with the same AR(10), without coupling, and

used as contrast condition. Below (Subsection 5.3.1), we reported the analysis performed on two interacting areas: Anterior Left Superior (ALS), that drives the Anterior Right Inferior (ARI) (see Haufe and Ewald (2016) for the eight different ROIs). We did not simulate any plausible biological connectivity since we were interested only in demonstrating the ability of the method to infer source interaction from pseudo-EEG recordings.

The source reconstruction was carried out with Fieldtrip toolbox, employing a different head model to avoid performing the reconstruction on the same head model, used to generate the data (see the so called inverse - crime in Colton and Kress (2012)). The sources were localized by a spatial filter Linear-constrained Minimum Variance beamformer (LCMV) on the time-domain and the voxels with highest power selected with ad-hoc threshold in both ROIs (ARI and ALS). We checked that the thresholded voxels, in each of the ROI, were clustered by visual inspection, otherwise a clustering analysis would have been recommended. Next, for these voxels, the source time series were reconstructed by multiplying the spatial filter with the original EEG data . For each voxel this reconstruction contains three channels (3D virtual channel) for the x-, the y-, and the z- component of the equivalent dipole source. Thus, to obtain a single time-series, we determined the largest (temporal) eigenvector using Singular Value Decomposition (SVD), and we recomputed the virtual channel time-series only for the dipole direction that has the most power. Finally, the time-series of the selected voxels were averaged and the causality analysis was performed on this two final virtual channels time-series (see A.2).

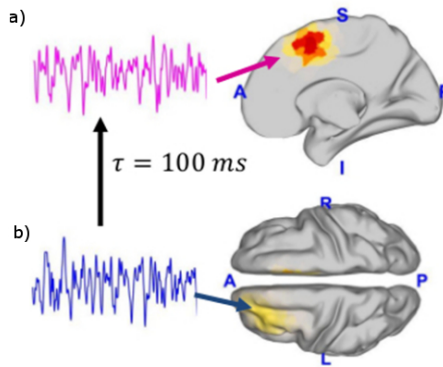


Figure 5.6: Simulation sources time-series. (a) First source placed in the Anterior Right Inferior. (b) Second sources placed in the Anterior Left Superior. The arrow indicates that the direction of interaction with a delay of 100 ms.

5.3.1 Result

```

1 %Structure with cfg parameters
2 cfg=struct();
3 cfg.delay=[0 200 10]
4 cfg.order=2;
5 cfg.verbosity='info_m';
6 cfg.es=[10 1];
7 cfg.subsampling=1
8 cfg.numvalidate=4000 ;
9 cfg.display='True';
10 cfg.testing='True';
11 cfg.channel={'ant_r','ant_l'};;
12 cfg.combination='manual';
13 cfg.time=[0.1 12]
14 cfg.sampling=0.001
15 %Analysis
16 data_prepare=DDIFT_prepare(cfg,vchan);
17 [Result]=DDIFT_delays_analysis(data_prepare);

```

Firstly, we were interested in localizing the simulated sources in the Anterior Left Superior and the Anterior Right Inferior octants of the brain using LCMV beamforming. Figure 5.7 shows the result of the performed source localization. In particular, in the lower part of Figure 5.7 both sources are visible.

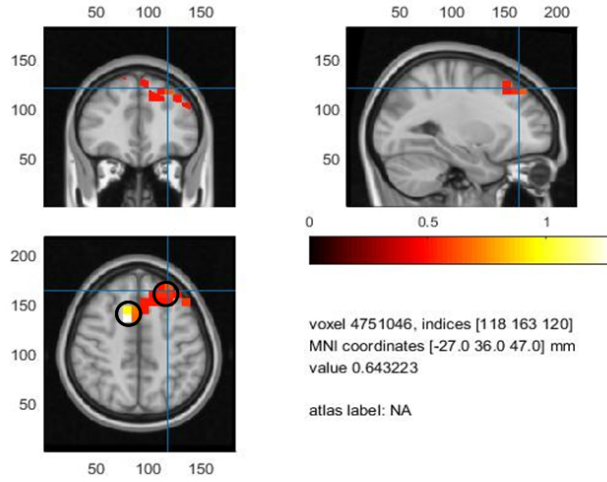


Figure 5.7: Visualization of the source localization via LCMV beamforming.

The method correctly estimates the direction of interaction between the two sources: the Anterior Left Superior drives the Anterior Right Inferior. Optimal results are reported for a second-order model with reconstruction dimension of $d = 10$. Figure 5.8 shows the REG plot after the casual analysis,

where the true delay of 100 ms is within the estimated confident area and almost perfectly recovered from the statistical method (Delay 99.88 ms, Figure 5.8).

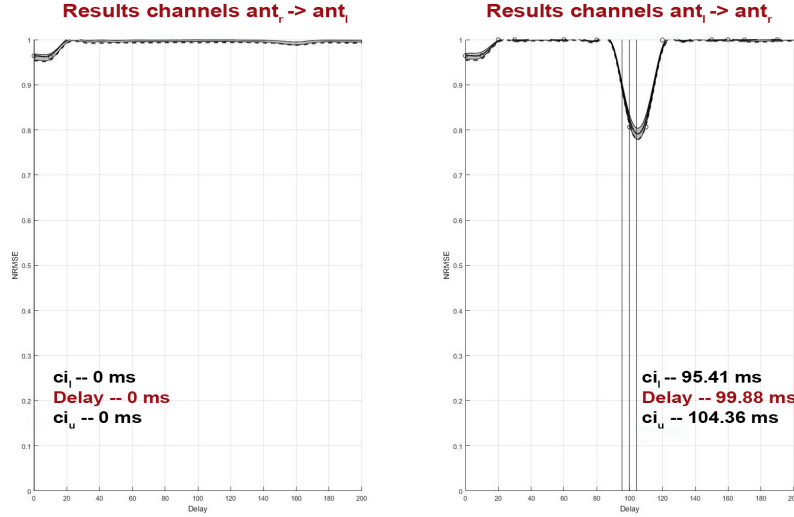


Figure 5.8: Result of simulated sources. REG plot after causal analysis on the reconstructed time series.

5.4 Discussion

In the present simulations we aimed to demonstrate the applicability of DIFTool for the quantification of effective connectivity from EEG signals at both sensors and sources level. Here, we considered different critical issues that might arise in analysis of neuroscientific data.

Particularly relevant for EEG and MEG analysis, we have shown that the statistical method is able to estimate correctly the direction of information flow and the true delay of two interacting signals when linear mixture of the original coupled signals were available (see Figure 5.2 and Table A.1). However, if the two signals are identical (or too close to being identical), due to volume conduction problem, the signals asymmetries are deteriorated and it's not possible to distinguish between driver and driven system. Furthermore, the case of two measurements reflect a single source signal with different noise levels (not shown here) can lead to a significant amount of false positive for method based on Granger principle of causality (Haufe et al., 2013) and see Nolte et al. (2008) or Vicente et al. (2011) for different simulations. We point out that this scenario, for the proposed statistical method, will lead a trivial result of mutually reconstructible measurements since the domain of

the measurements functions overlap and they will observe the same underlying system Schumacher et al. (2015). On the other hand, the presence of different noise realizations is not a concern since if there are not interacting subsystems, reconstruction of the driver measurements is not possible. Thus, an optimal experimental design to avoid overlapping measurements should be taken under consideration to perform causality analysis on a sensors level. Furthermore, we extended our simulations to the case of "cascade effect" and "common drive" input that drives two variables with different coupling delays (see Figure 5.3 and Figure 5.4). We were able to show that the first situation is recovered by the method. Similar to the Convergent cross map analysis (Ye et al., 2015) the indirect pathway showed the lowest NRMSE reconstruction with a delay estimates as a sum of the direct pathways. In the common drive scenario the method erroneously detects and interaction between the uncoupled nodes Y and Z (see Figure 5.4). However, if the common driver drives both nodes Y and Z simultaneously (same delay of coupling) the method will yield symmetric REGs between them. We advise to use a graph algorithm, specifically the one presented in Wollstadt et al. (2015), to correct large network of interactions after the casual analysis. In addition, we proposed an improvement of such algorithm and how it can be easily implemented.

Finally, we demonstrated the ability of the method to recover causal interaction between two sources with simulated EEG data (Figure 5.8). Using source-level time-courses allows a better interpretability of the effective connectivity compared to analysis at the sensors level since the obtained causal interaction estimated from a channels pairs can be a reflection of any of the multiple underlying sources. Furthermore, the number of reconstructed sources time-series is much smaller than the number of channels where a complete pairwise analysis of channels become computationally intractable.

Appendix A

A.1 Table of result

Table A.1: Results of linear mixing simulations.

	Coupling	Delay	Mixing	Estimated delay	
				$X \rightarrow Y$ (true interaction)	$Y \rightarrow X$
AR(10)	Linear	20 ms	0.1	19.85 [19;20.7]	0
AR(10)	Linear	20 ms	0.25	19.83 [19;20.6]	0
AR(10)	Linear	20 ms	0.4	0 [0;0]	0
AR(10)	Linear	50 ms	0.1	49.86 [49;50.7]	0
AR(10)	Linear	50 ms	0.25	49.83 [49;50.6]	0
AR(10)	Linear	50 ms	0.4	0 [0;0]	0
AR(10)	Threshold	20 ms	0.1	19.81 [19;20.6]	0
AR(10)	Threshold	20 ms	0.25	19.78 [19;20.5]	0
AR(10)	Threshold	20 ms	0.4	0 [0;0]	0

A.2 Matlab script

```
1 % compute covariance matrix to use later with LCMV for
2 % all data, baseline and experiment, separately.
3 cfg = [];
4 cfg.covariance = 'yes';
5 cfg.channel = 'EEG';
6 cfg.vartrllength = 1;
7 cfg.covariancwindow = 'all';
8 timelock = ft_timelockanalysis(cfg, data);
9
10 cfg = [];
11 cfg.covariance = 'yes';
12 cfg.channel = 'EEG';
13 cfg.vartrllength = 1;
14 cfg.covariancwindow = 'all';
15 timelock_exp = ft_timelockanalysis(cfg, dataexp);
16 timelock_base = ft_timelockanalysis(cfg, database);
17
18 % LCMV beamformer
19 cfg=[]
20 cfg.method='lcmv'
21 cfg.grid=grid;
22 cfg.headmodel=headmodel;
23 cfg.channel=data.label;
24 cfg.elec=sens;
25 cfg.lcmv.keepfilter = 'yes';
26 [source] = ft_sourceanalysis(cfg, timelock)
27
28 cfg=[]
29 cfg.method='lcmv'
30 cfg.grid=grid;
31 cfg.grid.filter=source.avg.filter;
32 cfg.headmodel=headmodel;
33 cfg.elec=sens;
34 cfg.lcmv.keepfilter = 'yes';
35
36 [source_exp] = ft_sourceanalysis(cfg, timelock_exp)
37 [source_base] = ft_sourceanalysis(cfg, timelock_base);
38
39 interdiff=source_exp;
40 interdiff.avg.pow=(source_exp.avg.pow)./source_base.avg.pow;
41 interdiff.pos=template_grid.pos;
42 % plot 5.7 source localization
43 cfg = [];
44 cfg.downsample = 2;
45 cfg.parameter = 'pow';
46 cfg.interpmethod = 'nearest';
47 source_int = ft_sourceinterpolate(cfg, interdiff, mri);
48
49
50 %% map beamformer source locations onto an anatomical label in an atlas
51 atlas = ft_read_atlas('C:\...\template\atlas\aal\ROI_MNI_V4.nii');
52 [atlas] = ft_convert_units(atlas, 'mm');
53 cfg = [];
54 cfg.interpmethod = 'nearest';
55 cfg.parameter = 'tissue';
56 sourcemodel2 = ft_sourceinterpolate( cfg, atlas, sourcemodel);
57
```

```

58 cfg = [];
59 cfg.atlas = atlas;
60 cfg.roi = atlas.tissuelabel(Right_idx);
61 cfg.inputcoord = 'mni';
62 mask_rigth = ft_volumelookup(cfg, sourcemodel2);
63 cfg = [];
64 cfg.atlas = atlas;
65 cfg.roi = atlas.tissuelabel(Left_idx);
66 cfg.inputcoord = 'mni';
67 mask_left = ft_volumelookup(cfg, sourcemodel2);
68 %apply mask
69 tmp= repmat(interdiff.inside,1,1);
70 tmp(tmp==1) = 0;
71 tmp(mask_rigth)=1;
72 rr=tmp;
73
74 tmp= repmat(interdiff.inside,1,1);
75 tmp(tmp==1) = 0;
76 tmp(mask_left)=1;
77 ll=tmp;
78 %take voxels of the two hemispheres separately
79 pow_rigth=interdiff.avg.pow(rr);
80 pow_left=interdiff.avg.pow(ll);
81 %voxels manually thresholded
82 rh=pow_rigth>=0.5;
83 lh=pow_left>=0.5;
84
85 max_voxel_r=(pow_rigth(rh));
86 max_voxel_l=(pow_left(lh));
87
88 [tf, loc_r] = ismember(max_voxel_r, interdiff.avg.pow);
89 [tfl, loc_l] = ismember(max_voxel_l, interdiff.avg.pow) ;
90
91 % structure data
92 vchan=[]
93 vchan.time = dataexp.time;
94 vchan.fsampl = dataexp.fsampl;
95 Ntr = numel(dataexp.trial);
96
97 vchan.trial = cell( 1, 1);
98 label={'ant_r','ant_l'};
99 %% reconstruct sources time series from spatial filter
100 for i =1:numel(label),
101     if i==1
102
103         ak=tf;
104     else
105         ak=tfl;
106     end
107     for f = 1:numel( (ak)),
108         if i==1
109
110             source_inx=loc_r(f);
111         else
112             source_inx=loc_l(f);
113         end
114         dipole_data{ 1} = interdiff.avg.filter{ source_inx} * ←
115             dataexp.trial{ 1}(:, :);
116         pos_voxel=interdiff.pos(source_inx,:);
117
118         % run svd on data, multiply data by the orientation of the ←
119         dipole in which it is strongest

```

```

118         time_series = cat( 2, dipole_data{ :});
119         [ U1, ~, ~] = svd( time_series, 'econ');
120         for tr = 1:1,
121             tt.trial{ tr}( f, :) = U1( :, 1)' * dipole_data{ tr};
122
123         end
124         tt.pos{1}( f, :)=pos_voxel;
125         clear source_inx dipole_data U1 timeseries
126     end
127
128     % mean channels within brain region
129     for tr = 1:1 %only one trial
130
131         vchan.trial{ tr}( i, :) = mean(tt.trial{ tr});
132     end
133     vchan.label( i) = label( i);
134     vchan.pos( i)=tt.pos;
135 end
136 %% Analysis
137
138 %Structure with cfg parameters
139 cfg=struct();
140 cfg.delay=[0 200 10]
141 cfg.order=2;
142 cfg.verbosity='info_m';
143 cfg.es=[10 1];
144 cfg.subsampling=1
145 cfg.numvalidate=4000 ;
146 cfg.display='True';
147 cfg.testing='True';
148 cfg.channel={'ant_r','ant_l'};;
149 cfg.combination='manual';
150 cfg.time=[0.1 12]
151 cfg.sampling=0.001
152 %Analysis
153 data_prepare=DDIFT_prepare(cfg,vchan);
154 [Result]=DDIFT_delays_analysis(data_prepare);

```


Bibliography

- Aeyels, Dirk (1981), “Generic observability of differentiable systems.” *SIAM Journal on Control and Optimization*, 19, 595–603.
- Bastos, André M and Jan-Mathijs Schoffelen (2015), “A tutorial review of functional connectivity analysis methods and their interpretational pitfalls.” *Frontiers in systems neuroscience*, 9.
- Berger, JO (1985), “Statistical decision theory and bayesian analysis.”
- Chicharro, Daniel and Anders Ledberg (2012), “When two become one: the limits of causality analysis of brain dynamics.” *PLoS One*, 7, e32466.
- Colton, David and Rainer Kress (2012), *Inverse acoustic and electromagnetic scattering theory*, volume 93. Springer Science & Business Media.
- Franz, Matthias O and Bernhard Schölkopf (2006), “A unifying view of wiener and volterra theory and polynomial kernel regression.” *Neural computation*, 18, 3097–3118.
- Fraser, Andrew M and Harry L Swinney (1986), “Independent coordinates for strange attractors from mutual information.” *Physical review A*, 33, 1134.
- Haufe, Stefan and Arne Ewald (2016), “A simulation framework for benchmarking eeg-based brain connectivity estimation methodologies.” *Brain topography*, 1–18.
- Haufe, Stefan, Vadim V Nikulin, Klaus-Robert Müller, and Guido Nolte (2013), “A critical assessment of connectivity measures for eeg data: a simulation study.” *Neuroimage*, 64, 120–133.
- Kennel, Matthew B, Reggie Brown, and Henry DI Abarbanel (1992), “Determining embedding dimension for phase-space reconstruction using a geometrical construction.” *Physical review A*, 45, 3403.

- Lindner, Michael, Raul Vicente, Viola Priesemann, and Michael Wibral (2011), “Trentool: A matlab open source toolbox to analyse information flow in time series data with transfer entropy.” *BMC neuroscience*, 12, 1.
- Manjunath, G, P Tino, and H Jaeger (2012), “Theory of input driven dynamical systems.” *dice. ucl. ac. be, number April*, 25–27.
- MATLAB (2016), *9.0.0.341360 (R2016a)*. The MathWorks Inc., Natick, Massachusetts.
- Nolte, Guido, Andreas Ziehe, Vadim V Nikulin, Alois Schlögl, Nicole Krämer, Tom Brismar, and Klaus-Robert Müller (2008), “Robustly estimating the flow direction of information in complex physical systems.” *Physical review letters*, 100, 234101.
- Oostenveld, Robert, Pascal Fries, Eric Maris, and Jan-Mathijs Schoffelen (2010), “Fieldtrip: open source software for advanced analysis of meg, eeg, and invasive electrophysiological data.” *Computational intelligence and neuroscience*, 2011.
- Rosasco, Lorenzo (2015), “Class 04: Reproducing kernel hilbert spaces.” <https://www.youtube.com/watch?v=KZZD5sBwGCA>.
- Schumacher, Johannes (2015), *Time Series Analysis informed by Dynamical Systems Theory*. Ph.D. thesis, Institute of Cognitive Science.
- Schumacher, Johannes, Thomas Wunderle, Pascal Fries, Frank Jäkel, and Gordon Pipa (2015), “A statistical framework to infer delay and direction of information flow from measurements of complex systems.” *Neural computation*.
- Stark, J, David S Broomhead, ME Davies, and J Huke (2003), “Delay embeddings for forced systems. ii. stochastic forcing.” *Journal of Nonlinear Science*, 13, 519–577.
- Sugihara, George, Robert May, Hao Ye, Chih-hao Hsieh, Ethan Deyle, Michael Fogarty, and Stephan Munch (2012), “Detecting causality in complex ecosystems.” *science*, 338, 496–500.
- Sugihara’s Lab (2012), “State space reconstruction: Takens’s theorem and shadow manifolds.” <https://www.youtube.com/user/sugiharalab>.
- Sundararajan, S and S Sathiya Keerthi (2001), “Predictive approaches for choosing hyperparameters in gaussian processes.” *Neural Computation*, 13, 1103–1118.

- Takens, Floris (1981), “Dynamical systems and turbulence (detecting strange attractors in fluid turbulence).”
- Takens, Floris (2002), “The reconstruction theorem for endomorphisms.” *Bulletin of the Brazilian Mathematical Society*, 33, 231–262.
- Ver Steeg, Greg (2000), “Non-parametric entropy estimation toolbox (npeet).”
- Vicente, Raul, Michael Wibral, Michael Lindner, and Gordon Pipa (2011), “Transfer entropy – A model-free measure of effective connectivity for the neurosciences.” *Journal of computational neuroscience*, 30, 45–67.
- Wang, Han and Michael Brady (1995), “Real-time corner detection algorithm for motion estimation.” *Image and vision computing*, 13, 695–703.
- Whitney, Hassler (1936), “Differentiable manifolds.” *Annals of Mathematics*, 645–680.
- Williams, Christopher KI and Carl Edward Rasmussen (2006), “Gaussian processes for machine learning.” *the MIT Press*, 2, 4.
- Wollstadt, Patricia, Ulrich Meyer, and Michael Wibral (2015), “A graph algorithmic approach to separate direct from indirect neural interactions.” *PloS one*, 10, e0140530.
- Ye, Hao, Ethan R Deyle, Luis J Gilarranz, and George Sugihara (2015), “Distinguishing time-delayed causal interactions using convergent cross mapping.” *Scientific reports*, 5.