

A Nonlinear Time Series Analysis Toolbox

User Manual

Version 1.2

Immo Weber

[Immo.weber@systemsneuroscience.de](mailto:Immo.weber@systemsneuroscience.de)



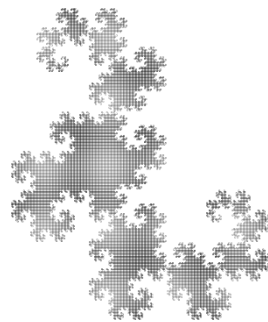
## Table of Contents

|   |    |
|---|----|
| 1. Introduction.....  | 5  |
| 2. Installation.....  | 6  |
| 3. General Workflow .....                                   | 6  |
| 3.1. Overview.....  | 6  |
| 3.1.1. Load Data.....                                       | 6  |
| 3.1.2. Prepare Data (optional).....                         | 7  |
| 3.1.3. Choose Method(s) .....                               | 7  |
| 3.1.4. Define/Optimize Embedding Parameters (optional)..... | 7  |
| 3.1.5. Define Method-Specific Parameters (optional) .....   | 7  |
| 3.1.6. Calculate .....                                      | 7  |
| 3.1.7. Plot Results.....                                    | 7  |
| 3.1.8. Save Results.....                                    | 8  |
| 3.2. GUI.....   | 8  |
| 3.2.1. Overview.....  | 8  |
| 3.2.2. Workflow .....                                       | 9  |
| 3.3. Batch-Editor.....                                      | 12 |
| 3.3.1. Overview.....  | 12 |
| 3.3.2. Workflow .....                                       | 13 |
| 3.4. Plotting Tool .....                                    | 14 |
| 3.4.1. Workflow .....                                       | 14 |
| 3.4.2. Topographic Representation of EEG or MEG-Data .....  | 15 |
| 3.5. Statistics GUI.....                                    | 18 |

|        |  |    |
|--------|--|----|
| 3.6.   | Custom-made Scripts .....                                | 20 |
| 3.6.1. | Overview.....  | 20 |
| 3.6.2. | Workflow .....   | 21 |
| 4.     | Table of Functions .....                                 | 22 |
| 4.1.   | Nonlinear Dynamics Measures.....                         | 22 |
| 4.1.1. | Correlation Dimension .....                              | 22 |
| 4.1.2. | Lyapunov Exponent .....                                  | 24 |
| 4.1.3. | Ragwitz Estimator.....                                   | 26 |
| 4.1.4. | False Nearest Neighbours.....                            | 27 |
| 4.1.5. | Space-Time Separation Plot .....                         | 29 |
| 4.1.6. | Time Inversion .....                                     | 30 |
| 4.1.7. | Hurst Exponent.....                                      | 31 |
| 4.1.8. | Detrended Fluctuation Analysis .....                     | 32 |
| 4.1.9. | Unstable Periodic Orbit Transform .....                  | 34 |
| 4.2.   | Recurrence Measures.....                                 | 36 |
| 4.2.1. | Recurrence Plot .....                                    | 36 |
| 4.2.2. | Joint Recurrence Plot.....                               | 39 |
| 4.2.3. | Cross Recurrence Plot.....                               | 41 |
| 4.2.4. | Recurrence Frequencies_over range of neighbourhoods..... | 43 |
| 4.2.5. | Windowed Recurrence Frequencies .....                    | 45 |
| 4.3.   | Information Theoretic Measures .....                     | 47 |
| 4.3.1. | Shannon Entropy (Bin Estimator) .....                    | 47 |

|        |   |    |
|--------|---|----|
| 4.3.2. | Differential Entropy (Kozachenko's Estimator) ..... | 48 |
| 4.3.3. | Mutual Information (Bin Estimator) .....            | 49 |
| 4.3.4. | Mutual Information (Kraskov's Estimator) .....      | 50 |
| 4.3.5. | Mutual Information Matrix (Bin & Kraskov) .....     | 52 |
| 4.3.6. | Auto-Mutual Information (Bin Estimator) .....       | 53 |
| 4.3.7. | Auto-Mutual Information (Kraskov's Estimator) ..... | 54 |
| 4.3.8. | Active Information Storage .....                    | 55 |
| 4.4.   | Related .....                                       | 56 |
| 4.4.1. | Phase Space reconstruction .....                    | 56 |
| 4.4.2. | Autocorrelation .....                               | 58 |
| 4.4.3. | Neighbours Distance (mex) .....                     | 58 |
| 4.4.4. | Optimize Embedding Parameters .....                 | 59 |
| 4.4.5. | Optimize Number of Bins for Histogram .....         | 61 |
| 4.4.6. | Prepare Data .....                                  | 61 |
| 4.5.   | Data .....  | 62 |
| 4.5.1. | Lorenz Data Generation .....                        | 62 |
| 4.5.2. | Sinusoid Generation .....                           | 63 |
| 4.5.3. | Logistic Map .....                                  | 64 |
| 4.5.4. | Lorenz Data .....                                   | 64 |
| 4.6.   | Statistics .....                                    | 64 |
| 4.6.1. | Generation of Surrogates .....                      | 64 |
| 4.6.2. | Monte-Carlo Test .....                              | 65 |

|        |   |    |
|--------|---|----|
| 4.6.3. | Cluster-based permutation analysis..... | 67 |
|--------|---|----|



Marburg, 2018

# 1. Introduction

NoLiTiA is a free open-source Matlab®-Toolbox for the analysis of time series with possible nonlinear origin i.e. the observed data was generated by a nonlinear system. A system is nonlinear if a change of its input is non-proportional to its output. The toolbox is designed with the intention to be both flexible for the experienced user and intuitive for beginners. The range of available methods originate from three main fields: 1) (classic) nonlinear dynamics, 2) recurrence quantification, 3) information theory (Fig. 1).

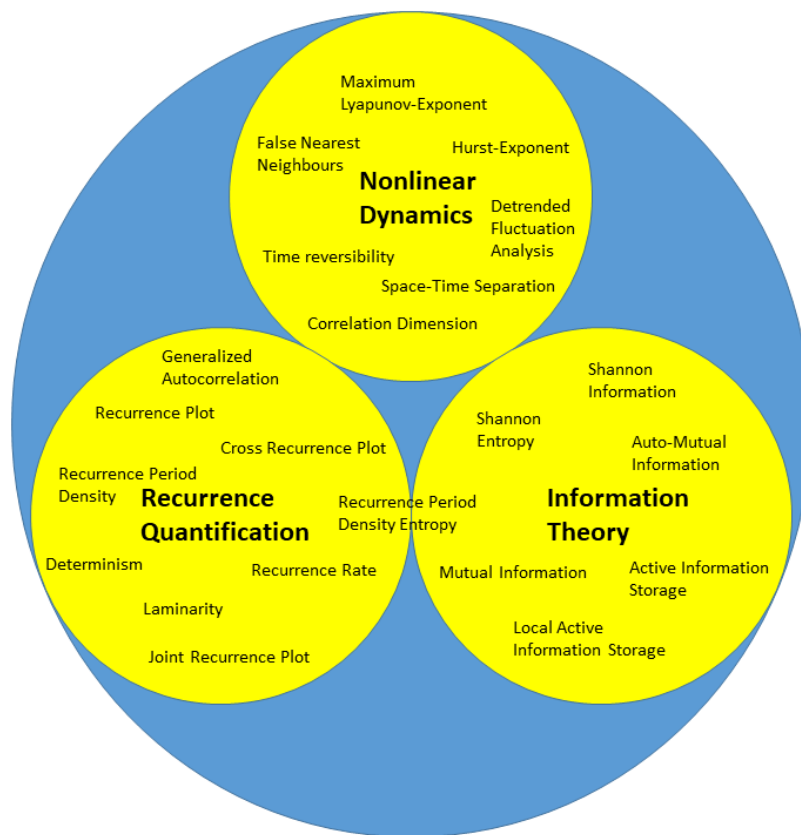


Figure 1: Topics covered by NoLiTiA.

## 2. Installation

To install the toolbox, download it from [www.nolitia.com](http://www.nolitia.com) and simply copy the complete folder of NoLiTiA onto your computer. Finally, run the “install\_nolitia.m” script within Matlab. NoLiTiA was developed using Matlab 2016b. Compatibility with older Matlab versions cannot be guaranteed.

## 3. General Workflow

### 3.1. Overview

The toolbox offers three distinct ways to analyse data, depending on the degree of needed flexibility, experience in programming and ease of use: 1) graphical user interface (GUI), 2) batch-editor, 3) custom-made Matlab-scripts.

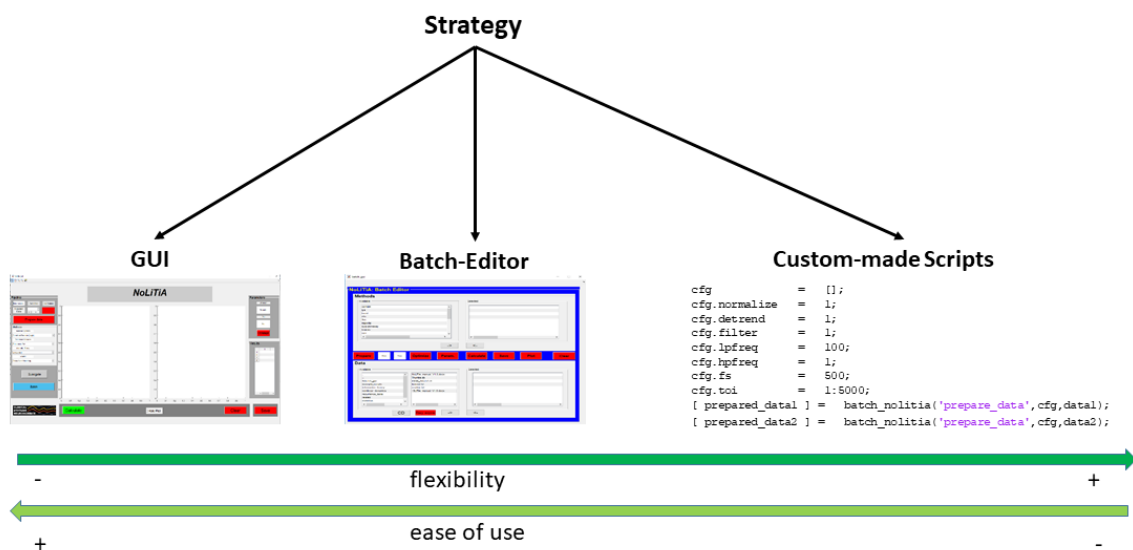


Figure 2: Analysis pathways.

No matter which of the three options the user chooses, all of them share more or less the same workflow:

#### 3.1.1. Load Data

Depending on the user's analysis pathway, data may either be loaded from a file or from workspace.

### 3.1.2. Prepare Data (optional)

NoLiTiA offers the possibility to pre-process your data. Options include the definitions of a time-region of interest, detrending, normalization and filtering (for filtering the signal processing toolbox needs to be installed). See Table 53 in section 3 for details.

### 3.1.3. Choose Method(s)

Choose methods from the three topics: 1) (classic) nonlinear dynamics, 2) recurrence based methods, 3) information theoretic measures. See section 4 for details.

### 3.1.4. Define/Optimize Embedding Parameters (optional)

For many methods, it is necessary to define embedding parameters (dim=dimension, tau=embedding delay) for state-space reconstruction. The three pathways offer either to define them ad-hoc by the user or to optimize them based on two different approaches ('deterministic', 'markov'). The optimization procedure should be chosen depending on whether your data was generated by a deterministic or a stochastic process (see section 3 for details).

### 3.1.5. Define Method-Specific Parameters (optional)

Every method has at least one parameter, which may be specified by the user. If a parameter is left unspecified, the default value is applied (see section 4 for a list of all parameters including default values).

### 3.1.6. Calculate

Run the analysis.

### 3.1.7. Plot Results

Depending on your analysis path, results are either plotted automatically (GUI), or the user may optionally choose to do so (batch, custom-made scripts, see section 3.3 and 3.6).



### 3.1.8. Save Results

You should be aware that results are not automatically saved. Either push the save button in the GUI or Batch-editor, or use the Matlab-command 'save' to save results.

## 3.2. GUI

### 3.2.1. Overview

The graphical user interface (GUI) is intended to be the most beginner's friendly option for analysis. The GUI can be invoked by typing "nolitia\_gui" in the command window (Fig. 3). The interface is composed of four main regions: on the left side, the user loads the input data, chooses whether and how to pre-process, specifies analysis methods, generates surrogate data and enters batch-mode. On the right side, the user may enter embedding parameters (dimension and tau) *ad-hoc* or choose to optimize them using two different approaches (see section 4.4.4). Method-specific results are displayed in the table below. In the middle of the interface, two axes display method-specific figures after calculation. The panel below the axes consists of three push buttons and a radar button. The button "Calculate" invokes the analysis of the input data with the pre-defined method. The button "Clear" clears the two main axes and by clicking the "Save"-button an UI opens, where the user may choose where to save results. By clicking the "Hold Plot" radar button the user may superimpose results of subsequent analyses. By toggling the "Record"-button, all main steps and commands done by the user are saved in a queue. Pressing the "Generate Script"-button automatically generates a

Matlab-script and prompts the user to enter a file name and saving directory. The recording queue can be deleted by pressing the “Clear Record”-button.



Figure 3: Graphical User Interface (GUI).

### 3.2.2. Workflow

#### *Data Import:*

The user may either choose to load a variable from workspace, from a “.mat-file” or to load some test data for training. Data must be an  $N \times 1$  vector for univariate methods or an  $N \times 2$  vector for bivariate methods like mutual information. The GUI only supports analysis of one dataset at a time. For stacked

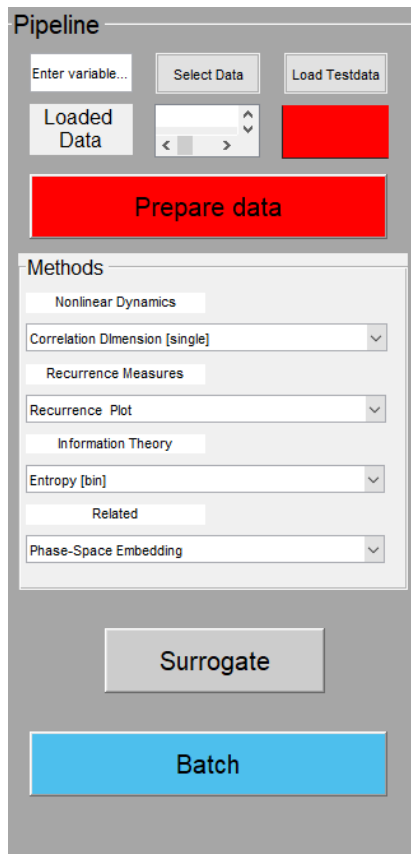


Figure 4: Pipeline of the GUI (left side).

analyses of multiple datasets, use the batch-editor (see section 2.3). Upon successful loading, the variable name is displayed in the field below “Select Data” and the red rectangle turns green. By clicking the “Surrogate”-Button the user may optionally transform the imported data into surrogate data using one of five different algorithms (see section 3, Table 59). After clicking, the button turns green indicating successful transformation. After clicking again on the button, the original data get reloaded, indicated by switching color from green to red.

#### *Prepare Data:*

The next optional step is to pre-process the input data. To begin,

click on the red “prepare data” -button to invoke a table

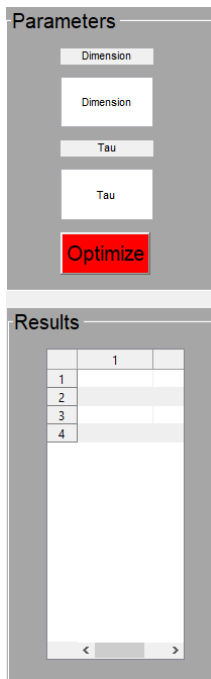
where pre-processing parameters may be specified. For a

list of parameters, see Table 51. Note that for filtering options, the signal processing toolbox must be installed.

#### *Select Method:*

After successful specification, the button “Prepare Data” turns green and the user may choose an analysis method from one of the three dropdown-menus on the left side of the GUI.

#### *Embedding Parameters:*



Depending on the chosen method, embedding parameters (dimension and tau) need to be specified to reconstruct the phase-space (see section 3: all functions with configuration structures requiring `cfg.dim` & `cfg.tau`). For this, the user may either choose to define both of them *ad-hoc* by typing them in the fields labeled “Dimension” and “Tau” on the right side of the GUI, or to optimize both using one of two different approaches by clicking the red “Optimize” button. For optimization parameters see section 4.4.4. After clicking “OK” the red “Optimize” -button turns green and optimum tau and dimension are displayed in the results table on the right side of the GUI. As optimized parameters are automatically defined, it is not necessary to enter them manually in the “Dimension” and “Tau” fields.

Figure 5: Embedding parameters and results table (right side)

*Calculate, Save & Clear:*

Finally, calculation can be invoked, by clicking the green “Calculate”-Button at the bottom of the GUI (Fig.6). Before calculation can proceed, the user is prompted to specify method-specific parameters (see section 3). Results are displayed in the table at the right of the GUI, while figures are presented in the two axes in the centre of the GUI. The left axis always displays the prepared time series. Results may be saved by clicking the red “Save”-button, while “Clear” clears the current axes and results table.

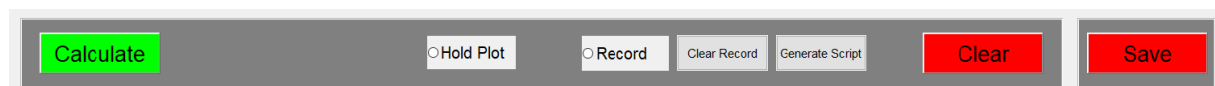


Figure 6: Plot, Clear and Save (bottom).

### 3.3. Batch-Editor

#### 3.3.1. Overview

The batch-editor is intended to be a compromise between the accessibility of the GUI and the flexibility of custom-made scripts. In contrast to the GUI it allows for a semi-automatized stacked analysis of multiple datasets and methods. The batch-editor can be invoked by either clicking on the batch button in the GUI (Fig. 4) or by typing “batch\_gui” in the command line of Matlab. The batch-editor is composed of three main parts. In the bottom half, the user loads datasets and chooses which data to analyse. In the top half, the user chooses which methods to use for analysis. In the center, linearly arranged buttons guide the user through the analysis pipeline (Fig. 7).

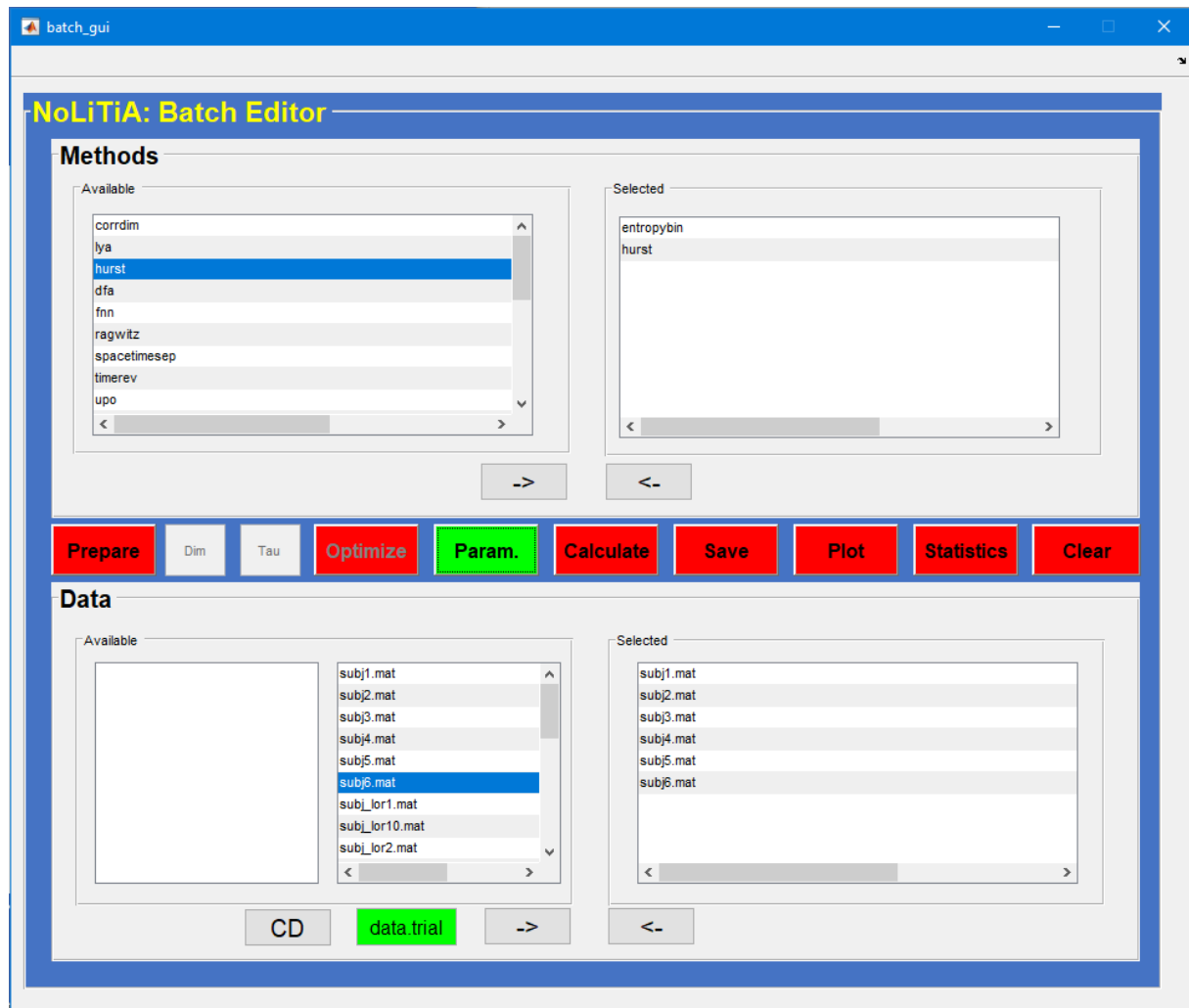


Figure 7: Batch-Editor.

### 3.3.2. Workflow

#### *Data Import*

The bottom part of the editor is dedicated to data import. The left table displays the current directory, while the center table shows the content of the current directory. To change the current directory either click on the dots at the top of the left table or click the CD-button. Before loading data, the user must specify a regular expression common to all datasets, in the red field next to the CD-button. For example, if the data to be analysed is saved in the field “trials” in the struct “data” the user enters: data.trials and hits enter. Upon successful specification, the field turns green. Next, the user sequentially chooses datasets by highlighting files with a left mouse-click and clicking on the right-pointing arrow at the bottom. Successfully loaded datasets appear on the right table. To unselect data, the user highlights the files in the right table and clicks the arrow pointing to the left. Data must be organized in cell arrays. Each cell may contain an  $n \times m$  matrix, with  $n$  being time points and  $m$  being the observables/variables. In the context of neuroscience research, cells may represent trials and the matrices time  $\times$  channel data of i.e. electrophysiological recordings.

#### *Prepare Data*

The next optional step is to pre-process the input data. To begin, the user clicks on the red “prepare” -button to invoke a table where pre-processing parameters may be specified. For a list with parameters, see Table 53. Note that for filtering options, the signal processing toolbox must be installed. The button turns green after pre-processing is finished.

#### *Select Methods*

The batch-editor allows for multiple analysis methods to be computed sequentially in an automatized fashion. To select a method, left-click on it in the top-left table and push the button with the right-pointing arrow. Upon successful selection, the method should appear in the top right table. To unselect a method, highlight it with a left-click and push the button with the arrow pointing to the left.

#### *Embedding Parameters*

Depending on the chosen methods, embedding parameters (dimension and tau) need to be specified to reconstruct the phase-space (see section 3: all functions with configuration structures requiring `cfg.dim` & `cfg.tau`). For this, the user may either choose to define both of them *ad-hoc* by typing them in the fields labeled “Dim” and “Tau”, or to optimize both for each dataset using one of two different approaches by clicking the red “Optimize” button. For optimization parameters see Table 51. The “Optimize”-Button turns green after the optimization procedure is finished. Note that optimization should not be applied if the user wants to plot time resolved topographic plots using the “plotting tool” (see section 3.4). Buttons and fields for embedding parameters are non-operable by default and become usable when selecting methods which necessitate phase-space reconstruction.

### *Define Parameters*

By clicking on the “Param.”-button, the user is sequentially prompted to specify method-specific parameters (see section 3). The button turns green after parameter specification is complete.

### *Calculation, Saving and Clearing*

Calculation can be invoked by clicking the “Calculate”-button. After successful calculation the button turns green and results may be saved by clicking the “Save”-button. Clicking on the “Clear”-Button reloads the batch-editor.

## 3.4. Plotting Tool

### 3.4.1. Workflow

Either clicking on the “Plot”-button or typing “`plot_batch_gui`” in the command line loads the “Plotting-Tool”. The plotting-tool is intended to be used for displaying results generated by the batch-editor. If the tool is loaded by clicking the “Plot”-button after computation in the batch-editor, the results of the first data set are automatically loaded into the plotting-tool. Alternatively, the user may load saved data by clicking the “Load”-button. Cell indices and matrix columns (observables) are displayed in the tables “Trials” and “Channel”, respectively. Available methods are represented by a hierarchical tree structure, which can be unfolded by clicking on “Methods” in the method panel. To display results,

left-click on a trial ID, channel ID, as well as on a method and push the “Plot”-button. Methods, which can be plotted using the “Plot”-button are indicated by a green arrow next to its name. Results are plotted in the center axis. By clicking the “Hold Plot” radar button the user may superimpose results of different datasets.

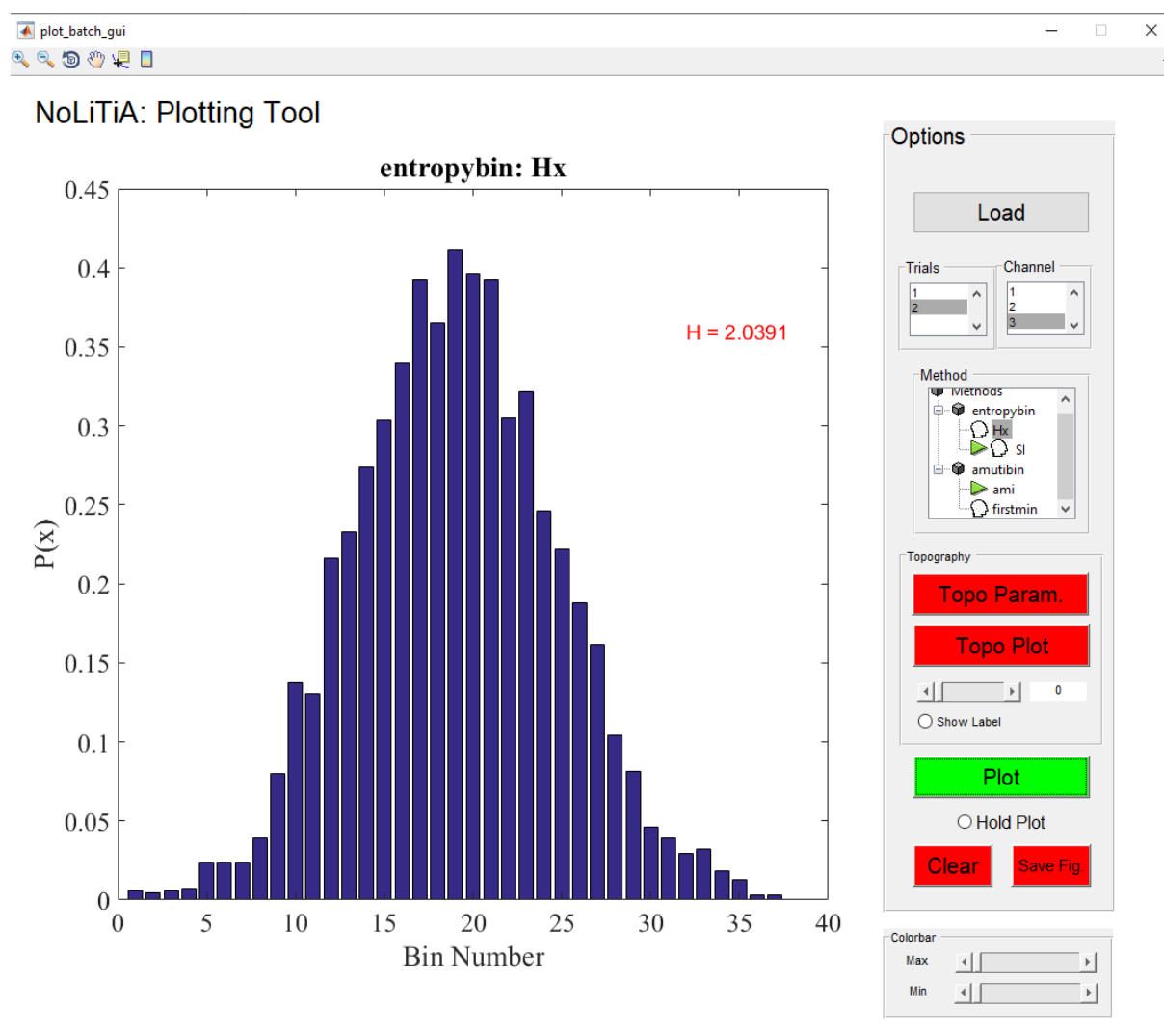


Figure 8: Plotting Tool.

### 3.4.2. Topographic Representation of EEG or MEG-Data

The plotting tool may be used to analyse electroencephalographic (EEG)- or magnetoencephalographic (MEG)-data, as each time series may represent one recorded channel. The plotting-tool allows for a



topographical representation of results per channel. By clicking on the red “Topo Param.”-button the user is prompted to select an “.sfp”-file containing the channel names and electrode positions. Additionally, the user may specify specific electrodes (1xnumelec) to plot, as well as a sampling frequency. Finally, after selecting a method from the list, the topographic plot is displayed after clicking the “Topo Plot”-button. Methods, which can be plotted using the “Topo Plot” button are indicated by a head shape next to its name. Colorbar limits can be adjusted by using the slider bar below the options panel. Methods with a green arrow, as well as a head shape symbol next to it may be plotted time resolved, by first clicking the “Topo Plot”-button and then using the slider below to scroll forward or backward in time. Supported function are listed in Table 1. For further function reference see section 3.

*Table 1: Supported functions for Topo Plot*

| Function              | Field    | Description                             |
|-----------------------|----------|---|
| <b>corrdim</b>        | Dtakens  | estimate of correlation dimension       |
| <b>lya</b>            | lle      | estimate of maximum Lyapunov exponent   |
| <b>ragwitz</b>        | dimopt   | estimate of Markov chain order          |
| <b>ragwitz</b>        | tauopt   | estimate of embedding delay             |
| <b>fnn</b>            | firstmin | estimate of optimal embedding dimension |
| <b>timerev</b>        | Qt       | Time reversibility score                |
| <b>timerev</b>        | zstat    | teststatistic for nonlinearity          |
| <b>timerev</b>        | sig      | H0 of linearity rejected                |
| <b>hurst</b>          | expo     | estimate of Hurst exponent              |
| <b>dfa</b>            | expo     | estimate of self-affinity               |
| <b>recurrenceplot</b> | rpde     | recurrence period density entropy       |
| <b>recurrenceplot</b> | det      | determinism                             |

|                          |          |   |
|--------------------------|----------|---|
| <b>recurrenceplot</b>    | lam      | laminarity                                  |
| <b>recurrenceplot</b>    | rr       | recurrence rate                             |
| <b>entropybin</b>        | Hx       | estimate of Shannon entropy                 |
| <b>entropybin</b>        | SI       | estimate of Shannon information             |
| <b>entropykozachenko</b> | Hx       | estimate of Shannon entropy                 |
| <b>entropykozachenko</b> | SI       | estimate of Shannon information             |
| <b>amutibin</b>          | firstmin | first minimum of auto-mutual<br>information |
| <b>amutiembknn</b>       | firstmin | first minimum of auto-mutual<br>information |
| <b>AIS</b>               | AIS      | active information storage                  |
| <b>AIS</b>               | IAIS     | local active information storage            |

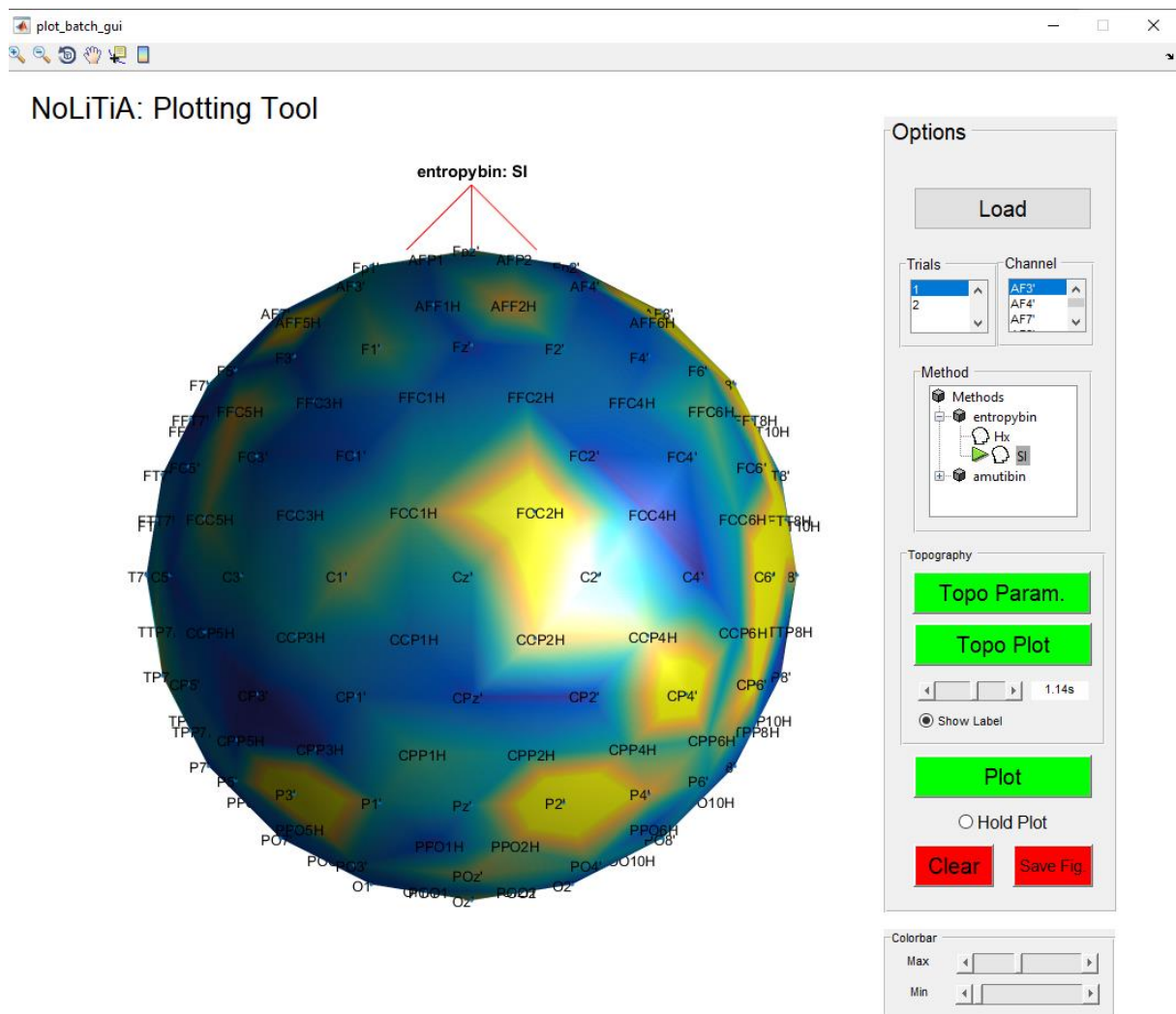


Figure 9: Topographic plot.

### 3.5. Statistics GUI

By clicking the “Statistics” button in the batch-editor or typing “nolitia\_statistics\_gui” in the command line it is possible to test statistical differences of estimated values between provided datasets. The “CD”-Button opens a pop-up window where the result-files of the batch computation may be selected. Pressing the arrow buttons (→,←) on the interface moves the highlighted files either to group 1 or group 2 which e.g., may represent different recording conditions. By clicking on any file in group 1 the method table on the left will display the available methods present in the data structure. Highlighting any of the methods will show the available variables in the variable table, which may be selected by

clicking on it. After choosing either “Monte-Carlo test” or “Cluster-based permutation analysis” (Maris and Oostenveld 2007) the computation may be initiated by clicking the “Compute” button. Results of the computation, including the p-values and whether or not the differences of datasets is significant, will be displayed in the “Output” table at the bottom. Results may be saved by clicking the “Save” button.

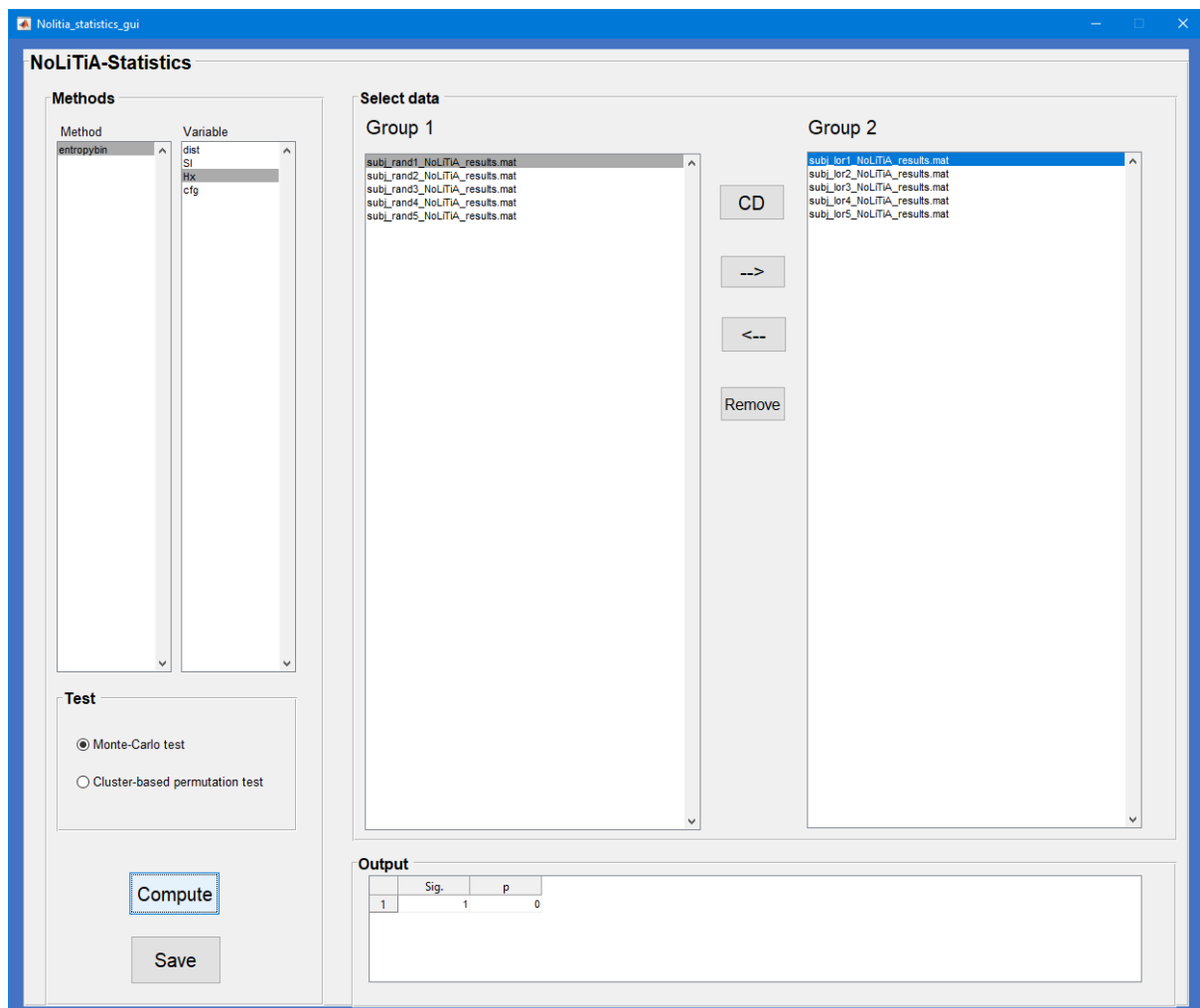
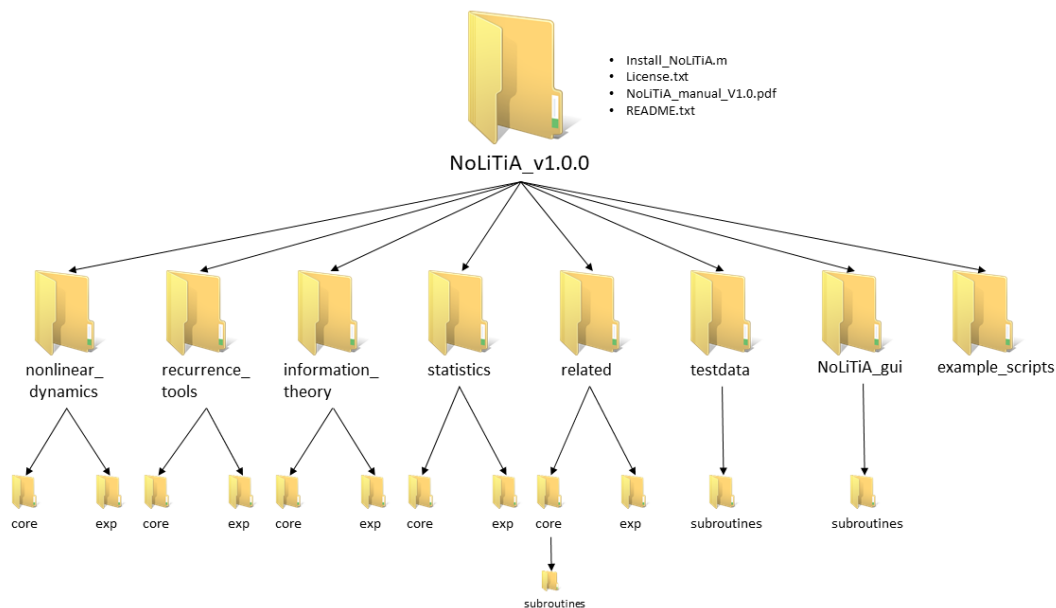


Figure 10: GUI of the statistics tool.

## 3.6. Custom-made Scripts

### 3.6.1. Overview

The final option to analyse data is by using custom-made scripts. This option offers the highest flexibility but requires some experience in Matlab-scripting. All core functions are built in the same way. Following the naming convention of other matlab toolboxes like Fieldtrip (Oostenveld et al. 2011) or SPM (The Wellcome Dept. of Imaging Neuroscience, London; [www.fil.ion.ucl.ac.uk/spm](http://www.fil.ion.ucl.ac.uk/spm)), NoLiTiA functions all start with the prefix “nta”. The user must provide up to two datasets (depending on whether the method is uni-or bivariate) and a configuration structure “cfg” containing method-specific parameters. Default values exist for all parameters. For a complete list of parameters per method see section 3. For beginners the GUI offers the possibility to record processing steps made in the GUI and to save them as a Matlab script (see section 3.2.1).



*Figure 10:* Folder structure. The main analysis functions are located in the „core“-folders. Content of the “exp”-folders is still under development and will be added to the core folder in future releases.

### 3.6.2. Workflow

The workflow is the same as previously described for the GUI and batch-editor. An example workflow is shown in figure 12.

**Hint:** Command line output can be suppressed by setting the `cfg.verbose` parameter to zero.

```
%% Load data %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

load('lorenz10000.mat')
data1      = x;
data2      = y;
results_final = []; %define output structure

%% Prepare data %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

cfg        = [];
cfg.normalize = 1;
cfg.detrend  = 1;
cfg.filter   = 1;
cfg.lpfreq   = 100;
cfg.hpfreq   = 1;
cfg.fs       = 500;
cfg.toi      = 1:5000;
[prepared_data1] = nta_prepare_data(data1,cfg);
[prepared_data2] = nta_prepare_data(data2,cfg);

%% Optimize embedding parameters %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

cfg        = [];
cfg.optimization = 'deterministic';
cfg.dims      = [2 9];
cfg.numbin    = 0; %Optimize bin size
[results_opt_emb] = nta_optimize_embedding(prepared_data1,cfg);
results_final.opt_emb = results_opt_emb;

%% Analyze data %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

cfg        = [];
cfg.minlength = 0;
cfg.dim       = results_opt_emb.optdim;
cfg.tau       = results_opt_emb.opttau;
cfg.plt       = 0;
[results_rec] = nta_recurrenceplot(prepared_data1,cfg);
results_final.rec = results_rec;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```

cfg          = [];
cfg.numbin   = 0;
[results_MIbin] = MIbin(prepared_data1,prepared_data2,cfg);
results_final.MIbin = results_MIbin;

%% Save data %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
save('results_NoLiTiA.mat','results_final')

```

**Figure 12:** Example analysis pipeline.

## 4. Table of Functions

### 4.1. Nonlinear Dynamics Measures

#### 4.1.1. Correlation Dimension

**Function:** *nta\_corrdim.m*

**Dependencies:** *nta\_autocorr.m*, *nta\_amutibin.m*, *nta\_phasespace.m*

**Description:** Calculation of the correlation sum  $C$  as a function of scales  $\varepsilon$  by which the correlation dimension  $D$  may be estimated (Grassberger and Procaccia 1983). Also calculates a maximum likelihood estimator for  $D$  by (Takens 1985):

$$C(\varepsilon) \sim \varepsilon^D \quad (1)$$

$$C(\varepsilon) = \frac{2}{N(N-1)} \sum_{i=1}^N \sum_{j=i+1}^N \theta(\varepsilon - \|\mathbf{x}_i - \mathbf{x}_j\|) \quad (2)$$

$$D_{Takens} = \frac{C(\varepsilon_0)}{\int_0^{\varepsilon_0} C(\varepsilon)/\varepsilon d\varepsilon} \quad (3)$$

$N$ : number of points,  $\|\cdot\|$ : distance norm,  $\theta$ : Heaviside-step function.

**Table 2:** Configuration Structure (*cfg*)

| Field Name | Description | Size | Data Type | Default |
|------------|-------------|------|-----------|---------|
|------------|-------------|------|-----------|---------|

|                |  |     |        |         |   |
|----------------|--|-----|--------|---------|---|
| <b>tau</b>     | embedding delay. If tau=0, the first minimum of the auto mutual information is used.   | 1x1 | int    | 0       |   |
| <b>dim</b>     | embedding dimension  | 1x1 | int    | 2       |   |
| <b>ens</b>     | min & max size of neighbourhood<br>(% of maximal attractor diameter)   | 1x2 | double | [1 100] |   |
| <b>nr</b>      | number of neighbourhoods   | 1x1 | int    | 10      |   |
| <b>th</b>      | Theiler window in samples. If th=0, 2*autocorrelation time is used.  | 1x1 | int    | 0       |   |
| <b>manual</b>  | manually define the linear scaling region by choosing two points when the crosshair appears. If manual=0 logE(1:length(logE)/3) is used for slope calculation. | 1x1 | int    | 0       |   |
| <b>resl</b>    | number of points used for slope calculation  | 1x1 | int    | 2       |   |
| <b>plt</b>     | plot results yes/no [1/0]  |     | 1x1    | int     | 0 |
| <b>verbose</b> | verbose level [1/0]  |     | 1x1    | int     | 1 |

Table 3: Output Structure (results)

| Field Name       | Description   | Size                  |
|------------------|---|-----------------------|
| <b>cfg</b>       | configuration structure   | struct                |
| <b>logE_logC</b> | 1. Column: log E (neighbourhood-size)<br>2. Column: log C (correlation sum) | nr x 2                |
| <b>diff_logC</b> | local slope of logC per logE  | 1xlength(1:resl:nr)-1 |



|                      |   |        |
|----------------------|---|--------|
| <b>Dtakens</b>       | estimate of the correlation dimension using Taken's estimator | 1x1    |
| <b>neighsizelist</b> | vector of neighbourhood-sizes used for calculation            | 1 x nr |

#### 4.1.2. Lyapunov Exponent

**Function:** *nta\_lya.m*

**Dependencies:** *nta\_autocorr.m, nta\_amutibin.m, nta\_phasespace.m, nta\_neighsearch.mexw64*

**Description:** Estimation of the largest Lyapunov exponent  $\lambda$  as described in (Kantz 1994) (5) and (Rosenstein et al. 1993) (6):

$$|\mathbf{x}(\Delta t)| \approx e^{\lambda t} |\mathbf{x}(t_0)| \quad (4)$$

By plotting

$$S(\Delta t) = \frac{1}{N} \sum_{t_0=1}^N \ln \left( \frac{1}{|U(\mathbf{x}_{t_0})|} \sum_{\mathbf{x}_t \in U(\mathbf{x}_{t_0})} \|\mathbf{x}_{t_0+\Delta t} - \mathbf{x}_{t+\Delta t}\| \right) \quad (5)$$

$$S(\Delta t) = \frac{1}{N} \sum_{t_0=1}^N \ln \left( \|\mathbf{x}_{t_0+\Delta t} - \mathbf{x}_{t+\Delta t}\| \right), \quad (6)$$

as a function of temporal separation  $\Delta t$ , one can estimate  $\lambda$  by calculating the slope of a reasonable linear fit of the expansion rate  $S$ .  $N$ : Number of samples,  $U$ : neighbourhood,  $\|\cdot\|$ : distance norm.

*Table 4: Configuration Structure (cfg)*

| Field Name    | Description  | Size | Data Type | Default |
|---------------|--|------|-----------|---------|
| <b>tau</b>    | embedding delay. If tau=0, the first minimum of the auto mutual information is used. | 1x1  | int       | 0       |
| <b>dim</b>    | embedding dimension  | 1x1  | int       | 2       |
| <b>method</b> | either choose Kantz ['Kantz']<br>or Rosenstein's ['Rosenstein'] algorithm            | char | char      | 'Kantz' |

|                |  |     |        |     |
|----------------|--|-----|--------|-----|
| <b>en</b>      | size of neighbourhood<br>(% of maximal attractor diameter)   | 1x1 | double | 5   |
| <b>numran</b>  | number of random points used<br>for calculation  | 1x1 | int    | 100 |
| <b>it</b>      | number of temporal iterations  | 1x1 | int    | 10  |
| <b>th</b>      | Theiler window in samples. If th=0,<br>2*autocorrelation time is used.   | 1x1 | int    | 0   |
| <b>resl</b>    | number of points used<br>for slope estimation  | 1x1 | int    | 2   |
| <b>manual</b>  | manually define the linear region by choosing two<br>points when the crosshair appears. If manual=0<br>1:it/3 is used for slope calculation. | 1x1 | int    | 0   |
| <b>plt</b>     | plot results yes/no [1/0]  | 1x1 | int    | 1   |
| <b>verbose</b> | verbose level [1/0]  | 1x1 | int    | 1   |

*Table 5: Output Structure (results)*

| Field Name        | Description                                  | Size                  |
|-------------------|--|-----------------------|
| <b>cfg</b>        | configuration structure                      | struct                |
| <b>lle</b>        | estimate of the largest<br>Lyapunov exponent | 1x1                   |
| <b>loglya</b>     | log of distances after it<br>iterations      | 1 x it+1              |
| <b>it</b>         | vector with temporal<br>iterations           | 1 x it+1              |
| <b>diffloglya</b> | local slope of loglya<br>per iteration       | 1 x length(1:resl:it) |

|                  |                           |            |
|------------------|---------------------------|------------|
| <b>residuals</b> | residuals of line fitting | 1 x (it/3) |
|------------------|---------------------------|------------|

#### 4.1.3. Ragwitz Estimator

**Function:** *nta\_ragwitz.m*

**Dependencies:** *nta\_autocorr.m, nta\_phasespace.m, nta\_neighsearch.mexw64*

**Description:** Local constant predictor used for estimation of phase-space embedding parameters  $d$  and  $\tau$  of Markov models (Ragwitz and Kantz 2002). Optimal choices for  $d$  and  $\tau$  comprise combinations of both for which the root mean squared predication error is minimum:

$$\hat{\mathbf{x}}_{t+1}^{d_x}(d, \tau) = \frac{1}{|U_\varepsilon(\mathbf{x}_t^{d_x})|} \sum_{\mathbf{x}_{t-\Delta t}^{d_x} \in U_\varepsilon(\mathbf{x}_t^{d_x})} \mathbf{x}_{t-\Delta t+1}^{d_x} \quad (7)$$

$$\text{RMSPE}(d, \tau) = \sqrt{\frac{\sum_{t=1}^N (\hat{\mathbf{x}}_{t+\Delta t}^{d_x} - \mathbf{x}_{t+\Delta t}^{d_x})^2}{N}} \quad (8)$$

$U$ : local neighbourhood,  $\varepsilon$ : neighbourhood-size,  $N$ : number of points.

*Table 6: Configuration Structure (cfg)*

| Field Name  | Description  | Size | Data Type | Default     |
|-------------|--|------|-----------|-------------|
| <b>taus</b> | vector of tau to scan either in<br>multiple of autocorrelation time<br>in percent  | 1xN  | int       | [10:10:100] |
| <b>dims</b> | min. & max. embedding<br>dimension   | 1x2  | int       | [2 9]       |
| <b>mode</b> | how to define tau (taus).<br>'samples': taus in samples or<br>'multi'=taus in multiple of<br>autocorrelation time in percent | char | char      | 'multi'     |
| <b>mass</b> | number of neighbours per point   | 1x1  | int       | 4           |

|                |  |            |      |           |
|----------------|--|------------|------|-----------|
| <b>hor</b>     | prediction horizon                     | 1x1        | int  | 1         |
| <b>metric</b>  | distance norm 'euclidean' or 'maximum' | 1x7 or 1x9 | char | 'maximum' |
| <b>plt</b>     | plot results yes/no [1/0]              | 1x1        | int  | 0         |
| <b>verbose</b> | verbose level [1/0]                    | 1x1        | int  | 1         |

*Table 7: Output Structure (results)*

| Field Name    | Description   | Size                                   |
|---------------|---|--|
| <b>cfg</b>    | configuration structure   | struct                                 |
| <b>tauo</b>   | optimum tau   | 1x1                                    |
| <b>dimopt</b> | optimum dimension   | 1x1                                    |
| <b>rmspe</b>  | mean squared prediction error<br>normalized to standard deviation of data | length(dims(1):dims(2)) x length(taus) |
| <b>prges</b>  | first rows: predicted values, second rows: original values                | 2 x dims x taus                        |

#### 4.1.4. False Nearest Neighbours

**Function:** *nta\_fnn.m*

**Dependencies:** *nta\_amutibin.m, nta\_phasespace.m*

**Description:** False Nearest Neighbours Algorithm by (Kennel et al. 1992). Calculates the number of false nearest neighbours  $fnn$  as a function of embedding dimensions  $d$ . False neighbours arise due to projections caused by an insufficient embedding dimension:

$$fnn(d) = \sum_i \theta \left( \frac{|x_{i+1} - n(x_i)_{j+1}|}{\|x_i^d - n_{xi}^d\|} - Rtol \right) \wedge \theta \left( \sqrt{(\|x_i^d - n_{xi}^d\|)^2 - (|x_{i+1} - n(x_i)_{j+1}|)^2} - STD(X) * Atol \right) \quad (9)$$

$n_x$ : next neighbour of  $x$ ,  $\| \cdot \|$ : distance norm,  $\theta$ : Heaviside-step function,  $Rtol$ : distance threshold,  $Atol$ : loneliness threshold,  $STD$ : standard deviation.

*Table 8: Configuration Structure (cfg)*

| Field Name     | Description  | Size | Data Type | Default |
|----------------|--|------|-----------|---------|
| <b>tau</b>     | embedding delay. If tau=0, the first minimum of the auto mutual information is used. | 1x1  | int       | 0       |
| <b>dim</b>     | max embedding dimensions   | 1x1  | int       | 9       |
| <b>Rtol</b>    | threshold parameter for distance   | 1x1  | double    | 10      |
| <b>Atol</b>    | threshold parameter for loneliness   | 1x1  | double    | 2       |
| <b>thr</b>     | threshold parameter for first minimum of false neighbours (%)                        | 1x1  | double    | 1       |
| <b>plt</b>     | plot results yes/no [1/0]  | 1x1  | int       | 0       |
| <b>verbose</b> | verbose level [1/0]  | 1x1  | int       | 1       |

*Table 9: Output Structure (results)*

| Field Name | Description             | Size   |
|------------|-------------------------|--------|
| <b>cfg</b> | configuration structure | struct |

|                 |   |                             |
|-----------------|---|-----------------------------|
| <b>fnn</b>      | first row: % of false nearest<br>neighbours, second row: tested<br>dimensions | 2 x length(dims(1):dims(2)) |
| <b>firstmin</b> | dimension at first zero<br>crossing   | 1x1                         |

#### 4.1.5. Space-Time Separation Plot

**Function:** *nta\_spacetimesep.m*

**Dependencies:** *nta\_amutibin.m, nta\_phasespace.m, nta\_neighsearch.mexw64*

**Description:** Space-Time-Separation-Plot. Can be used to determine Theiler-window (Provenzale et al. 1992). It shows the proportion of points closer than a distance  $\epsilon$  at a given time separation  $\Delta t$  as a function of  $\Delta t$ :

$$H_{to}(\Delta t) = p(||\mathbf{x}(t + \Delta t) - \mathbf{x}(t)|| < \epsilon) \quad (10)$$

$||..||$ : distance norm.

*Table 10: Configuration Structure (cfg)*

| Field Name    | Description  | Size       | Data Type | Default   |
|---------------|--|------------|-----------|-----------|
| <b>tau</b>    | embedding delay  | 1x1        | int       | 1         |
| <b>dim</b>    | embedding dimension  | 1x1        | int       | 2         |
| <b>nr</b>     | Number of classes (succeeding classes contain an increasing proportion of points in phase-space) | 1x1        | int       | 5         |
| <b>maxlag</b> | maximum number of temporal lags  | 1x1        | int       | 100       |
| <b>metric</b> | distance norm 'euclidean' or 'maximum'   | 1x7 or 1x9 | char      | 'maximum' |
| <b>plt</b>    | plot results yes/no [1/0]  | 1x1        | int       | 1         |

|                |                     |     |     |   |
|----------------|---------------------|-----|-----|---|
| <b>verbose</b> | verbose level [1/0] | 1x1 | int | 1 |
|----------------|---------------------|-----|-----|---|

Table 11: Output Structure (results)

| Field Name         | Description   | Size        |
|--------------------|---|-------------|
| <b>cfg</b>         | configuration structure   | struct      |
| <b>hto</b>         | space-time-matrix:<br><br>matrix containing proportions of state-space<br><br>vector with a given distance at a specific time lag | nr x maxlag |
| <b>lagvector</b>   | vector containing<br><br>temporal lags  | 1 x maxlag  |
| <b>classvector</b> | vector containing<br><br>histogram intervals of distances   | 1 x nr      |

#### 4.1.6. Time Inversion

**Function:** *nta\_timerev.m*

**Dependencies:** *nta\_surrogates.m*

**Description:** Test for time reversibility. In conjunction with appropriate surrogates, timerev can be used to test for nonlinearity ( $H_0$  = linear dynamics,  $\alpha=0.05$ ) (Schreiber and Schmitz 1997).

$$Q_t = \langle (x_t - x_{t-1})^3 \rangle \quad (11)$$

<...>: mean

Table 12: Configuration Structure (cfg)

| Field Name | Description | Size | Data Type | Default |
|------------|-------------|------|-----------|---------|
| <b>lag</b> | delay       | 1x1  | int       | 1       |

|                 |   |     |     |     |
|-----------------|---|-----|-----|-----|
| <b>numsurr</b>  | number of surrogates  | 1x1 | int | 100 |
| <b>surrmode</b> | mode==1: random shuffling<br><br>mode==2: phase randomization<br><br>mode==3: amplitude adjusted phase randomization<br><br>mode==4: cut time series at random point and flip second half<br><br>mode==5: cut time series at random point and switch halves | 1x1 | int | 2   |
| <b>numit</b>    | Number of iterations for amplitude adjustment (mode==3)   | 1x1 | int | 10  |
| <b>verbose</b>  | verbose level [1/0]   | 1x1 | int | 1   |

*Table 13: Output Structure (results)*

| Field Name   | Description                  | Size   |
|--------------|------------------------------|--------|
| <b>cfg</b>   | configuration structure      | struct |
| <b>Qt</b>    | time reversibility statistic | 1x1    |
| <b>zstat</b> | z-score of surrogate test    | 1x1    |
| <b>sig</b>   | H0 rejected [1] or not [0]   | 1x1    |

#### 4.1.7. Hurst Exponent

**Function:** *nta\_hurst.m*

**Dependencies:** -



**Description:** Estimation of Hurst exponent H as a measure for self-affinity of a time series (Hurst 1951):

$$\frac{R(n)}{STD(n)} = C * n^H \quad (12)$$

$$R(n) = \max \left( \sum_{i=1}^t (x_t - \bar{x})_i \right)_t - \min \left( \sum_{i=1}^t (x_t - \bar{x})_i \right)_t, \quad t = 1..n \quad (13)$$

STD: standard deviation, n: window size,  $\bar{x}$ : average.

*Table 14: Configuration Structure (cfg)*

| Field Name     | Description               | Size | Data Type | Default |
|----------------|---------------------------|------|-----------|---------|
| <b>scale</b>   | max. number of windows    | 1x1  | int       | 10      |
| <b>plt</b>     | plot results yes/no [1/0] | 1x1  | int       | 1       |
| <b>verbose</b> | verbose level [1/0]       | 1x1  | int       | 1       |

*Table 15: Output Structure (results)*

| Field Name           | Description               | Size      |
|----------------------|---------------------------|-----------|
| <b>cfg</b>           | configuration structure   | struct    |
| <b>expo</b>          | estimate of the exponent  | 1x1       |
| <b>lognges</b>       | log of window sizes       | 1 x scale |
| <b>logmeanrr</b>     | log of rescaled ranges    | 1 x scale |
| <b>residuals</b>     | residuals of line fitting | 1 x scale |
| <b>meanresiduals</b> | mean of residuals         | 1x1       |

#### 4.1.8. Detrended Fluctuation Analysis

**Function:** *nta\_dfa.m*

**Dependencies:** -

**Description:** Calculation of detrended fluctuation analysis (DFA) to detect long-range correlations i.e. self-affinity, similar to the Hurst exponent  $H$  (Peng et al. 1994). In contrast to  $H$  DFA, can also be applied to nonstationary data:

$$X_t = \sum_{i=1}^t x_i - \bar{x} \quad (14)$$

$$F(n) = \sqrt{\frac{1}{N} \sum_{t=1}^N (X_t - Y_t)^2} \quad (15)$$

$$F(n) \propto n^a \quad (16)$$

$X_t$ : unbounded process,  $\bar{x}$ : mean of  $x$ ,  $n$ : window size,  $N$ : number of windows,  $Y_t$ : local linear fit of each window,  $a$ : scaling exponent.

Table 16: Configuration Structure (cfg)

| Field Name     | Description                 | Size | Data Type | Default |
|----------------|-----------------------------|------|-----------|---------|
| <b>scales</b>  | min & max number of windows | 1x2  | int       | [2 10]  |
| <b>plt</b>     | plot results yes/no [1/0]   | 1x1  | int       | 1       |
| <b>verbose</b> | verbose level [1/0]         | 1x1  | int       | 1       |

Table 17: Output Structure (results)

| Field Name       | Description                  | Size                            |
|------------------|------------------------------|---------------------------------|
| <b>cfg</b>       | configuration structure      | struct                          |
| <b>expo</b>      | estimate of the exponent     | 1x1                             |
| <b>logbox</b>    | log of window sizes          | 1 x length(scales(1):scales(2)) |
| <b>logF</b>      | log of temporal fluctuations | 1 x length(scales(1):scales(2)) |
| <b>residuals</b> | residuals of line fitting    | 1 x length(scales(1):scales(2)) |

|                      |                   |     |
|----------------------|-------------------|-----|
| <b>meanresiduals</b> | mean of residuals | 1x1 |
|----------------------|-------------------|-----|

#### 4.1.9. Unstable Periodic Orbit Transform

**Function:** *nta\_upo.m*

**Dependencies:** *nta\_amutibin.m, nta\_phasespace.m, nta\_surrogates.m*

**Description:** Function to detect unstable periodic orbits in 2-dimensional phase-space (So et al. 1996). The algorithm concentrates points on nearby unstable period-one fixpoints. Significance of concentrated fixpoints may be tested by performing a surrogate test.

$$\hat{\mathbf{z}}_n = (\mathbf{1} - \mathbf{S}_n)^{-1}(\mathbf{z}_{n+1} - \mathbf{S}_n \mathbf{z}_n) \quad (17)$$

$$\mathbf{S}_n = \begin{pmatrix} a_n^1 & \dots & a_n^d \\ \mathbf{1} & & \mathbf{0} \end{pmatrix} + \kappa \mathbf{R} \|\mathbf{z}_{n+1} - \mathbf{z}_n\| \quad (18)$$

$$\begin{pmatrix} a_n^1 \\ \vdots \\ a_n^d \end{pmatrix} = \begin{pmatrix} (\mathbf{z}_n - \mathbf{z}_{n-1})^\dagger \\ \vdots \\ (\mathbf{z}_{n-(d-1)} - \mathbf{z}_{n-d})^\dagger \end{pmatrix}^{-1} \begin{pmatrix} z_{n+1}^1 - z_n^1 \\ \vdots \\ z_{n-(d-2)}^1 - z_{n-(d-1)}^1 \end{pmatrix} \quad (19)$$

$\mathbf{z}_n$ : nth phase-space vector,  $\hat{\mathbf{z}}_n$ : transformed phase-space vector,  $\mathbf{1}$ : identity matrix,  $d$ : dimension,  $\kappa$ : diffusion parameter,  $\mathbf{R}$ :  $d \times d$  random matrix drawn from the uniform interval  $[-1, 1]$ ,  $\|\cdot\|$  distance norm,  $\dagger$ : transpose,  $z_n^1$ : first element of nth phase-space vector.

Table 18: Configuration Structure (cfg)

| Field Name | Description | Size | Data Type | Default |
|------------|-------------|------|-----------|---------|
|------------|-------------|------|-----------|---------|

|                 |   |     |     |         |
|-----------------|---|-----|-----|---------|
| <b>tau</b>      | embedding delay   | 1x1 | int | 1       |
| <b>numit</b>    | number of iterations for upo transform  | 1x1 | int | 200     |
| <b>bins</b>     | number of bins per dimension  | 1x2 | int | [10 10] |
| <b>numsurr</b>  | number of surrogates  | 1x1 | int | 0       |
| <b>surrmode</b> | mode==1: random shuffling<br>mode==2: phase randomization<br>mode==3: amplitude adjusted phase randomization<br>mode==4: cut time series at random point and flip second half<br>mode==5: cut time series at random point and switch halves | 1x1 | int | 2       |
| <b>plt</b>      | plot results yes/no [1/0]   | 1x1 | int | 1       |
| <b>verbose</b>  | verbose level [1/0]   | 1x1 | int | 1       |

Table 19: Output Structure (results)

| Field Name        | Description   | Size                         |
|-------------------|---|------------------------------|
| <b>cfg</b>        | configuration structure                                   | struct                       |
| <b>bincenters</b> | center coordinates of<br>histogram bins                   | {1 x bins(1)}, {1 x bins(2)} |
| <b>countorig</b>  | normalized histogram counts<br>of original phase-space    | bins(1) x bins(2)            |
| <b>counttrans</b> | normalized histogram counts<br>of transformed phase-space | bins(1) x bins(2)            |

|                  |  |                   |
|------------------|--|-------------------|
| <b>countsurr</b> | normalized histogram counts<br>of surrogates                             | bins(1) x bins(2) |
| <b>statcount</b> | z-score map of transformed<br>phase space                                | bins(1) x bins(2) |
| <b>histdiag</b>  | zscores of center diagonal<br>of original and transformed<br>phase-space | bins(1) x 2       |

## 4.2. Recurrence Measures

### 4.2.1. Recurrence Plot

**Function:** *nta\_recurrenceplot.m*

**Dependencies:** *nta\_amutibin.m, nta\_phasespace.m, nta\_neighsearch.mexw64*

**Description:** Calculates a recurrence matrix  $M$  as well as different recurrence based quantities (Eckmann, J-P., S. Oliffson Kamphorst, and David Ruelle. 1987; Little et al. 2007; Marwan et al. 2002; Romano et al. 2005).

$$M_{t,t+\Delta t} = \Theta(\varepsilon - ||\mathbf{x}_t - \mathbf{x}_{t+\Delta t}||) \quad (20)$$

$$RR = \frac{1}{N^2} \sum_{i,j=1}^N M(i,j) \quad (21)$$

$$DET = \frac{\sum_{l=\min l}^N lp(l)}{\sum_{l=1}^N lp(l)} \quad (22)$$

$$LAM = \frac{\sum_{v=\min l}^N vp(v)}{\sum_{v=1}^N vp(v)} \quad (23)$$

$$T = (t + \Delta t - \Delta t^{\text{exit}}) - (t + \Delta t^{\text{enter}}) \quad (24)$$

$$P(T) = \frac{R(T)}{\sum_{i=T_{\min}}^{T_{\max}} R(i)}, T = T_{\min} \dots T_{\max} \quad (25)$$

$$R_*(T) = \frac{R(T)}{T_{\max} - (T - 1)}, \quad (26)$$

$$RPDE = -(\log_2 T_{\max})^{-1} \sum_{t=1}^{T_{\max}} P(T) \log_2 P(T) \quad (27)$$

$\Theta$ : Heaviside-step function,  $||..||$ : distance norm,  $\epsilon$ : neighbourhood-size,  $N$ : number of points,  $\min l$ : minimum length of diagonal/vertical lines,  $p$ : frequency distribution of line segments,  $T$ : recurrence time,  $\Delta t^{\text{enter}}$ : sample difference between  $\mathbf{x}_t$  reentering  $U_\epsilon$  and  $\mathbf{x}_{t+\Delta t}$ ,  $\Delta t^{\text{exit}}$ : difference in samples between  $\mathbf{x}_t$  and  $\mathbf{x}_t$  first leaving  $U_\epsilon$ ,  $R(T)$ : histogram with bin size equal to 1 sample and a number of bins equal to the longest recurrence time  $T_{\max}$ ,  $R(T)^*$ : normalized recurrence histogram,  $P(T)$ : recurrence period

probabilities as a function of recurrence times, RR: recurrence rate, DET: determinism, LAM: laminarity, RPDE: recurrence period density entropy.

*Table 20: Configuration Structure (cfg)*

| Field Name          | Description  | Size | Data Type | Default |
|---------------------|--|------|-----------|---------|
| <b>tau</b>          | embedding delay  | 1x1  | int       | 0       |
| <b>dim</b>          | embedding dimension  | 1x1  | int       | 2       |
| <b>en</b>           | neighbourhood-size in % std of data  | 1x1  | double    | 0       |
| <b>rr</b>           | recurrence rate in % of total possible rr  | 1x1  | double    | 5       |
| <b>minlengthdet</b> | minimum length of diagonal lines used for determinism estimation   | 1x1  | int       | 0       |
| <b>minlengthlam</b> | minimum length of vertical lines used for laminarity estimation  | 1x1  | int       | 0       |
| <b>minmaxRecPD</b>  | minimum and maximum recurrence periods. Maximum must be smaller than $\text{length}(\text{input}) - (\text{tau} * \text{dim} * ((\text{dim} - 1) / \text{dim}))$ . | 1x2  | int       | [0 0]   |
| <b>singlenei</b>    | use only nearest neighbour for calculation of recurrence periods<br>yes/no [1/0]   | 1x1  | int       | 1       |
| <b>norm</b>         | normalize recurrence frequencies by total possible number of recurrences per period bin yes/no [1/0]   | 1x1  | int       | 0       |

|                |  |            |      |           |
|----------------|--|------------|------|-----------|
| <b>metric</b>  | distance norm 'euclidean' or 'maximum' | 1x7 or 1x9 | char | 'maximum' |
| <b>plt</b>     | plot results yes/no [1/0]              | 1x1        | int  | 1         |
| <b>verbose</b> | verbose level [1/0]                    | 1x1        | int  | 1         |

Table 21: Output Structure (results)

| Field Name     | Description  | Size                                      |
|----------------|--|---|
| <b>cfg</b>     | configuration structure                                  | struct                                    |
| <b>rr</b>      | recurrence rate  | 1x1                                       |
| <b>det</b>     | determinism  | 1x1                                       |
| <b>lam</b>     | laminariy  | 1x1                                       |
| <b>ratio</b>   | ratio of determinism and rr                              | 1x1                                       |
| <b>sumdist</b> | thresholded recurrence matrix                            | NxN, N=size(embedding matrix,1)           |
| <b>alldist</b> | untresholded recurrence matrix                           | NxN, N=size(embedding matrix,1)           |
| <b>ht</b>      | recurrence period probabilities as a function of periods | 1 x length(minmaxRecPD(1):minmaxRecPD(2)) |
| <b>gaco</b>    | generalized autocorrelation as a function of lags        | 1xN, N=size(embedding matrix,1)           |
| <b>rpde</b>    | recurrence period density entropy                        | 1x1                                       |

#### 4.2.2. Joint Recurrence Plot

**Function:** *nta\_jointrecurrenceplot.m*

**Dependencies:** *nta\_recurrenceplot.m*



**Description:** Calculates a joint recurrence matrix and recurrence based measures (Romano et al. 2004; Little et al. 2007; Marwan et al. 2002; Romano et al. 2005). It is defined as the Hadamard product of the individual recurrence plots of multiple time series:

$$JR_{t,t+\Delta t} = \Theta(\varepsilon - ||\mathbf{x}_t - \mathbf{x}_{t+\Delta t}||) * \Theta(\varepsilon - ||\mathbf{y}_t - \mathbf{y}_{t+\Delta t}||) \quad (28)$$

$\Theta$ : Heaviside-step function,  $||..||$ : distance norm,  $\varepsilon$ : neighbourhood-size.

*Table 22: Configuration Structure (cfg)*

| Field Name          | Description   | Size | Data Type | Default |
|---------------------|---|------|-----------|---------|
| <b>taus</b>         | embedding delays  | 1x2  | int       | [0 0]   |
| <b>dims</b>         | embedding dimensions  | 1x2  | int       | [2 2]   |
| <b>ens</b>          | neighbourhood-sizes in % std of data  | 1x2  | int       | [0 0]   |
| <b>rrs</b>          | recurrence rates in % of total possible rr  | 1x2  | int       | [5 5]   |
| <b>minmaxRecPD</b>  | minimum and maximum recurrence periods. Maximum must be smaller than length(input)-(tau*dim*((dim-1)/dim)). | 1x2  | int       | [0 0]   |
| <b>minlengthdet</b> | minimum length of diagonal lines used for determinism estimation  | 1x1  | int       | 0       |
| <b>minlengthlam</b> | minimum length of vertical lines used for laminarity estimation   | 1x1  | int       | 0       |
| <b>singlenei</b>    | use only nearest neighbour for calculation of recurrence periods yes/no [1/0]                               | 1x1  | int       | 1       |
| <b>norm</b>         | normalize recurrence frequencies by total possible number of recurrences per period bin yes/no [1/0]        | 1x1  | int       | 0       |

|                |  |            |      |           |
|----------------|--|------------|------|-----------|
| <b>metric</b>  | distance norm 'euclidean' or 'maximum' | 1x7 or 1x9 | char | 'maximum' |
| <b>plt</b>     | plot results yes/no [1/0]              | 1x1        | int  | 1         |
| <b>verbose</b> | verbose level [1/0]                    | 1x1        | int  | 1         |

Table 23: Output Structure (results)

| Field Name     | Description   | Size                                      |
|----------------|---|---|
| <b>cfg</b>     | configuration structure                                 | struct                                    |
| <b>rr</b>      | recurrence rate   | 1x1                                       |
| <b>det</b>     | determinism   | 1x1                                       |
| <b>lam</b>     | laminariy   | 1x1                                       |
| <b>ratio</b>   | ratio of determinism and rr                             | 1x1                                       |
| <b>sumdist</b> | thresholded recurrence matrix                           | NxN, N=size(embedding matrix,1)           |
| <b>alldist</b> | untresholded recurrence matrix                          | NxN, N=size(embedding matrix,1)           |
| <b>ht</b>      | recurrence period probabilitas as a function of periods | 1 x length(minmaxRecPD(1):minmaxRecPD(2)) |
| <b>gaco</b>    | generalized autocorrelation as a function of lags       | 1xN, N=size(embedding matrix,1)           |
| <b>rpde</b>    | recurrence period density entropy                       | 1x1                                       |

#### 4.2.3. Cross Recurrence Plot

**Function:** *nta\_crossrecurrenceplot.m*

**Dependencies:** *nta\_amutibin.m, nta\_phasespace.m, nta\_recurrenceplot.m*

**Description:** Calculates a cross recurrence matrix and recurrence based measures (Marwan and Kurths 2002; Little et al. 2007; Marwan et al. 2002; Romano et al. 2005). For generation of a cross recurrence plot, multiple time series are embedded into the same phase

space. A recurrence matrix is then calculated for the joint embedding space according to:

$$CR_{t,t+\Delta t} = \Theta(\varepsilon - ||\mathbf{x}_t - \mathbf{y}_{t+\Delta t}||) \quad (29)$$

$\Theta$ : Heaviside-step function,  $||..||$ : distance norm,  $\varepsilon$ : neighbourhood-size.

Table 24: Configuration Structure (cfg)

| Field Name          | Description   | Size | Data Type | Default |
|---------------------|---|------|-----------|---------|
| <b>tau</b>          | embedding delay   | 1x1  | int       | 1       |
| <b>dim</b>          | embedding dimension   | 1x1  | int       | 2       |
| <b>en</b>           | neighbourhood-size in % std of data   | 1x1  | double    | 0       |
| <b>rr</b>           | recurrence rate in % of total possible rr   | 1x1  | double    | 5       |
| <b>minmaxRecPD</b>  | minimum and maximum recurrence periods.<br><br>Maximum must be smaller than<br><br>length(input)-(tau*dim*((dim-1)/dim)). | 1x2  | int       | [0 0]   |
| <b>minlengthdet</b> | minimum length of diagonal lines used for<br><br>determinism estimation   | 1x1  | int       | 0       |
| <b>minlengthlam</b> | minimum length of vertical lines used for<br><br>laminarity estimation  | 1x1  | int       | 0       |
| <b>singlenei</b>    | use only nearest neighbour for calculation of<br><br>recurrence periods yes/no [1/0]                                      | 1x1  | int       | 1       |
| <b>norm</b>         | normalize recurrence frequencies by total possible<br><br>number of recurrences per period bin yes/no [1/0]               | 1x1  | int       | 0       |
| <b>plt</b>          | plot results yes/no [1/0]   | 1x1  | int       | 1       |
| <b>verbose</b>      | verbose level [1/0]   | 1x1  | int       | 1       |

Table 25: Output Structure (results)

| Field Name     | Description   | Size                                     |
|----------------|---|--|
| <b>cfg</b>     | configuration structure                                     | struct                                   |
| <b>rr</b>      | recurrence rate   | 1x1                                      |
| <b>det</b>     | determinism   | 1x1                                      |
| <b>lam</b>     | laminariy   | 1x1                                      |
| <b>ratio</b>   | ratio of determinism and rr                                 | 1x1                                      |
| <b>sumdist</b> | thresholded recurrence matrix                               | NxN, N=size(embedding matrix,1)          |
| <b>alldist</b> | untresholded recurrence matrix                              | NxN, N=size(embedding matrix,1)          |
| <b>ht</b>      | recurrence period probabilities as a<br>function of periods | 1x length(minmaxRecPD(1):minmaxRecPD(2)) |
| <b>gaco</b>    | generalized autocorrelation as a<br>function of lags        | 1xN, N=size(embedding matrix,1)          |
| <b>rpde</b>    | recurrence period density entropy                           | 1x1                                      |

#### 4.2.4. Recurrence Frequencies\_over range of neighbourhoods

**Function:** *nta\_recfreq\_en\_scan.m*

**Dependencies:** *nta\_recurrenceplot.m*

**Description:** Calculates recurrence frequencies as well as different recurrence based quantities over a range of neighbourhood-sizes (Little et al. 2007; Marwan et al. 2002; Romano et al. 2005; Eckmann, J-P., S. Oliffson Kamphorst, and David Ruelle. 1987).

$$P(T, \varepsilon) = \frac{R(T)}{\sum_{i=T_{\min}}^{T_{\max}} R(i)}, T = T_{\min} \dots T_{\max} \quad (30)$$

T: recurrence time,  $\epsilon$ : neighbourhood-size, R(T): histogram with bin size equal to 1 sample and a number of bins equal to the longest recurrence time  $T_{\max}$ , P(T): recurrence period probabilities as a function of recurrence times.

Table 26: Configuration Structure (cfg)

| Field Name         | Description  | Size             | Data Type | Default   |
|--------------------|--|------------------|-----------|-----------|
| <b>tau</b>         | embedding delay  | 1x1              | int       | 0         |
| <b>dim</b>         | embedding dimension  | 1x1              | int       | 2         |
| <b>ens</b>         | Neighbourhood-size in % std of data.<br><br>Specify minimum size, step-size and maximum size.  | 1x3              | double    | [1 1 101] |
| <b>minmaxRecPD</b> | minimum and maximum recurrence periods.<br><br>Maximum must be smaller than<br><br>$\text{length}(\text{input}) - (\text{tau} * \text{dim} * ((\text{dim} - 1) / \text{dim}))$ | 1x2              | int       | [0 0]     |
| <b>singlenei</b>   | use only nearest neighbour for calculation of recurrence periods yes/no [1/0]  | 1x1              | int       | 1         |
| <b>norm</b>        | normalize recurrence frequencies by total possible number of recurrences per period bin yes/no [1/0]   | 1x1              | int       | 0         |
| <b>metric</b>      | distance norm 'euclidean' or 'maximum'   | 1x7<br>or<br>1x9 | char      | 'maximum' |
| <b>plt</b>         | plot results yes/no [1/0]  | 1x1              | int       | 1         |
| <b>verbose</b>     | verbose level [1/0]  | 1x1              | int       | 1         |

Table 27: Output Structure (results)

| Field Name  | Description                      | Size                                  |
|-------------|----------------------------------|---------------------------------------|
| <b>cfg</b>  | configuration structure          | struct                                |
| <b>rr</b>   | recurrence rate                  | 1x length([ens(1):ens(2):ens(3)])     |
| <b>ht</b>   | recurrence period probabilitites | length([ens(1):ens(2):ens(3)]) x      |
|             | as a function of periods         | length(minmaxRecPD(1):minmaxRecPD(2)) |
| <b>gaco</b> | generalized autocorrelation as   | length([ens(1):ens(2):ens(3)]) x N,   |
|             | a function of lags               | N=size(embedding matrix,1)            |
| <b>rpde</b> | recurrence period density        | 1x length([ens(1):ens(2):ens(3)])     |
|             | entropy                          |                                       |

#### 4.2.5. Windowed Recurrence Frequencies

**Function:** *nta\_wind\_recfreq.m*

**Dependencies:** *nta\_recurrenceplot.m*

**Description:** Calculates windowed recurrence frequencies as well as other recurrence based quantities with an overlap of half a window (Little et al. 2007; Marwan et al. 2002; Romano et al. 2005; Eckmann, J-P., S. Oliffson Kamphorst, and David Ruelle. 1987).

$$P(T, w_n) = \frac{R(T, w_n)}{\sum_{i=T_{\min}}^{T_{\max}} R(i, w_n)}, T = T_{\min} \dots T_{\max} \quad (31)$$

T: recurrence time,  $w_n$ : nth temporal window of input time series  $x [x_n \dots x_{\text{window size}}]$ , R(T): histogram with bin size equal to 1 sample and a number of bins equal to the longest recurrence time  $T_{\max}$ , P(T): recurrence period probabilities as a function of recurrence times.

Table 28: Configuration Structure (cfg)

| Field Name          | Description  | Size       | Data Type | Default   |
|---------------------|--|------------|-----------|-----------|
| <b>tau</b>          | embedding delay  | 1x1        | int       | 0         |
| <b>dim</b>          | embedding dimension  | 1x1        | int       | 2         |
| <b>en</b>           | neighbourhood-size in % std of data  | 1x1        | double    | 5         |
| <b>window</b>       | window size in samples.Parameter must be even number   | 1x1        | int       | 1/10      |
| <b>fs</b>           | Sampling rate  | 1x1        | int       | -         |
| <b>minlengthdet</b> | minimum length of diagonal lines used for determinism estimation   | 1x1        | int       | 0         |
| <b>minlengthlam</b> | minimum length of vertical lines used for laminarity estimation  | 1x1        | int       | 0         |
| <b>minmaxRecPD</b>  | minimum and maximum recurrence periods. Maximum must be smaller than $\text{window} - (\text{tau} * \text{dim} * ((\text{dim} - 1) / \text{dim}))$ . | 1x2        | int       | [0 0]     |
| <b>singlenei</b>    | use only nearest neighbour for calculation of recurrence periods yes/no [1/0]  | 1x1        | int       | 1         |
| <b>norm</b>         | normalize recurrence frequencies by total possible number of recurrences per period bin yes/no [1/0]   | 1x1        | int       | 0         |
| <b>metric</b>       | distance norm 'euclidean' or 'maximum'   | 1x7 or 1x9 | char      | 'maximum' |
| <b>plt</b>          | plot results yes/no [1/0]  | 1x1        | int       | 1         |
| <b>verbose</b>      | verbose level [1/0]  | 1x1        | int       | 1         |

Table 29: Output Structure (results)

| Field Name   | Description  | Size  |
|--------------|--|---|
| <b>cfg</b>   | configuration structure                                      | struct  |
| <b>time</b>  | time vector in samples                                       | time x 1  |
| <b>resht</b> | recurrence period probabilitites as a<br>function of periods | length(minmaxRecPD(1):minmaxRecPD(2)) x<br>time |
| <b>rr</b>    | recurrence rate  | 1 x number of windows                           |
| <b>det</b>   | determinism  | 1 x number of windows                           |
| <b>lam</b>   | laminariy  | 1 x number of windows                           |
| <b>ratio</b> | ratio of determinism and rr                                  | 1 x number of windows                           |
| <b>gaco</b>  | generalized autocorrelation as a<br>function of lags         | 1xN, N=size(embedding matrix,1)                 |
| <b>rpde</b>  | recurrence period density entropy                            | 1 x number of windows                           |

## 4.3. Information Theoretic Measures

### 4.3.1. Shannon Entropy (Bin Estimator)

**Function:** *nta\_entropybin.m*

**Dependencies:** *nta\_freddiac.m*

**Description:** Calculates the Shannon entropy H and Shannon information SI using a binning estimator (Shannon CE 1949).

$$H_{\text{bin}}(X) = - \sum_x p_X(X = x) \log_2 p_X(X = x) \quad (32)$$

$$SI_{\text{bin}}(x) = - \log_2 p(x) \quad (33)$$



$P_x$ : probability of variable X taking the value x.

Table 30: Configuration Structure (cfg)

| Field Name     | Description   | Size | Data Type | Default |
|----------------|---|------|-----------|---------|
| <b>numbin</b>  | number of bins. If numbin=0, the number of bins gets optimized according to the Freedman-Diaconis rule. | 1x1  | int       | 0       |
| <b>verbose</b> | verbose level [1/0]   | 1x1  | int       | 1       |

Table 31: Output Structure (results)

| Field Name  | Description              | Size                 |
|-------------|--------------------------|----------------------|
| <b>cfg</b>  | configuration structure  | struct               |
| <b>Hx</b>   | Shannon entropy [bit]    | 1x1                  |
| <b>SI</b>   | Shannon information      | 1xN, N=length(input) |
| <b>dist</b> | Probability distribution | 1 x numbin           |

#### 4.3.2. Differential Entropy (Kozachenko's Estimator)

**Function:** *nta\_entropykozachenko.m*

**Dependencies:** *nta\_amutibin.m, nta\_phasespace.m, nta\_neighsearch.mexw64*

**Description:** Calculation of differential entropy H and local differential entropy SI using a nearest neighbour estimator (Kozachenko, L. F., and Nikolai N. Leonenko. 1987).

$$H_K(X) = -\psi(\text{mass}) + \psi(N) + \frac{d}{N} * \sum_{i=1}^N \varepsilon(i) \quad (34)$$

$$SI_K(x) = -\psi(\text{mass}) + \psi(N) + d * \varepsilon(x) \quad (35)$$

$\Psi$ : digamma function, mass: number of nearest neighbours, N: total number of points, d: dimension,  $\varepsilon$ : neighbourhood-diameter.

Table 32: Configuration Structure (cfg)

| Field Name     | Description  | Size       | Data Type | Default   |
|----------------|--|------------|-----------|-----------|
| <b>tau</b>     | embedding delay  | 1x1        | int       | 1         |
| <b>dim</b>     | embedding dimension  | 1x1        | int       | 2         |
| <b>mass</b>    | number of neighbours used for probability density estimation | 1x1        | int       | 4         |
| <b>metric</b>  | distance norm 'euclidean' or 'maximum'                       | 1x7 or 1x9 | char      | 'maximum' |
| <b>verbose</b> | verbose level [1/0]  | 1x1        | int       | 1         |

Table 33: Output Structure (results)

| Field Name | Description                 | Size                 |
|------------|-----------------------------|----------------------|
| <b>cfg</b> | configuration structure     | struct               |
| <b>Hx</b>  | Differential entropy [nats] | 1x1                  |
| <b>SI</b>  | Shannon information         | 1xN, N=length(input) |

#### 4.3.3. Mutual Information (Bin Estimator)

**Function:** *nta\_Mlbin.m*

**Dependencies:** *nta\_freddiac.m*

**Description:** Calculation of mutual information MI and local mutual information IMI between two signals using a binning estimator (Cover, Thomas M., and Joy A. Thomas. 1991).

$$MI_{\text{bin}}(X; Y) = H_{\text{bin}}(X) + H_{\text{bin}}(Y) - H_{\text{bin}}(X; Y) \quad (36)$$

$$IMI_{\text{bin}}(x; y) = SI_{\text{bin}}(y) + SI_{\text{bin}}(y) - SI_{\text{bin}}(x; y) \quad (37)$$

H: Shannon entropy.

*Table 34: Configuration Structure (cfg)*

| Field Name     | Description   | Size | Data Type | Default |
|----------------|---|------|-----------|---------|
| <b>numbin</b>  | number of bins. If numbin=0, the number of bins gets optimized according to the Freedman-Diaconis rule. | 1x1  | int       | 0       |
| <b>verbose</b> | verbose level [1/0]   | 1x1  | int       | 1       |

*Table 35: Output Structure (results)*

| Field Name | Description              | Size                 |
|------------|--------------------------|----------------------|
| <b>cfg</b> | configuration structure  | struct               |
| <b>MI</b>  | mutual information [bit] | 1x1                  |
| <b>IMI</b> | Local mutual information | 1xN, N=length(input) |

#### 4.3.4. Mutual Information (Kraskov's Estimator)

**Function:** *nta\_Mlkraskov.m*

**Dependencies:** *nta\_amutibin.m, nta\_phasespace.m, nta\_neighsearch.mexw64*

**Description:** Calculation of mutual information and local mutual information between two signals using a nearest neighbour estimator (Kraskov et al. 2004).

$$MI_K(X, Y) = \psi(\text{mass}) + \psi(N) + \frac{\sum_{i=1}^N \psi(n_{xi}) + \psi(n_{yi})}{N} \quad (38)$$

$$lMI_K(x, y) = \psi(\text{mass}) + \psi(N) + \psi(n_x) + \psi(n_y) \quad (39)$$

$\Psi$ : digamma function, mass: number of nearest neighbours, N: total number of points, n: number of neighbours.

Table 36: Configuration Structure (cfg)

| Field          | Description  | Size       | Data | Default   |
|----------------|--|------------|------|-----------|
| Name           |  |            | Type |           |
| <b>taus</b>    | embedding delays                                     | 1x2        | int  | [0 0]     |
| <b>dims</b>    | embedding dimensions                                 | 1x2        | int  | [2 2]     |
| <b>mass</b>    | number of nearest neighbours used for PDF estimation | 1x1        | int  | 4         |
| <b>metric</b>  | distance norm 'euclidean' or 'maximum'               | 1x7 or 1x9 | char | 'maximum' |
| <b>verbose</b> | verbose level [1/0]                                  | 1x1        | int  | 1         |

Table 37: Output Structure (results)

| Field Name | Description             | Size   |
|------------|-------------------------|--------|
| <b>cfg</b> | configuration structure | struct |

|            |                           |                      |
|------------|---------------------------|----------------------|
| <b>MI</b>  | mutual information [nats] | 1x1                  |
| <b>IMI</b> | Local mutual information  | 1xN, N=length(input) |

#### 4.3.5. Mutual Information Matrix (Bin & Kraskov)

**Function:** `nta_Mlmatrix.m`

**Dependencies:** `nta_Mlkraskov.m`, `nta_Mlbin.m`

**Description:** Column-wise calculation of mutual information using two estimators (Cover, Thomas M., and Joy A. Thomas. 1991; Kraskov et al. 2004). See 4.3.3 & 4.3.4 for details on algorithms.

*Table 38: Configuration Structure (cfg)*

| Field Name     | Description   | Size       | Data Type | Default   |
|----------------|---|------------|-----------|-----------|
| <b>mode</b>    | estimator type: bin ['bin'] or nearest neighbour ['nn']   | 1x2<br>1/3 | char      | 'bin'     |
| <b>numbin</b>  | number of bins. If numbin=0, the number of bins gets optimized according to the Freedman-Diaconis rule. | 1x1        | int       | 0         |
| <b>taus</b>    | embedding delays  | 1x2        | int       | [1 1]     |
| <b>dims</b>    | embedding dimensions  | 1x2        | int       | [2 2]     |
| <b>mass</b>    | number of nearest neighbours used for PDF estimation  | 1x1        | int       | 4         |
| <b>metric</b>  | distance norm 'euclidean' or 'maximum'  | 1x7 or 1x9 | char      | 'maximum' |
| <b>verbose</b> | verbose level [1/0]   | 1x1        | int       | 1         |

Table 39: Output Structure (results)

| Field Name      | Description                                 | Size            |
|-----------------|---|-----------------|
| <b>cfg</b>      | configuration structure                     | struct          |
| <b>MIMaabs</b>  | mutual information Matrix (absolute values) | numbin x numbin |
| <b>MIManorm</b> | mutual information Matrix (normalized)      | numbin x numbin |

#### 4.3.6. Auto-Mutual Information (Bin Estimator)

**Function:** *nta\_amutibin.m*

**Dependencies:** *nta\_Mlbin.m*

**Description:** Calculates the auto-mutual information as a function of lags using a binning estimator (Cover, Thomas M., and Joy A. Thomas. 1991).

$$AMI_{bin}(X; X + \Delta t) = H_{bin}(X) + H_{bin}(X + \Delta t) - H_{bin}(X, X + \Delta t) \quad (40)$$

H: Shannon entropy

Table 40: Configuration Structure (cfg)

| Field Name     | Description               | Size | Data Type | Default          |
|----------------|---------------------------|------|-----------|------------------|
| <b>numbin</b>  | number of bins            | 1x1  | int       | 10               |
| <b>maxlag</b>  | maximum number of lags    | 1x1  | int       | half data length |
| <b>plt</b>     | plot results yes/no [1/0] | 1x1  | int       | 1                |
| <b>verbose</b> | verbose level [1/0]       | 1x1  | int       | 1                |

Table 41: Output Structure (results)

| Field Name | Description             | Size   |
|------------|-------------------------|--------|
| <b>cfg</b> | configuration structure | struct |

|                 |   |            |
|-----------------|---|------------|
| <b>ami</b>      | auto mutual information as a function of time [bit] | 1 x maxlag |
| <b>firstmin</b> | first minimum of auto-mutual information            | 1x1        |

#### 4.3.7. Auto-Mutual Information (Kraskov's Estimator)

**Function:** *nta\_amutiembknn.m*

**Dependencies:** *nta\_Mdelayunimulti.m*

**Description:** Calculates the auto-mutual information as a function of lags  $\Delta t$  using a nearest neighbours estimator (Kraskov et al. 2004).

$$AMI_K(X, X + \Delta t) = \psi(mass) + \psi(N) + \frac{\sum_{i=1}^N \psi(n_{xi}) + \psi(n_{X+\Delta ti})}{N} \quad (41)$$

$\Psi$ : digamma function, mass: number of nearest neighbours, N: total number of points, n: number of neighbours.

Table 42: Configuration Structure (cfg)

| Field Name    | Description  | Size       | Data Type | Default          |
|---------------|--|------------|-----------|------------------|
| <b>tau</b>    | embedding delay                                      | 1x1        | int       | 1                |
| <b>dim</b>    | embedding dimension                                  | 1x1        | int       | 2                |
| <b>mass</b>   | number of nearest neighbours used for PDF estimation | 1x1        | int       | 4                |
| <b>maxlag</b> | maximum number of lags                               | 1x1        | int       | half data length |
| <b>metric</b> | distance norm 'euclidean' or 'maximum'               | 1x7 or 1x9 | char      | 'maximum'        |
| <b>plt</b>    | plot results yes/no [1/0]                            | 1x1        | int       | 1                |

|                |                     |     |     |   |
|----------------|---------------------|-----|-----|---|
| <b>verbose</b> | verbose level [1/0] | 1x1 | int | 1 |
|----------------|---------------------|-----|-----|---|

Table 43: Output Structure (results)

| Field Name      | Description                                 | Size       |
|-----------------|---|------------|
| <b>cfg</b>      | configuration structure                     | struct     |
| <b>embAMI</b>   | auto-Mutual information per time lag [nats] | 1 x maxlag |
| <b>firstmin</b> | first minimum of embAMI                     | 1x1        |
| <b>AIS</b>      | active information storage                  | 1x1        |

#### 4.3.8. Active Information Storage

**Function:** *nta\_AIS.m*

**Dependencies:** *nta\_Mldelayunimulti.m*

**Description:** Calculates the active information storage using a nearest neighbours estimator (Lizier et al. 2012). Active information storage is defined as the mutual information between the current time point  $t$  of a random variable and the next past state at  $t-1$ . It is a measure for the average past information currently in use at each time point :

$$AIS(X) = MI_K(\mathbf{X}_{t-1}^{d_x}, X_t), \quad (42)$$

MI: mutual information, d: embedding dimension.

Table 44: Configuration Structure (cfg)

| Field Name | Description         | Size | Data Type | Default |
|------------|---------------------|------|-----------|---------|
| <b>tau</b> | embedding delay     | 1x1  | int       | 1       |
| <b>dim</b> | embedding dimension | 1x1  | int       | 2       |



|                |  |            |      |           |
|----------------|--|------------|------|-----------|
| <b>mass</b>    | number of nearest neighbours used for PDF estimation | 1x1        | int  | 4         |
| <b>metric</b>  | distance norm 'euclidean' or 'maximum'               | 1x7 or 1x9 | char | 'maximum' |
| <b>verbose</b> | verbose level [1/0]                                  | 1x1        | int  | 1         |

Table 45: Output Structure (results)

| Field Name  | Description                       | Size                 |
|-------------|-----------------------------------|----------------------|
| <b>cfg</b>  | configuration structure           | struct               |
| <b>AIS</b>  | active information storage [nats] | 1x1                  |
| <b>IAIS</b> | Local active information storage  | 1xN, N=length(input) |

## 4.4. Related

### 4.4.1. Phase Space reconstruction

**Function:** *nta\_phasespace.m*

**Dependencies:** *neighsearch.mexw64*

**Description:** Embedding of time series in phase-space using Taken's delay embedding (Takens 1981). By time-shifting a univariate time series  $d$  times by a factor  $\tau$ , the time series can be embedded in a  $d$ -dimensional phase-space:

$$\mathbf{x}_t^{d_x} = [x_{t-(d_x-1)\tau}, x_{t-(d_x-2)\tau}, \dots, x_{t-\tau}, x_t]^T \quad (43)$$

T: transpose

The function also allows to perform two nonlinear noise reduction schemes: 1) a simple nonlinear noise reduction where each vector is replaced by a weighted average of its closest neighbours and 2)

a projection scheme, where the local neighbourhood gets projected onto a local manifold along the eigenvectors with the largest eigenvalues.

*Table 46: Configuration Structure (cfg)*

| Field Name          | Description   | Size | Data Type | Default |
|---------------------|---|------|-----------|---------|
| <b>tau</b>          | embedding delay   | 1x1  | int       | 1       |
| <b>dim</b>          | embedding dimension   | 1x1  | int       | 2       |
| <b>filter</b>       | mode of nonlinear noise reduction<br>[none/simple/projection]   |      | char      | none    |
| <b>en</b>           | neighbourhood size for nonlinear noise reduction  | 1x1  | double    | 0       |
| <b>ensize</b>       | determine whether neighbourhood-size are<br>absolute or relative values in % of attractor<br>diameter [rel/abs] |      | char      | abs     |
| <b>Q</b>            | number of largest components to use for noise<br>reduction  | 1x1  | int       | 2       |
| <b>keepfiltTS</b>   | keep filtered univariate time series [1/0]  | 1x1  | int       | 1       |
| <b>keepresidual</b> | keep residual univariate time series [1/0]  | 1x1  | int       | 1       |
| <b>verbose</b>      | verbose level [1/0]   | 1x1  | int       | 1       |

*Table 47: Output Structure (results)*

| Field Name      | Description             | Size       |
|-----------------|-------------------------|------------|
| <b>cfg</b>      | configuration structure | struct     |
| <b>embTS</b>    | phase-space coordinates | time x dim |
| <b>residual</b> | extracted noise         | time x 1   |
| <b>filtTS</b>   | Filtered time series    | time x 1   |

#### 4.4.2. Autocorrelation

**Function:** *nta\_autocorr.m*

**Dependencies:** *xcorr.m*

**Description:** Calculates the autocorrelation  $R$  as a function of delays  $\tau$ :

$$R(\tau) = \sum_{t=0}^{N-\tau-1} x(t)x(t+\tau) \quad (44)$$

$N$ : length of time series,  $\tau$ : time lag

*Table 48: Configuration Structure (cfg)*

| Field Name | Description                        | Size | Data Type | Default          |
|------------|------------------------------------|------|-----------|------------------|
| <b>lag</b> | number of temporal lags in samples | 1x1  | int       | half data length |

*Table 49: Output Structure (results)*

| Field Name | Description  | Size    |
|------------|--|---------|
| <b>Rmm</b> | one-sided autocorrelation as a function of temporal lag in samples | 1 x lag |

#### 4.4.3. Neighbours Distance (mex)

**Function:** *nta\_neighsearch.mex64*

**Dependencies:** -

**Description:** Calculates a distance matrix of phase-space states using the Euclidean or maximum norm.

$$\text{Dist} = ||\mathbf{x}_t - \mathbf{x}_{t+\Delta t}|| \quad (45)$$

$||..||$ : distance norm.

Table 50: Input

| Inputs                  | Description             | Size          | Data Type | Default |
|-------------------------|-------------------------|---------------|-----------|---------|
| <b>Embedding matrix</b> | Output of phasespace.m  | samples x dim | double    | -       |
| <b>Qx1 vector</b>       | Indices of query points | Qx1 vector    | int       | -       |

Table 51: Output Structure (results)

| Output         | Description                 | Size                 |
|----------------|-----------------------------|----------------------|
| <b>sumdist</b> | Distance matrix (Euclidean) | NxN, N=size(input,1) |

#### 4.4.4. Optimize Embedding Parameters

**Function:** *nta\_optimize\_embedding.m*

**Dependencies:** *nta\_fnn.m, nta\_amutibin.m, nta\_ragwitz.m, nta\_phasespace.m*

**Description:** This function optimizes embedding parameters (dimension and tau (Cao 1997; Ragwitz and Kantz 2002)). See 4.1.3, 4.1.4 & 4.3.6 for details on algorithms.

Table 52: Configuration Structure (cfg)

| Field Name          | Optimization | Description   | Size | Data Type | Default        |
|---------------------|--------------|---|------|-----------|----------------|
| <b>optimization</b> | -            | choose optimization procedure<br>'deterministic' (false nearest neighbours (dimension) & auto-mutual information (tau)) | char | char      | deterministic' |

|               |               |   |           |        |             |  |
|---------------|---------------|---|-----------|--------|-------------|--|
|               |               | or 'markov' (Ragwitz estimator (dimension & tau))   |           |        |             |  |
| <b>dims</b>   | deterministic | min and max embedding dimensions  | 1x2       | int    | [2 9]       |  |
| <b>R</b>      | deterministic | Threshold Parameter for initial distance  | 1x1       | double | 10          |  |
| <b>numbin</b> | deterministic | number of bins  | 1x1       | int    | 0           |  |
| <b>dims</b>   | markov        | min and max embedding dimensions  | 1x2       | int    | [2 9]       |  |
| <b>taus</b>   | markov        | vector of tau to scan either in multiple of autocorrelation time in percent   | 1xN       | int    | [10:10:100] |  |
| <b>mass</b>   | markov        | number of neighbours per point  | 1x1       | int    | 4           |  |
| <b>hor</b>    | markov        | prediction horizon  | 1x1       | int    | 1           |  |
| <b>mode</b>   | markov        | how to define tau (taus). 'samples': taus in samples or 'multi'=taus in multiple of autocorrelation time in percent | 1x5   1x7 | char   | multi'      |  |
| <b>embed</b>  | -             | embed input data yes/no   | 1x1       | int    | 0           |  |
|               |               | [1/0]   |           |        |             |  |

Table 53: Output Structure (results)

| Field Name    | Description                   | Size   |
|---------------|-------------------------------|--------|
| <b>cfg</b>    | configuration structure       | struct |
| <b>optdim</b> | optimized embedding dimension | 1x1    |
| <b>opttau</b> | optimized embedding delay     | 1x1    |

#### 4.4.5. Optimize Number of Bins for Histogram

**Function:** *nta\_freddiac.m*

**Description:** Low-level function to optimize histogram bins (Freedman, David, and Persi Diaconis 1981).

$$\text{Number of bins} = \frac{\max(x) - \min(x)}{2 * \text{IQR} * \sqrt[3]{N}} \quad (46)$$

IQR: inter quartile range, N: total number of points.

#### 4.4.6. Prepare Data

**Function:** *nta\_prepare\_data.m*

**Description:** Preprocesses data by normalization, trend correction and filtering.

Table 54: Configuration Structure (cfg)

| Field Name       | Description                        | Size | Data Type | Default |
|------------------|------------------------------------|------|-----------|---------|
| <b>toi</b>       | time points of interest in samples | 1xN  | int       | 0       |
| <b>normalize</b> | z-normalization yes/no [1/0]       | 1x1  | int       | 1       |
| <b>detrend</b>   | remove linear trend yes/no [1/0]   | 1x1  | int       | 0       |
| <b>filter</b>    | filter yes/no [1/0]                | 1x1  | int       | 0       |

|               |                                  |     |     |      |
|---------------|----------------------------------|-----|-----|------|
| <b>lpfreq</b> | lowpass frequency in Hz          | 1x1 | int | 100  |
| <b>hpfreq</b> | highpass frequency in Hz         | 1x1 | int | 1    |
| <b>causal</b> | apply causal filter yes/no [1/0] | 1x1 | int | 0    |
| <b>fs</b>     | Sampling frequency in Hz         | 1x1 | int | 1000 |

Table 55: Output Structure (results)

| Field Name       | Description   | Size     |
|------------------|---------------|----------|
| <b>prep_data</b> | prepared data | time x 1 |

## 4.5. Data

### 4.5.1. Lorenz Data Generation

**Function:** *nta\_lorenzgen.m*

**Description:** Generates x,y and z component of the Lorenz-system with typical parameters for chaotic dynamics (Lorenz 1963).

$$\begin{aligned}
 \dot{X} &= a(Y - X) \\
 \dot{Y} &= X(b - Z) - Y \\
 \dot{Z} &= XY - cZ
 \end{aligned}
 \tag{47}$$

$a=10, b=28, c=8/3$

Table 56: Input

| Inputs   | Description   | Size    | Data Type | Default |
|----------|---|---------|-----------|---------|
| <b>L</b> | time vector at which the Lorenz-function should be evaluated e.g. 1:0.01:1000 | 1x time | double    | -       |

Table 57: Output

| Output   | Description                  | Size     |
|----------|------------------------------|----------|
| <b>x</b> | x component of Lorenz system | time x 1 |
| <b>y</b> | y component of Lorenz system | time x 1 |
| <b>z</b> | z component of Lorenz system | time x 1 |

#### 4.5.2. Sinusoid Generation

**Function:** `nta_signalgen.m`

**Description:** Generates superimposed sinusoids.

$$y(x) = \begin{pmatrix} a^1 \\ \vdots \\ a^n \end{pmatrix} * \begin{pmatrix} \sin(x + \phi^1) \\ \vdots \\ \sin(x + \phi^n) \end{pmatrix} + \eta \quad (48)$$

$a^n$ : amplitude of nth sinusoid,  $\phi^n$ : phase of nth sinusoid,  $\eta$ : uniform white noise.

Table 58: Input

| Inputs           | Description                                   | Size | Data Type | Default |
|------------------|---|------|-----------|---------|
| <b>signalinf</b> | amplitude, frequency, phase                   | nx3  | double    | -       |
| <b>sec</b>       | time length in seconds                        |      | double    | -       |
| <b>fs</b>        | sampling frequency in Hz                      |      | double    | -       |
| <b>noise</b>     | amplitude of added noise in % STD of sinusoid |      | double    | -       |

Table 59: Output Structure (results)

| Output   | Description        | Size     |
|----------|--------------------|----------|
| <b>y</b> | oscillatory signal | time x 1 |



### 4.5.3. Logistic Map

**Function:** *nta\_logisticmap.m*

**Description:** This function computes the logistic map or "Verhulst equation".

$$f(x) = rx(1 - x) \quad (49)$$

r: control parameter.

| Inputs | Description  | Size | Data Type | Default |
|--------|--|------|-----------|---------|
| numit  | number of iterations                               | 1x1  | int       | -       |
| r      | control parameter (chaotic behavior @ $r > 3.57$ ) | 1x1  | double    | -       |

| Output | Description          | Size        |
|--------|----------------------|-------------|
| y      | iterated time series | numit+1 x 1 |

### 4.5.4. Lorenz Data

**Mat-File:** *lorenz5000.mat* (5000 samples of x-component of the Lorenz-function)

*lorenz10000.mat* (10000 samples of x,y,z-components of the Lorenz-function)

## 4.6. Statistics

### 4.6.1. Generation of Surrogates

**Function:** *nta\_surrogates.m*

**Dependencies:** -

**Description:** Generates surrogate data using five different algorithms (Schreiber and Schmitz 1996).

Table 60: Configuration Structure (cfg)

| Field Name   | Description  | Size | Data Type | Default |
|--------------|--|------|-----------|---------|
| <b>mode</b>  | mode==1 random shuffling<br><br>mode==2 phase randomization<br><br>mode==3 amplitude adjusted phase randomization<br><br>mode==4 cut time series at random point and flip second half<br><br>mode==5 cut time series at random point and switch halves | 1x1  | int       | 1       |
| <b>numit</b> | Number of iterations for amplitude adjustment (mode==3)  | 1x1  | int       | 0       |

Table 61: Output Structure (results)

| Field Name  | Description             | Size                   |
|-------------|-------------------------|------------------------|
| <b>cfg</b>  | configuration structure | struct                 |
| <b>surr</b> | surrogate time series   | length(input data) x 1 |

#### 4.6.2. Monte-Carlo Test

**Function:** *nta\_nolitia\_monte.m*

**Dependencies:** -

**Description:** Tests statistical differences of provided datasets using nonparametric Monte-Carlo simulations.

Table 62: Input

| Inputs       | Description                         | Size | Data Type  | Default |
|--------------|-------------------------------------|------|------------|---------|
| <b>data1</b> | input data (cell array of subjects) | 1xN  | cell array | -       |
| <b>data2</b> | input data (cell array of subjects) | 1xN  | cell array | -       |

Table 63: Configuration Structure (cfg)

| Field Name       | Description                              | Size | Data Type | Default |
|------------------|--|------|-----------|---------|
| <b>numperm</b>   | Number of permutations                   | 1x1  | int       | 1000    |
| <b>vars</b>      | Variables to test                        | 1xN  | int       | 1       |
| <b>outputvar</b> | Specific output variable of vars to test | 1xN  | char      | -       |
| <b>alpha</b>     | Alpha level to test                      | 1x1  | double    | 0.05    |
| <b>bonf</b>      | Bonferroni correction yes/no [1/0]       | 1x1  | int       | 0       |
| <b>verbose</b>   | Verbose level [1/0]                      | 1x1  | int       | 1       |
| <b>method</b>    | Name of method to be tested              | 1xN  | char      | -       |

Table 64: Output Structure (results)

| Field Name            | Description  | Size   |
|-----------------------|--|--------|
| <b>cfg</b>            | configuration structure                              | struct |
| <b>P</b>              | p-value  | 1x1    |
| <b>sig</b>            | Significant yes/no [1/0]                             | 1x1    |
| <b>diffsurr_grand</b> | grand mean of differences between surrogate datasets | 1x1    |
| <b>difforig_grand</b> | grand mean of differences between original datasets  | 1x1    |

### 4.6.3. Cluster-based permutation analysis

**Function:** *nta\_nolitia\_cbpa.m*

**Dependencies:** image processing toolbox (for fast calculations), connsets7 (alternative),  
nolitia\_t\_values

**Description:** Tests statistical differences of provided datasets using nonparametric cluster-based permutation analysis (Maris and Oostenveld 2007).

*Table 65: Input*

| Inputs       | Description                         | Size | Data Type  | Default |
|--------------|-------------------------------------|------|------------|---------|
| <b>data1</b> | input data (cell array of subjects) | 1xN  | cell array | -       |
| <b>data2</b> | input data (cell array of subjects) | 1xN  | cell array | -       |

*Table 66: Configuration Structure (cfg)*

| Field Name        | Description   | Size | Data Type | Default |
|-------------------|---|------|-----------|---------|
| <b>numperm</b>    | Number of permutations  | 1x1  | int       | 1000    |
| <b>vars</b>       | Variables to test   | 1xN  | int       | 1       |
| <b>clustalpha</b> | Alpha level for clusters  | 1xN  | char      | 0.05    |
| <b>alpha</b>      | Final alpha level to test   | 1x1  | double    | 0.05    |
| <b>maxcluster</b> | How to define clustervalue: sizec=maximum<br>size of cluster, sumt= sum of t-values | 1xN  | char      | 'sizec' |
| <b>verbose</b>    | Verbose level [1/0]   | 1x1  | int       | 1       |
| <b>method</b>     | Name of method to be tested   | 1xN  | char      | -       |

*Table 67: Output Structure (results)*

| Field Name | Description | Size |
|------------|-------------|------|
|------------|-------------|------|

|                    |                                       |                      |
|--------------------|---------------------------------------|----------------------|
| <b>cfg</b>         | configuration structure               | struct               |
| <b>pos_cluster</b> | Statistics of positive clusters found | 1xvars               |
| <b>neg_cluster</b> | Statistics of negative clusters found | 1xvars               |
| <b>p</b>           | p-value                               | 1xnumber of clusters |
| <b>ids</b>         | Indices of cluster samples            | 1xnumber of clusters |
| <b>sig</b>         | Significant yes/no [1/0]              | 1xnumber of clusters |
| <b>averages</b>    | Mean values of input data             | nx2                  |

## Literature

Cao, Liangyue (1997): Practical method for determining the minimum embedding dimension of a scalar time series. In *Physica D: Nonlinear Phenomena* 110 (1-2), pp. 43–50. DOI: 10.1016/S0167-2789(97)00118-8.

Cover, Thomas M., and Joy A. Thomas. (1991): Elements of information theory.

Eckmann, J-P., S. Oliffson Kamphorst, and David Ruelle. (1987): Recurrence plots of dynamical systems. In *EPL (Europhysics Letters)* (973).

Freedman, David, and Persi Diaconis (1981): On the histogram as a density estimator: L 2 theory. In *Zeitschrift für Wahrscheinlichkeitstheorie und verwandte Gebiete*, 453-476.

Grassberger, Peter; Procaccia, Itamar (1983): Measuring the strangeness of strange attractors. In *Physica D: Nonlinear Phenomena* 9 (1), pp. 189–208. DOI: 10.1016/0167-2789(83)90298-1.

Hurst, H. E. (1951): Long-term storage capacity of reservoirs. In *Trans. Amer. Soc. Civil Eng.*, pp. 770–808.

Kantz, Holger (1994): A robust method to estimate the maximal Lyapunov exponent of a time series. In *Physics Letters A* 185 (1), pp. 77–87. DOI: 10.1016/0375-9601(94)90991-1.

Kennel, Matthew B.; Brown, Reggie; Abarbanel, Henry D. I. (1992): Determining embedding dimension for phase-space reconstruction using a geometrical construction. In *Phys. Rev. A* 45 (6), pp. 3403–3411. DOI: 10.1103/PhysRevA.45.3403.

Kozachenko, L. F., and Nikolai N. Leonenko. (1987): Sample estimate of the entropy of a random vector. In *Problemy Peredachi Informatsii*, pp. 9–16.

Kraskov, Alexander; Stögbauer, Harald; Grassberger, Peter (2004): Estimating mutual information. In *Phys. Rev. E* 69 (6 Pt 2), p. 66138. DOI: 10.1103/PhysRevE.69.066138.

Little, Max A.; McSharry, Patrick E.; Roberts, Stephen J.; Costello, Declan A. E.; Moroz, Irene M. (2007): Exploiting nonlinear recurrence and fractal scaling properties for voice disorder detection. In *Biomedical engineering online* 6, p. 23. DOI: 10.1186/1475-925X-6-23.

Lizier, Joseph T.; Prokopenko, Mikhail; Zomaya, Albert Y. (2012): Local measures of information storage in complex distributed computation. In *Information Sciences* 208, pp. 39–54. DOI: 10.1016/j.ins.2012.04.016.

Lorenz, Edward N. (1963): Deterministic nonperiodic flow. In *Journal of the atmospheric sciences*, 130-141.

Maris, Eric; Oostenveld, Robert (2007): Nonparametric statistical testing of EEG- and MEG-data. In *Journal of Neuroscience Methods* 164 (1), pp. 177–190. DOI: 10.1016/j.jneumeth.2007.03.024.

Marwan, Norbert; Kurths, Jürgen (2002): Nonlinear analysis of bivariate data with cross recurrence plots. In *Physics Letters A* 302 (5-6), pp. 299–307. DOI: 10.1016/S0375-9601(02)01170-2.

Marwan, Norbert; Wessel, Niels; Meyerfeldt, Udo; Schirdewan, Alexander; Kurths, Jürgen (2002): Recurrence-plot-based measures of complexity and their application to heart-rate-variability data. In *Phys. Rev. E* 66 (2 Pt 2), p. 26702. DOI: 10.1103/PhysRevE.66.026702.

Oostenveld, Robert; Fries, Pascal; Maris, Eric; Schoffelen, Jan-Mathijs (2011): FieldTrip: Open source software for advanced analysis of MEG, EEG, and invasive electrophysiological data. In *Computational Intelligence and Neuroscience* 2011, p. 156869. DOI: 10.1155/2011/156869.

- Peng, C.-K.; Buldyrev, S. V.; Havlin, S.; Simons, M.; Stanley, H. E.; Goldberger, A. L. (1994): Mosaic organization of DNA nucleotides. In *Phys. Rev. E* 49 (2), pp. 1685–1689. DOI: 10.1103/physreve.49.1685.
- Provenzale, A.; Smith, L. A.; Vio, R.; Murante, G. (1992): Distinguishing between low-dimensional dynamics and randomness in measured time series. In *Physica D: Nonlinear Phenomena* 58 (1-4), pp. 31–49. DOI: 10.1016/0167-2789(92)90100-2.
- Ragwitz, M.; Kantz, H. (2002): Markov models from data by simple nonlinear time series predictors in delay embedding spaces. In *PHYSICAL REVIEW E* 6505 (5), p. 6201.
- Romano, M. C.; Thiel, M.; Kurths, J.; Kiss, I. Z.; Hudson, J. L. (2005): Detection of synchronization for non-phase-coherent and non-stationary data. In *Europhys. Lett.* 71 (3), pp. 466–472. DOI: 10.1209/epl/i2005-10095-1.
- Romano, M. Carmen; Thiel, Marco; Kurths, Jürgen; Bloh, Werner von (2004): Multivariate recurrence plots. In *Physics Letters A* 330 (3-4), pp. 214–223. DOI: 10.1016/j.physleta.2004.07.066.
- Rosenstein, Michael T.; Collins, James J.; De Luca, Carlo J. (1993): A practical method for calculating largest Lyapunov exponents from small data sets. In *Physica D: Nonlinear Phenomena* 65 (1), pp. 117–134. DOI: 10.1016/0167-2789(93)90009-P.
- Schreiber; Schmitz (1996): Improved Surrogate Data for Nonlinearity Tests. In *Physical review letters* 77 (4), pp. 635–638. DOI: 10.1103/PhysRevLett.77.635.
- Schreiber, Thomas; Schmitz, Andreas (1997): Discrimination power of measures for nonlinearity in a time series. In *Phys. Rev. E* 55 (5), pp. 5443–5447. DOI: 10.1103/PhysRevE.55.5443.
- Shannon CE, Weaver W. (1949): The mathematical theory of communication. In *Urbana*.



So; Ott; Schiff; Kaplan; Sauer; Grebogi (1996): Detecting unstable periodic orbits in chaotic experimental data. In *Physical review letters* 76 (25), pp. 4705–4708. DOI: 10.1103/PhysRevLett.76.4705.

Takens, Floris. (1981): Dynamical systems and turbulence. Detecting strange attractors in turbulence. Berlin, Heidelberg: Springer.

Takens, Floris. (1985): Dynamical systems and bifurcations. On the numerical determination of the dimension of an attractor. Berlin, Heidelberg: Springer.