

A large, light blue wireframe graphic of a human brain, composed of numerous overlapping lines, serves as a background for the central text.

g. *BSanalyze*

BIOSIGNAL ANALYSIS

advanced biosignal processing and analysis

<http://www.gtec.at>
office@gtec.at

VERSION 5.16.02

USER MANUAL

Copyright 2017 g.tec medical engineering GmbH

CONTENT:

PREFACE	7
RELEASE NOTES	8
REQUIRED PRODUCTS.....	9
RELATED PRODUCTS	10
USING THIS GUIDE	11
CONVENTIONS.....	13
INSTALLATION AND CONFIGURATION.....	14
HARDWARE AND SOFTWARE REQUIREMENTS	15
FILES ON YOUR COMPUTER	20
QUICKSTART	21
RUNNING G.BSANALYZE.....	22
LOADING AND VIEWING DATA.....	23
MERGING DATA	24
TRIGGERING DATA	26
ASSIGNING ATTRIBUTES.....	28
COMPUTING POWER SPECTRAL DENSITY	30
DISPLAYING RESULTS AND PRINTING.....	32
CLOSING RESULT2D AND G.BSANALYZE.....	33
THE DATA EDITOR	34
DISPLAY, PRESENT AND SHOW.....	36
MARKERS AND ATTRIBUTES	39
EPOCHING, TOOLS AND COMMENT.....	40
FILE.....	46
LOAD DATA	47
LOAD CLASS INFORMATION	48
LOAD MARKER.....	50
LOAD HEADER INFORMATION	53
LOAD SCORING DATA	54
SAVE AND SAVE AS	55
SAVE SCORING DATA.....	56
IMPORT.....	57
IMPORT EDF	58
IMPORT ASCII	59
IMPORT RDF	61
IMPORT BKR.....	62
IMPORT CNT	63
IMPORT TFM.....	64
IMPORT BLOCK.....	65
IMPORT G.MOBI LAB	67
IMPORT MIT.....	68
IMPORT BCI2000	69
IMPORT BDF	70
EXPORT ASCII.....	71

PRINTER PREVIEW	73
PRINT	74
CLOSE ALL FIGURES AND EXIT	75
HEADER INFORMATION	76
DAQ	77
AMPLIFIER.....	78
CHANNEL CONFIGURATION	80
CALIBRATION	82
GEOMETRY	85
MARKER.....	87
EPOCH	89
SUBJECT	90
SESSION/PARADIGM	91
COMPANY.....	92
VIEW	93
SCALING.....	95
TRANSFORM	97
CUT SAMPLES.....	98
CUT TRIALS AND CHANNELS.....	100
SELECT TRIALS AND CHANNELS.....	102
SORT TRIALS AND CHANNELS	103
ARITHMETIC.....	104
MERGE	106
TRIGGER.....	108
UNTRIGGER	111
PRE-PROCESSING.....	112
SOURCE DERIVATION	113
DC CORRECTION	118
BASELINE CORRECTION	120
RECTIFY AND SMOOTH.....	121
MOVING WINDOW FILTER.....	123
REMOVE DRIFT	126
DETREND.....	128
DOWN- AND UPSAMPLING.....	130
SPATIAL FILTER	131
FILTER.....	138
TOOLS	142
REACTION TIME	143
SINGLE TRIAL ANALYSIS.....	147
TRIGGER FINDER	152
ARTIFACT	155
EVENTFINDER.....	156
SIGNAL QUALITY CHECK	160
TMS DISCHARGE ARTIFACT REMOVAL	163
ANALYZE	166

DATA QUALITY	170
AVERAGE	175
SPECTRUM.....	181
MEAN FREQUENCY.....	186
ERD	191
ERD - MAPS.....	198
WAVELET TRANSFORMATION	203
COMMON SPATIAL PATTERNS	206
PRINCIPAL COMPONENT ANALYSIS	214
INDEPENDENT COMPONENT ANALYSIS.....	216
CANONICAL CORRELATION ANALYSIS (CCA)	221
COHERENCE	225
EVENT RELATED COHERENCE.....	229
PERI-STIMULUS TIME HISTOGRAM.....	232
P300 ACCURACY.....	235
PARAMETER EXTRACTION.....	241
HJORTH	245
BARLOW.....	249
AAR	253
VARIANCE.....	264
BANDPOWER	266
MINIMUM ENERGY	268
EXPONENTIAL WINDOW	270
RUNNING FRACTAL DIMENSION.....	272
TEMPORAL AND SPATIAL COMPLEXITY.....	274
CROSS CORRELATION.....	276
CROSS CORRELATION TEMPLATE MATCHING	278
CANONICAL CORRELATION	282
PHASE LOCKING VALUE.....	285
CALCULATE SPEED.....	289
CLASSIFICATION	291
GENERATE CLASSIFIER	292
MULTI-CLASS LDA	298
MINIMUM DISTANCE CLASSIFIER.....	300
APPEARANCE SETTINGS	301
VIDEO.....	303
GRESULT2D.....	308
DISPLAY OPTIONS	309
PRESENTATION OPTIONS	310
MENU EDIT AND SETTINGS	311
RESULT MEASURE.....	312
PRINT AND PREVIEW	314
SAVE RESULTS	315
EXPORT ASCII.....	316
CLOSING GRESULT2D	317
MONTAGE CREATOR.....	318
STARTING THE MONTAGE CREATOR	319

SELECT ELECTRODE GRID	320
DEFINE ELECTRODE POSITIONS	323
DEFINE CONSTELLATIONS FOR SOURCE DERIVATIONS.....	324
READ ELECTRODE POSITIONS USING THE POLHEMUS PATRIOT DIGITIZER	326
IMPORT POLHEMUS	329
SAVE MONTAGE AND EXIT.....	330
HELP	331
ACCESSING DATA	332
USING THE GET COMMAND.....	335
USING THE SET COMMAND	336
USER EXTENSIONS	337
BATCH MODE	338
DATA-SETS	341
MOVEMENT IMAGINATION	342
RIGHT AND LEFT HAND MOVEMENT IMAGINATION	343
INDEX FINGER MOVEMENT	344
EVOKED POTENTIAL.....	345
PRODUCT PAGE	346

To the Reader

Welcome to g.tec's world of medical and electrical engineering!

Discover the only professional biomedical signal processing platform under MATLAB and Simulink. Your ingenuity finds the appropriate tools in the g.tec elements and systems.

Choose and combine flexibly the elements for biosignal amplification, signal processing and stimulation to perform even real-time feedback.

Our team is prepared to find the better solution for your needs.

Take advantage of our experience!

Dr. Christoph Guger

Dr. Guenter Edlinger

Researcher and Developer

Reduce development time for sophisticated real-time applications from month to hours.

Integrate g.tec's open platform seamlessly into your processing system.

g.tec's rapid prototyping environment encourages your creativity.

Scientist

Open new research fields with amazing feedback experiments.

Process your EEG/ECG/EMG/EOG data with g.tec's biosignal analyzing tools.

Concentrate on your core problems when relying on g.tec's new software features like ICA, AAR or online Hjorth's source derivation.

Study design and data analysis

You are planning an experimental study in the field of brain or life sciences? We can offer consultation in experimental planning, hardware and software selection and can even do the measurements for you. If you have already collected EEG/ECG/EMG/EOG, g.tec can analyze the data starting from artifact control, do feature extraction and prepare the results ready for publication.

Preface

This section includes the following topics:

[Release Notes](#)

[Required Products](#)

[Related Products](#) - g.Recorder, g.HIsys, MATLAB-API, API / device driver package, Pocket PC recording software

[Using This Guide](#) - Suggestions for reading the handbook

[Conventions](#) - Text formats in the handbook

Release Notes

Release notes bring to your attention new features and changes of g.BSanalyze software when upgrading to a newer version.

Version: 5.16.02 (November, 2017)

New features

- Artifact - TMS Discharge Artifact Removal;
- BCIBatch now checks for the session run number.

Changes

- Support of Windows 10 and MATLAB 2015a

Required Products

g.BSanalyze uses:

MATLAB – as basic matrix operation platform

Signal Processing Toolbox - to give access to standard signal analysis tools



NOTE

Proper operation of g.BSanalyze only possible when using the specified version of MATLAB and Signal Processing Toolbox. Any other version may cause the software to malfunction.

Related Products

g.tec provides several biosignal analysis elements that are especially relevant to the kinds of tasks you perform with g.BSanalyze.

For more detailed information on any of our elements, up-dates or new extensions please visit our website www.gtec.at or just send us an email to office@gtec.at

Using This Guide

Chapter "[Installation and Configuration](#)" lists hardware and software requirements and explains the installation of the software.

Chapter "[Quick Start](#)" introduces basic features and capabilities of g.BSanalyze. The chapter will get you working with hands-on exercises.

Chapter "[The Data Editor](#)" explains how to visualize data, scroll through data-sets and to analyze data epochs.

Chapter "[File](#)" describes the importing, exporting and printing of data.

Chapter "[Header](#)" demonstrates the handling of information stored from the data acquisition device. Header contains information about DAQ boards, biosignal amplifier settings, channels, calibration, geometry, marker, subject, session and company settings.

Chapter "[View](#)" explains tools for data visualization such as axis scaling, toolbar commands, data status, jumper and player.

Chapter "[Transform](#)" explains how to cut samples, trials and channels and how to sort, merge and trigger data-sets.

"[Pre-Processing](#)" deals with functions used a priori to the actual analysis such as source derivation, spatial filters, DC-correction, Down- /Upsampling.

Chapter "[Tools](#)" explains the reaction time window and horizontal averaging.

The "[Artifact](#)" section describes the usage of the event-finder for the detection of overflows, zero-lines and other events.

Chapter "[Analyze](#)" discusses biosignal analysis tools like event-related desynchronization (ERD), independent component analysis (ICA) up to data quality. These are functions which produce a specific graphical output that can be viewed with Result2D.

"[Parameter Extraction](#)" explains tools to extract specific features like bandpower or Hjorth parameters of your data.

"[Options](#)" shows important settings to change the appearance of your data in g.BSanalyze.

"[Result2D](#)" describes the 2 dimensional result viewer for topographical plots of means, spectral data up to ICA plots.

The next section describes the "[Montage Creator](#)" which is used to define electrode montage settings and to define constellations for performing source derivations.

Chapter "[Help](#)" explains the usage of the on-line help, the printable documentation and the function help.

Chapter "[Accessing Data](#)" explains how to have access to the data object.

“[User Extensions](#)” demonstrates how to include user-specific MATLAB M-files into g.BSanalyze.

Chapter “[Batch-Mode](#)” shows how to use the g.BSanalyze commands from the MATLAB command line.

”[Data-sets](#)” explains the experimental procedures used to acquire the biosignal data.

Conventions

Item	Format	Example
MATLAB code	Courier	to start simulink, type simulink
String variables	<i>Courier italics</i>	set(P_C, 'PropertyName', ...)
Menu items	Boldface	Select Save from the File menu.

Installation and Configuration

This chapter includes the following sections:

[Hardware and Software Requirements](#)

[Installation from a CD](#)

[g.BSanalyze apps installation](#)

[Files on your Computer](#)

Hardware and Software Requirements

Hardware Requirements

g.BSanalyze requires a PC compatible desktop, notebook workstation running Microsoft Windows.

The table below lists optimal settings:

Hardware	Properties
CPU	Intel or AMD x64 working at minimum 2600 MHz
Harddisk	>= 100 gigabyte
RAM	>=4 gigabyte
USB slot	1-free connector

NOTE: The size of the data-set which can be loaded into g.BSanalyze is determined by the RAM size of your computer. The RAM size should be approximately 5 times bigger than the data-set size. Try to work with as small data-sets as possible by splitting the recording experiment in shorter segments.

Software Requirements

g.BSanalyze requires the installation of MATLAB and of the Signal Processing Toolbox. Make sure that this installation works correctly before installing g.BSanalyze.

Software	Version
MATLAB	Release 2015a
Signal Processing Toolbox	Release 2015a
Windows	Windows 10 Professional English Win64
Acrobat Reader	DC 2015.009.20069



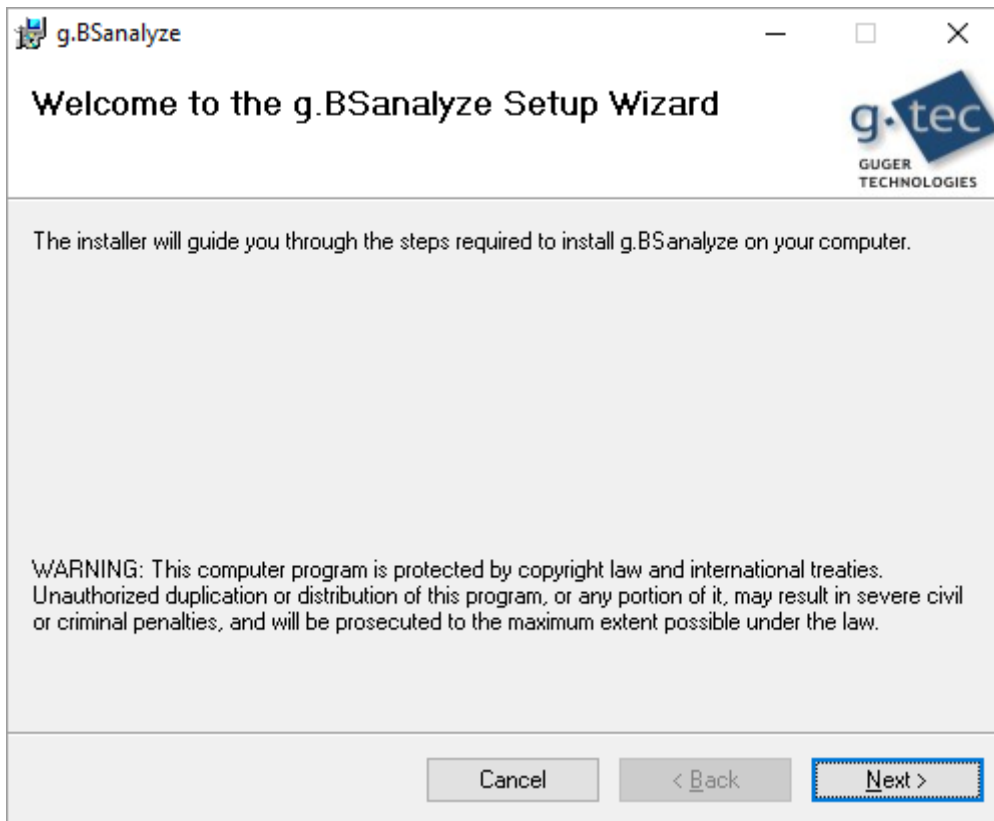
NOTE

This version of g.BSanalyze will not work properly Matlab versions earlier than 2015a. It has not been tested on newer Matlab versions thus can cause errors or may not work at all.

Installation from a CD

The installation of g.BSanalyze consists of three steps

1. Insert the g.tec product CD into the CD-drive and change to the g.BSanalyze directory of your CD-drive and double-click on the `Setup.exe` file. The installation starts and displays the welcome message. Follow the instructions on the screen.



Please read the License Agreement for g.BSanalyze and if you agree with the terms, click **I Agree** and **Next**. Then just follow the steps on the screen.

2. Installation of the Hardlock

The g.BSanalyze installation installs automatically the driver software for the Hardlock. For manual installation click the `HASPUserSetup.exe` in the `Prerequisites/HaspHL` folder on your g.tec CD.

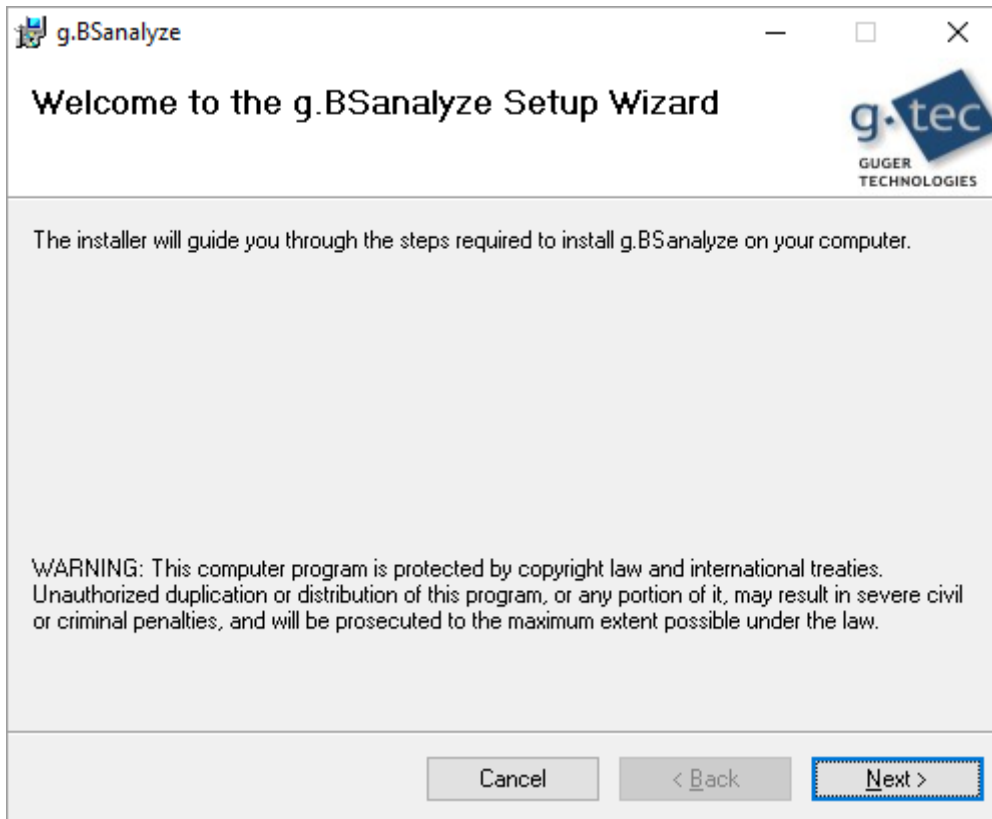
Note that you have to uninstall the Hardlock driver separately if you uninstall the g.BSanalyze software.

3. Set MATLAB path

To make the path settings start MATLAB and open the **Set Path** window in the **File** menu. Then click on the **Add with Subfolders** button and select

C:\Program Files\gtec\gBSanalyze

to add all subdirectories.



Click **Save** and **Close** to finish the installation.

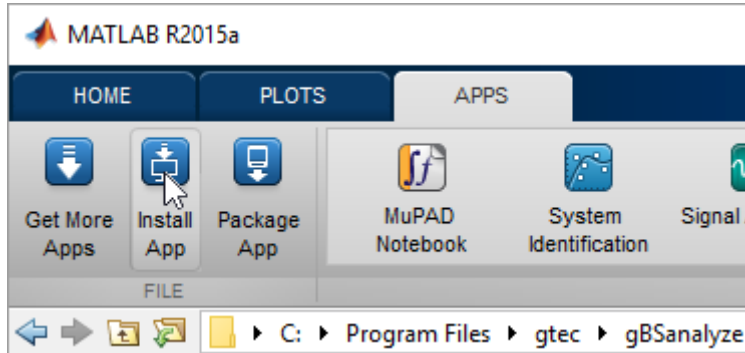
Insert the Hardlock into a free USB slot of your PC or notebook. The light must be on if the installation was successful.

To uninstall g.BSanalyze use the corresponding function in **Control Panel -> Uninstall a program**.

After uninstalling g.BSanalyze the application folder C:\Program Files\gtec\gBSanalyze might not be empty. Any remaining files there have to be deleted manually.

g.BSanalyze apps installation

To install g.BSanalyze MATLAB apps (for MATLAB versions R2012b or higher) click on **Install App** button under **APPS** tab:

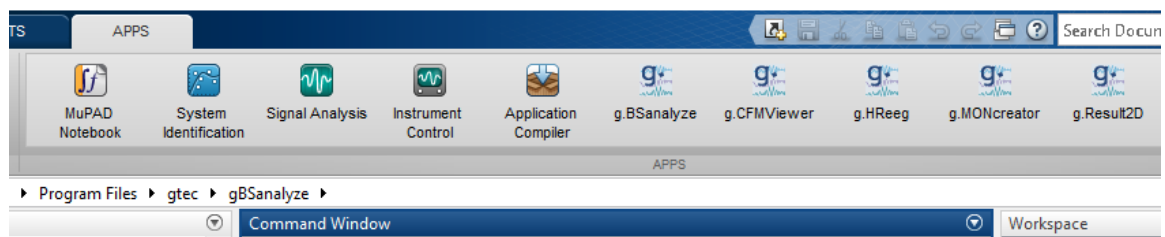


Navigate to

```
C:\Program Files\gtec\gBSanalyze\gBSanalyzeApps\
```

and select all the app installer files stored in this folder. Click on the **Open** button. In the App Installer dialog box, click **Install**.

After installation, you can run the new apps from the apps gallery:



To uninstall an app, in the apps gallery right-click the app and select **Uninstall**. MATLAB deletes the app code from disk and removes the app from the apps gallery.



NOTE

When the windows font-size was set to a value different from 100% at the time when Matlab was installed, distorted menu entries and dialogs reaching beyond the screen boundaries may be observed, even if windows font-size is reset to 100%. This can be only be solved by setting windows font-size to 100% uninstalling and reinstalling Matlab before reverting font-size to a value differing from the recommended windows default of 100%.

Files on your Computer

g.BSanalyze program files - program files are stored under

C:\Program Files\gtec\gBSanalyze

Example Data-sets - are stored in the subdirectory

Documents\gtec\gBSanalyze\testdata

Example data analysis batches – are stored under

Documents\gtec\gBSanalyze\user

Help - help files are stored under

C:\Program Files\gtec\gBSanalyze\Help

Quickstart

The following section gives a short introduction in

[Running g.BSanalyze](#)

[Loading and Viewing Data](#)

[Merging Data](#)

[Triggering Data](#)

[Assigning Attributes](#)

[Computing Power Spectral Density](#)

[Displaying Results and Printing](#)

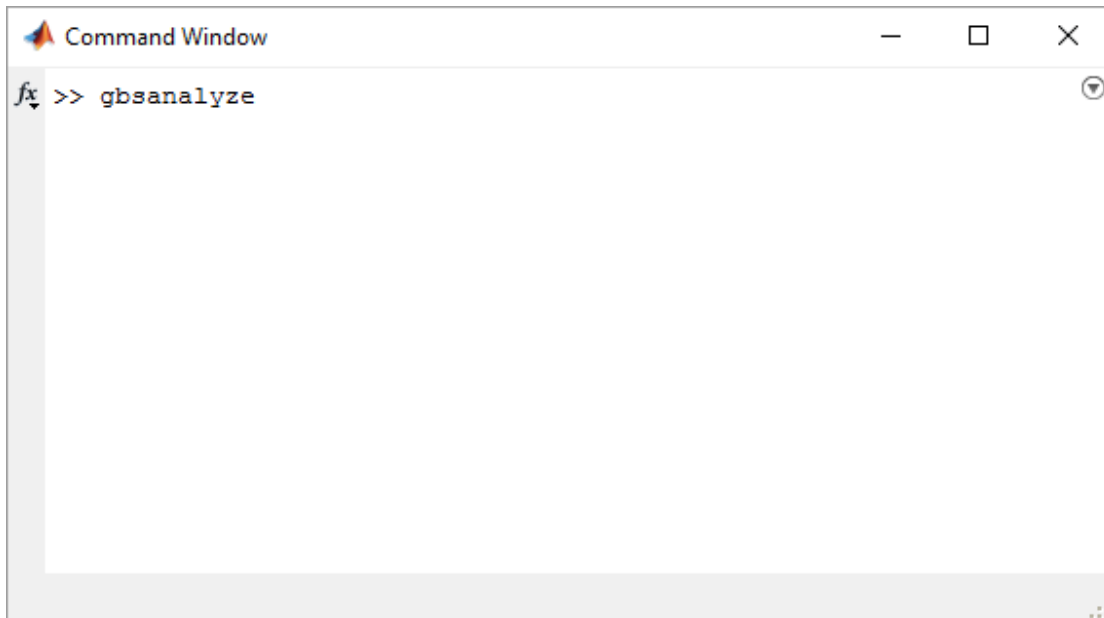
[Closing Result2D and g.BSanalyze](#)

Running g.BSanalyze

After starting MATLAB and setting the correct path, type:

```
gbsanalyze
```

in the MATLAB command line



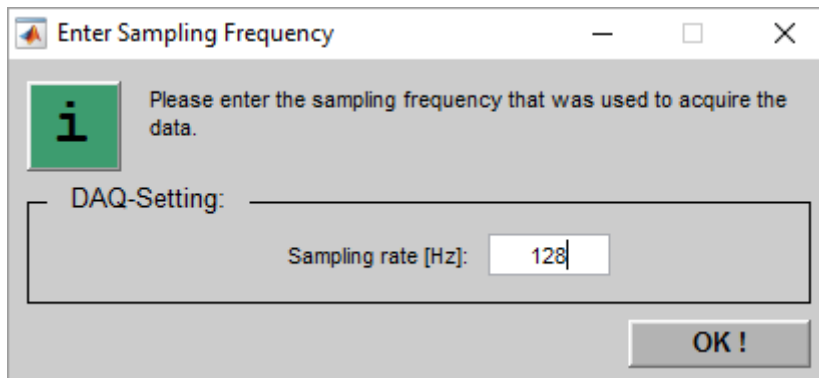
`g.BSanalyze` starts with a blank data window.

Loading and Viewing Data

1. Select **Load Data** in the **File** menu to open a data-set by selecting `session1.mat` from the directory

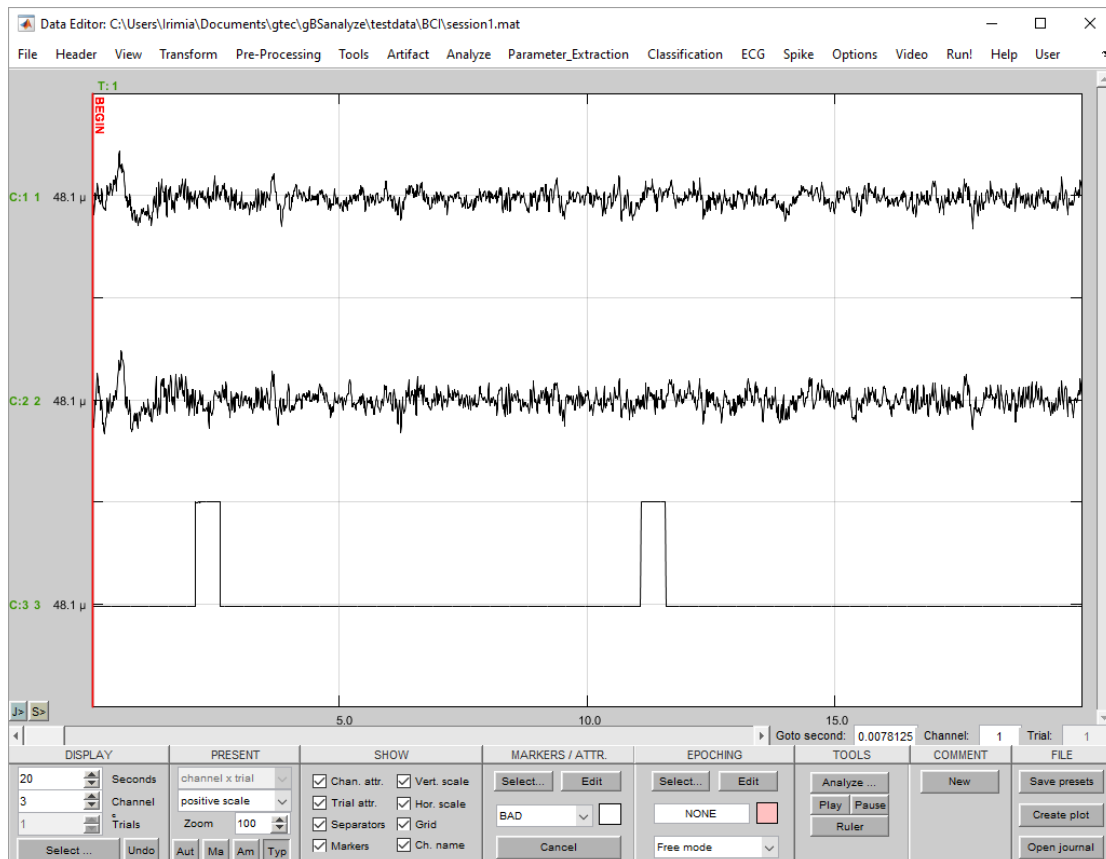
`Documents\gtec\gBSanalyze\testdata\bci`

2. If you are asked, enter a sampling frequency of 128 Hz in the window



3. After pressing the **OK** button the data are displayed

The title of Data Editor show the currently loaded file including the path.



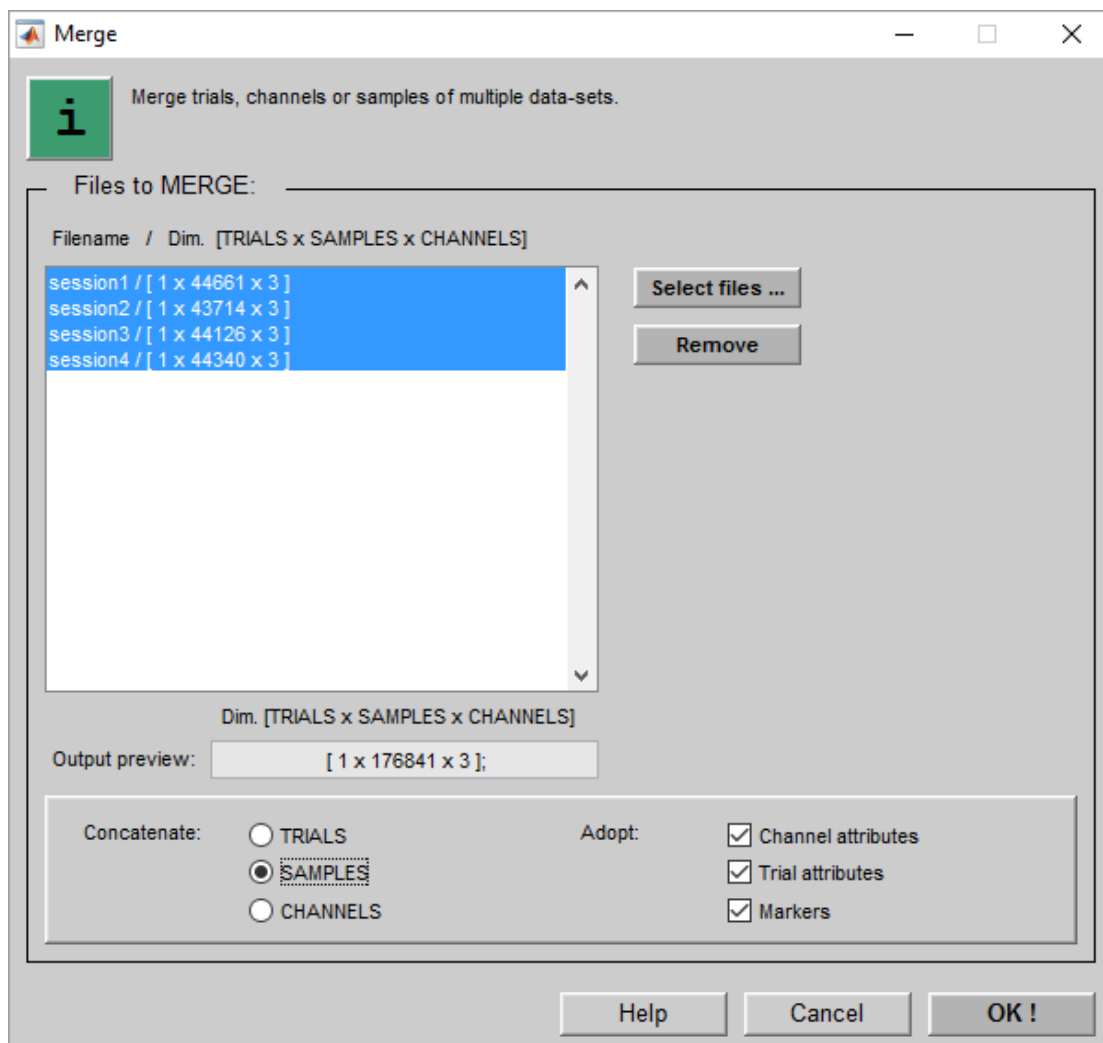
Merging Data

The data was recorded from a subject which participated in 4 recording sessions and had the task to imagine 20 left hand movements and 20 right hand movements per session. This gives in total 160 imaginations. Each session was stored to harddisk as separate file (session1.mat, session2.mat, session3.mat, session4.mat).

For the further analysis of all the data a concatenation of all sessions is necessary.

Perform the following steps:

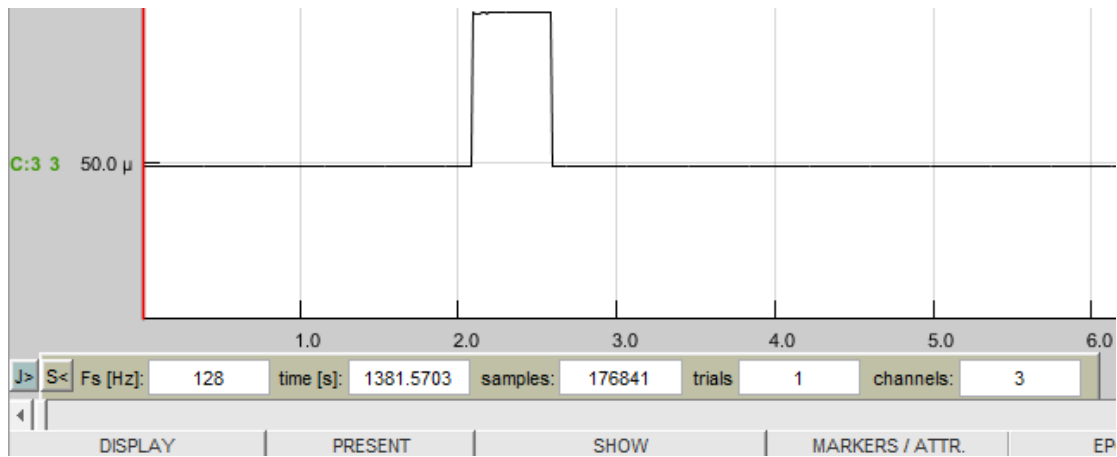
1. Select the option **Merge** in the menu **Transform**
2. Press **Select files ...**



3. Choose `session2.mat`
4. Repeat Step 3 also for `session3.mat` and `session4.mat`.
5. Select **Concatenate SAMPLES**. **Output preview** shows the expected result of the merging process. Press the button **OK**.

The new data-set consists now of 3 channels, 1 trial and about 4 times more samples as a single data-set (176841 samples).

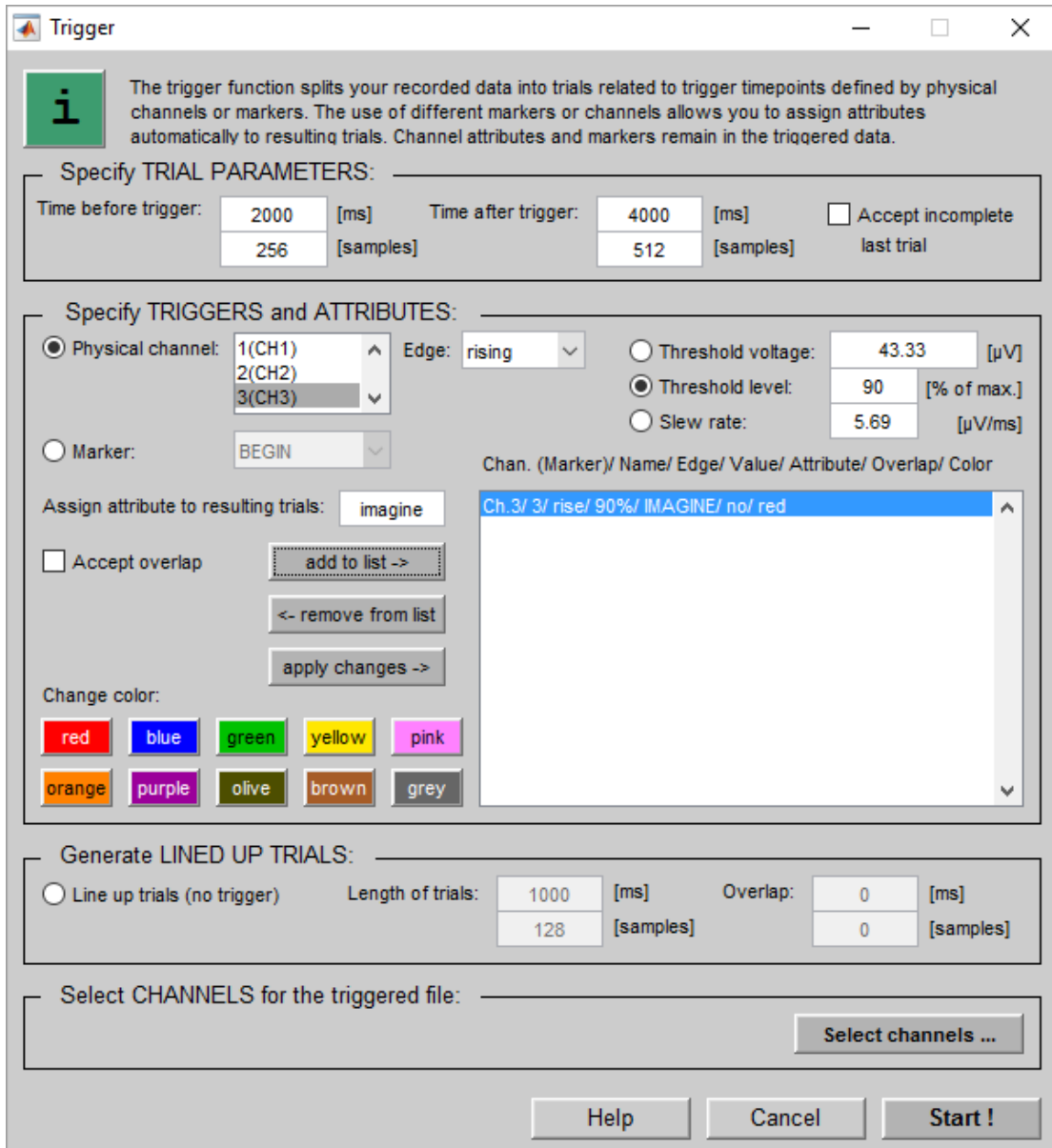
Open **Status** under the **View** menu.



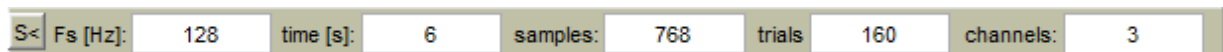
Triggering Data

For many calculations performed with biosignal data epoching or triggering of data-sets is necessary. Therefore, a specific time marker or trigger channel with special events is required. The loaded data-set has 160 TTL-impulses on channel 3 which can be used to split the data into equally sized trials.

1. Select **Trigger** in the **Transform** menu
2. Define the **Time before trigger** as 2000 ms and **Time after trigger** as 4000 ms
3. Select channel 3 in **Physical channel** as trigger channel and set the **Threshold level** to 90 % of maximum.
4. Select e.g. the name `imagine` in **Assign attribute to resulting trials** and press button **add to list ->**
5. Press button **Start!** to perform the triggering



The data are now triggered and the status window displays a total of 160 trials for 3 channels. Each trial has a length of $6 \cdot 128 = 768$ samples.

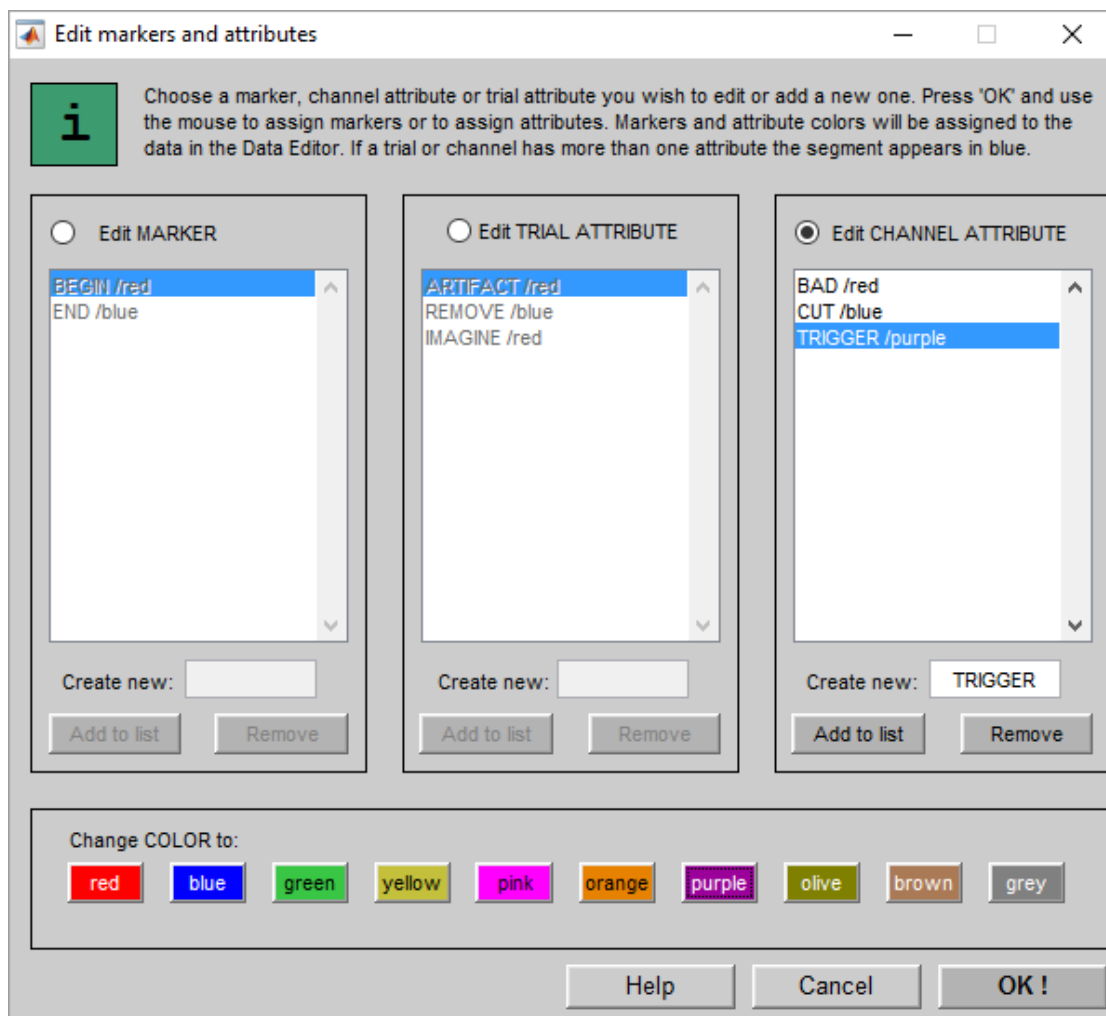


Assigning Attributes

g.BSanalyze allows to assign channel attributes and trial attributes which are used for further calculations to include or exclude specific trials or channels. Trials with EOG, EMG or overflow artifacts can be marked with the trial attribute ARTIFACT to be excluded from further processing or with the attribute REMOVE to be deleted. Also channels which are not relevant for further processing such as the trigger signal after triggering or channels with noise can be marked with BAD to be excluded. On the other hand it is also possible to assign a trial attribute like LEFT to indicate a left hand movement and to include only trials with the LEFT attribute in further calculations.

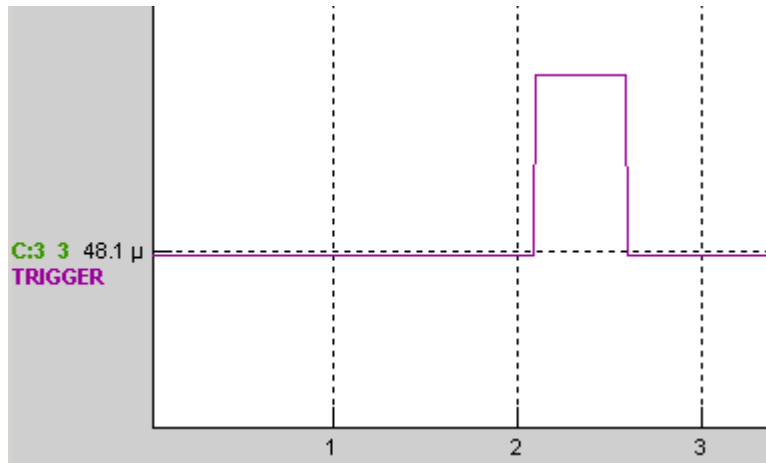
To assign attributes and markers perform the following steps:

1. Press **Select** in section **MARKERS/ATTR.**
2. Click on **Edit CHANNEL ATTRIBUTE** and enter TRIGGER under **Create new.** Press the button **Add to list.** Click on button purple to assign a new color.



3. Close the window with **OK**.

4. Assign the attribute `TRIGGER` to channel 3 by clicking onto the line which represents channel 3. The attribute `TRIGGER` is indicated at the left border of the window.



Computing Power Spectral Density

1. In order to compute the power spectra (PSD) density for each individual channel select **Spectrum** from the menu **Analyze**
2. Choose in **Length of interval(s) to analyze** 1000 ms. Set the **Action interval starts at** to 4500 ms.

Compute power spectra averaged over trials for specified intervals. If a reference interval and an action interval are analyzed a significance test is applied to identify reactive frequency bands.

Specify INTERVAL(S):

Length of interval(s) to analyze: 1000 [ms] 128 [samples] Reference interval starts at: 7.8125 [ms] 1 [samples] Action interval starts at: 4500 [ms] 576 [samples]

Compute action spectra and compare to reference

Specify type of DATA WINDOW:

Data window type: boxcar

Data DOWNSAMPLING:

Reduce number of samples to limit frequency scale range Factor = 4

Select TRIALS and CHANNELS:

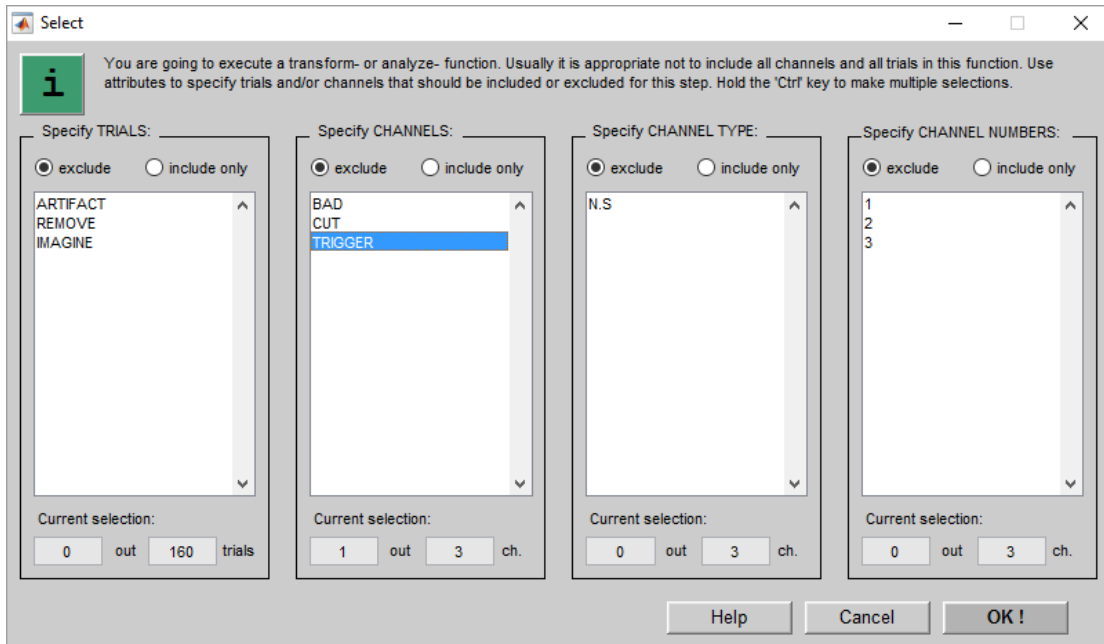
Select trials / chan.

Result procedure: Show with Result2D Show with Result3D Save results Automatic treemaker is: enabled

Filename: enter filename

Help Cancel Start !

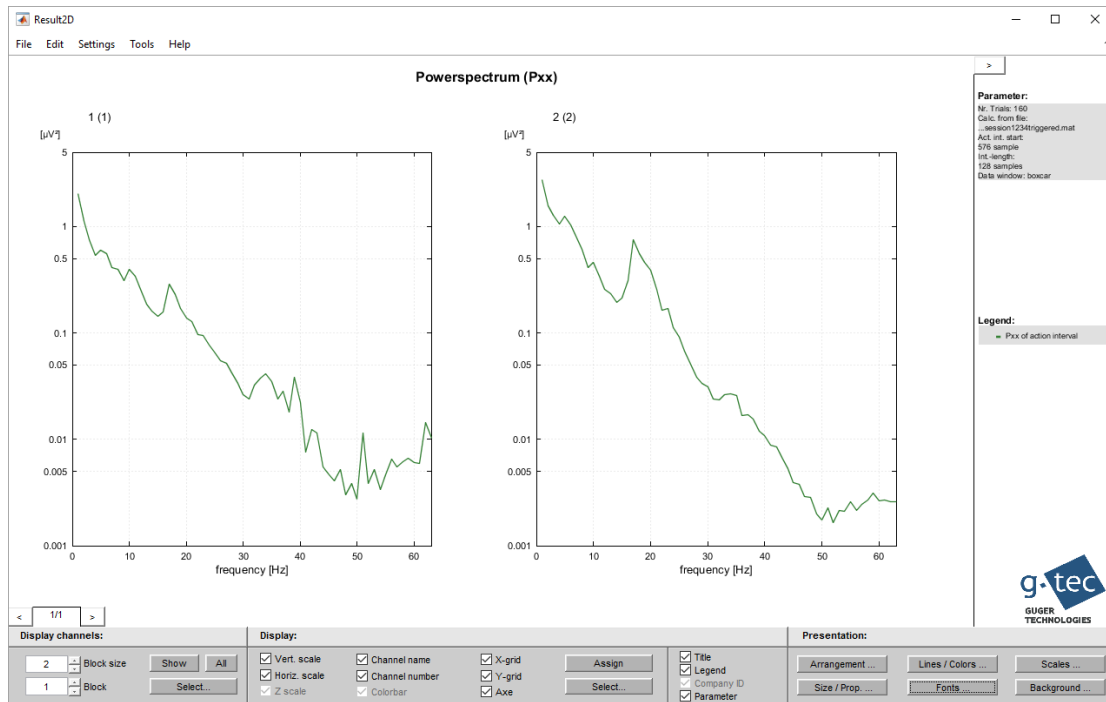
3. To exclude the trigger channel from the computation press **Select trials / chan.**
4. Select **exclude** under **Specify CHANNELS** and select TRIGGER



5. Confirm the selection with the **OK** button and start the spectrum calculation. The PSDs of channels 1 and 2 are calculated.

Displaying Results and Printing

After calculation Result2D opens automatically. The right diagram displays a prominent EEG activity around 18 Hz.



For printing the results select

1. Select **Print Preview** under the **File** menu
2. Adjust the printing result and press button **Print...**
3. Select a printer and press button **OK** to print the result.

Closing Result2D and g.BSanalyze

To close the applications select in Result2D in menu **File** the entry **Close Result2D** and select in the Data Editor under the menu **File** the entry **Exit**.

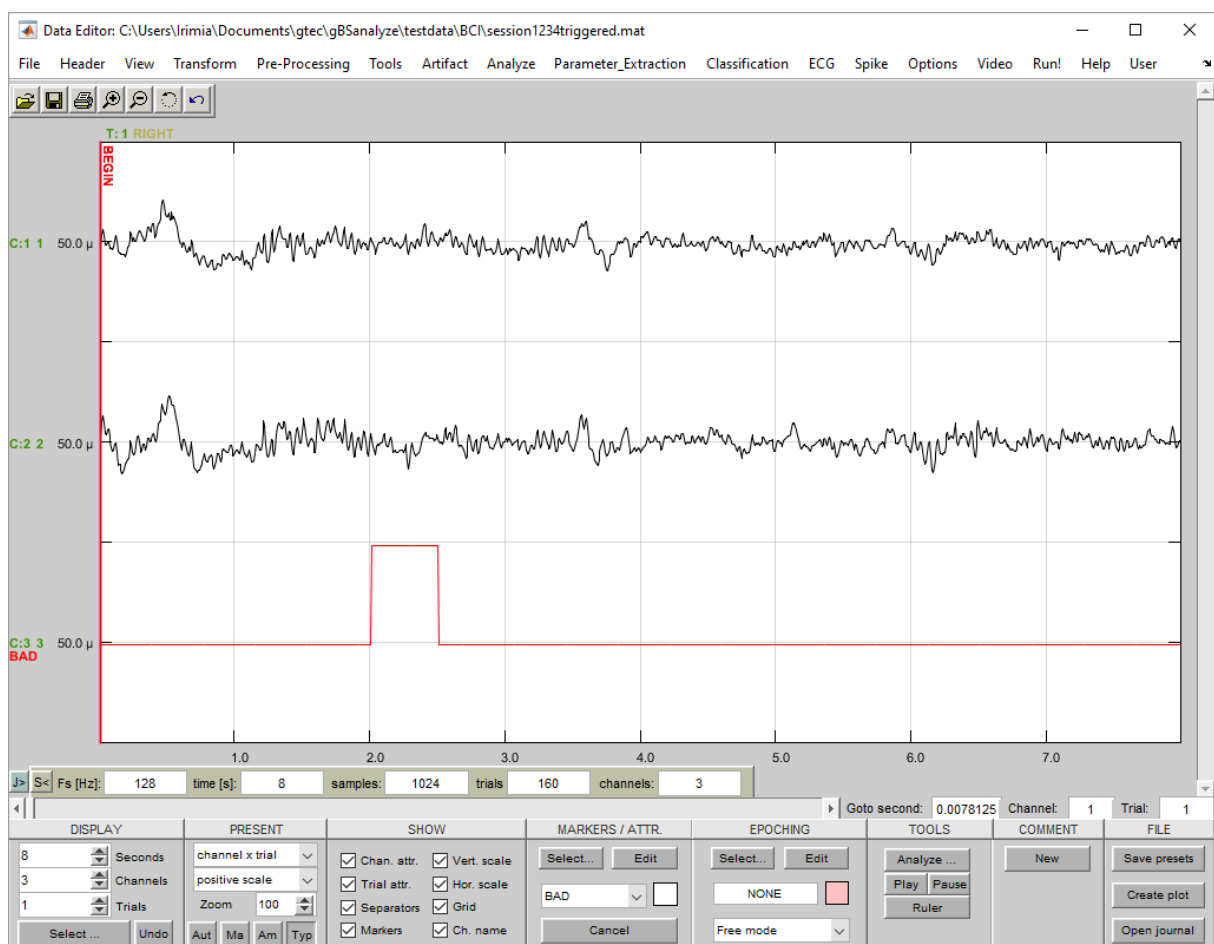
The Data Editor

The Data Editor is a tool for displaying and scrolling through data-sets with different dimensions and sampling frequencies.

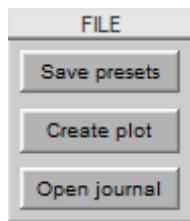
Load file `session1234triggered.mat` of

`Documents\gtec\gBSanalyze\testdata\BCI`

by clicking on **Load Data** under the **File** menu. The Data Editor displays two EEG channels and one Trigger channel.



The buttons in the **File** field are:



Save presets – stores Data Editor settings like the position of the window, filter settings,...
The path must be defined under **Options – Appearance** settings and must be on the MATLAB search path.

Create plot – open a new MATLAB window with the current plot

Open journal – opens the journal file which displays all the output which was plotted in the MATLAB command window. This is a very useful function to inspect and store the data analysis steps. The name and path of the journal file can be defined under **Options – Appearance Settings**.

Display, Present and Show

The options given in the **DISPLAY** field are

DISPLAY		PRESENT		SHOW			
8	Seconds	channel x trial		<input checked="" type="checkbox"/> Chan. attr.	<input checked="" type="checkbox"/> Vert. scale		
3	Channels	positive scale		<input checked="" type="checkbox"/> Trial attr.	<input checked="" type="checkbox"/> Hor. scale		
1	Trials	Zoom	100	<input checked="" type="checkbox"/> Separators	<input checked="" type="checkbox"/> Grid		
Select ...	Undo	Aut	Ma	Am	Typ	<input checked="" type="checkbox"/> Markers	<input checked="" type="checkbox"/> Ch. name

- **Seconds** - seconds visible on the screen
- **Channels** - channels visible on the screen
- **Trials** - trials visible on the screen

The **PRESENT** field allows to make the following changes to the presentation of the data

- channel x trial – view channels as rows and trials as columns
trial x channel – view trials as rows and channels as columns
- positive scale or negative scale
- Zoom – zoom into the data
- Auto – press the button to enable the auto-scaling modus. In this mode each channel is scanned for its maximum and each channels is appropriately scaled.
- Manu – press the button to enable manual scaling. In this mode each channel is scaled according to the settings in the window **Scaling** under the **View** menu.
- Ampl – press the button to enable the amplifier sensitivity scaling. In this mode each channel is scaled according to the amplifier sensitivity in the **Amplifier** window under the **Header** menu.
- Typ – enable type specific scaling. Channels of the same type (e.g. EEG) are scaled with the same amplitude. Use the **Channel Configuration** Window under **Header** to edit the channel type.

The **SHOW** area allows to display the following information:

Chan. attr. - each channel can have channel attributes such as BAD

Trial attr. - trials can have trial attributes, e.g. ARTIFACT

Separators – appear between trials in channel x trial mode or between channels in trial x channel mode

Markers – time markers, e.g. BEGIN

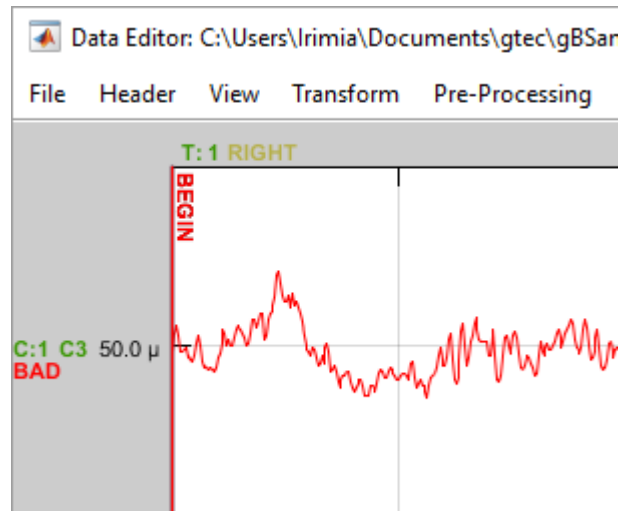
Vert. scale – vertical data scaling, e.g. 50.0 μ

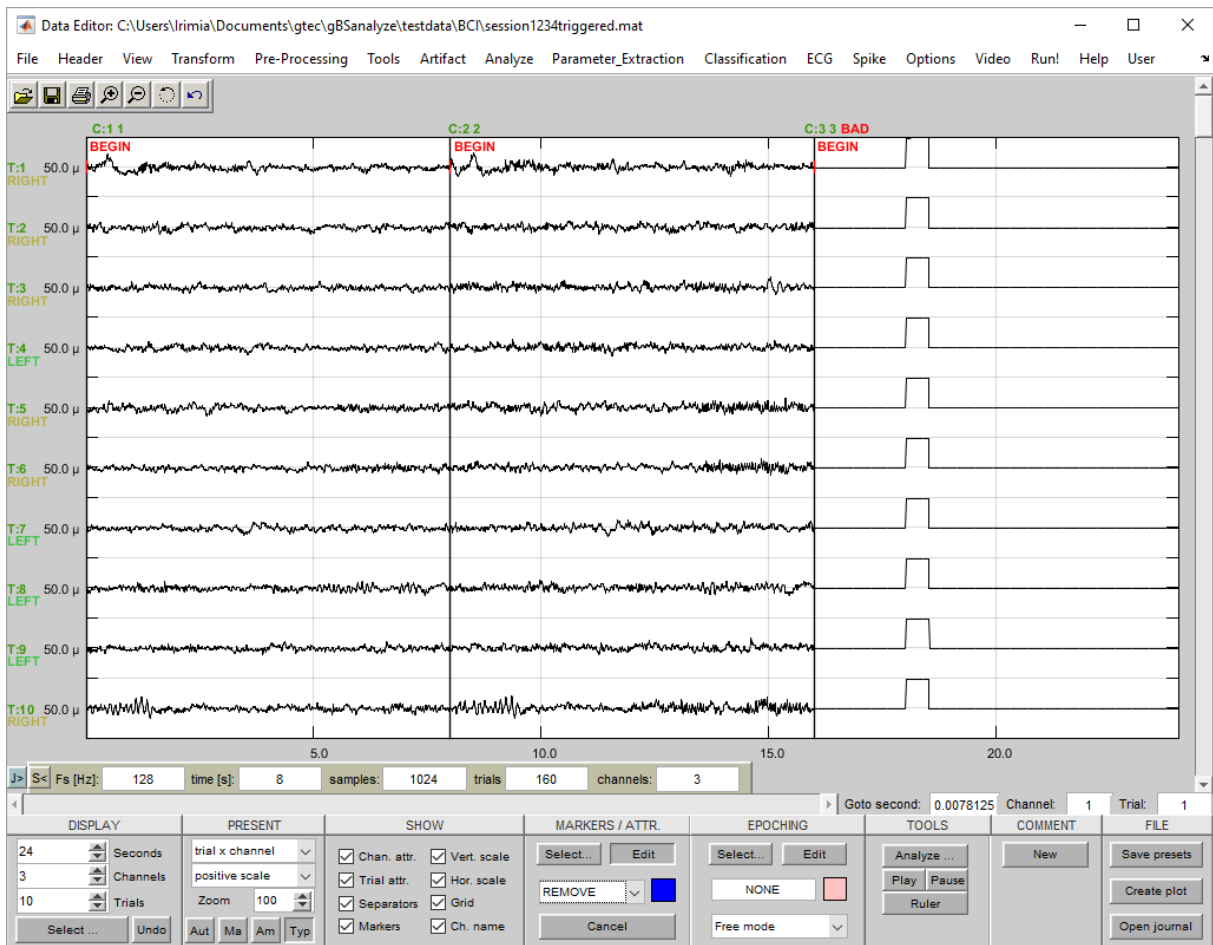
Hor. scale – time scaling in seconds

Grid – horizontal and vertical grid lines

Ch. name – electrode/channel names, e.g. C3

The channel number is indicated as C: number, the trial number as T: number.

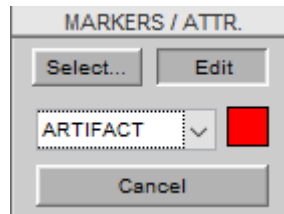




Select the presentation **trial x channel** under the **PRESENT** control area to view trials as rows and channels as columns and display 3 channels.

Markers and Attributes

Click on **Select...** under the **MARKERS / ATTR.** field and select the trial attribute **ARTIFACT**. Close the window to start editing trial attributes of your data.



To search and mark artifacts in your data display only the 2 EEG signals and use the vertical scroll-bar to inspect the data. Scroll down to trial 25 and click on the trial to assign the trial attribute **ARTIFACT**. The click must be within 3 pixels around the channel. To delete the attribute click again on the channel.

To stop the attribute editing de-activate the **Edit** toggle button.

Epoching, Tools and Comment

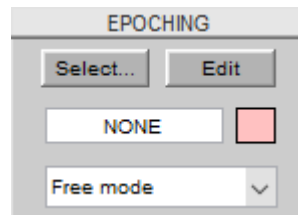
Epoching and Analyze

The **EPOCHING** field allows to mark specific areas of your data in 3 modes:

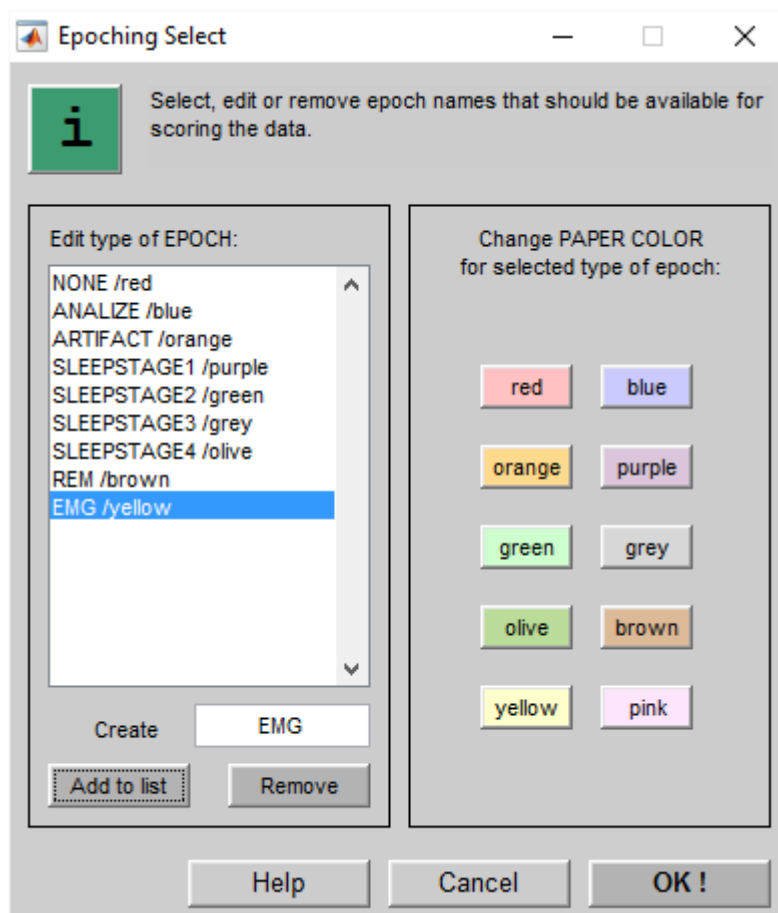
Free – mark epochs of single trials and single channels

Multi chan. – mark epochs over all channels

Multi trial – mark epochs over all trials

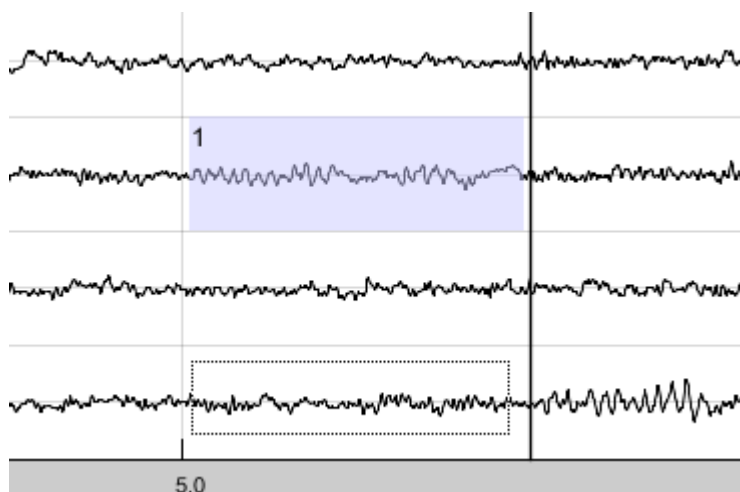


Select the **Free mode** from the listbox and press the **Select...** button.

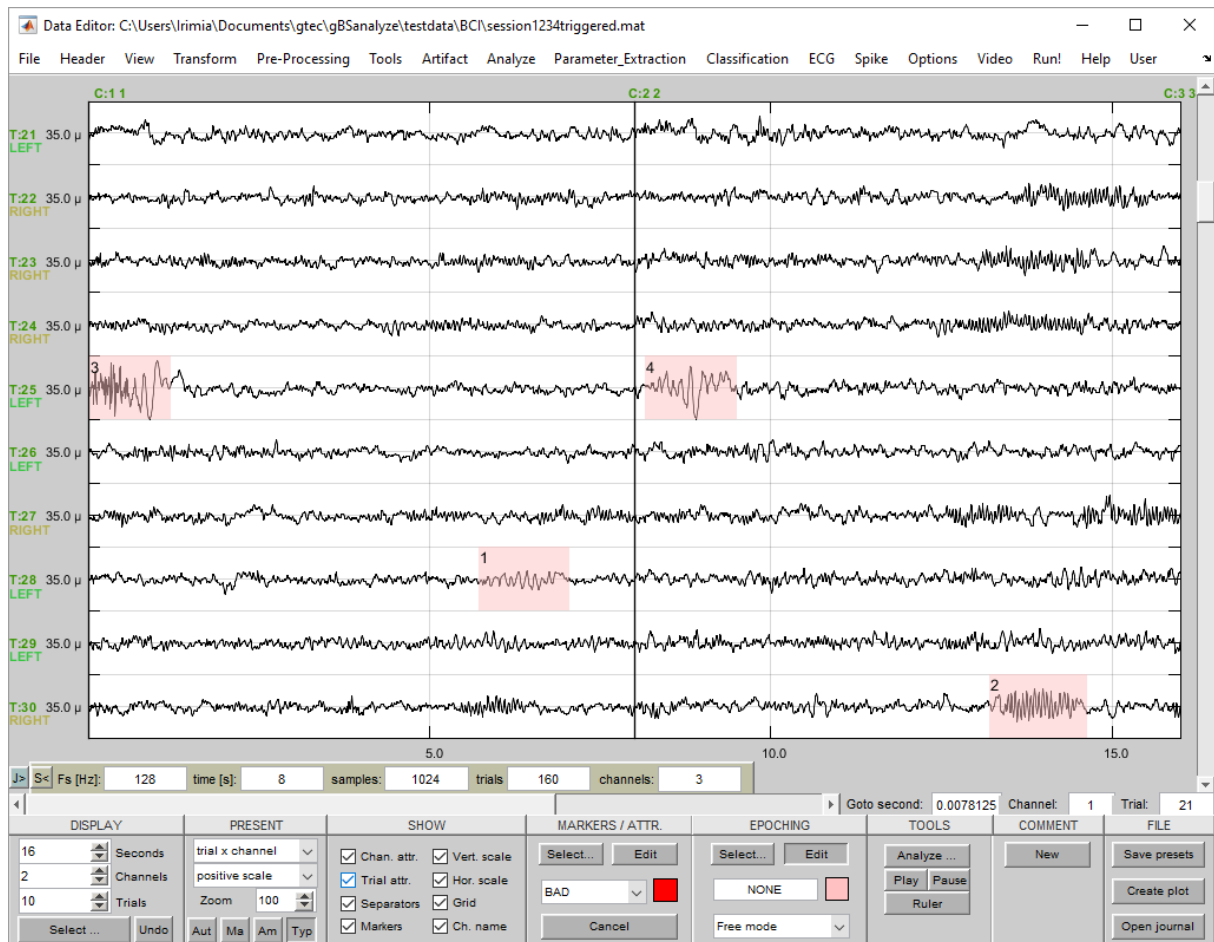


Create the new epoch name EMG by entering the name under **Create new**, pressing the **Add to list** button and quitting the window with OK.

To start **EPOCHING** click into the data axis to mark different epochs by dragging a dotted rectangle over the specific region. Each marked epoch gets a unique number for identification.



Proceed in the same way and mark all interesting regions in the Data Editor.

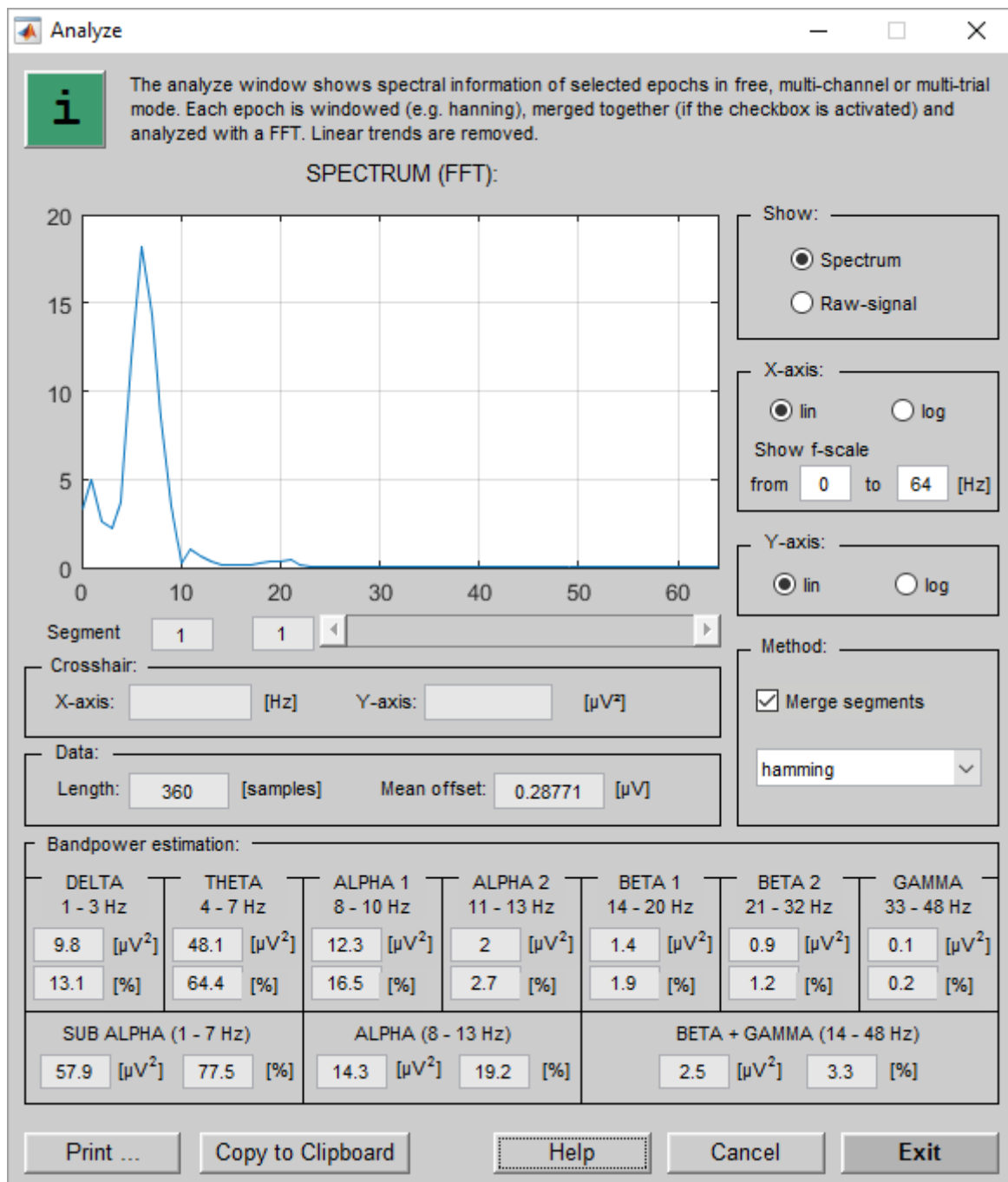


Click on the **Analyze** button under **TOOLS** to open a window for a detailed investigation of the specific regions.

The window allows viewing the raw-signal or the spectral information with linear or log-scaling. If the **Merge segments** box is checked all marked epochs are merged together using a special windowing function. The default windowing function can be selected under menu **Options** in **Appearance Settings**.

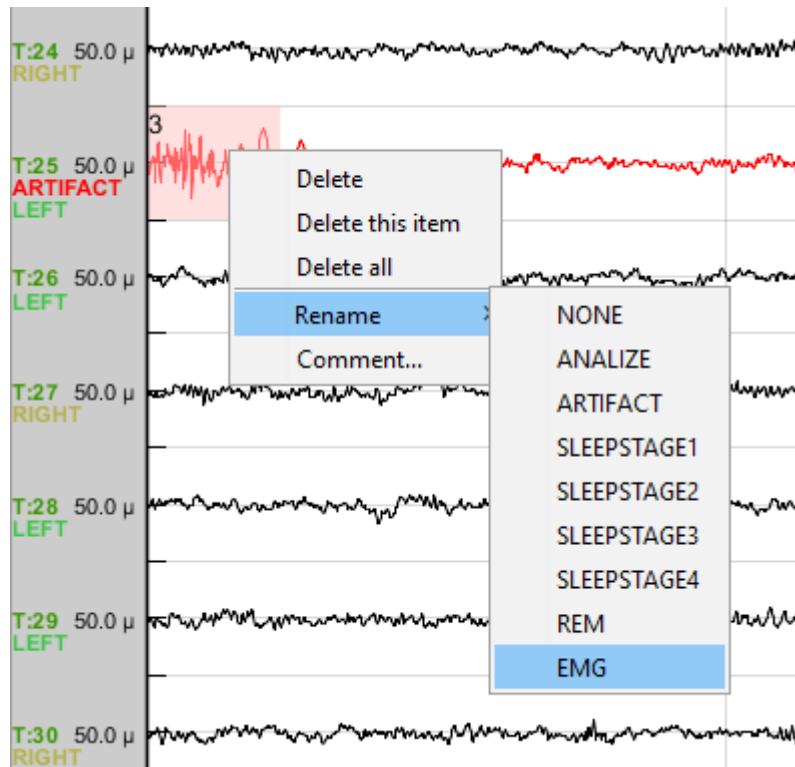
Uncheck the **Merge segments** box to view the spectral information for each epoch separately. Use the scroll bar to step through the marked epochs.

Select Epoch number 1 to show the spectrum which has a prominent alpha peak around 9-10 Hz. About 64 % of the power is between 8 and 10 Hz in the **ALPHA 1** region.

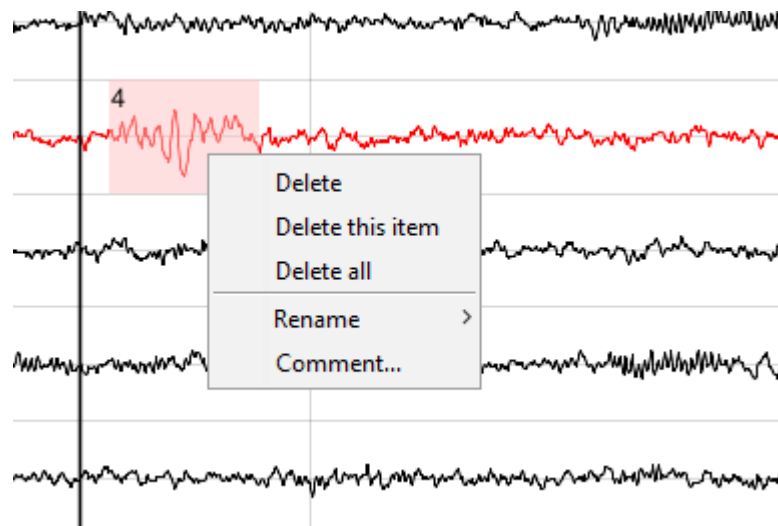


Close the **Analyze** window with **Exit**.

Right click on segment 3 to open the context menu, go to **Rename** and assign the epoch name EMG to the artefact.



To delete epochs right click on the specific segment and select **Delete** for a single epoch, **Delete this item** to delete all epochs with the same name and **Delete all** to delete all epochs. **Comment...** allows to add user-specific text to each epoch.



When selecting the **Multi trial** mode the free epochs are hidden.

Click on **Save as** under the **File** menu to store the edited attributes and epochs to the data-set.

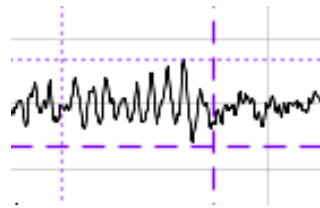
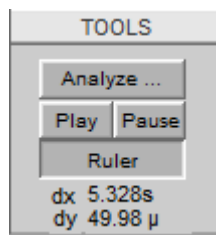


NOTE

Multi-trial mode can only be used when the data display is set to *trial x channel* and *Multi-chan* mode can only be used when the data display is set to *channel x trial*

Ruler

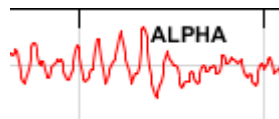
Click on **Ruler** under the **TOOLS** area to activate a horizontal and vertical ruler.



The rulers can be dragged to specific data elements to measure the amplitude and time. Under **Options Appearance Settings** the layout of the ruler lines can be changed.

Comment

Click on **New** under the **COMMENT** area, edit the text and confirm with **Position** (the **Comment** window closes). Position the cursor on the line where the comment should appear and click on the line. A right mouse button context menu allows to edit and delete the text.



File

The **File** menu is grouped into five parts:

The first part contains dialog windows for loading data, class and header information:

- [Load Data](#) - load saved data from harddisk/disk/CD
- [Load Class Information](#) - load class information and assign attributes to trials
- [Load Marker](#) - load time markers from a file
- [Load Header Information](#) - load header information and assign specific entries to the data-set
- [Load Scoring Data](#) - load epochs from a file to the data-set

The second part contains dialog windows for saving data to harddisk/disk/CD:

- [Save](#) - save current data under same name
- [Save as](#) - save current data under a new name
- [Save Scoring Data](#) - save epochs to a file

The third part contains dialog windows for importing and exporting data:

- [Import](#) - import and convert data saved under a different file format
- [Export](#) - export and save current data to a different file format

The fourth part contains dialog windows for printing and page positioning:

- [Print Preview](#) - show a printer preview on the screen
- [Print](#) - show print menu for selecting and configuring the printer and print actual figure

The fifth part contains dialog windows for closing GUIs and the program itself:

- [Close all figures](#) - close all GUIs displayed on the screen except the main Data Editor window
- [Exit](#) - close all windows and exit g.BSanalyze

Load Data

The **Load Data** window allows to load data files created with:

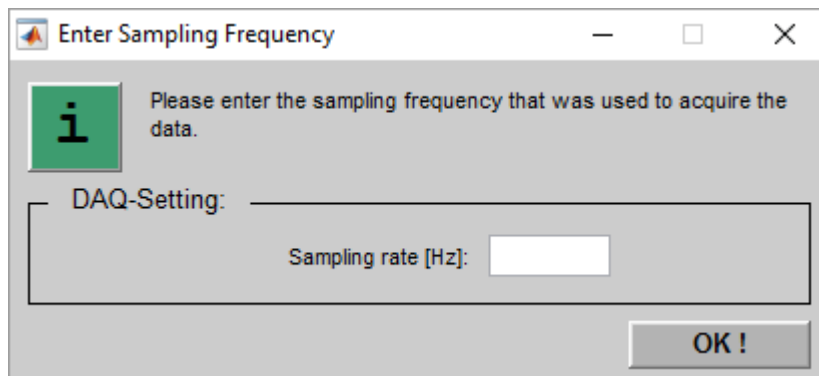
g.BSanalyze - BIOSIGNAL ANALYZIS,
g.DAQsys - DATA ACQUISITION SYSTEM,
g.Recorder – BIOSIGNAL RECORDING SOFTWARE
MATLAB - matrix with dimension [samples x channels] or
[trials x samples x channels]
into g.BSanalyze.

1. Click **Load Data** under the **File** menu, select the directory and the data file `session1.mat` (*.mat, *.hdf5) under

`Documents\gtec\gBSanalyze\testdata\BCI`

2. Click **Open** to load the signal into g.BSanalyze

If the sampling frequency was not stored in the file the following window asks for the sampling rate used to acquire the data.



3. Enter a **Sampling rate** of 128 Hz.

The following code shows how to perform the example demonstrated above from the MATLAB command line.

```
%Load the demo file
P_C=data;
File=['C:\Users\' getenv('USERNAME')
'\Documents\gtec\gBSanalyze\testdata\BCI\session1.mat'];
P_C=load(P_C,File);
```

If you load a file without sampling frequency set

```
P_C.SamplingFrequency=128;
```

Load Class Information

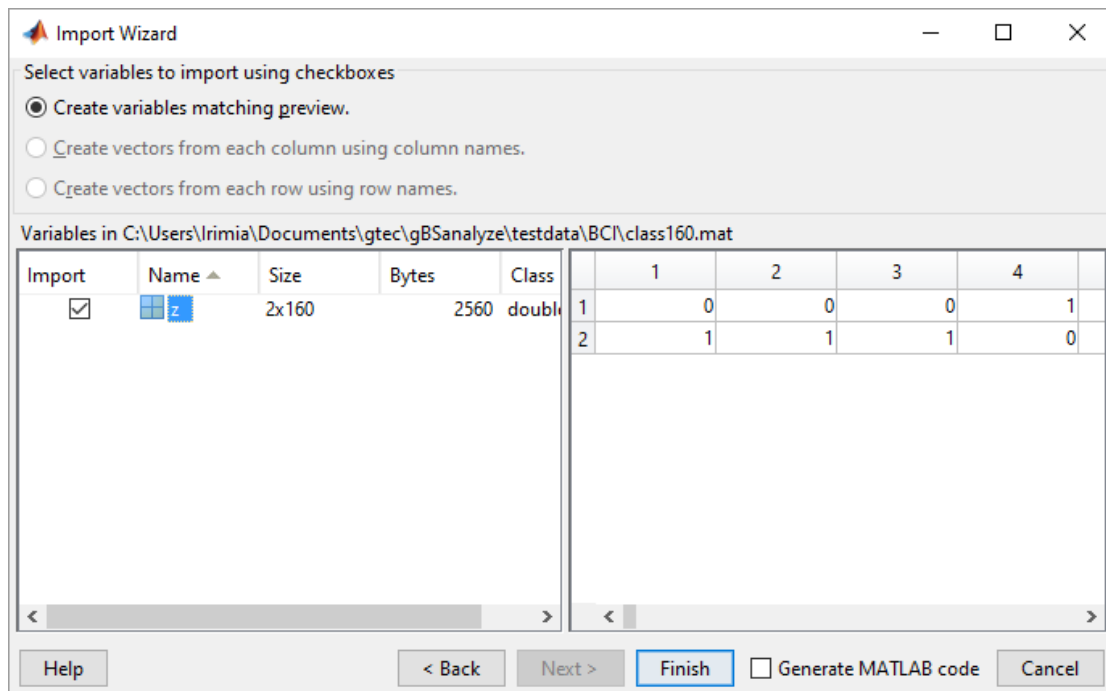
The **Load Class Information** window is used to assign trial attributes to the data from a given *.mat, or ASCII-file. The loaded class information vector must match the number of trials. Assume that you have loaded a data file with 3 trials into g.BSanalyze. An appropriate class information vector would be 0 1 0. Load class information would assign an attribute to the second trial.

1. Load the file `session1234triggered.mat` under

`Documents\gtec\gBSanalyze\testdata\BCI`

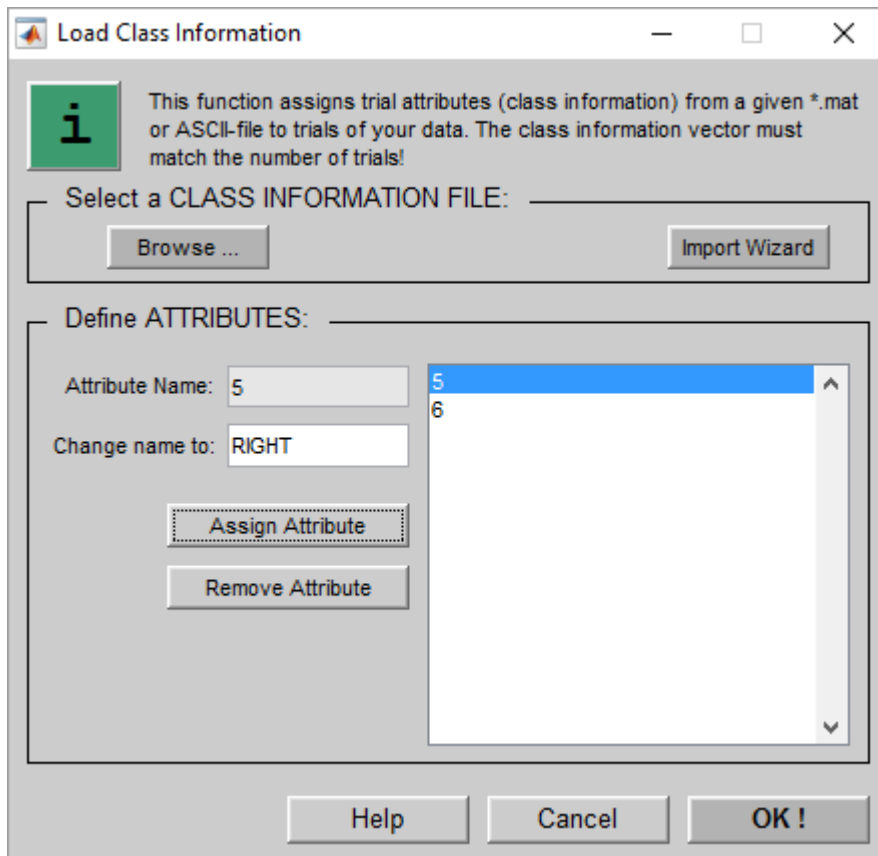
2. Select **Load Class Information** from the **File** menu.

3. Click the **Import Wizard** and search for the class information file `class160.mat` in the same directory. The Import Wizard shows the variables stored in the file and gives a truncated preview of the class information.



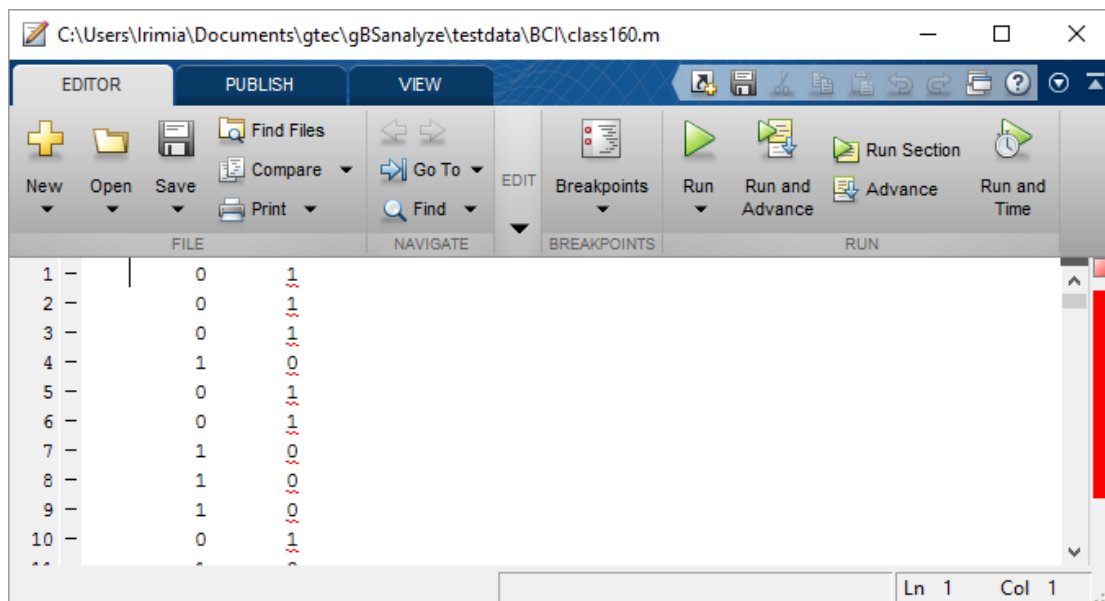
3. Click **Finish** to assign the attributes to the data trials. The window shows the attribute name 5 for the first row of the loaded class matrix and 6 for the second row.

4. Select the new loaded attribute 5, enter the attribute name `Right` and click **Assign Attribute**. Click on 6 and assign the name `LEFT`.



5. Close the window with **OK**. The attributes are assigned to your trials.

To load class information from an ASCII file click the **Browse** button and select the file `class160.m` which can be viewed with the MATLAB Editor. Each column in the file corresponds to a specific attribute and each row to a trial. The number of rows of the class information file and the number of trials in the Data Editor must be equal.



Trial 1 to 3 will receive the second trial attribute, trial 4 the first attribute, ...

Load Marker

Load Marker allows to import time markers into g.BSanalyze. The time markers can be stored as MATLAB file (*.mat) or in ASCII format.

Perform the following steps:

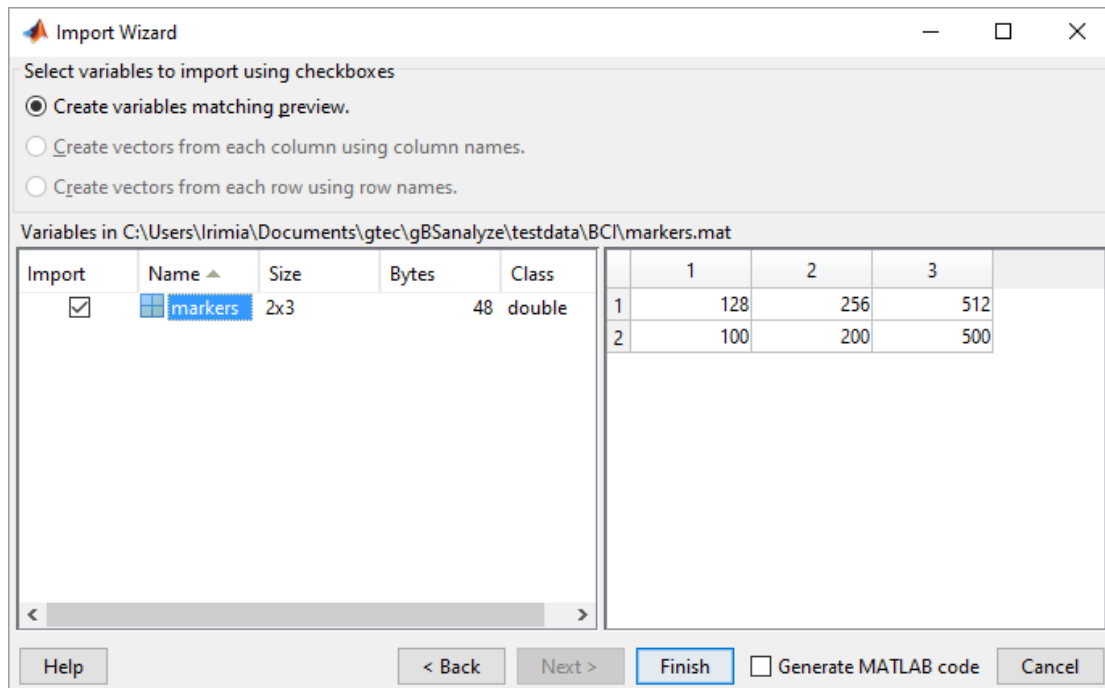
1. Load the file `session1.mat` from the following path

`Documents\gtec\gBSanalyze\testdata\bci`

2. Click on **Load Marker** under the **File** menu to open the following window

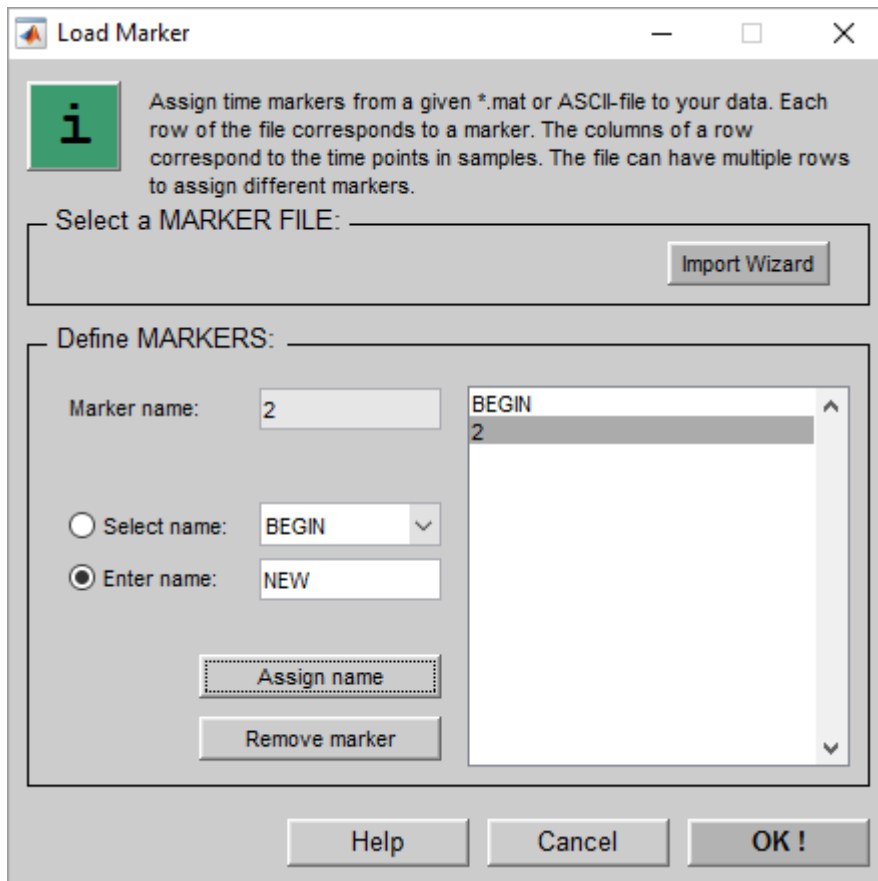
3. Click on **Import Wizard** and browse for the file

`Documents\gtec\gBSanalyze\testdata\bci\markers.mat`

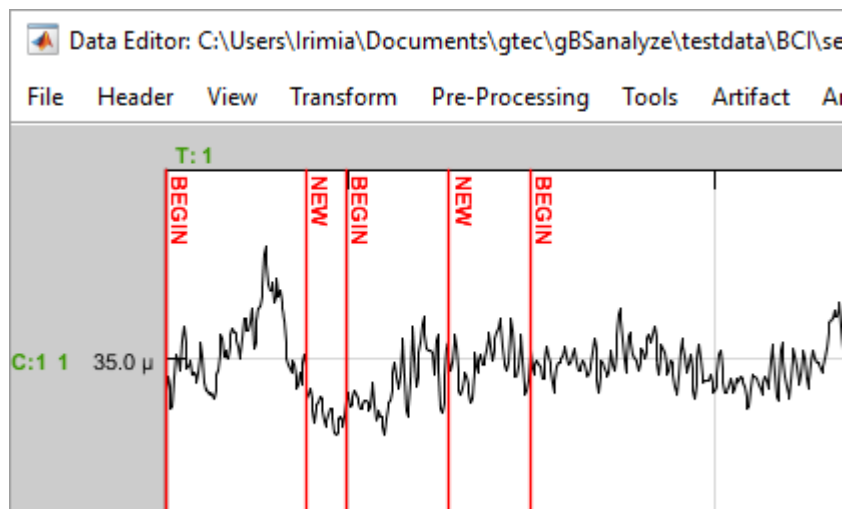


The **Import Wizard** gives a preview of the marker definition file. Each row represents a new kind of marker. Each column represents a new time point. Click on **Finish** to start the import.

4. Click on 1 in the listbox, select the name `BEGIN` and press **Assign name** to assign the marker name `BEGIN` to the first row. Click on 2 in the listbox, enter the name `NEW` and press **Assign name**. This assigns the marker name `NEW` to the second row.



5. Click **OK** to import the marker information. The **Data Editor** shows the imported markers:



To perform the example demonstrated above from the MATLAB command line use the following code:

```
%Load the demo data-set
P_C=data;
File= ['C:\Users\' getenv('USERNAME')
'\Documents\gttec\gBSanalyze\testdata\BCI\session1.mat'];
P_C=load(P_C,File);
P_C.SamplingFrequency=128;
```

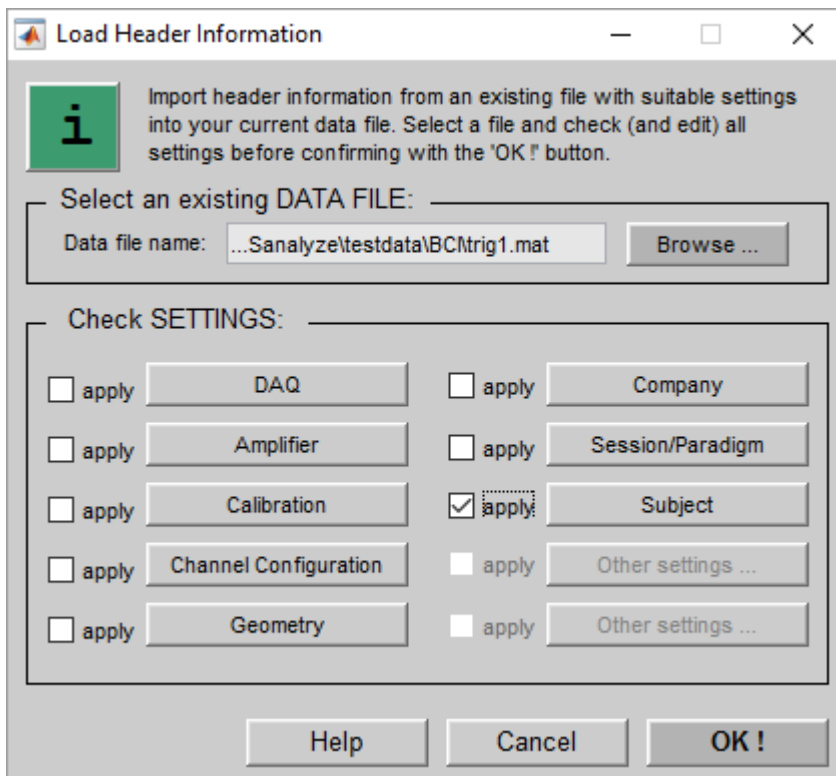
```
%Load Marker  
FileName=[];  
Markers=[128 256 512;100 200 500];  
MarkerName={'BEGIN'; 'NEW'};  
MarkerIndex=[1 2];  
P_C=gBSloadmarker(P_C,FileName,Markers,MarkerName,MarkerIndex);
```

Load Header Information

Use the **Load Header Information** window to assign information of already edited files to the current data-set in g.BSanalyze.

Perform the following steps:

1. Select **Load Header Information** from the **File** menu
2. Click the **Browse** button and open the already edited data-set



3. Click on the buttons and inspect the settings. If the settings are correct for a specific kind of header information check the box on the left side of the corresponding button.

Note that **DAQ**, **Amplifier**, **Calibration**, **Channel Configuration** and **Geometry** can only be loaded if the number of channels of the currently open data-set in g.BSanalyze corresponds to the number of channels of the loaded file. The sampling frequency of your data-set in g.BSanalyze must match to load **DAQ** settings. There are no limitations for **Company**, **Session/Paradigm** and **Subject**.

4. Click the **OK** button to import **Subject** data.

Load Scoring Data

Load Scoring Data is used to assign scoring information to your data-set in g.BSanalyze. It loads marked epochs edited in free mode, multi-trial mode or multi-channel mode (see Section The Data Editor – Epoching and Comment). The **Save Scoring Data** function is used to store the epochs.

Perform the following steps:

1. Load the file `session1.mat` from the following path

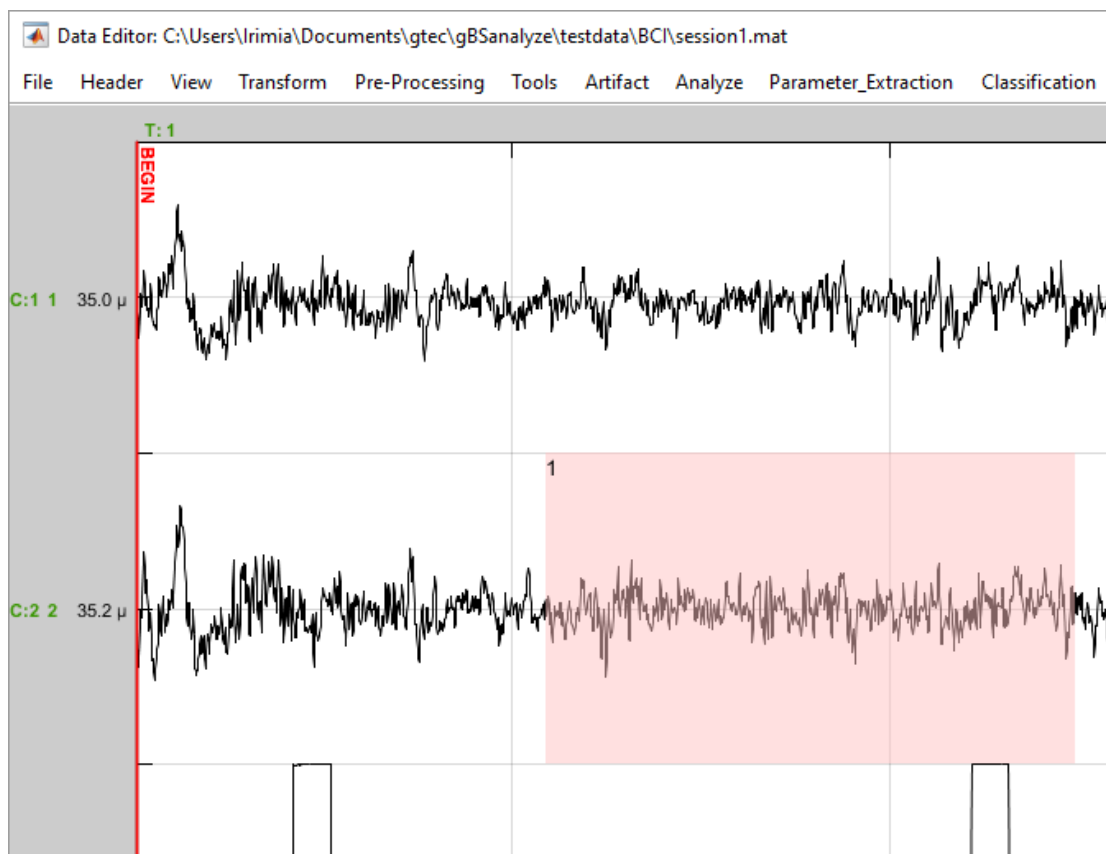
`Documents\gtec\gBSanalyze\testdata\bci`

The data-set does not contain any scoring information.

2. Click on **Load Scoring Data** under the **File** menu and select

`session1_score.mat`

Click **Open** to assign the epoch to the data-set.

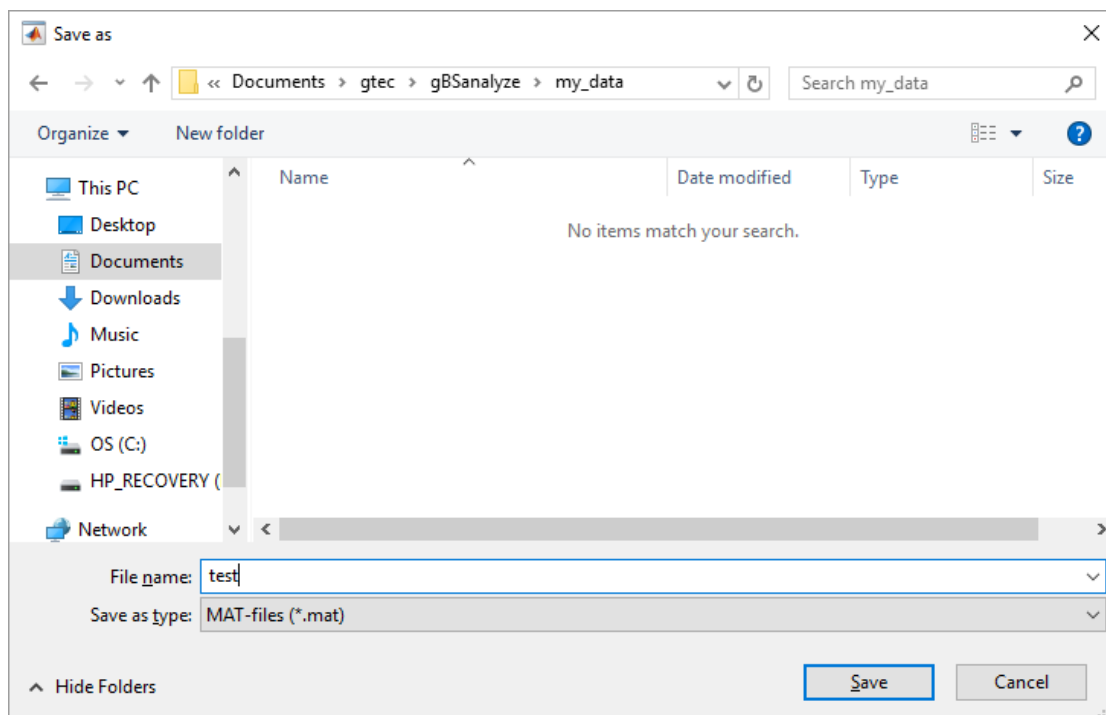


Save and Save as

g.BSanalyze allows to store the edited data including all header information as MATLAB-file onto your harddisk. **Save** stores the data under the current filename. **Save as** allows to store the data under a new filename.

Perform the following steps:

1. Select **Save as** from the **File** menu
2. Change to the directory where the data should be stored, enter `test` into the **File name** box and click **Save**.



If the file already exists you are asked to overwrite the older file.

The following code shows how to perform the example demonstrated above from the MATLAB command line.

```
%Save the demo file  
P_C_S=P_C;  
save(['C:\Users\' getenv('USERNAME')  
'\Documents\gtec\gBSanalyze\my_data\test.mat'], 'P_C_S');
```

Save Scoring Data

Save Scoring Data allows to store only the epochs edited in free-mode, multi-trial mode and multi-channel mode to harddisk. The scoring information can be loaded with **Load Scoring Data** into g.BSanalyze.

Import

g.BSanalyze allow to import third party data-sets with special filters. The current version supports:

European data format (EDF & EDFplus)

ASCII data

RDF data

BKR data

Neuroscan (CNT) data

Task Force Monitor (TFM) data

Block Import for BKR, CNT, EDF, BDF, GDF data

g.MOBILab – the mobile laboratory from g.tec on a Pocket PC

MIT data

BCI2000 data

Micromed data

Biopack data

BDF data

Axona data

mindBEAGLE data

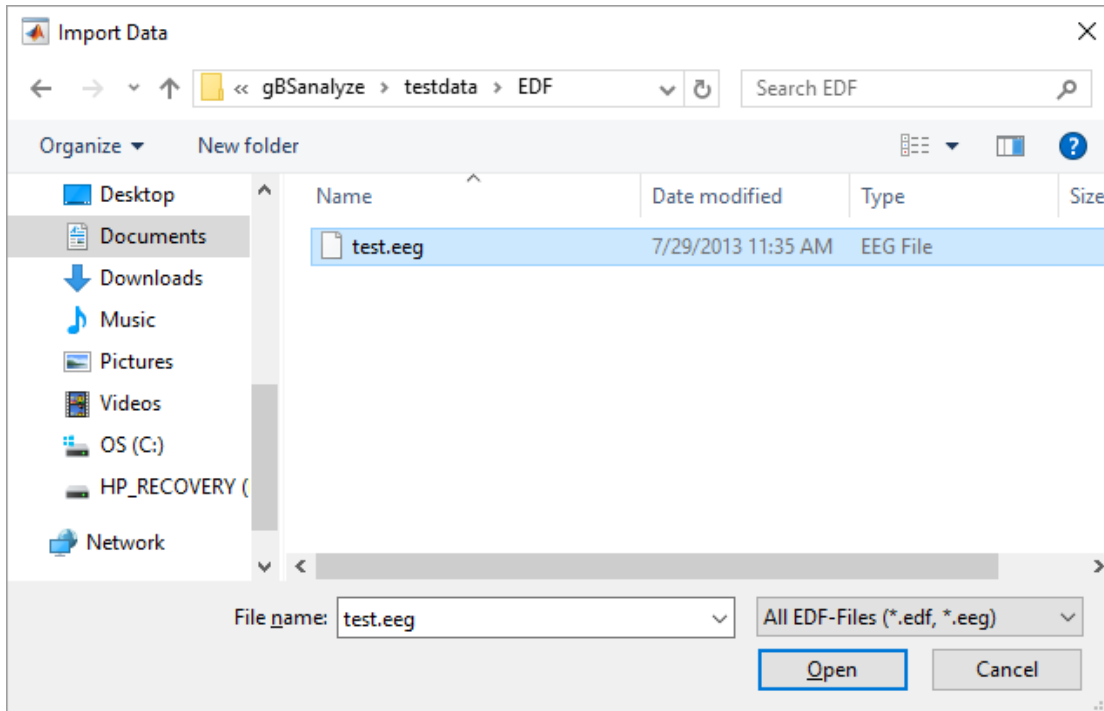
cortiQ data

recoveriX data

Import EDF

The **Import Data** window allows to load data-sets stored in European data format into g.BSanalyze. The extension can be '.edf' or '.eeg'.

1. Click on **Import EDF** from the **File** menu to open the **Import Data** window



2. Change to the directory

Documents\gtec\gBSanalyze\testdata\edf

3. Select the file `test.eeg` and click **Open** to load the data-set into g.BSanalyze

Reference

Kemp, G., Värri, A., Rosa, A.C., Nielsen, K.D., Gade, J., "A simple format for exchange of digitized polygraphic recordings," Electroencephalography and Clinical Neurophysiology, vol. 82, pp. 391-393, 1992.

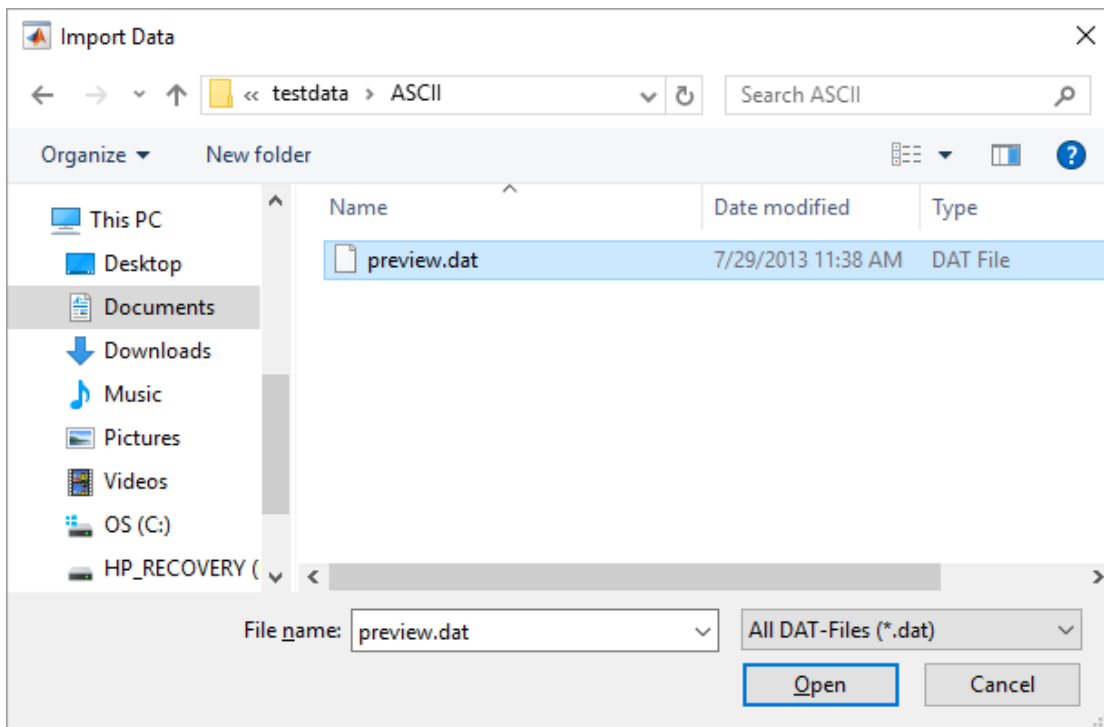
To perform the operation from the MATLAB command line, use:

```
%Import EDF-data
P_C=data;
File= ['C:\Users\' getenv('USERNAME')
'\Documents\gtec\gBSanalyze\testdata\edf\test.eeg'];
P_C=import(P_C, 'EDF', File);
```

Import ASCII

The **Import ASCII** window allows to load data-sets stored in ASCII format into g.BSanalyze. The extension must be '.dat'.

1. Click on **Import ASCII** from the **File** menu to open the **Import Data** window



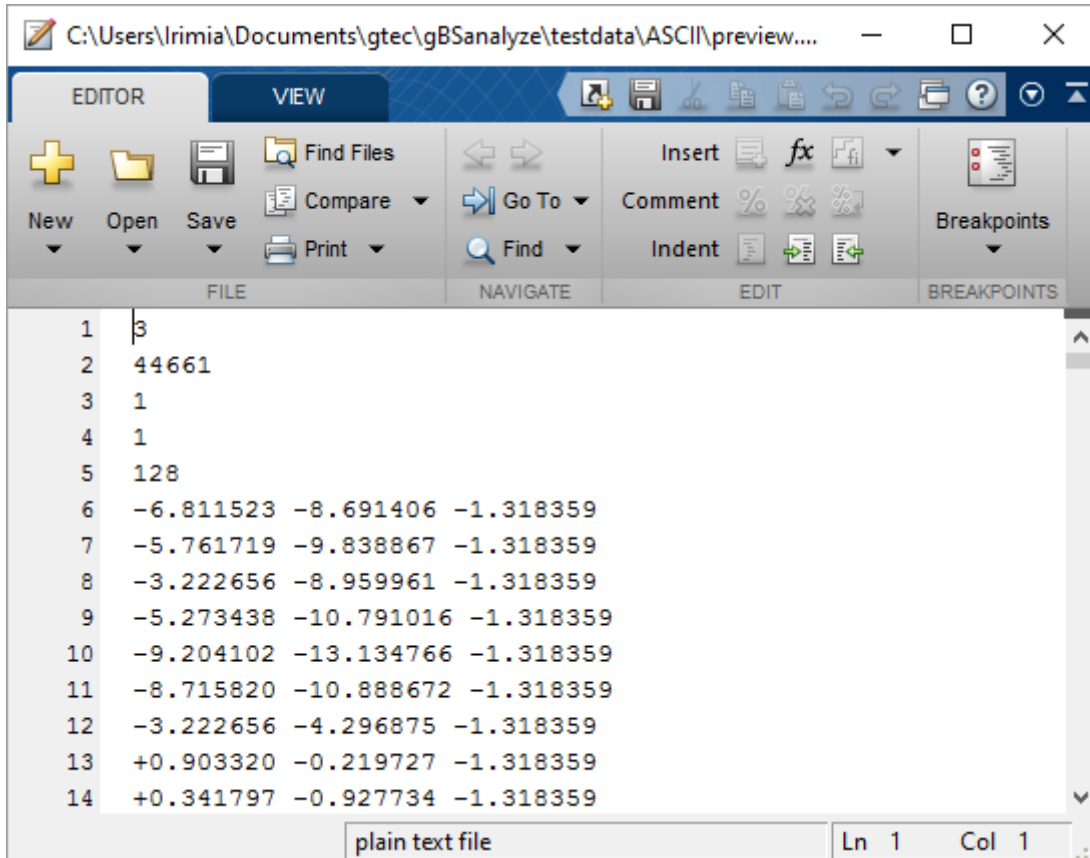
2. Change to the directory

Documents\gtec\gBSanalyze\testdata\ASCII

3. Select the file `preview.dat` and click **Open** to load the data-set into g.BSanalyze

The ASCII matrix must have the following format:

1. row: channels
2. row: samples
3. row: trials
4. row: 1 – samples are rows
2 – samples are columns
5. row: sampling frequency in Hz
6. to Nth row: data



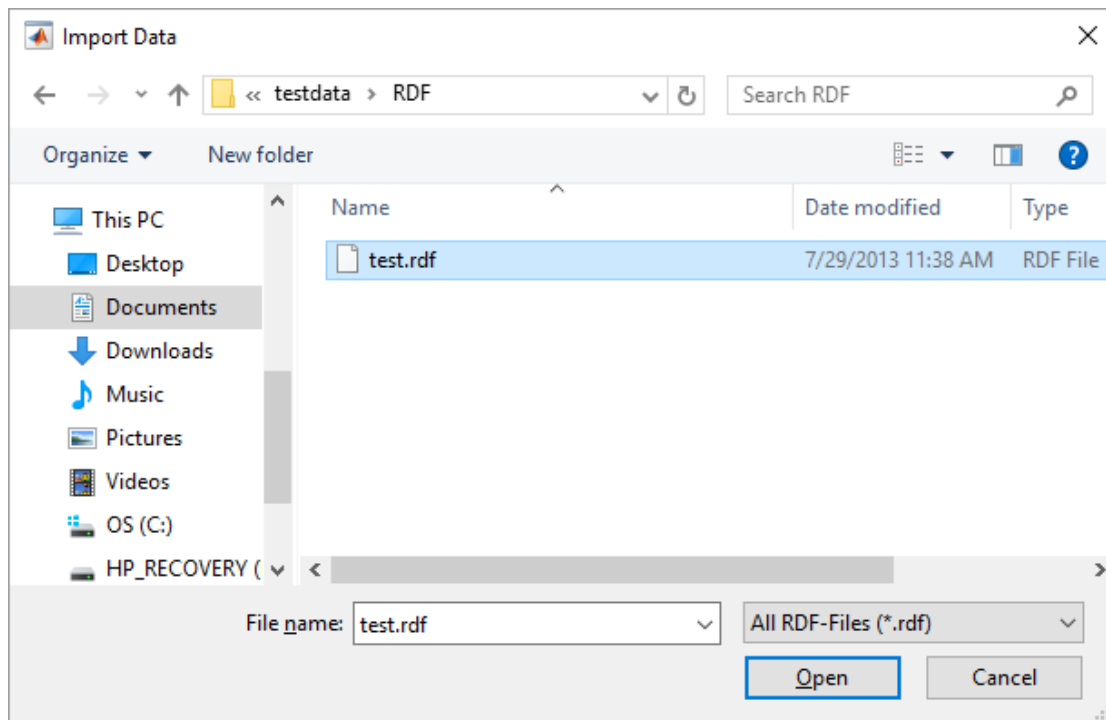
To perform the operation from the MATLAB command line, use:

```
%Import ASCII-data  
P_C=data;  
File=['C:\Users\' getenv('USERNAME')  
'\Documents\gtec\gBSanalyze\testdata\ASCII\preview.dat'];  
P_C=import(P_C, 'ASCII', File);
```

Import RDF

The **Import Data** window allows to load data-sets stored in RDF data format into g.BSanalyze. The extension must be '.rdf'.

1. Click on **Import RDF** from the **File** menu to open the **Import Data** window



2. Change to the directory

Documents\gtec\gBSanalyze\testdata\rdf

3. Select the file `test.rdf` and click **Open** to load the data-set into g.BSanalyze

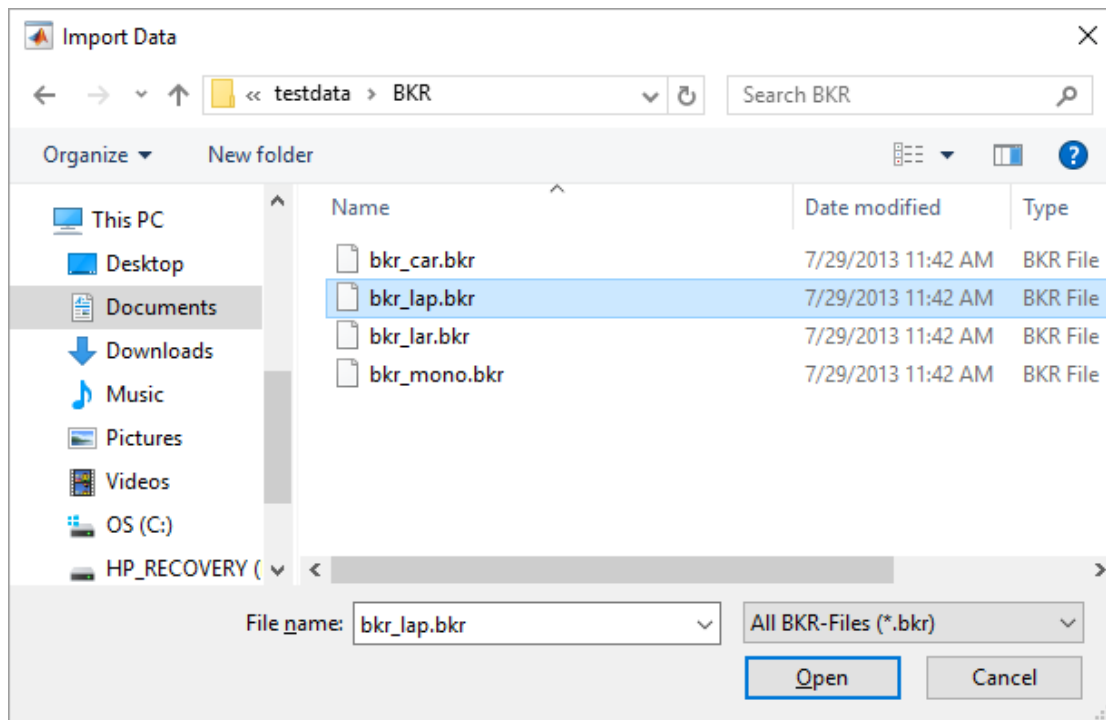
To perform the operation from the MATLAB command line, use:

```
%Import RDF-data  
P_C=data;  
File= ['C:\Users\' getenv('USERNAME')  
'\Documents\gtec\gBSanalyze\testdata\RDF\test.rdf'];  
P_C=import(P_C, 'RDF', File);
```

Import BKR

The **Import Data** window allows to load data-sets stored in BKR data format into g.BSanalyze. The extension must be '.bkr'.

1. Click on **Import BKR** from the **File** menu to open the **Import Data** window



2. Change to the directory

Documents\gtec\gBSanalyze\testdata\bkr

3. Select the file `bkr_lap.bkr` and click **Open** to load the data-set into g.BSanalyze

To perform the operation from the MATLAB command line, use:

```
%Import BKR-data
P_C=data;
File= ['C:\Users\' getenv('USERNAME')
'\Documents\gtec\gBSanalyze\testdata\bkr\bkr_lap.bkr'];
P_C=import(P_C, 'BKR', File);
```

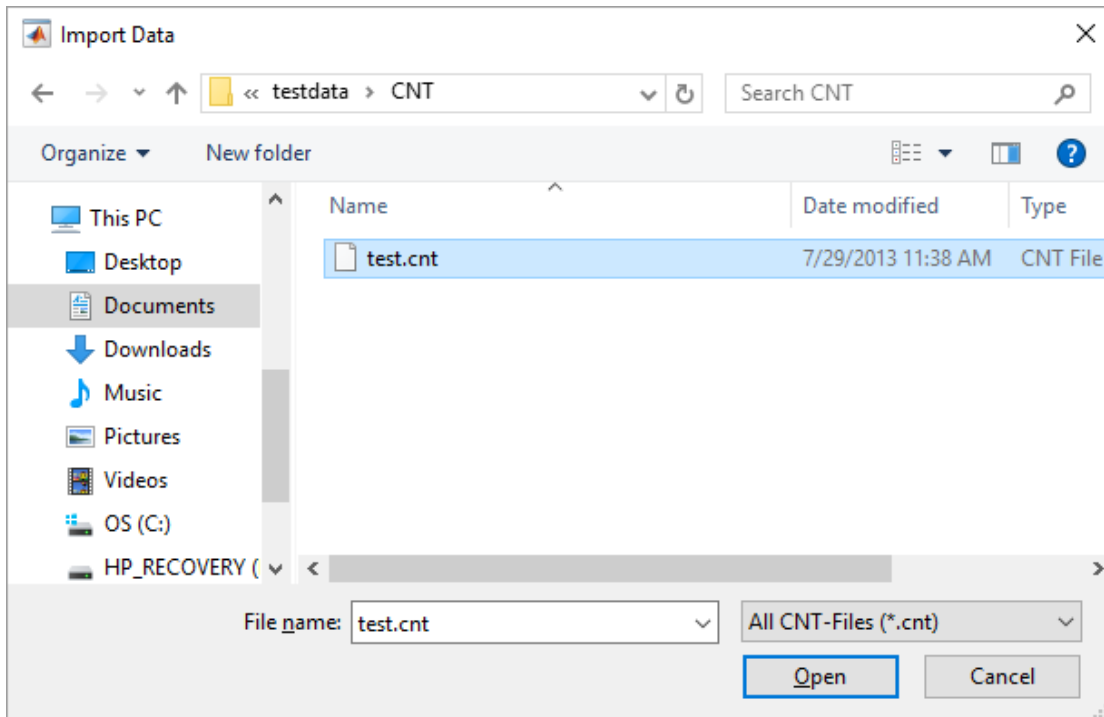
Import CNT

The **Import Data** window allows to load data-sets stored in CNT data format from Neuroscan into g.BSanalyze. The extension can be '.cnt'.

1. Click on **Import CNT** from the **File** menu to open the **Import Data** window
2. Change to the directory

Documents\gtec\gBSanalyze\testdata\CNT

3. Select the file `test.cnt` and click **Open** to load the data-set into g.BSanalyze



To perform the operation from the MATLAB command line, use:

```
%Import CNT-data  
P_C=data;  
File= ['C:\Users\' getenv('USERNAME')  
'\Documents\gtec\gBSanalyze\testdata\CNT\test.cnt'];  
P_C=import(P_C, 'CNT', File);
```

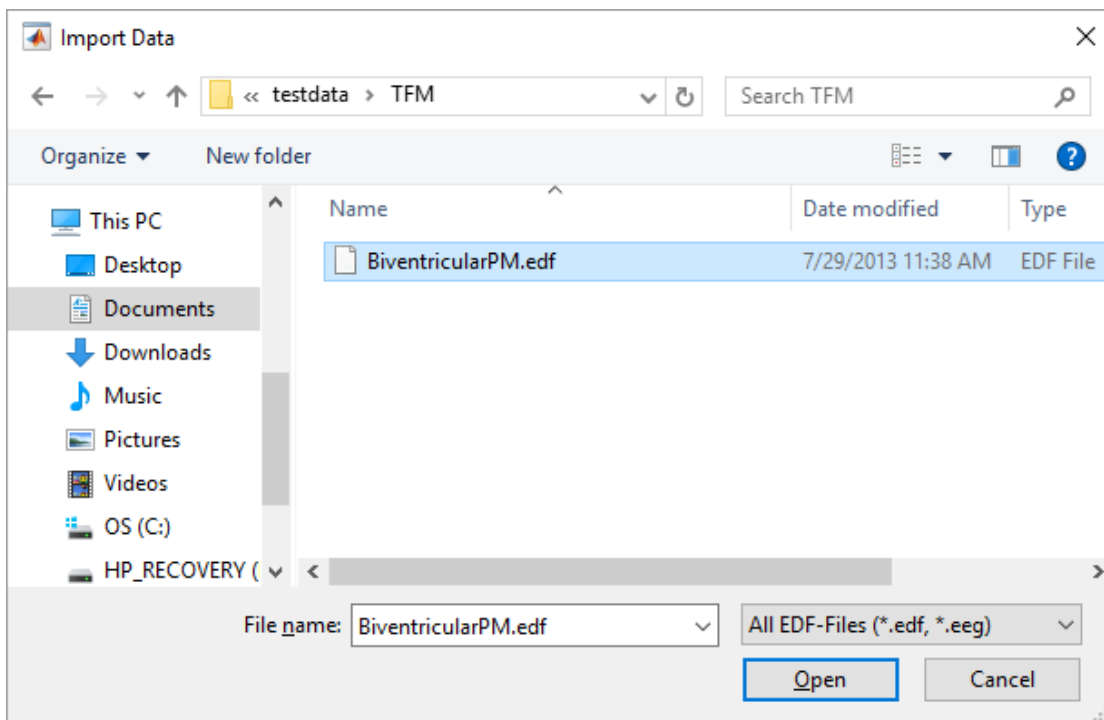
Import TFM

The **Import Data** window allows to load data-sets stored in EDF data format from the Task Force Monitor from CNSystems (Graz, Austria) into g.BSanalyze. The extension can be '.edf'.

1. Click on **Import TFM** from the **File** menu to open the **Import Data** window
2. Change to the directory

Documents\gtec\gBSanalyze\testdata\TFM

3. Select the file `BiventricularPM.edf` and click **Open** to load the data-set into g.BSanalyze



To perform the operation from the MATLAB command line, use:

```
%Import TFM-data  
P_C=data;  
File= ['C:\Users\' getenv('USERNAME')  
'\Documents\gtec\gBSanalyze\testdata\TFM\BiventricularPM.edf'];  
P_C=import(P_C, 'TFM', File);
```


Import Block

Block import allows to load segments (blocks) of BKR, CNT, EDF, BDF, and GDF data. The function has the following 3 modes:

- 1) **ALL** – load the whole data-set
- 2) **PERIOD** – load only a smaller part of the data-set
- 3) **BLOCK** – divide the data-set into equal parts and load each part separately

Block Import

Select a data file and specify the blocksize. Rognized data formats are BKR, CNT, EDF, BDF and GDF.

Select a DATA FILE: _____

Data file name: ...BSanalyze\testdata\CNT\test.cnt Browse ...

Data PROPERTIES: _____

Number of channels: 32 Number of: 31270

Sampling frequency: 250 [Hz] Total time: 125.08 [s]

Specify BLOCKSIZE: _____

ALL

PERIOD

from _____ to _____ [s]

_____ [sample]

BLOCK

Block size: 30 [s]

7500 [samples]

Resulting number of 5 Select block: 1

Selection: _____

from 0.004 to 30 [s]

1 7500 [sample]

Select CHANNELS: _____

Select channels ...

Help Cancel Import !

The **PERIOD** and **BLOCK** modes allow to work with very large data-sets in MATLAB.

1. Click on **Import Block** from the **File** menu to open the **Block Import** window

2. Click on **Browse** and change to the directory

```
Documents\gtec\gBSanalyze\testdata\CNT
```

3. Select the file `test.cnt`

4. Enable the radio button **BLOCK** and specify a **Block size** of 30 s. **Select block** number 1 from the pull-down menu. The **Selection** field gives a preview of the data fragment that should be loaded into g.BSanalyze.

5. Press **Import**

To switch to the next segment enter the **Block Import** window again and select the appropriate segment from the **Select block** pull-down menu.

The source code of the **Block Import** window functions is available under

```
C:\Program Files\gtec\gBSanalyze\user\t200
```

Note that the **Select channels...** function works only for BKR data.

To perform the operation from the MATLAB command line, use:

```
%Import Data
P_C=data;
File=['C:\Users\' getenv('USERNAME')
'\Documents\gtec\gBSanalyze\testdata\CNT\test.cnt'];
ChannelInclude=[1  2  3  4  5  6  7  8  9  10  11  12  13  14  15
16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32];
Period=[1  7500];
P_C=import(P_C, 'BLOCK', File, ChannelInclude, Period);
```

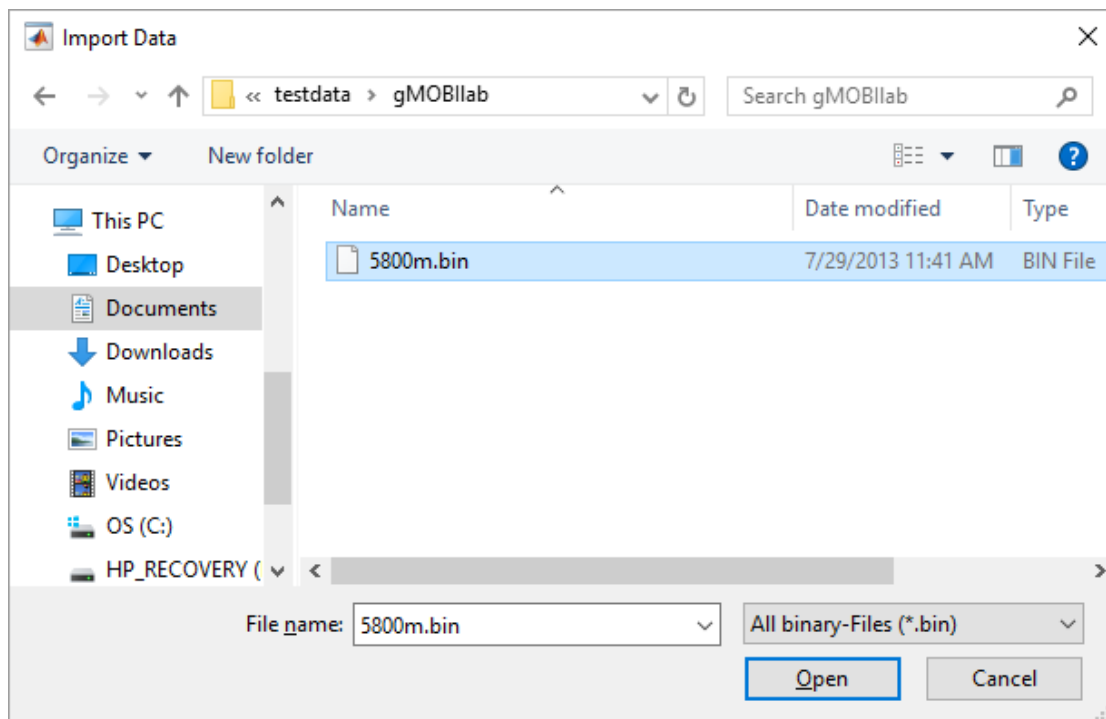
Import g.MOBILab

The **Import Data** window allows to load data-sets stored in ASCII data format from g.MOBILab into g.BSanalyze. The extension can be '.txt'.

1. Click on **Import gMOBILab** from the **File** menu to open the **Import Data** window
2. Change to the directory

Documents\gtec\gBSanalyze\testdata\gMOBILab

3. Select the file 5800m.txt and click **Open** to load the data-set into g.BSanalyze



To perform the operation from the MATLAB command line, use:

```
%Import gMOBILab-data
P_C=data;
File= ['C:\Users\' getenv('USERNAME')
'\Documents\gtec\gBSanalyze\testdata\gMOBILab\5800m.bin'];
P_C=import(P_C, 'GMOBILAB', File);
```

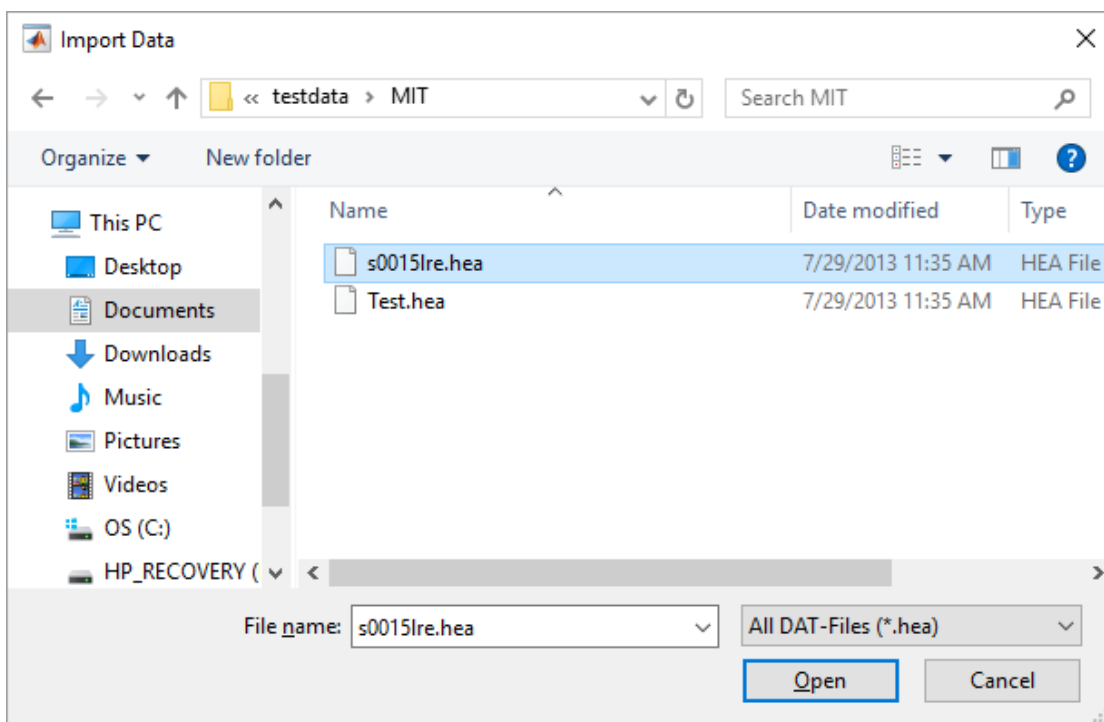
Import MIT

The **Import Data** window allows to load data-sets stored in MIT data format into g.BSanalyze. The extension can be '.hea'.

1. Click on **Import MIT** from the **File** menu to open the **Import Data** window
2. Change to the directory

Documents\gtec\gBSanalyze\testdata\MIT

3. Select the file `s00151re.he` and click **Open** to load the data-set into g.BSanalyze



To perform the operation from the MATLAB command line, use:

```
%Import Data  
P_C=data;  
File= ['C:\Users\' getenv('USERNAME')  
'\Documents\gtec\gBSanalyze\testdata\MIT\s00151re.he'];  
P_C=import(P_C, 'MIT', File);
```

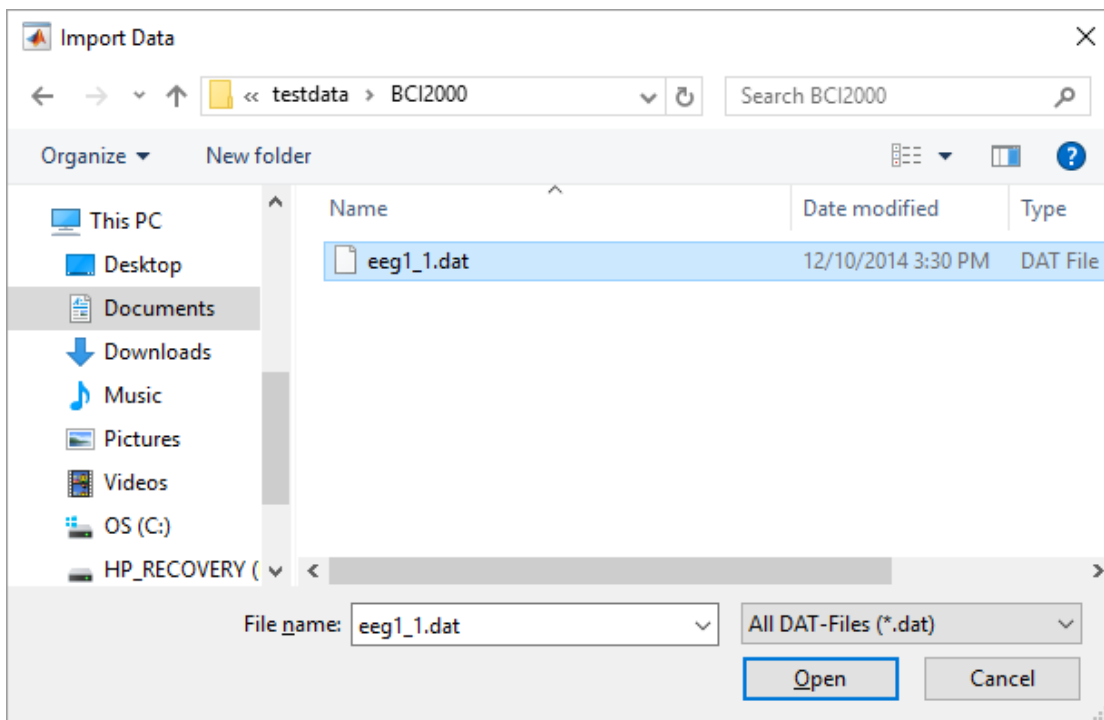
Import BCI2000

The **Import Data** window allows to load data-sets stored in BCI2000 data format into g.BSanalyze. The extension can be '.dat'.

1. Click on **Import BCI2000** from the **File** menu to open the **Import Data** window
2. Change to the directory

Documents\gtec\gBSanalyze\testdata\BCI2000

3. Select the file `eeg1_1.dat` and click **Open** to load the data-set into g.BSanalyze.



To perform the operation from the MATLAB command line, use:

```
%Import Data  
P_C=data;  
File= ['C:\Users\' getenv('USERNAME')  
'\Documents\gtec\gBSanalyze\testdata\BCI2000\eeg1_1.dat'];  
P_C=import(P_C, 'BCI2000', File);
```

Import BDF

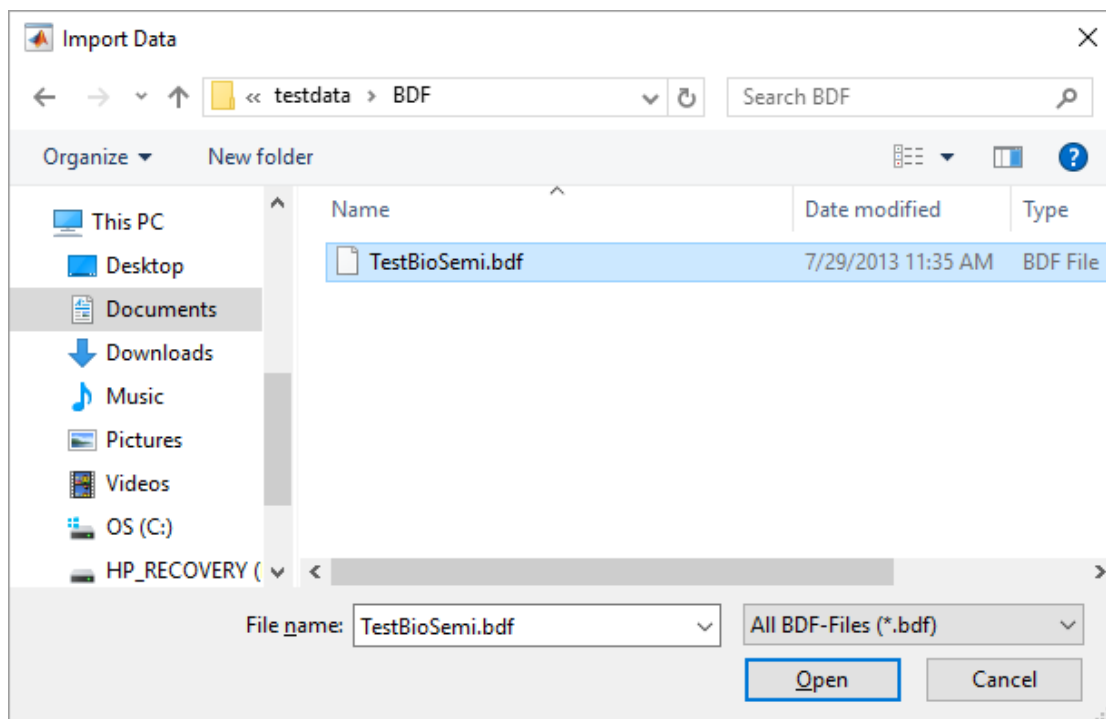
The **Import Data** window allows to load data-sets stored in BDF data format into g.BSanalyze. The extension can be '.bdf'.

1. Click on **Import BDF** from the **File** menu to open the **Import Data** window

2. Change to the directory

Documents\gtec\gBSanalyze\testdata\BDF

3. Select the file `TestBioSemi.bdf` and click **Open** to load the data-set into g.BSanalyze



To perform the operation from the MATLAB command line, use:

```
%Import Data  
P_C=data;  
File= ['C:\Users\' getenv('USERNAME')  
'\Documents\gtec\gBSanalyze\testdata\BDF\TestBioSemi.bdf'];  
P_C=import(P_C, 'BDF', File);
```

Export ASCII

The **Export ASCII** dialog box allows to export biosignal data and header information to ASCII format. Select the information that should be stored in ASCII format.

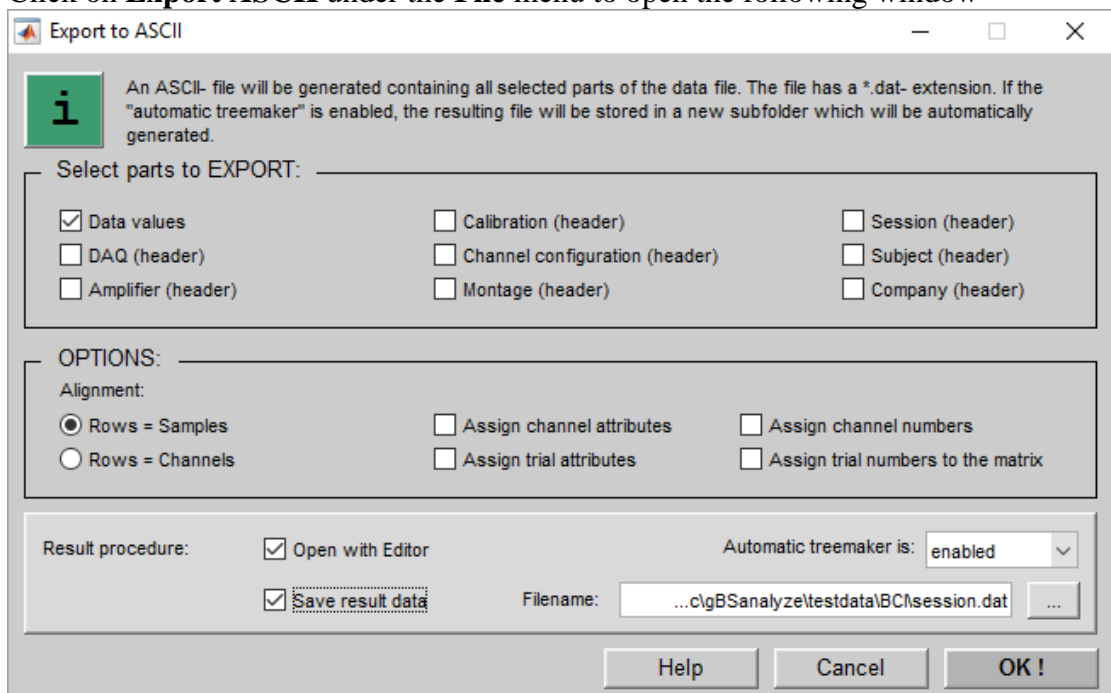
Use the **Open with Editor** checkbox to open the **MATLAB Editor** and to view the ASCII converted data.

Perform the following steps:

1. Load the file `session1.mat` from the following path

`Documents\gtec\gBSanalyze\testdata\bci`

2. Click on **Export ASCII** under the **File** menu to open the following window



3. Check **Data values** to convert the data matrix into ASCII and select as **Alignment Rows = Samples**
4. Check the **Open with Editor** and the **Save result data** box and select as filename `session1.dat`. The ASCII subdirectory is automatically created by the **Automatic treemaker** under the data directory.
5. Click **OK** to convert the data and to open the MATLAB Editor with the ASCII data

To perform the example demonstrated above from the MATLAB command line use the following code:

```
%Load the demo data-set  
P_C=data;
```

```

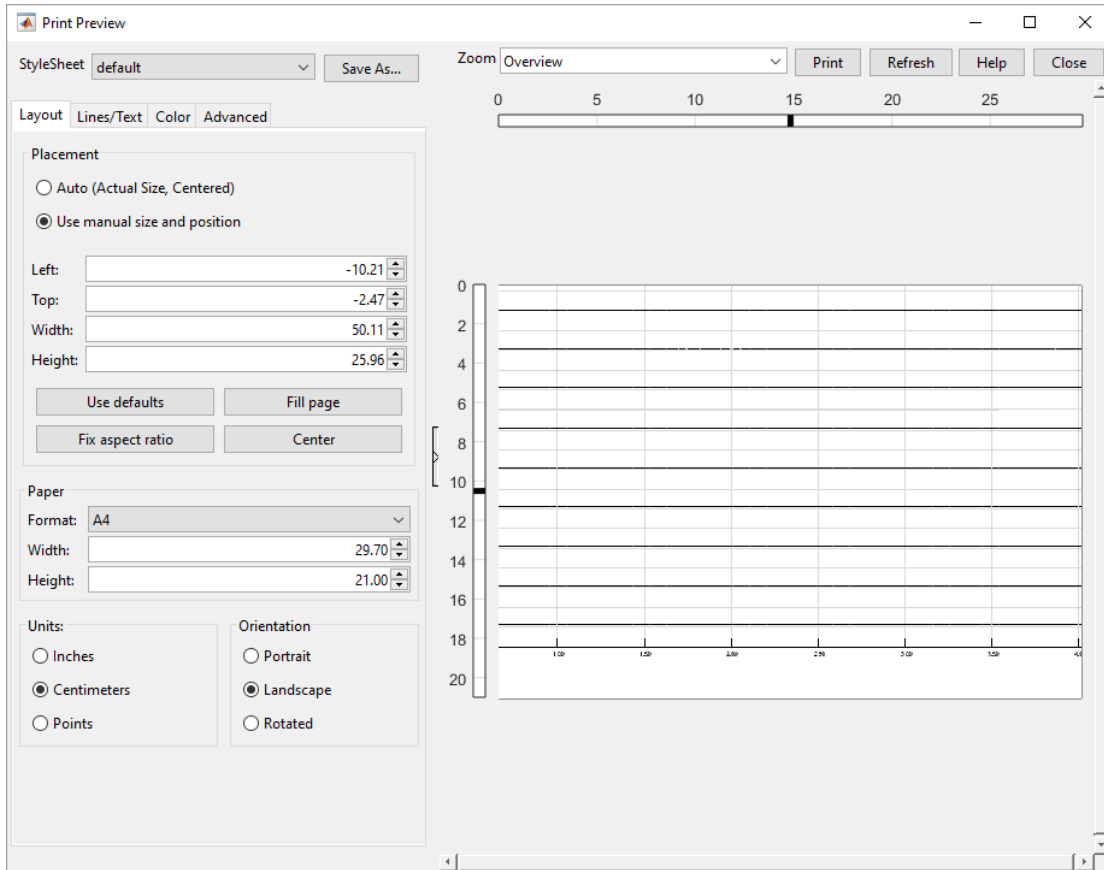
File= ['C:\Users\' getenv('USERNAME')
'\Documents\gtec\gBSanalyze\testdata\BCI\session1.mat'];
P_C=load(P_C,File);
P_C.SamplingFrequency=128;

%Save as ASCII
S.StoreData=1;           %convert data
S.RowsSamples=1;
S.RowsChannels=0;
S.AssignChNr=0;
S.AssignTrNr=0;
S.AssignChNr=0;
S.AssignTrAt=0;
S.AssignChAt=0;
S.DAQ=0;
S.Amplifier=0;
S.ChannelConfig=0;
S.Calibration=0;
S.Montage=0;
S.Subject=0;
S.Session=0;
S.Company=0;
FileName= ['C:\Users\' getenv('USERNAME')
'\Documents\gtec\gBSanalyze\testdata\BCI\ASCII\session1.dat'];
P_C=gBSexportascii(P_C,S,FileName,0);

```


Printer Preview

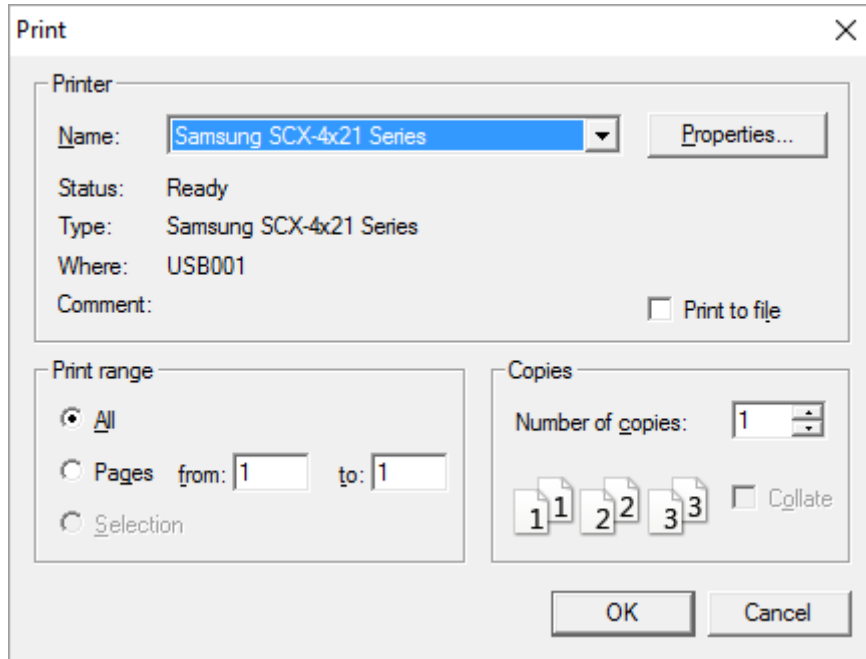
The dialog window gives a preview of the printed figure in g.BSanalyze. Use this dialog to control the layout and appearance of figure before sending it to a printer or print file.



For details, see MATLAB > Graphics > Printing and Exporting > `printpreview` help topics.

Print

1. Select **Print** from the **File** menu to open the dialog:



2. Chose a printer and confirm the printout with the **OK** button.

A printout into a file can be made if the **Print to file** checkbox is selected.

Close all Figures and Exit

To close all dialog windows select **Close all Figures** and to exit g.BSanalyze select **Exit** from the **File** menu. You are asked for confirmation if the checkbox **Ask for confirmation before exit** in the **Appearance Settings** dialog box in the **Options** menu is checked.

Header Information

There are two parts of header information that could be assigned in g.BSanalyze.

The first part contains dialog windows with information about the hardware and experimental settings of the data acquisition system:

- [DAQ](#) - information about the data acquisition boards
- [Amplifier](#) - information about the signal conditioning system
- [Channel Configuration](#) - biosignal electrode configuration
- [Calibration](#) - information about the calibration of the data
- [Geometry](#) - experimental location of biosignal electrodes
- [Marker](#) - external or keyboard markers set during data acquisition
- [Epoch](#) - epochs edited in free-mode, multi-channel or multi-trial mode

The second part contains dialog windows with general experimental settings. The windows are independent of the hardware settings:

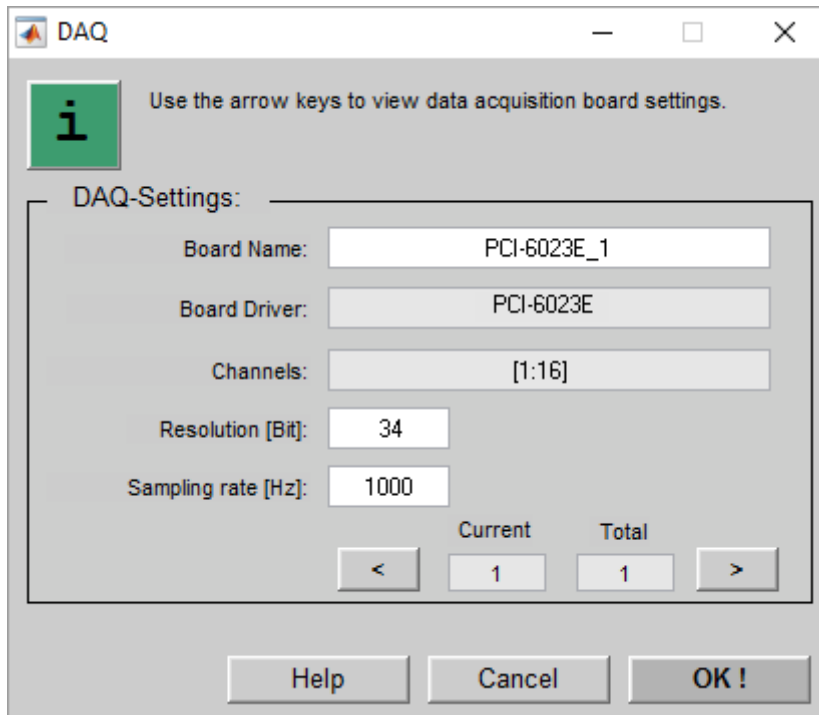
- [Subject](#) - information about the subject which participates in studies
- [Session/Paradigm](#) - session and paradigm information
- [Company](#) - information about the company and people who perform the experiments

If you are going to analyse a series of data files (e.g. for the same study) you can use the function "Load Header Information" to get header information from an already existing data file.

DAQ

The **DAQ** window contains the following information about analog input channels of the data acquisition boards:

- **Board Name** - name of the data acquisition board
- **Board Driver** - data acquisition driver associated with the data acquisition board
- **Channels** - ID of recorded channels (notation: [1:16], [1 5], [1 2 3 6 8 10], ...)
- **Resolution [Bit]** - resolution of the data acquisition board
- **Sampling rate [Hz]** - sampling frequency of the data acquisition board



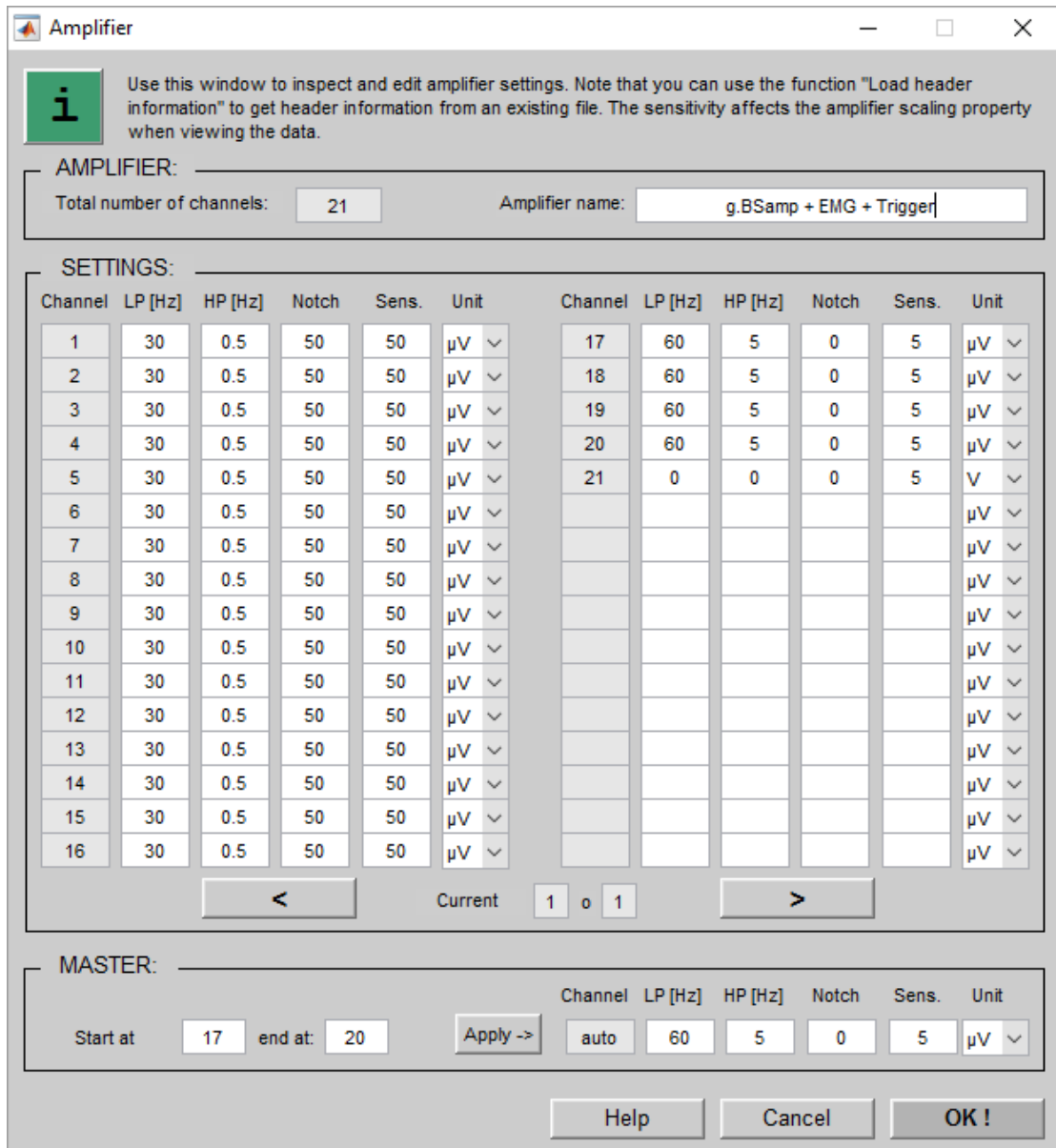
Use the arrow keys to scroll through all DAQ boards. The sampling frequency must be equal for all data acquisition boards. The **Resolution** is used for the **Data Quality** calculation under the **Analyze** menu.

Amplifier

The **Amplifier** dialog window is used to describe the settings of external conditioning systems. In the following example channels 1-16 are connected to g.Bsamp (1-12: EEG, 13-16: EOG), channels 17-20 are connected to a custom EMG amplifier and channel 21 is a trigger signal from g.STIMunit.

The dialog box has the following entries:

- **Total number of channels (status)** – number of channels acquired
- **Amplifier name** - enter the IDs of the biosignal conditioning systems
- **Channel (status)** - channel number according to the order of the data acquisition boards
- **LP [Hz]** - settings of the lowpass filter of the amplifier
- **HP [Hz]** - settings of the highpass filter of the amplifier
- **Notch [Hz]** - center frequency of the notch filter of the amplifier
- **Sens.** - sensitivity of the amplifier input (affects the data scaling in amplifier scaling mode)
- **Unit** - unit of the sensitivity of the amplifier input



If your data has more than 32 channels use the arrow keys to scroll through the pages.

Please follow these steps to achieve the settings as shown above.

1. Enter **Start at channel:** 1 and **end at:** 12 in the **MASTER** field
2. Enter a **LP** of 30, a **HP** of 0.5, a **Notch** of 50, a **Sens.** of 50 and set the **Unit** to µV
3. Click **Apply** in the **MASTER** field to assign the settings
4. Repeat steps 1 to 3 for the EOG and EMG settings
5. Click into the **Sens.** edit box of channel 21 and edit 5. Set the Unit to V

Channel Configuration

The **Channel** configuration dialog box is used to define channel names and channel types. In the following example channels 1-16 are connected to g.BSamp (1-12: EEG, 13-16: EOG), channels 17-20 are connected to a custom EMG amplifier and channel 21 is a trigger signal from g.STIMunit.

The dialog box has the following entries:

- **Total number of channels (status)** - information according to the data acquisition board
- **Chan. number (status)** - channel number according to the order of the data acquisition boards
- **Name** - of the channel (electrode) used to identify the signal
- **Channeltype** - type of biosignal or other recorded signal

Channel Configuration

Use this window to inspect and edit your channel configuration. Note that you can use the function "Load header information" to get header information from an existing file. To load an electrode configuration created with the g.MONcreator use the Browse button.

CONFIGURATION:

Total number of channels: Montage Creator

SETTINGS:

Chan. number:	Name:	Channeltype:	Chan. number:	Name:	Channeltype:
1	FP1	EEG	17	EMG1	EMG
2	FP2	EEG	18	EMG2	EMG
3	F7	EEG	19	EMG3	EMG
4	F3	EEG	20	EMG4	EMG
5	Fz	EEG	21	STIM	TRG
6	F4	EEG			N.S.
7	F8	EEG			N.S.
8	T7	EEG			N.S.
9	C3	EEG			N.S.
10	Cz	EEG			N.S.
11	C4	EEG			N.S.
12	T8	EEG			N.S.
13	VEOG1	EOG			N.S.
14	HEOG1	EOG			N.S.
15	VEOG2	EOG			N.S.
16	HEOG2	EOG			N.S.

< Current 1 of 1 >

MASTER:

Start at channel: end at: Chan. number: Name: Channeltype:

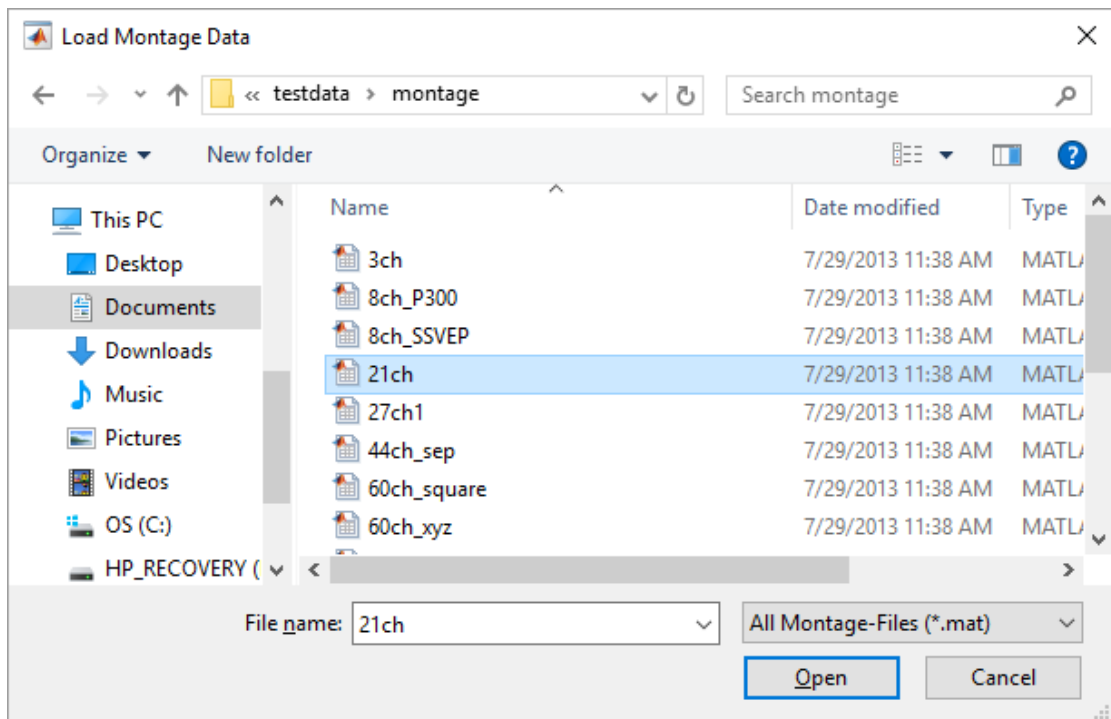
If your data has more than 32 channels use the arrow keys for scrolling.

Please follow these steps to achieve the settings as shown above:

1. Enter **Start at channel:** 1 and **end at:** 12 in the **MASTER** field
2. Select EEG as **Channeltype**
3. Click **Apply** in the **MASTER** field to assign the settings
4. Repeat steps 1 to 3 for the EOG and EMG settings
5. For channel 21 select TRG in the **Channeltype** pull-down menu
6. Go to the **Name** boxes for each channel to edit the names

To load an electrode configuration created with the g.MONcreator perform these steps:

1. Click on the **Browse** button and select the montage file you created with g.MONcreator for 21 input channels.



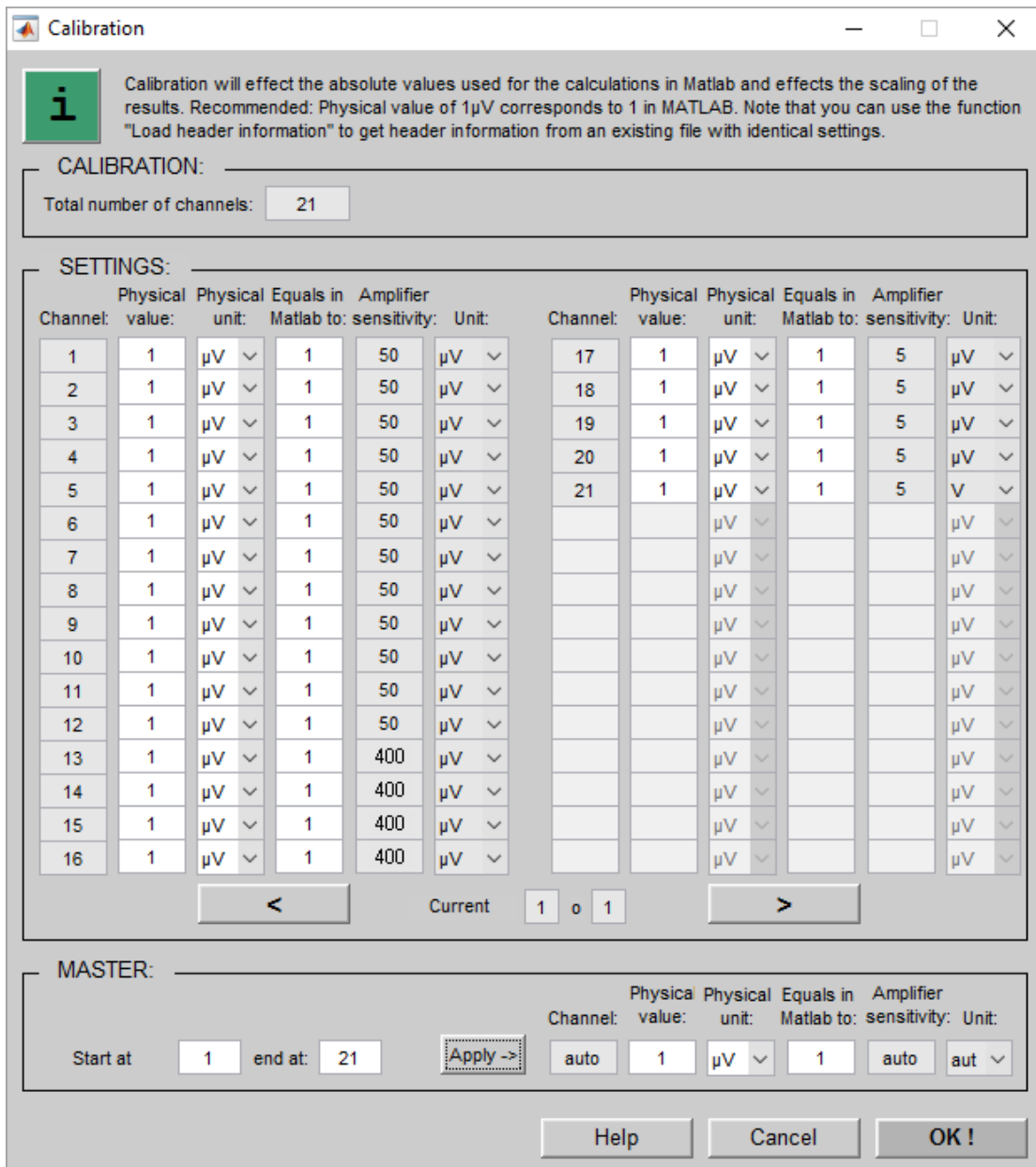
2. Select the file and click **Open**. Note that the number of channels in g.BSanalyze must match the number of channels of the montage file.
3. Confirm the settings and close the window with **OK**.

Calibration

The **Calibration** dialog box is used to define how a physical value is presented in MATLAB.

The dialog box has the following entries:

- **Total number of channels (status)**
- **Channel (status)** - channel number according to the order of the data acquisition boards
- **Physical value** and **Physical unit** - the value (magnitude) of the physical signal (e.g. 1 μV)
- **Equals in Matlab** - calibration value of how the physical signal is presented in MATLAB (e.g. 1)
- **Amplifier sensitivity (status)** - sensitivity of the amplifier input
- **Unit (status)** - unit of the sensitivity of the amplifier input



If your data has more than 32 channels use the arrow keys for scrolling.

To define settings for more channels at once, please follow these steps to achieve the settings as shown above:

1. Enter **Start at channel:** 1 and **end at:** 21 in the **MASTER** field
2. Select a **Physical value** of 1 μ V **equals in Matlab** to 1
3. Click **Apply** in the **MASTER** field to assign the settings
4. Confirm the settings and close the window with **OK**

The following code show how to perform the example demonstrated above from the MATLAB command line.

```
%Calibration  
PhysicalValue=[1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1  
1];  
PhysicalUnit=[1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1  
1];  
MatlabValue=[1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1  
1];  
P_C=gBscalibrate(P_C,PhysicalValue,PhysicalUnit,MatlabValue);
```

Geometry

The **Geometry** dialog box contains information about x-, y- and z-co-ordinates of the electrode montage. Use the g.MONcreator to assign electrode names and co-ordinates and store the montage. Geometry data will be used in g.BSanalyze for calculating **Source Derivations** where the electrodes are weighted according to their distance between them (e.g. weighted average reference) and for topographic presentation in Result2d.

To assign the created montage to the data in g.BSanalyze follow these steps:

1. Click on **Browse** and search for the montage file you created

GEOMETRY:

Total number of channels: 21 Montage Creator: ...\testdata\montage\21ch.mat Browse ...

SETTINGS:

Chan. number:	Name:	Pos. X / Y / Z [mm]	Chan. number:	Name:	Pos. X / Y / Z [mm]
1	FP1	-0.31 / 0.95 / 0	17	EMG1	0 / 0 / 0
2	FP2	0.31 / 0.95 / 0	18	EMG2	0 / 0 / 0
3	F7	-0.81 / 0.59 / 0	19	EMG3	0 / 0 / 0
4	F3	-0.57 / 0.7 / 0.44	20	EMG4	0 / 0 / 0
5	Fz	0 / 0.71 / 0.71	21	STIM	0 / 0 / 0
6	F4	0.57 / 0.7 / 0.44			
7	F8	0.81 / 0.59 / 0			
8	T7	-1 / 0 / 0			
9	C3	-0.71 / 0 / 0.71			
10	Cz	0 / 0 / 1			
11	C4	0.71 / 0 / 0.71			
12	T8	1 / 0 / 0			
13	VEOG1	-0.27 / 0.82 / -0.5			
14	HEOG1	-0.27 / 0.82 / -0.5			
15	VEOG2	0.27 / 0.82 / -0.5			
16	HEOG2	0.27 / 0.82 / -0.5			

Current 1 of 1

EDIT Montage settings:

g.MONcreator ...

Help Cancel OK !

2. Select the file which contains electrode information for 21 channels. Click **Open** to assign the co-ordinates to your data.
3. To change x-, y- or z-co-ordinates go directly to the editor box and adapt the values
4. Confirm the settings and close the window with **OK**

Marker

The **Marker** dialog box includes information about time markers. Time markers are assigned either during recording of the data with g.DAQsys or in g.BSanalyze.

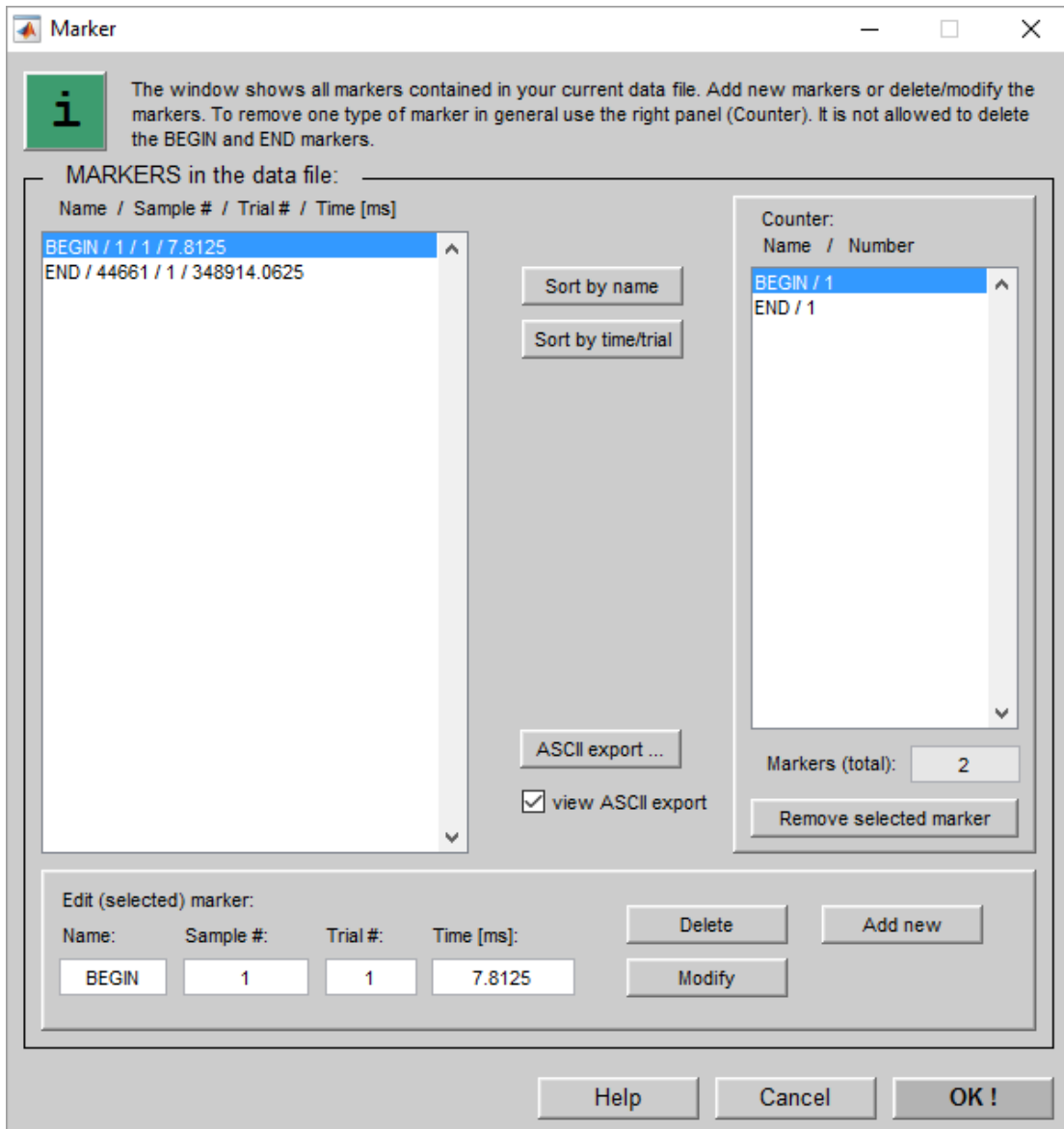
The dialog box has the following fields:

Markers - listbox shows all assigned markers. Use **Sort by name** and **Sort by time/trial** button to sort the marker list.

Counter - listbox shows all marker names and the total number of each marker. Use the **Remove selected marker** button to delete all markers with a specific name.

ASCII export - store the marker list to a ASCII file. If **view ASCII export** is checked the markers are shown with the MATLAB editor.

Edit (selected) marker - allows to **Add new** markers, to **Delete** already defined markers and to **Modify** the name, sample and time of an already defined marker.

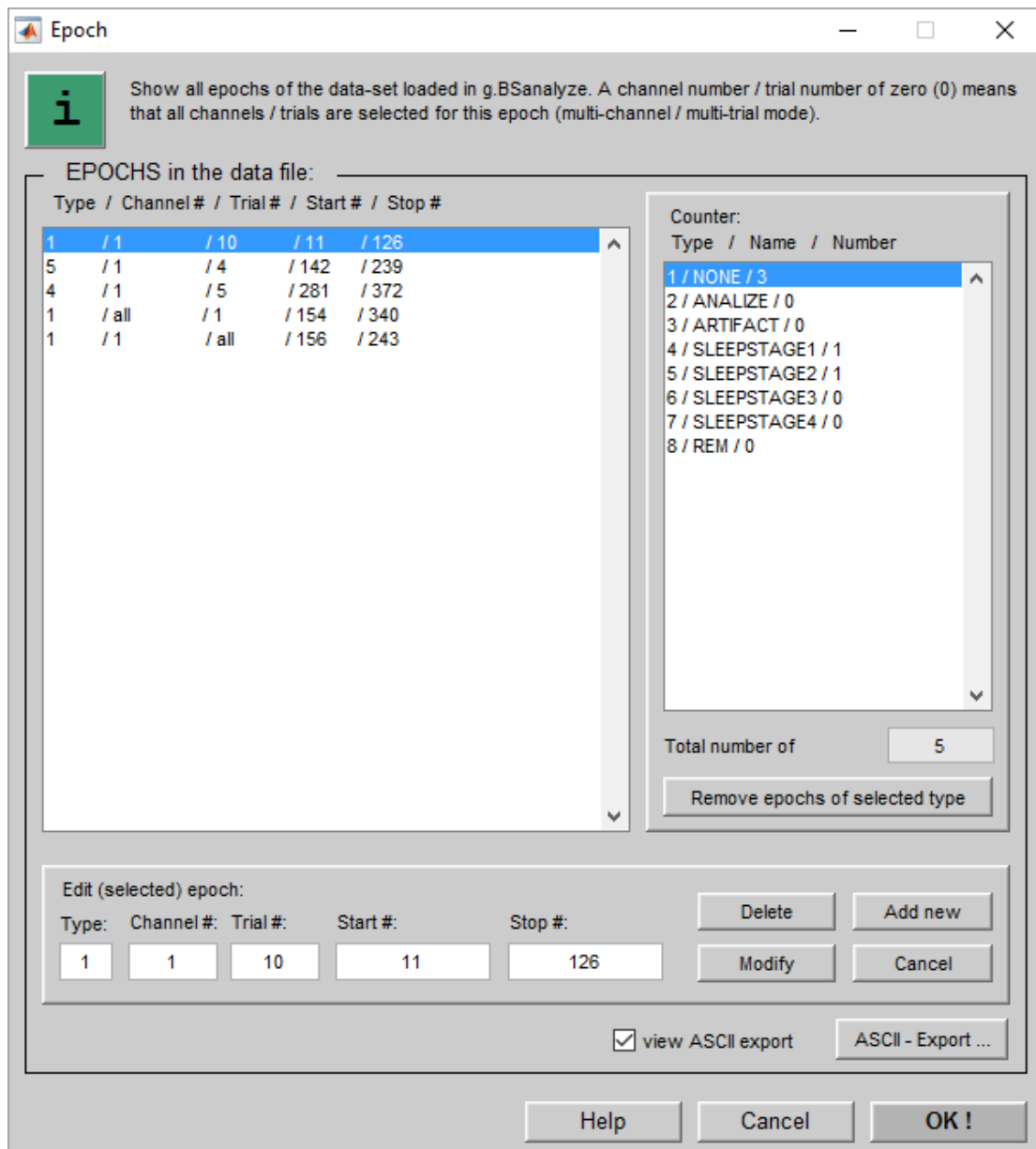


Note that it is not allowed to delete the BEGIN and END markers which show the begin and the end of the data-set.

Epoch

The **Epoch** dialog box allows to inspect and edit the scoring/epoching information of the loaded data-set. The epochs can be assigned in free-mode, multi-channel mode or multi-trial mode (see Section [Epoching, Tools and Comment](#)). The listbox **EPOCHS in the data file** shows all assigned epochs. If the **Channel #** is set to all the epoch was edited in multi-channel mode, if the **Trial #** is all then the epoch was assigned in multi-trial mode.

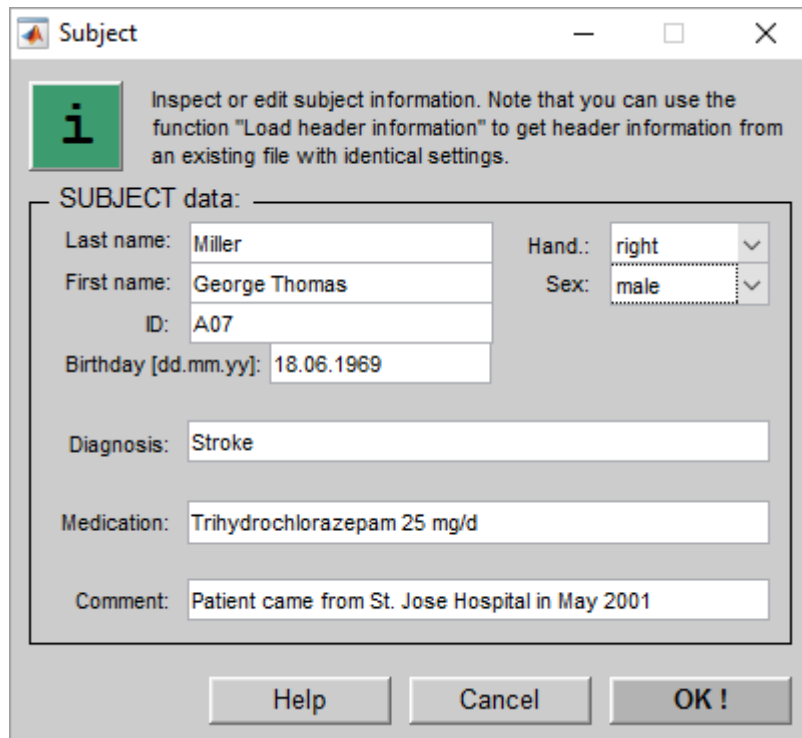
To delete or modify a specific entry click on the epoch in the listbox and change the values in the **Edit (selected) epoch** section. Select in the Counter listbox a specific name and click **Remove epochs of selected type** to delete all epochs with this name.




The **ASCII-Export** button allows to store the epoch list to harddisk. Check **view ASCII export** to open the MATLAB Editor with the epoch list.

Subject

The **Subject** dialog box includes information about the subject who participated in the study. Enter all important data which should be stored in the header of the data file.



Subject

 Inspect or edit subject information. Note that you can use the function "Load header information" to get header information from an existing file with identical settings.

SUBJECT data:

Last name:	Miller	Hand.:	right
First name:	George Thomas	Sex:	male
ID:	A07		
Birthday [dd.mm.yy]:	18.06.1969		
Diagnosis:	Stroke		
Medication:	Trihydrochlorazepam 25 mg/d		
Comment:	Patient came from St. Jose Hospital in May 2001		

Help Cancel OK !

Session/Paradigm

The **Session/Paradigm** dialog box includes information about the experimental session and the executed paradigm. Enter all important data which should be stored in the header of the data file.

Session/Paradigm

i Inspect or edit session and paradigm information. Note that you can use the function "Load header information" to get header information from an existing file with identical settings.

SESSION data:

Number:	3
Time [hh:mm]:	4:20 PM
Date [dd:mm:yy]:	19.12.2013
Studyname:	SEP medianus

Comment:
motor threshold: 2 mA

PARADIGM data:

Paradigm name:	SEP_01
Run number:	1
Condition:	eyes open, subthr.
ISI:	1.25 s

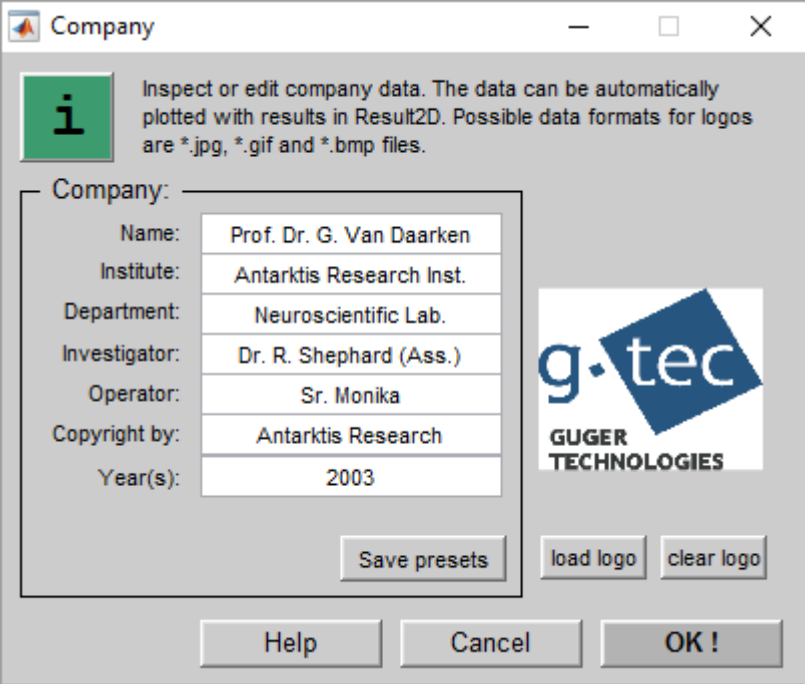
Comment:
right hand stimulation

Help Cancel OK!

Company

The **Company** dialog box includes information about your company and company members. Enter all important data which should be stored in the header of the data file you are going to analyse.

Load a logo (*.jpg, *.gif or *.bmp format) to plot it automatically together with your company data on calculation results.



The screenshot shows the 'Company' dialog box with the following content:

- Title Bar:** Company
- Information Icon:** A green square with a white 'i'.
- Text:** Inspect or edit company data. The data can be automatically plotted with results in Result2D. Possible data formats for logos are *.jpg, *.gif and *.bmp files.
- Company Form:**

Name:	Prof. Dr. G. Van Daarken
Institute:	Antarktis Research Inst.
Department:	Neuroscientific Lab.
Investigator:	Dr. R. Shephard (Ass.)
Operator:	Sr. Monika
Copyright by:	Antarktis Research
Year(s):	2003
- Logo Preview:** A blue square logo with the text 'g·tec' and 'GUGER TECHNOLOGIES' below it.
- Buttons:** 'Save presets', 'load logo', 'clear logo', 'Help', 'Cancel', 'OK!'.

Use the **Save presets** button to store this settings as default company settings to the Data Editor. The next time gBSanalyze is started and a data file with an empty **Company Name** is loaded the default settings are used.

View

The **View** menu allows to display and hide the following tools:

[Toolbar](#)

[Status](#)

[Player](#)

[Jumper](#)

[Scaling](#)

The **Toolbar** has the following commands:



File open - load data files using the **File Open** window

Save - save current data to file

Print - print current figure

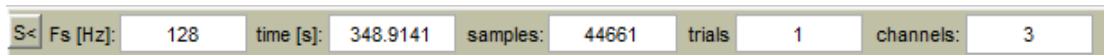
Zoom in - toggle button for zooming into the data using the mouse

Zoom out - toggle button for zooming out using the mouse

Rotate 3D - rotate current graph using the mouse

Undo - undo the last step

Status shows the following information:



Fs [Hz] - sampling frequency of the current data in Hertz

time [s] - recording time per trial in seconds

samples - data samples per trial and per channel

trials - number of trials

channels - number of channels

Player allows to view the data similar to a video. The following commands are supported:



Speed - define the speed of viewing the data-set. Enter 1 to update every second, 2 to update every 500 ms,...

Step [s] - define the step size in seconds

<< play backward as fast as possible

< play backward

> play forward

>> play forward as fast as possible

Jumper allows to jump directly to markers or attributes. Following commands are supported:



<< first jump to the first marker, attribute in the data-set with the specified name

< prev jump to the previous one

> next jump to the next one

last >> jump to the last one



select the marker or attribute name

Scaling

The **Scaling** window is used to set manually the y-axis scaling for each channel in the Data Editor. The settings will be activated when the window is closed or when the **Ma** (Manual) button under **PRESENT** in the Data Editor is pressed.

For the given number of channels the scale and unit can be chosen.

- **Scale** - set individual scaling for the specific channel
- **Unit** - select the correct unit for the specific channel

1. Click on **Scaling** under the **View** menu

Define the scaling for each channel. Change to the Manual scaling mode under PRESENT in the Data Editor to activate it.

AMPLIFIER: _____
Total number of channels:

SETTINGS:

Channel:	Scale:	Unit:	Channel:	Scale:	Unit:
1	200	μV			μV
2	200	μV			μV
3	200	μV			μV
4	200	μV			μV
5	400	μV			μV
6	400	μV			μV
7	2	mV			μV
8	5	V			μV
9	5	V			μV
		μV			μV
		μV			μV
		μV			μV
		μV			μV
		μV			μV
		μV			μV
		μV			μV
		μV			μV
		μV			μV

< Current 1 of 1 >

MASTER: _____
Start at channel: end at: Channel: Scale: Unit:

2. To define the scaling for the first 4 channels select **Start at channel** 1 and **end at** 4. After entering the **Scale** 200 and **Unit** μV press **Apply**.
3. Press the **OK** button to confirm the settings

Transform

There are two parts of transform operations:

The first part contains dialog windows to select or cut individual channels or trials for further computation:

[Cut Samples](#) – delete data segments of un-triggered and triggered data

[Cut Trials and Channels](#) – delete specific trials and channels

[Select Trials and Channels](#) – select trials and channels with specific attributes for further processing (window can be started e.g. from **Cut Trials and Channels**)

[Sort Trials and Channels](#) – sort trials and channels according to a specific criteria

The second part contains dialog windows with basic operations:

[Arithmetic](#) – perform basic arithmetic operations

[Merge](#) – combine multiple data-sets

[Trigger](#) – split recorded data into trials (epochs)

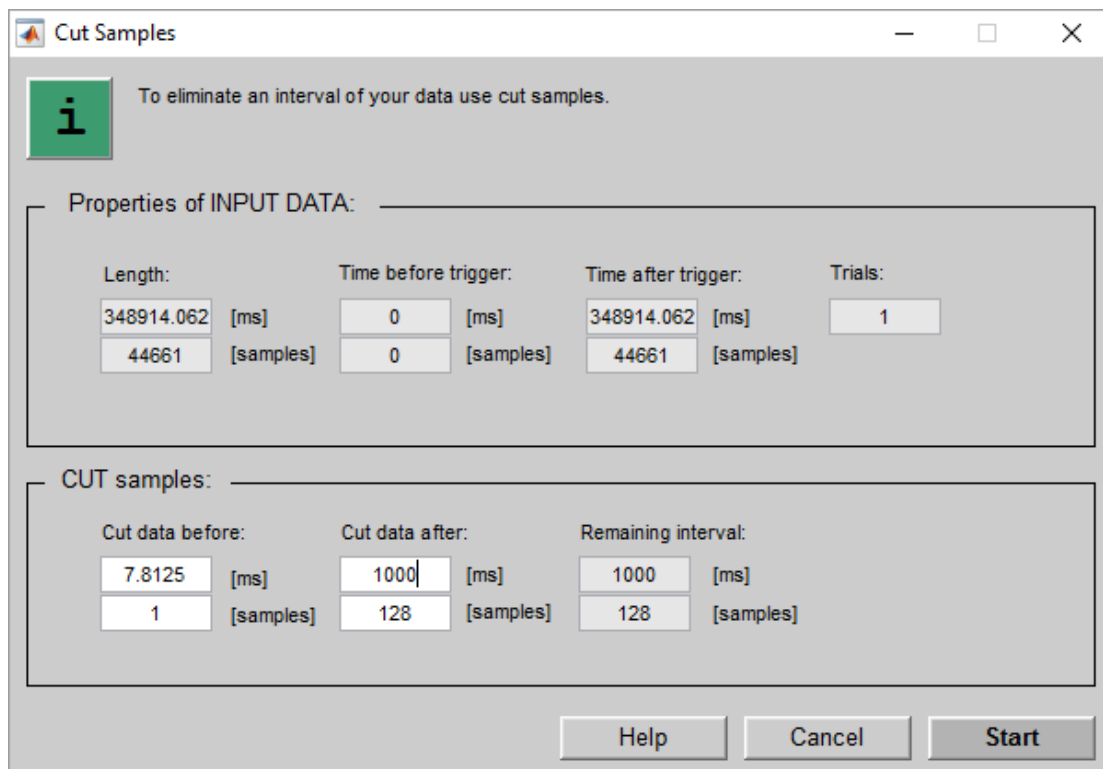
[Untrigger](#) – align trials (epochs) to raw-data format

Cut Samples

The **Cut Samples** window allows to drop data segments of un-triggered and triggered data-sets.

The **Properties of the INPUT DATA** section contains status information about the loaded data:

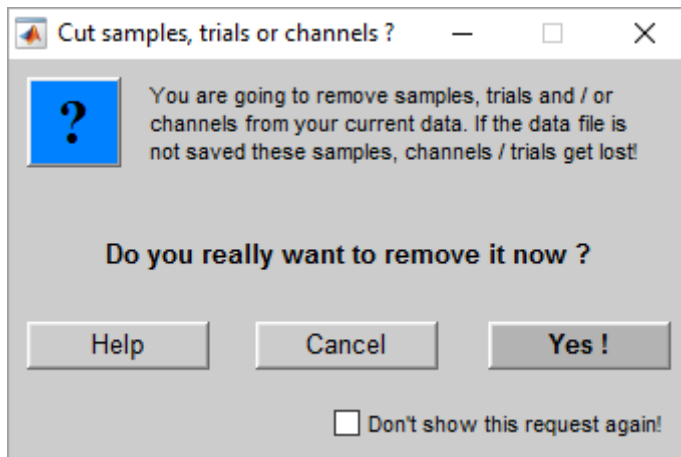
- **Length** - length of the data-set
- **Time before trigger** - time before trigger edge
- **Time after trigger** - time after trigger edge
- **Trials** - number of trials



To extract an interval of 1 second of the data follow these steps:

1. Load data-set `session1.mat` from
`Documents\gtec\gBSanalyze\testdata\Aircraft_Simulator`
2. Open **Cut Samples** from the **Transform** menu
3. Set **Cut data before** to sample 1
4. Enter 1000 ms into **Cut data after**
5. Press **Start** to extract a data segment of 1000 ms.
Note that the Markers of your data are deleted.

6. A confirmation dialog appears:



To confirm to cut samples, trials or channels press the **Yes** button or **Cancel** the operation.

Note that all dropped samples, trials and channels get lost!

Check the box **Don't show this request again** if the confirm dialog should not appear for future operations. To re-activate the confirm dialog go to **Appearance Settings** in the **Options** menu and check the corresponding box.

The following code shows how to perform the example demonstrated above from the MATLAB command line.

```
%Load File
P_C=data;
File=['C:\Users\' getenv('USERNAME')
'\Documents\gtec\gBSanalyze\testdata\Aircraft_Simulator\Session1.MAT'];
P_C=load(P_C,File);

%Cut Samples
SampleStart=1;           %interval-begin in samples
SampleEnd=128;          %interval-end in samples
P_C=gBScutsamples(P_C,SampleStart,SampleEnd);
```

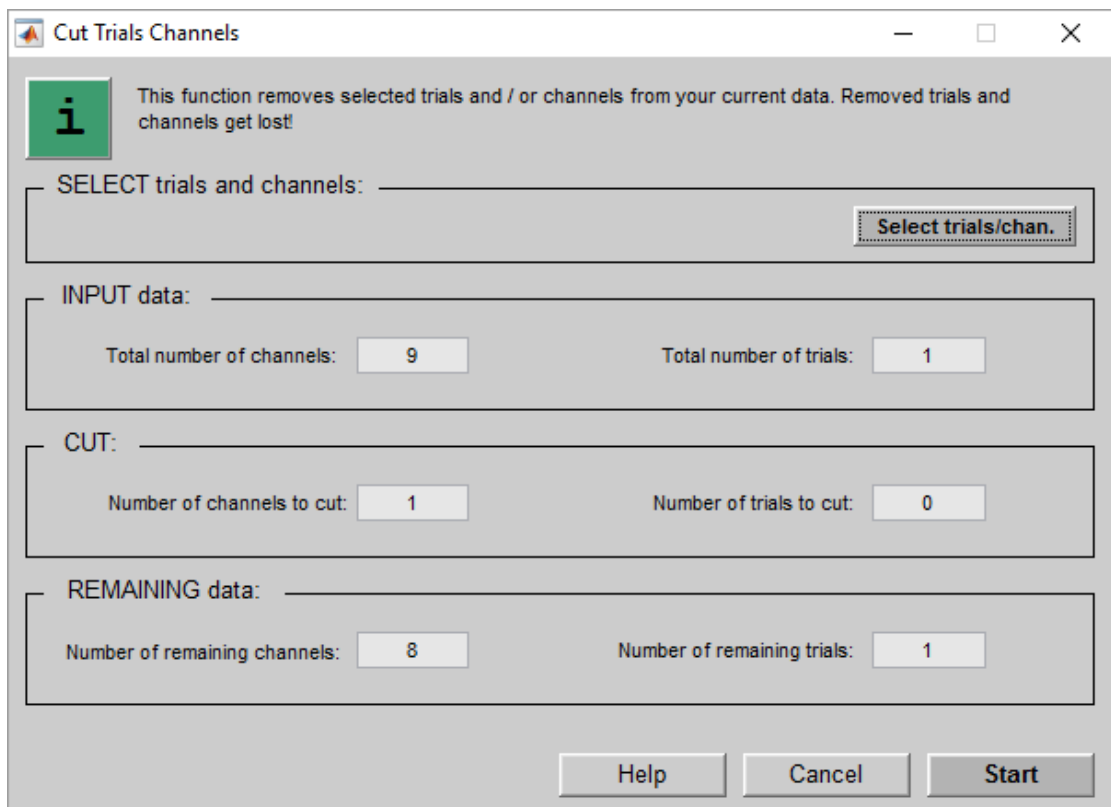
Cut Trials and Channels

allows to drop trials and channels with specific attributes (e.g. ARTIFACT), channel-types (e.g. EEG, EMG,...) or channel numbers.

1. Load data-set `session1.mat` from

`Documents\gtec\gBSanalyze\testdata\Aircraft_Simulator`

2. Open **Cut Trials Channels** from the **Transform** menu and click **Select trials/chan.** to select the trials and channels to drop (see Chapter Select Trials and Channels)



3. **INPUT data** show the existing number of channels and trials. **CUT** specifies how many channels and trials are marked for cutting and **REMAINING data** gives a preview of how many channels and trials are kept for further processing.
4. Confirm the selection with the **Start** button.
Note that cut channels and trials get lost!

The following code shows how to perform the example demonstrated above from MATLAB command line.

```
%Load File
P_C=data;
File= ['C:\Users\' getenv('USERNAME')
'\Documents\gtec\gBSanalyze\testdata\Aircraft_Simulator\Session1.MAT'];
P_C=load(P_C,File);

%Select Trials and Channels
trial_id=[];
channel_id=[1];           %select all channels with attribute BAD
type_id=[];
channelnr_id=[];
flag_tr='tr_exc';
flag_ch='ch_exc';        %exclude the selected channels
flag_type='type_exc';
flag_nr='nr_exc';
[TrialExclude,ChannelExclude]=gBSselect(P_C,trial_id,flag_tr,...
channel_id,flag_ch,type_id,flag_type,channelnr_id,flag_nr);

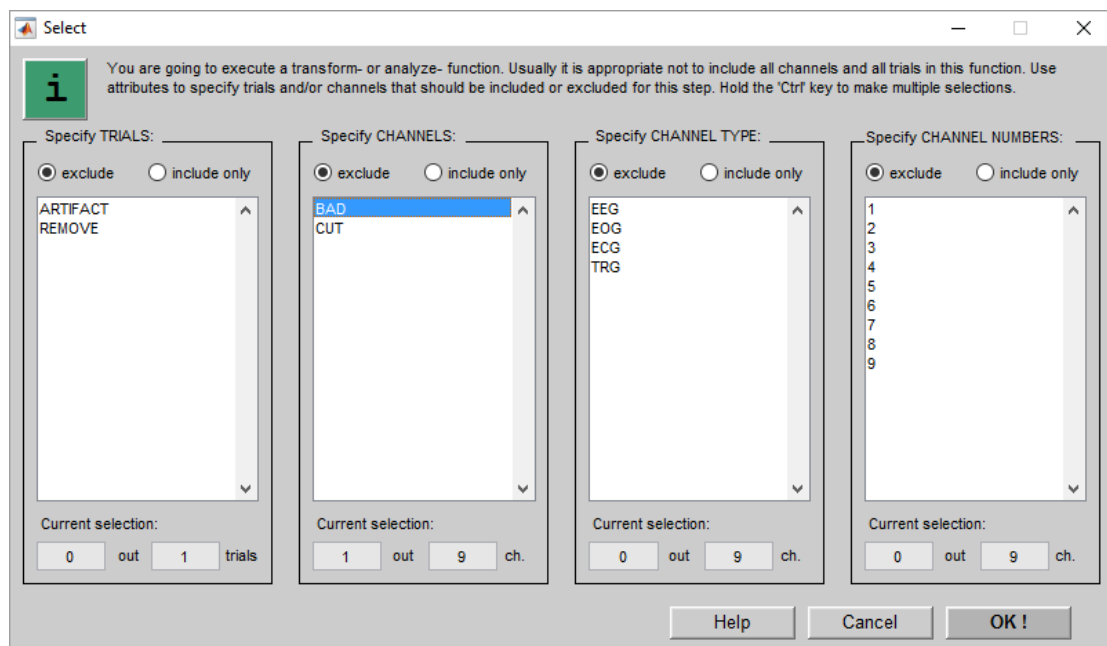
%Cut Trials and Channels
TrialExclude=[];        %do not exclude any trial
ChannelExclude=[9];     %exclude channel 9
P_C=gBScuttrialschannels(P_C,TrialExclude,ChannelExclude);
```

Select Trials and Channels

allows to mark trials and channels with specific attributes (e.g. ARTIFACT) or channel-type (e.g. ECG) for further processing. Not selected trials/channels will not be considered for further operations. The window can be called from several operations (e.g. **Cut Trials Channels**).

Perform the following steps:

1. Open e.g. **Cut Trials Channels** from the **Transform** menu and click **Select trials/chan.** to open the following window:



2. Select to exclude/include trials with a specific attribute or channels with a specific attribute/channel-type (e.g. EEG, EOG,...).
Click on radiobutton **exclude** channels and mark the **BAD** attribute. The **Current selection** gives a preview of how many channels are selected. Use the STRG key to select multiple attributes.
3. Confirm the selection with the **OK** button

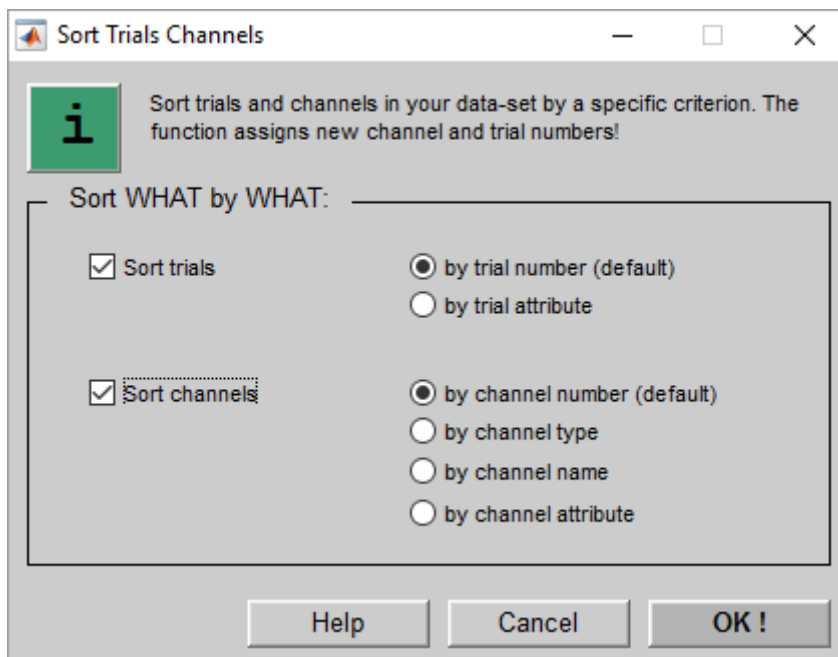
An example for batch processing is given in Section **Cut Trials and Channels**.

Sort Trials and Channels

1. Open **Sort Trials Channels** from the **Transform** menu

Trials and channels can be sorted by the following criteria:

- **by trial number (default)**
- **by trial attribute** - according to the **Edit markers and attributes** window
- **by channel number (default)**
- **by channel type** - according to the channel type in **Channel Configuration**
- **by channel name** - in alphabetical order
- **by channel attribute** - according to the **Edit markers and attributes** window



2. Check **Sort trials by trial number** and **sort channels by channel number**
3. Click **OK**

Note that **Sort Trials / Channels** assigns new trial and channel numbers.

The following code shows how to perform the example demonstrated above from MATLAB command line.

```
%Sort Trials Channels  
Type={  
'TrialNr'           %sort according to Trial Number  
'ChNr'};           %sort according to Channel Number  
P_C=gBSsorttrialschannels(P_C,Type);
```

Arithmetic

Arithmetic performs basic arithmetic operations on selected channels and trials.

Apply on multiple channels

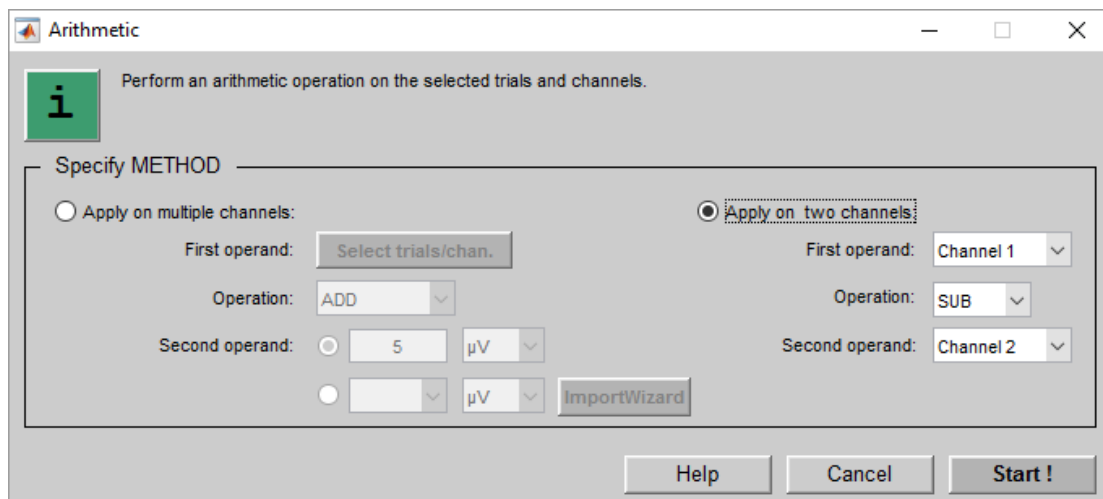
The operation is carried out between channels and a constant.
Selectable operations are:

- 'ADD' for addition,
- 'SUB' for subtraction,
- 'MULT' for multiplication,
- 'DIV' for division,
- 'Z-MEAN' for base line correction, subtracts from each sample the mean of the selected channels at sample N
- 'Z-MEDIAN' for base line correction, subtracts from each sample the median of the selected channels at sample N
- 'NORM' for normalization, divides each sample N by the mean of the selected channels at sample N
- 'SQR' for power of two,
- 'SQRT' for square root,
- 'LOG' for natural logarithm and
- 'LOG10' for logarithm with base 10.

The **Second operand** can be a scalar value or a vector imported from a file or from the clipboard. The vector length must be equal to the number of channels of the loaded data-set.

Apply on two channels

The operation is carried out between two different channels.



Selectable operations are:

- 'ADD' for addition,
- 'SUB' for subtraction and
- 'MULT' for multiplication.

1. Load `session1.mat` from

`Documents\gtec\gBSanalyze\testdata\aircraft_simulator`

2. Open **Arithmetic** from the **Transform** menu to calculate a bipolar derivation of channel 1 and channel 2.

3. Select **Apply on two channels** and select the operation `Channel 1 - Channel 2`.

4. Click **Start**

Note: The result is stored in the first operand. It is not possible to choose the same channel as first and second operand.

The following code shows how to perform the example demonstrated above from MATLAB command line.

```
%Load demo file
P_C=data;
File= ['C:\Users\' getenv('USERNAME')
'\Documents\gtec\gBSanalyze\testdata\Aircraft_Simulator\Session1.MAT'];
P_C=load(P_C,File);

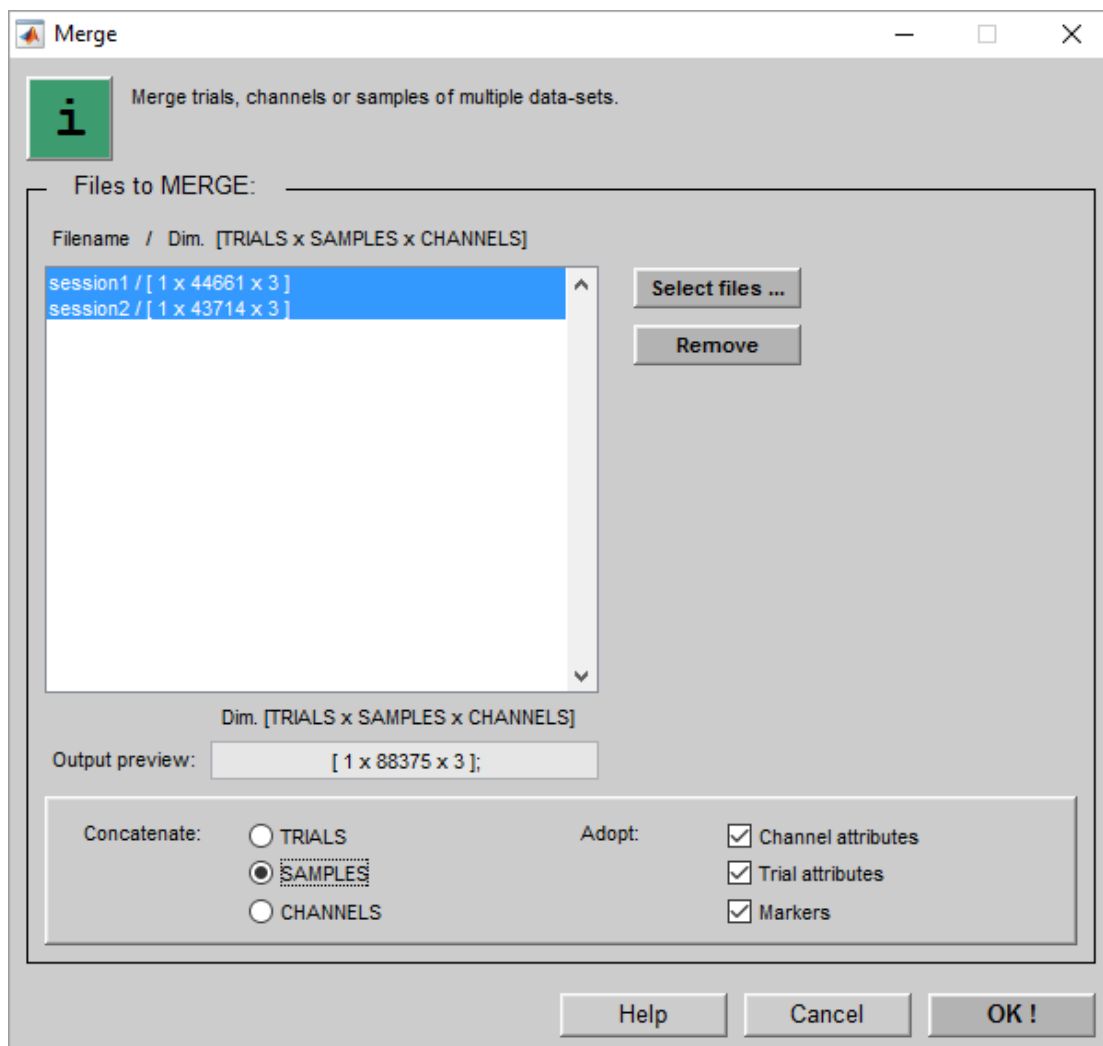
% Transform
ApplyOn = 'two channels';
ChannelExclude_mult = [];
TrialExclude_mult = [];
Operation_mult = 'ADD';
SecondOperand_mult(1) = 5;
Unit_mult = 'µV';
FirstOperand_two = 1;
Operation_two = 'SUB';
SecondOperand_two = [2];
ProgressBarFlag = 0;
P_C = gBSarithmetic(P_C, ApplyOn, ChannelExclude_mult,...
    TrialExclude_mult, Operation_mult, SecondOperand_mult,...
    Unit_mult, FirstOperand_two, Operation_two,...
    SecondOperand_two, ProgressBarFlag);
```

Merge

The **Merge** window allows to link MATLAB (*.mat) files together. The input files must have the same sampling frequency.

Perform the following steps:

1. Assume that `session1 [1 x 44661 x 3]` is loaded in g.BSanalyze
2. Click on **Merge** under the **Transform** menu



3. **Select files...** allows to browse for *.mat files that should be linked to `session1`.
Select `session2` from
`Documents\gtec\gBSanalyze\testdata\bci`
4. Click on **SAMPLES** to concatenate the samples of `session1` and `session2`
5. **Output preview** shows the expected data-set size

6. Check **Channel attributes**, **Trial attributes** and **Markers** to assign also these information to the new data-set
7. Perform the operation by pressing the **OK** button

The following code shows how to perform the example demonstrated above from MATLAB command line.

```
%Load session1.mat
P_C=data;
File=['C:\Users\' getenv('USERNAME')
'\Documents\gtec\gBSanalyze\testdata\BCI\session1.mat'];
P_C=load(P_C,File);
P_C.SamplingFrequency=128;    %assign the sampling frequency

%Merge session2.mat
FileName={'C:\Users\' getenv('USERNAME')
'\Documents\gtec\gBSanalyze\testdata\BCI\session2.mat'}];
Concatenate='Samples';      %merge over samples
AdoptChAttr=1;              %merge channel attributes
AdoptTrialAttr=1;          %merge trial attributes
AdoptMarkers=1;            %merge markers
P_C=gBSmerge(P_C,FileName,Concatenate,AdoptChAttr,AdoptTrialAttr,...
AdoptMarkers);
```

Trigger

The **Trigger** window is used to create trials (epochs) with a specific length of your data-set. The function is searching for specific events in the data (e.g. TTL impulses) which are used as synchronization time-points.

The window allows the following settings:

- **Time before trigger** - included time period before trigger time-point
- **Time after trigger** - included time period after trigger time-point
- **Accept incomplete last trials** - accept also incomplete last trials (if not enough data is available after the last trigger event). If the box is not checked the trial is rejected.

The trigger function splits your recorded data into trials related to trigger timepoints defined by physical channels or markers. The use of different markers or channels allows you to assign attributes automatically to resulting trials. Channel attributes and markers remain in the triggered data.

Specify TRIAL PARAMETERS:

Time before trigger: 2000 [ms] 256 [samples] Time after trigger: 4000 [ms] 512 [samples] Accept incomplete last trial

Specify TRIGGERS and ATTRIBUTES:

Physical channel: 1(CH1) 2(CH2) 3(CH3) Edge: rising Threshold voltage: 43.3 [µV] Threshold level: 90 [% of max.] Slew rate: 5.69 [µV/ms]

Marker: BEGIN

Assign attribute to resulting trials:

Accept overlap

Change color: red blue green yellow pink orange purple olive brown grey

Chan.	(Marker)	Name	Edge	Value	Attribute	Overlap	Color
Ch.3/	3/	rise/	90%/	-/	no/	red	

Generate LINED UP TRIALS:

Line up trials (no trigger) Length of trials: 1000 [ms] 128 [samples] Overlap: 0 [ms] 0 [samples]

Select CHANNELS for the triggered file:

- **Physical channel** - select the physical trigger channel (can be e.g. a TTL trigger pulse)
- **Edge** - choose to trigger on a falling or rising trigger edge
- **Threshold voltage [µV]** - specify voltage level in µV for triggering

- **Threshold level [%]** - specify voltage level in % of maximum
- **Slew rate [$\mu\text{V}/\text{ms}$]** - specify slew rate for triggering
- **Marker** - select an available marker as trigger (e.g. eye blink marker)
- **Assign attribute to resulting trials** - assign a new attribute to the new created trials
- **Accept overlap** - enable to consider also overlapped trials (trigger events are closer together as selected trial length)
- **Line up trials** - select this for splitting data into trials with specified length and with specified overlap without searching for trigger events

Note: The first trial is rejected if it is not long enough

Perform the following steps to trigger a data-set on an external TTL-trigger which was acquired simultaneously with the biosignals:

1. Load `session1` into `g.BSanalyze` from the following path

`Documents\gtec\gBSanalyze\testdata\bci`

The data-set consists of 2 EEG channels and 1 TTL-trigger channel

2. Click on **Trigger** under the **Transform** menu
3. Select channel 3 as **Physical channel** for triggering
4. Set **Threshold level** to 90 % of max and add this new trigger criterion to the trigger list with the **add to list** button
5. Click **Start** to perform the triggering. The trigger function is now searching on channel 3 for pulses with exceed the threshold of 90 % of the maximum value of the channel.
6. Open the **Status** information in the Data Editor to find size information of the triggered data-set. The data consists now of 40 trials with a length of 768 samples (6 seconds) and 3 channels.

S<	Fs [Hz]:	128	time [s]:	6	samples:	768	trials	40	channels:	3
----	----------	-----	-----------	---	----------	-----	--------	----	-----------	---

The following code shows how to perform the example demonstrated above from MATLAB command line.

%Load File

```
P_C=data;
File=['C:\Users\' getenv('USERNAME')
'\Documents\gtec\gBSanalyze\testdata\BCI\session1.mat'];
P_C=load(P_C,File);
P_C.SamplingFrequency=128;
```

%Trigger

```
New_tm{1}={3 1 '1' 90 0};    %search for events which exceed 90 %
SamplesBefore=256;
SamplesAfter=512;
Uncomplete=0;
ChannelExclude=[];
P_C=gBStrigger(P_C,New_tm,SamplesBefore,SamplesAfter,Uncomplete,...
ChannelExclude);
```

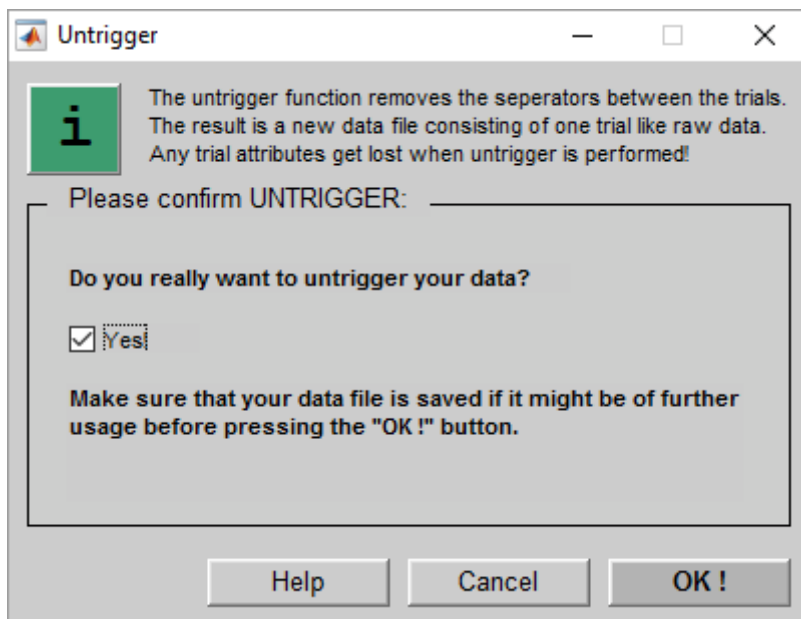
Untrigger

The **Untrigger** window is used to align trials (epochs) to raw-data format.

Consider a data-set with the following dimension:

S<	F _s [Hz]:	128	time [s]:	6	samples:	768	trials:	40	channels:	3
----	----------------------	-----	-----------	---	----------	-----	---------	----	-----------	---

1. Click on **Untrigger** under the **Transform** menu



2. Confirm the untrigger procedure by checking **Yes** and press **OK**. The function creates a data-set with the following dimension:

S<	F _s [Hz]:	128	time [s]:	240	samples:	30720	trials:	1	channels:	3
----	----------------------	-----	-----------	-----	----------	-------	---------	---	-----------	---

Note that trial attributes are deleted.

The following code shows how to perform the example demonstrated above from MATLAB command line.

```
%Untrigger
ProgressBarFlag=0;
P_C=gBSuntrigger(P_C,ProgressBarFlag);
```

Pre-Processing

The **Pre-Processing** menu contains functions to perform

[Source Derivation](#) - compute common average reference, local average reference, Laplacian, weighted average reference and bipolar derivations

[DC Correction](#) - eliminate DC offsets of the data

[Baseline Correction](#) – subtracts the mean value of a baseline interval

[Rectify and Smooth](#) - rectify the data and perform smoothing

[Moving Window Filter](#) – smoothes the data with a moving average window of a specific length

[Remove Drift](#) - remove electrode and amplifier drifts with a moving average window

[Detrend](#) - correct linear trends

[Down- and Upsampling](#) - resample the data

[Spatial Filter](#) - perform ICA-, PCA- and CSP-transformations

[Filter](#) - apply predefined filters (lowpass, highpass, bandpass and bandstop) to the data

Source Derivation

Method

On the scalp no position with zero potential can be found which would be perfectly suitable as reference. For this reason several methods are available which try to eliminate the unknown potential of the reference from the EEG derivation. Often the EEG derivation is performed as monopolar recording, which uses e.g. the right ear as reference, and afterwards the reference-independent derivation is calculated.

Common Average Reference (CAR) is computed according to

$$V_{CAR(i)} = V_{(i)} - \frac{1}{N} \sum_{j=1}^N V_{(j)}$$

where N is the total number of electrodes used for the recording and $V_{(i)}$ is the potential between the active electrode i and the reference. The CAR reference subtracts the mean of all electrodes (including the active electrode) from the active electrode and gives a reference independent recording.

Small or Large Laplacian Reference uses nearest-neighbor or nest-nearest neighbor

$$V_{LAP(i)} = V_{(i)} - \sum_{j \in S(i)} V_{(j)}$$

where $S_{(i)}$ is the subset of electrodes surrounding the electrode i (north, east, south, west). Note that the electrodes are not weighted according to their distance.

Local Average Reference (LAR) takes the average of the surrounding electrodes and subtracts it from the active electrode. The electrodes are weighted according to the distance from the active electrode.

The LAR is calculated according to the following equations

$$V_{LAR(i)} = V_{(i)} - \sum_{j \in S(i)} g_{(i,j)} V_{(j)}$$

where i is the channel number and $S_{(i)}$ the subset of surrounding electrodes for channel i.

The weighting of each electrode is calculated according to

$$\sum_{j \in S(i)} g_{(i,j)} = 1$$

and

$$g_{(i,j)} = \frac{1}{d_{(i,j)}} \left(\sum_{j \in S(i)} \frac{1}{d_{(i,j)}} \right)^{-1} \text{ with } d_{(i,j)} \text{ being the distance between channel } i \text{ and } j.$$

Weighted Average Reference (WAR) is calculated like LAR, but all N electrodes are included (except the active electrode) and weighted according to the distance.

$$V_{WAR(i)} = V_{(i)} - \sum_{\substack{j=1 \\ j \neq i}}^N g_{(i,j)} V_{(j)}$$

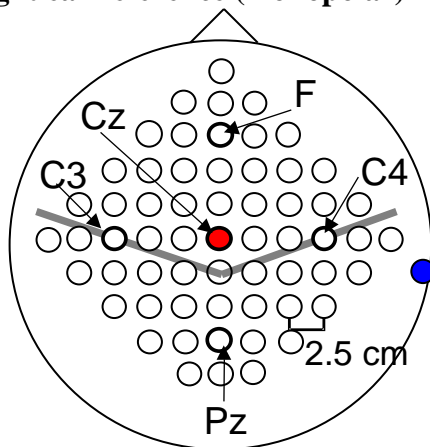
with

$$g_{(i,j)} = \frac{1}{d_{(i,j)}} \left(\sum_{\substack{j=1 \\ j \neq i}}^N \frac{1}{d_{(i,j)}} \right)^{-1}$$

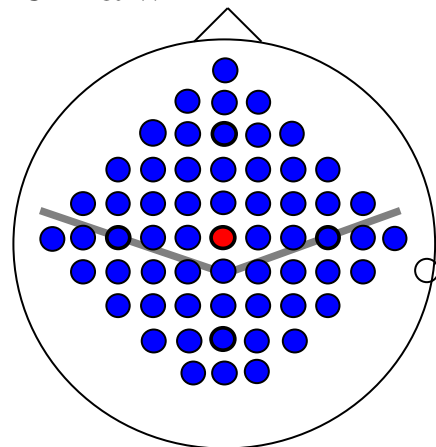
A **Bipolar Reference** is calculated by subtracting the secondary electrode j from the active electrode i.

$$V_{bipolar(i)} = V_{(i)} - V_{(j)}$$

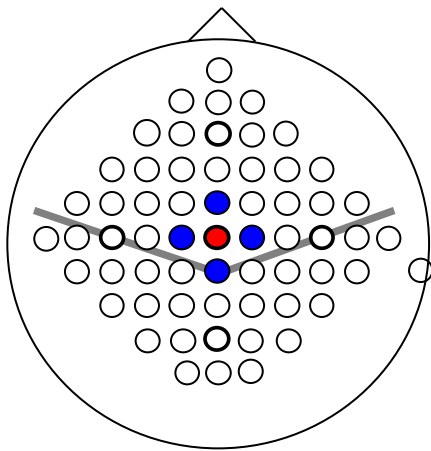
Right ear reference (monopolar)



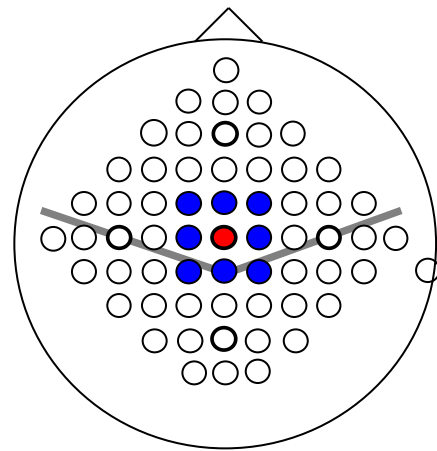
CAR & WAR



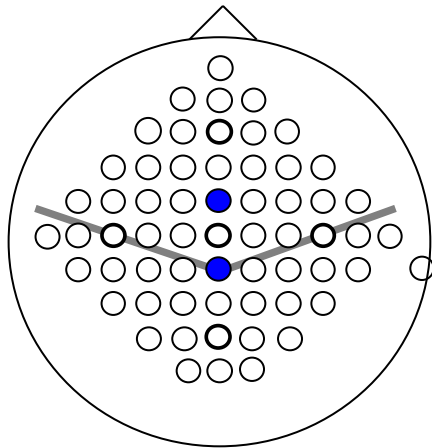
Laplacian



LAR



Bipolar



For the calculation of CAR, Laplacian and bipolar derivations no electrode positions are required. The electrode positions necessary for the calculation of LAR and WAR are imported from the montage file created with the Montage Creator g.MONcreator.

Perform the following steps to calculate a CAR derivation:

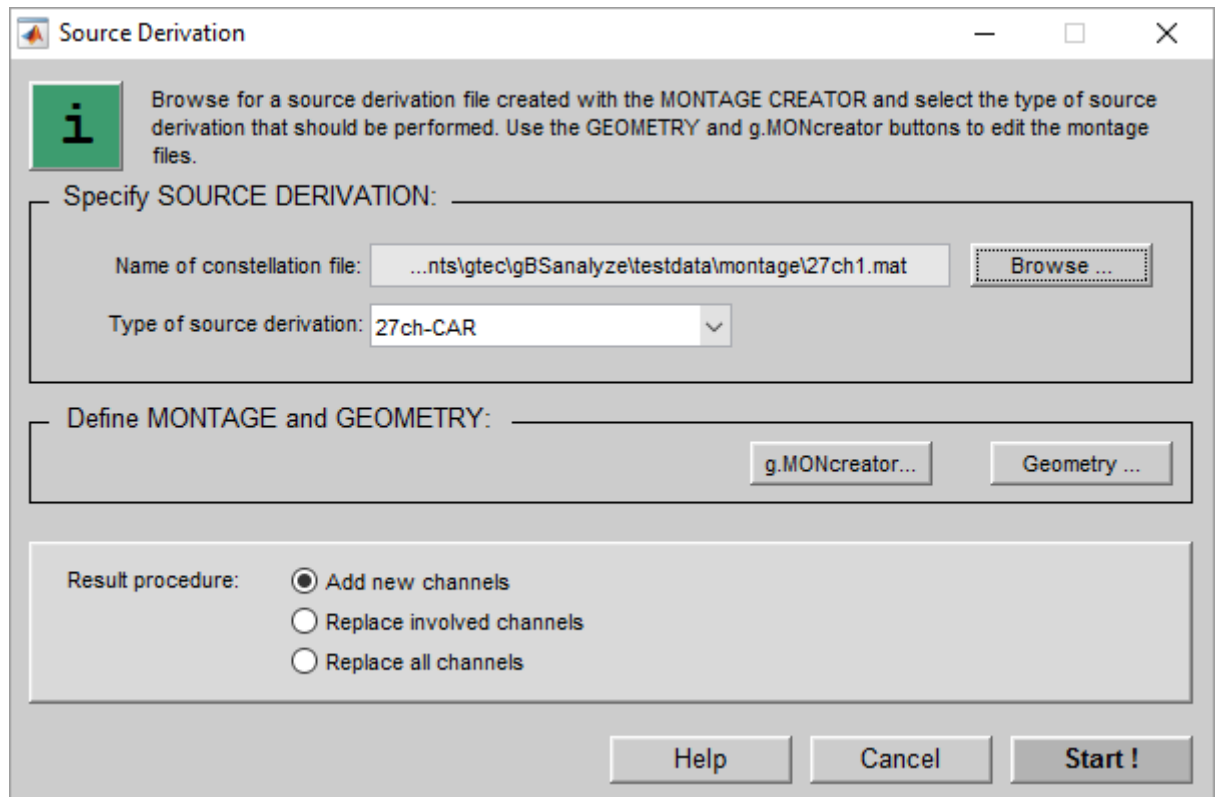
1. Import the monopolar recording data file `bkr_mono.bkr` with 27 channels from the following directory

```
Documents\gtec\gBSanalyze\testdata\bkr
```

2. Open the **Source Derivation** window under the **Pre-Processing** menu entry
3. Select the constellation file `27ch1.mat` created with **g.MONcreator** by pressing the **Browse** button and searching in the following directory:

```
Documents\gtec\gBSanalyze\testdata\montage
```

- To create a new constellation or inspect a constellation file press the **g.MONcreator** button. Click **Geometry** to edit the electrode positions of your montage.



- Select the **Type of source derivation** from the pull-down menu. To perform a CAR - calculation select the pre-defined 27ch-CAR constellation.
- Select **Add new channels** to append the CAR channels to your data-set
- Click **Start** to perform the operation. The common average reference channels are appended to the data and the channel-attribute CAR is assigned to the new channels. This allows to compare the monopolar data to the new CAR derivations in order to investigate the effects of the re-referencing.

The following code show how to perform the example demonstrated above from the MATLAB command line.

```
%Import File
P_C=data;
File=['C:\Users\' getenv('USERNAME')
'\Documents\gtec\gBSanalyze\testdata\BKR\bkr_mono.bkr'];
P_C=import(P_C, 'BKR', File);

%Load Montage
Mon=montage;
Mon=load(Mon, ['C:\Users\' getenv('USERNAME')
'\Documents\gtec\gBSanalyze\testdata\montage\27ch1.mat']);
```

%Source Derivation

```
Cst_nr=1;  
Replace='add channels';  
P_C=gBSsourcederivation(P_C,Mon,Cst_nr,Replace);
```

DC Correction

DC-correction subtracts DC offsets of selected trials and channels by selecting

Subtract MEAN – correct the mean value

Subtract MEDIAN – correct the median value

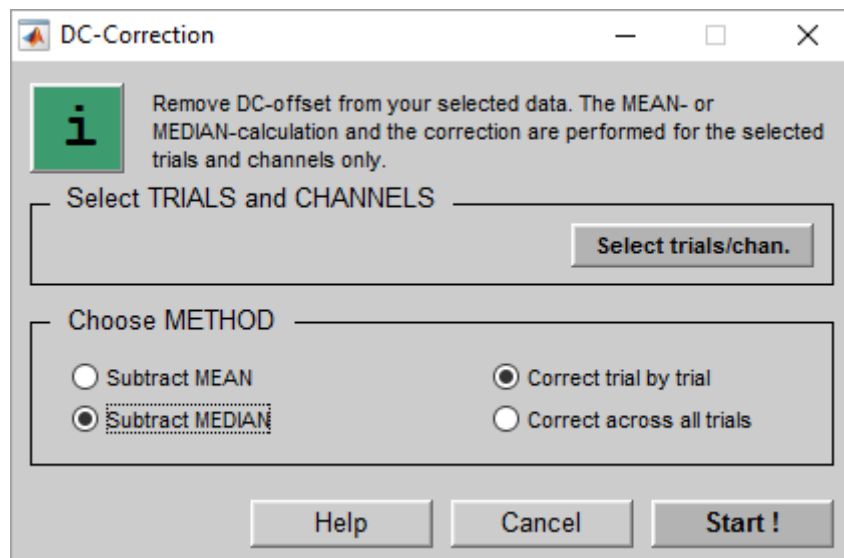
Correct trial by trial – the mean or median value is calculated from a single trial and the same trial is corrected with this value

Correct across all trials - all trials of one channel are combine for the computation of the mean or median value. The obtained result is subtracted across all trials of the current channel.

1. Load `trig1.mat` from

Documents\gtec\gBSanalyze\testdata\BCI

2. Open **DC-Correction** from the **Pre-Processing** menu



3. Select **Subtract MEDIAN** and **Correct trial by trial**

4. Click **Start** to eliminate the offset

The following code shows how to perform the example demonstrated above from the MATLAB command line.

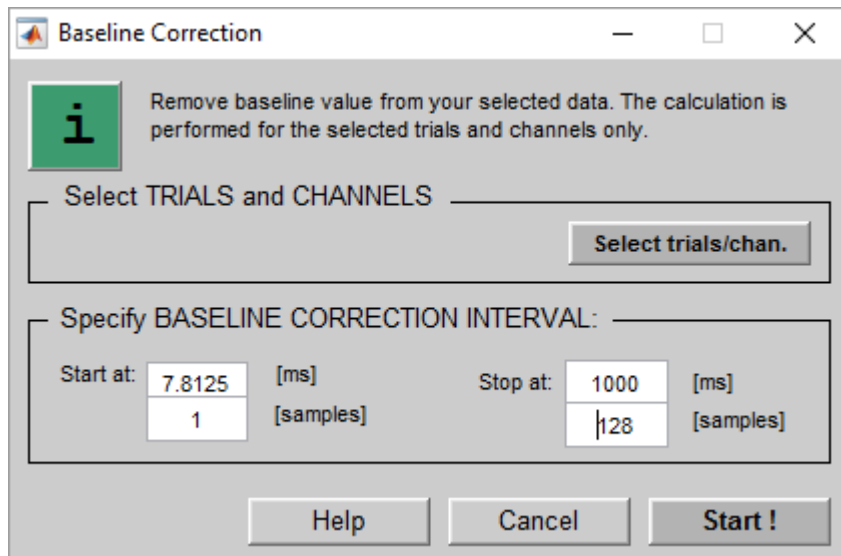
```
%Load File
P_C=data;
File=['C:\Users\' getenv('USERNAME')
'\Documents\gtec\gBSanalyze\testdata\BCI\trig1.mat'];
P_C=load(P_C,File);

%DC-Correction
TrialExclude = [];
ChannelExclude = [];
Method = 'median';
Parameter = 'singleTrial';
ProgressBarFlag = 0;
[P_C, DC_Cell] = gBSdccorrection(P_C, TrialExclude, ChannelExclude,...
                                Method, Parameter, ProgressBarFlag);
```

Baseline Correction

Baseline Correction subtracts the mean value of the baseline interval of the selected trials and channels. The baseline interval is defined with **Start** and **Stop** values.

1. Load data-set `trig1.mat` under
`Documents\gtec\gBSanalyze\testdata\bci`
2. Open **Baseline Correction** from the **Pre-Processing** menu



3. Set **Start at:** 1 samples, and **Stop at:** to 1000 ms.
4. Click **Start**.

The following code shows how to perform the example demonstrated above from the MATLAB command line.

```
%Load File
P_C=data;
File= ['C:\Users\' getenv('USERNAME')
'\Documents\gtec\gBSanalyze\testdata\BCI\trig1.mat'];
P_C=load(P_C,File);

% Baseline Correction
Interval = [1 256];
ChannelExclude = [];
TrialExclude = [];
ProgressBarFlag = 0;
P_C = gBSbaselinecorrection(P_C, Interval, ChannelExclude,...
    TrialExclude, ProgressBarFlag);
```

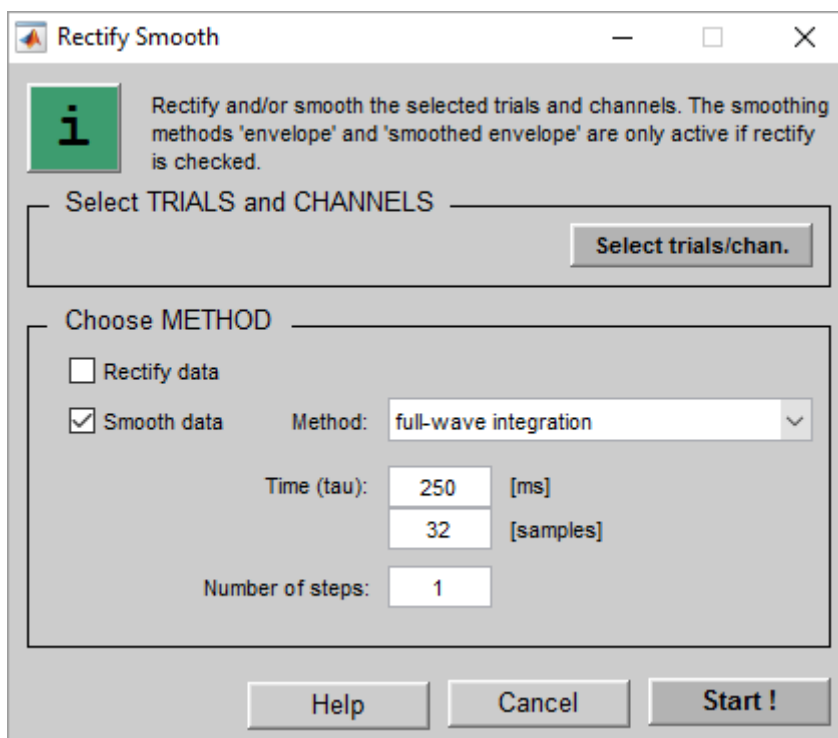

Rectify and Smooth

The window allows rectifying and smoothing the data by selecting

Rectify data – check to take the absolute value of each sample

Smooth data – check to select from a list of smoothing algorithms

- **Full-wave integration** - an integration over the specified time window is performed. The result is assigned to the element in the middle of the integration window.
- **Hanning smoothing** - the selected record is convoluted with a Hanning window. The result is assigned to the element in the middle of the integration window.
- **Exponential window** - the selected trials are convoluted with an exponential window. The result is assigned to the element at the end of the integration window.
- **Envelope** - computes the Hilbert transform of the input sequence
- **Smoothed envelope** - computes the Hilbert transform and performs a full-wave integration



Time (tau) – specifies the window length

Number of steps – smoothing repetitions

Perform the following steps:

1. Click on **Rectify Smooth** under the **Pre-Processing** menu
2. Check **Rectify data** and **Smooth data**
3. Select the smoothing **Method** from the pull-down menu, the **Time** window and the **Number of steps**
4. Click **Start**

The following code shows how to perform the example demonstrated above from the MATLAB command line.

```
%Load File
P_C=data;
File= ['C:\Users\' getenv('USERNAME')
'\Documents\gtec\gBSanalyze\testdata\BCI\trig1.mat'];
P_C=load(P_C,File);

%Rectify Smooth
TrialExclude = [];
ChannelExclude = [];
Method = 'rectify&smooth';
ListItem = 'full-wave integration';
IntervalLength = 32;
NumSteps = 1;
ProgressBarFlag = 0;
P_C = gBSrectifysmooth(P_C, TrialExclude, ChannelExclude, Method,...
                        ListItem, IntervalLength, NumSteps,...
                        ProgressBarFlag);
```

Moving Window Filter

This function smoothes the data of the selected channels by applying one of the following two backwards oriented moving window filters onto the selected data.

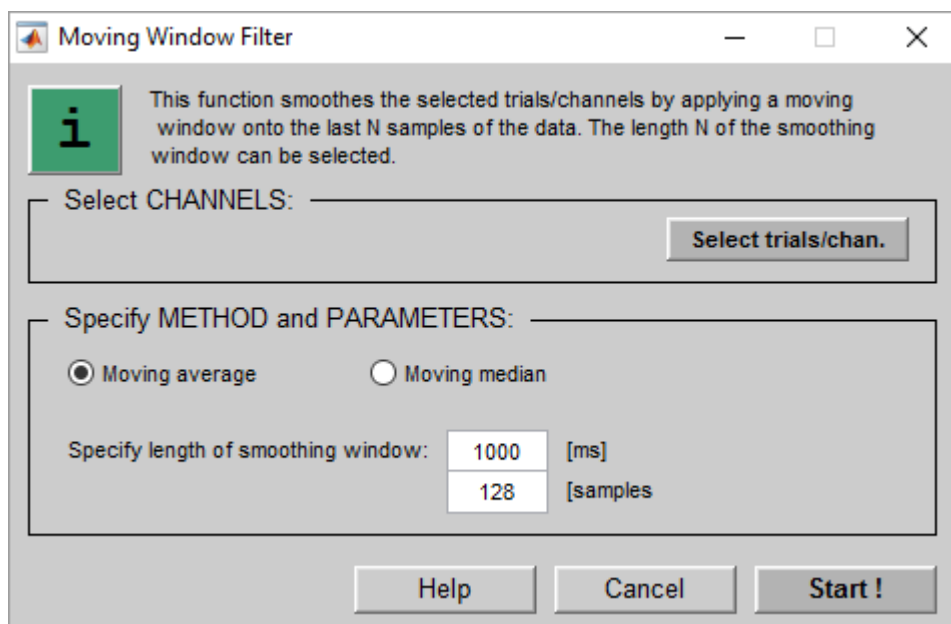
- **Moving average** - replaces each sample by the average of the past L samples including the one to be replaced using the following difference equation:

$$y(n) = \frac{1}{L}x(n) + \frac{1}{L}x(n-1) + \dots + \frac{1}{L}x(n-L+1)$$

- **Moving median** -. replaces each sample by the median value of the past L samples including the one to be replaced using the medfilt1 function of MATLAB.

The window allows the following inputs:

- **Select trials / chan.** - apply the filter only to specific data channels and trials
- **Moving average** - calculate the moving average window with a window size specified by the **Specify length of smoothing window** editor box
- **Moving median** - calculate the moving median window with a window size specified by the **Specify length of smoothing window** editor box
- **Specify length of smoothing window** - specifies the backwards oriented window length of the selected filter.



Example:

Perform the following steps to apply the moving window filter:

1. Load data-set `trig1.mat` under

`Documents\gttec\gBSanalyze\testdata\bci`

2. Click on **Moving Window Filter** under the **Pre-Processing** menu
3. Select **Moving average**
4. **Specify length of smoothing window** as 1000 **ms**. This window is moved through the data-set. The result is assigned to the last element of it.
The window is backwards oriented in case of Moving mean and centred at the current sample for Moving median.
5. Click on **Select trials / chan** to exclude channel 3 and confirm the settings with the **OK** button.
6. Press **Start** to apply the windowing function.

The following code shows how to perform the example demonstrated above from the MATLAB command line.

```
%Load Data
P_C=data;
File=['C:\Users\' getenv('USERNAME')
'\Documents\gtec\gBSanalyze\testdata\BCI\trig1.mat'];
P_C=load(P_C,File);

%Select Trials and Channels
trial_id=[];
channel_id=[];
type_id=[];
channelnr_id=[3];
flag_tr='tr_exc';
flag_ch='ch_exc';
flag_type='type_exc';
flag_nr='nr_exc';
[TrialExclude,
ChannelExclude]=gBSselect(P_C,trial_id,flag_tr,channel_id,flag_ch,...
type_id,flag_type,channelnr_id,flag_nr);

% Moving Window Filter
TrialExclude = [];
ChannelExclude = [3];
Method = 'median';
IntervalLength = 128;
ProgressBarFlag = 0;
P_C = gBSmovingwindowfilter(P_C, TrialExclude, ChannelExclude, Method,...
IntervalLength, ProgressBarFlag);
```

The following code shows how to repeat the example demonstrated above from the MATLAB command line computing the **Moving median** instead of the **Moving average**.

```
%Load Data
P_C=data;
File=['C:\Users\' getenv('USERNAME')
'\Documents\gtec\gBSanalyze\testdata\BCI\trig1.mat'];
P_C=load(P_C,File);

%Select Trials and Channels
trial_id=[];
channel_id=[];
type_id=[];
```

```

channelnr_id=[3];
flag_tr='tr_exc';
flag_ch='ch_exc';
flag_type='type_exc';
flag_nr='nr_exc';
[TrialExclude,
ChannelExclude]=gBSselect(P_C,trial_id,flag_tr,channel_id,flag_ch,...
type_id,flag_type,channelnr_id,flag_nr);

% Moving Window Filter
TrialExclude = [];
ChannelExclude = [3];
Method = 'median';
IntervalLength = 128;
ProgressBarFlag = 0;
P_C = gBSmovingwindowfilter(P_C, TrialExclude, ChannelExclude, Method,...
IntervalLength, ProgressBarFlag);

```

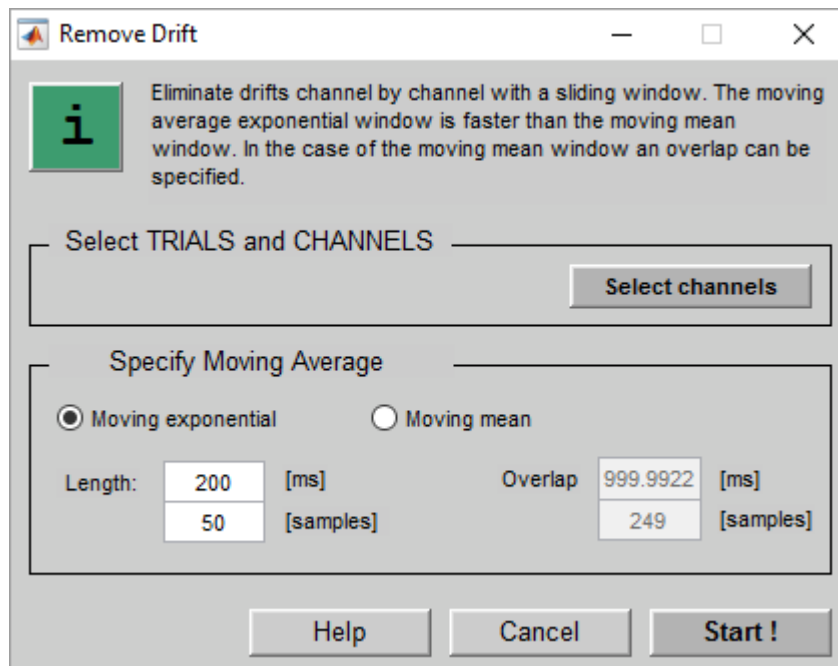
Remove Drift

The **Remove Drift** window allows to correct electrode and amplifier drifts from the loaded data-set. Three algorithms are integrated in g.BSanalyze for drift correction:

- **Moving exponential window** (Remove Drift window) - a moving exponential window of a backwards window is calculated and subtracted from the raw data
- **Moving mean** (Remove Drift window) - the moving mean is calculated from a backwards window and subtracted from the raw data
- FIR filter '*Remove Drift*' (Filter window) - the raw data is highpass filtered with an FIR filter

The window allows the following inputs:

- **Select channels** - chose the channels for the operation
- **Moving exponential** - calculate the moving exponential window with a window size specified by the **Length** editor box
- **Moving mean** - calculate the moving mean with a window size specified by the **Length**. **Overlap** specifies the stepsize of the moving mean window. Reducing the overlap speeds up the calculation
- **Length** - specifies the backwards window length
- **Overlap** - specifies the step size of the backwards window

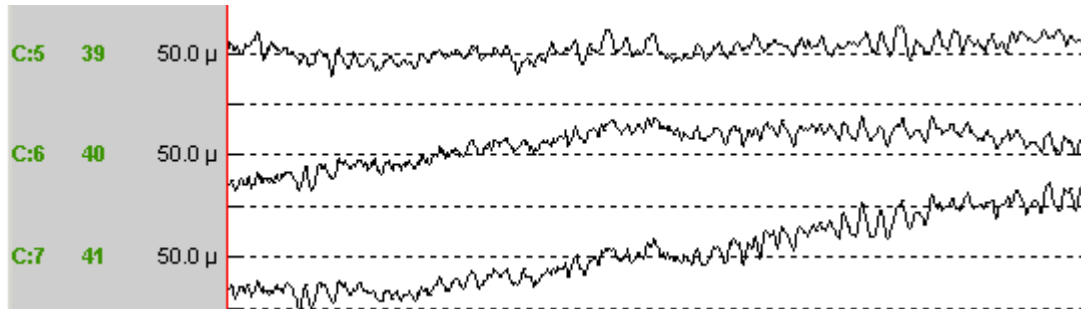


Perform the following steps:

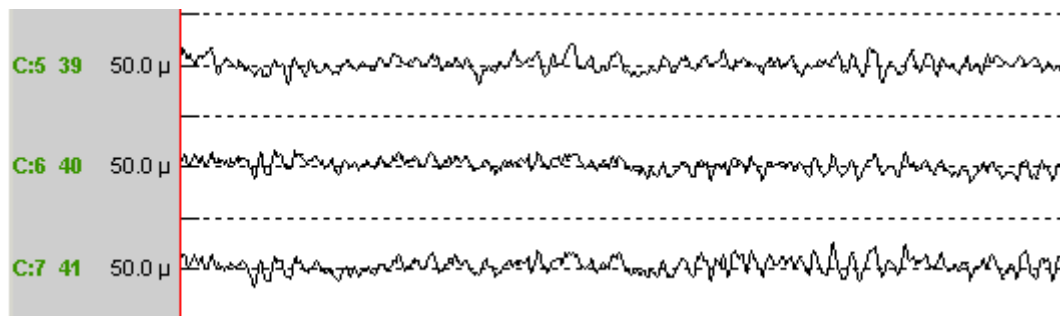
1. Load the data-file `test.mat` under

`Documents/gtec/gBSanalyze/testdata/Drift`

Channel 6 and 7 show a clear DC drift.



2. Click on **Remove Drift** under the **Pre-Processing** menu
3. Select **Moving exponential** with a **Length** of 50 samples and press **Start** to get the following drift corrected result.



The following code shows how to perform the example demonstrated above from the MATLAB command line.

```
%Load Data
P_C=data;
File=['C:\Users\' getenv('USERNAME')
'\Documents\gtec\gBSanalyze\testdata\Drift\test.mat'];
P_C=load(P_C,File);

%Remove Drift
ChannelExclude = [];
Method = ['Exponential'];
IntervalLength = 50;
Overlap = 249;
ProgressBarFlag = 0;
P_C = gBSremovedrift(P_C, ChannelExclude, Method, ...
IntervalLength, Overlap, ProgressBarFlag);
```

Detrend

Detrend performs the elimination of linear trends across selected trials and channels by selecting the

Estimation interval

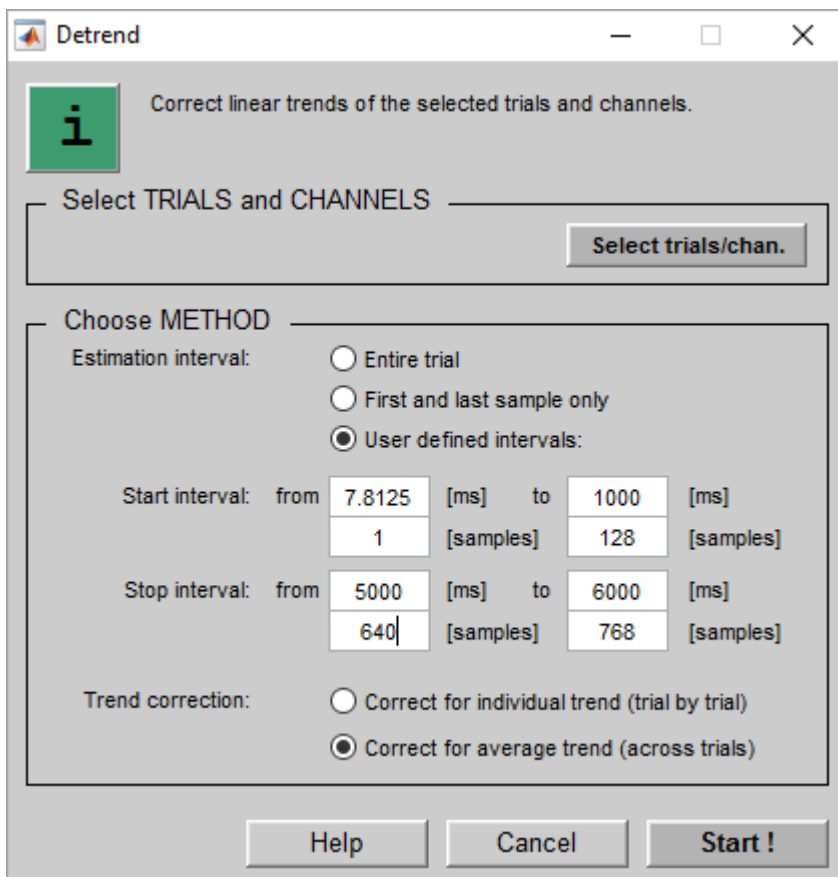
Entire trial - the best straight-line fitting through all samples of a trial is computed

First and last sample only - the trend line is placed through the first and the last element of the trial

User defined intervals - define a start and stop interval within the range of the data. The trend line is placed through the median of both intervals.

Trend correction

The data is either de-trended **trial by trial** (individual trend) or **across all trials** (average trend). The average trend is computed by calculating the trend for each trial and averaging the trends over trials.



Perform the following steps:

1. Load `trig1.mat` from
`Documents\gttec\gBSanalyze\testdata\BCI`
2. Click on **Detrend** under the **Pre-Processing** menu

3. Select **User defined intervals**. Set **Start interval** from 1 to 128 samples and the **Stop interval** to 640 to 768 samples.
4. Select **Correct for average trend (across trials)** to calculate the trend of each trial and to average all trends
5. Click **Start** to correct each trial with the average trend.

The following code shows how to perform the example demonstrated above from the MATLAB command line.

```
%Load File
P_C=data;
File= ['C:\Users\' getenv('USERNAME')
'\Documents\gtec\gBSanalyze\testdata\BCI\trig1.mat'];
P_C=load(P_C,File);

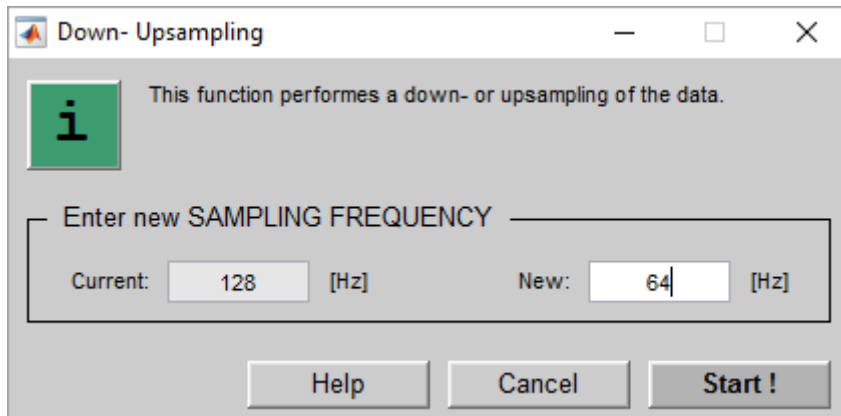
% Detrend
TrialExclude = [];
ChannelExclude = [];
EstimationInterval = 'user';
StartInterval = [1 128];
StopInterval = [640 768];
TrendCorrection = 'average';
ProgressBarFlag = 0;
[P_C, TrendCell] = gBSdetrend(P_C, TrialExclude, ChannelExclude,...
    EstimationInterval, StartInterval, StopInterval,...
    TrendCorrection, ProgressBarFlag);
```

Down- and Upsampling

Down- Upsampling changes the current sampling rate to a specified new one.

To change the sampling frequency of your data:

1. Click on **Down- Upsampling** from the **Pre-Processing** menu



2. Enter the **New** sampling frequency
3. Click **Start** to down-sample the data to 64 Hz.

The following code shows how to perform the example demonstrated above from the MATLAB command line.

```
% Down- Upsampling  
NewSmpFreq = 64;  
ProgressBarFlag = 0;  
P_C = gBSdownupsampling(P_C, NewSmpFreq, ProgressBarFlag);
```

Spatial Filter

The **Spatial Filter** window allow to move raw-data into the ICA-, PCA- or CSP-space with the corresponding filters calculated under the **Analyze** menu. Furthermore, the window allows to remove specific components from the raw data.

Method

Assume an EEG data-set with N channels represented as rows in matrix X . The rows of the output data matrix $U = WX$ are time courses of the activation of e.g. ICA components. Then the columns of the inverse matrix W^{-1} give the projection strengths of the components on the scalp electrodes. The scalp maps allow to investigate the physiological feasibility of the projections. For example eye activity should result in mainly frontal projections. Muscle activity should result in temporal projections. After investigating the maps corrected EEG channels X_{new} can be calculated as

$$X_{new} = (W)^{-1}U_{corrected}$$

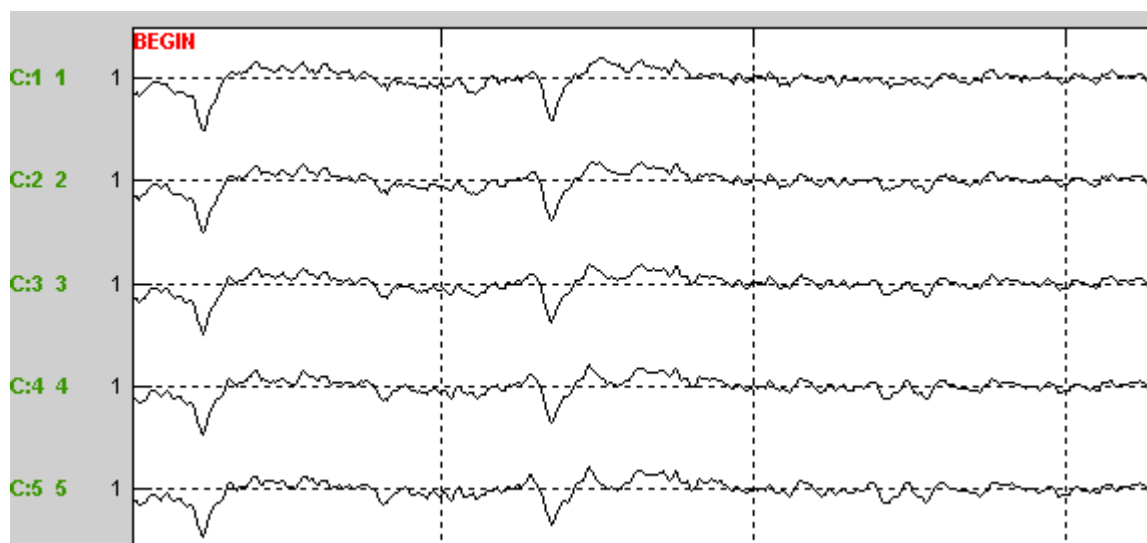
where $U_{corrected}$ is matrix U with the rows representing artifactual components set to zero. This results in a lower rank of X_{new} compared to the original data X .

Jung, T.P., Humphries, C., Lee, T.W., Makeig, S., McKeown, M.J., Iragui, V., Sejnowski, T., "Removing Electroencephalographic Artifacts: Comparison between ICA and PCA," Neural Networks for Signal Processing.

Perform the following steps:

1. Load the 27 EEG electrode data-set `run1234_geom.mat` from

```
Documents\gtec\gBSanalyze\testdata\RightLeftImagination
```

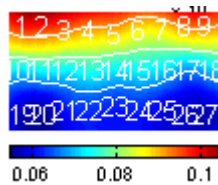


The figure shows a 3.2 second portion of the recorded EEG time series from 5 electrodes, all referred to the right ear. Two EOG artifacts within the first 2 seconds on all channels can be found.

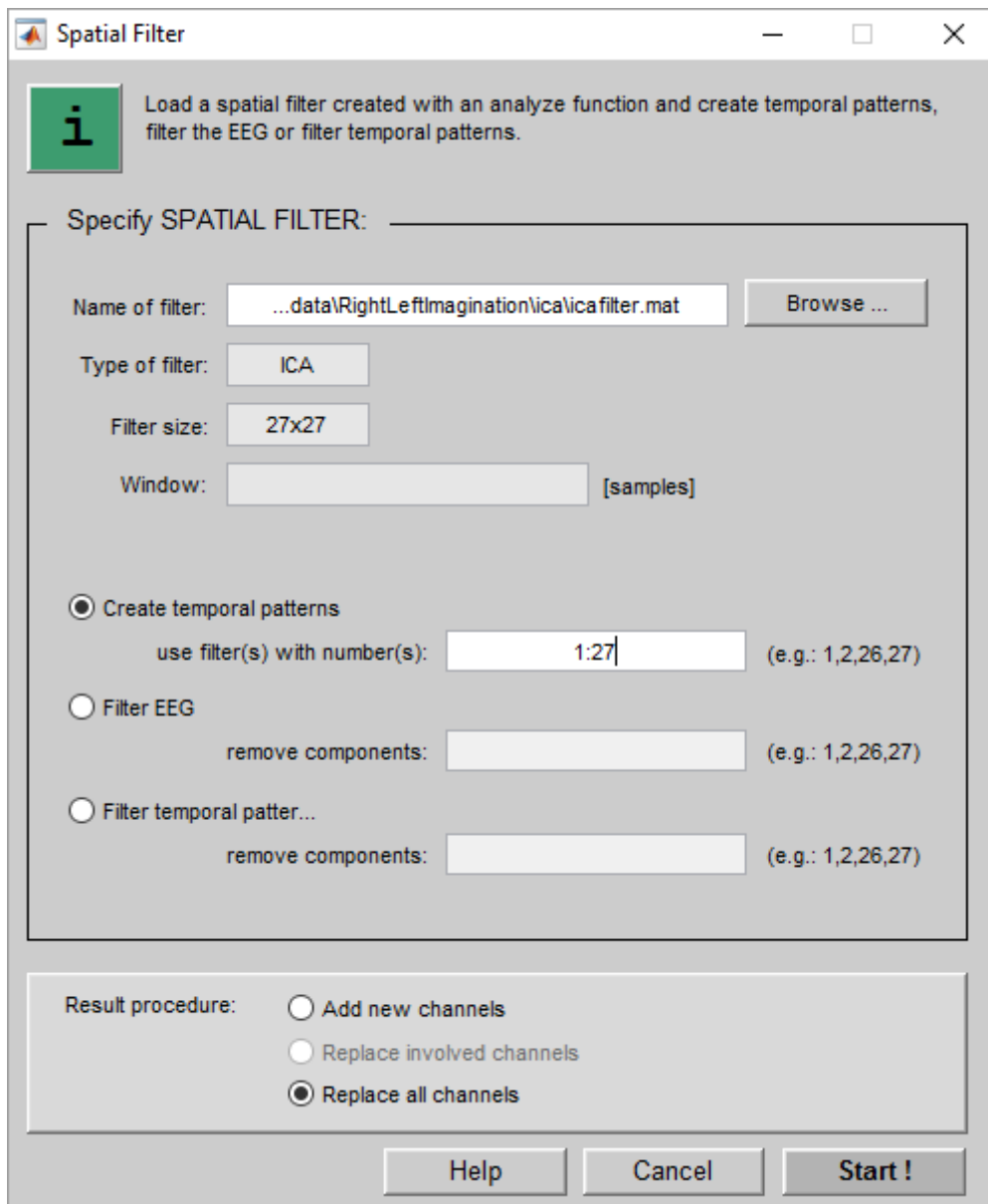
2. Click on **ICA** under the **Analyze** menu to perform an independent component analysis of the data (for details see section ICA under chapter Analyze). Store the calculated ICA filter under `icafilter.mat` in the directory

`Documents\gtec\gBSanalyze\testdata\RightLeftImagination\ica`

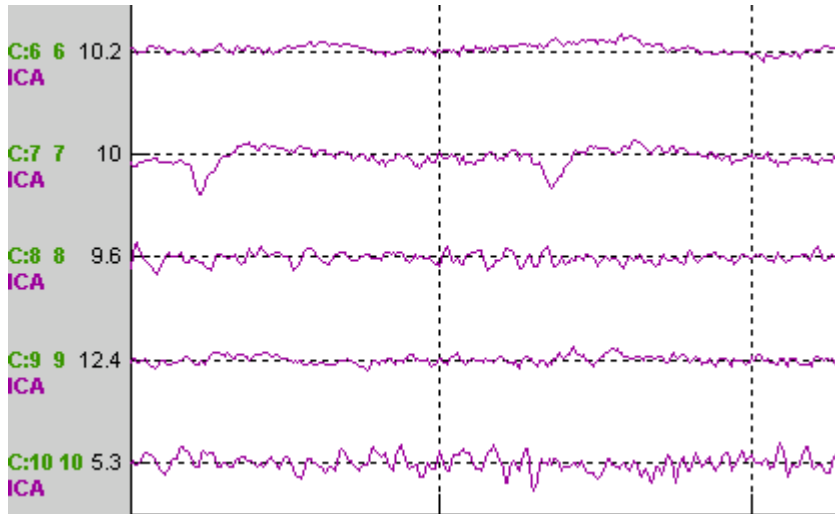
3. Start the calculation and Result2D opens automatically with 27 ICA filter projections. The columns of the inverse matrix W^{-1} give the projection strengths of the components on the scalp electrodes. Filter number 7 with strong frontal activity is presented:



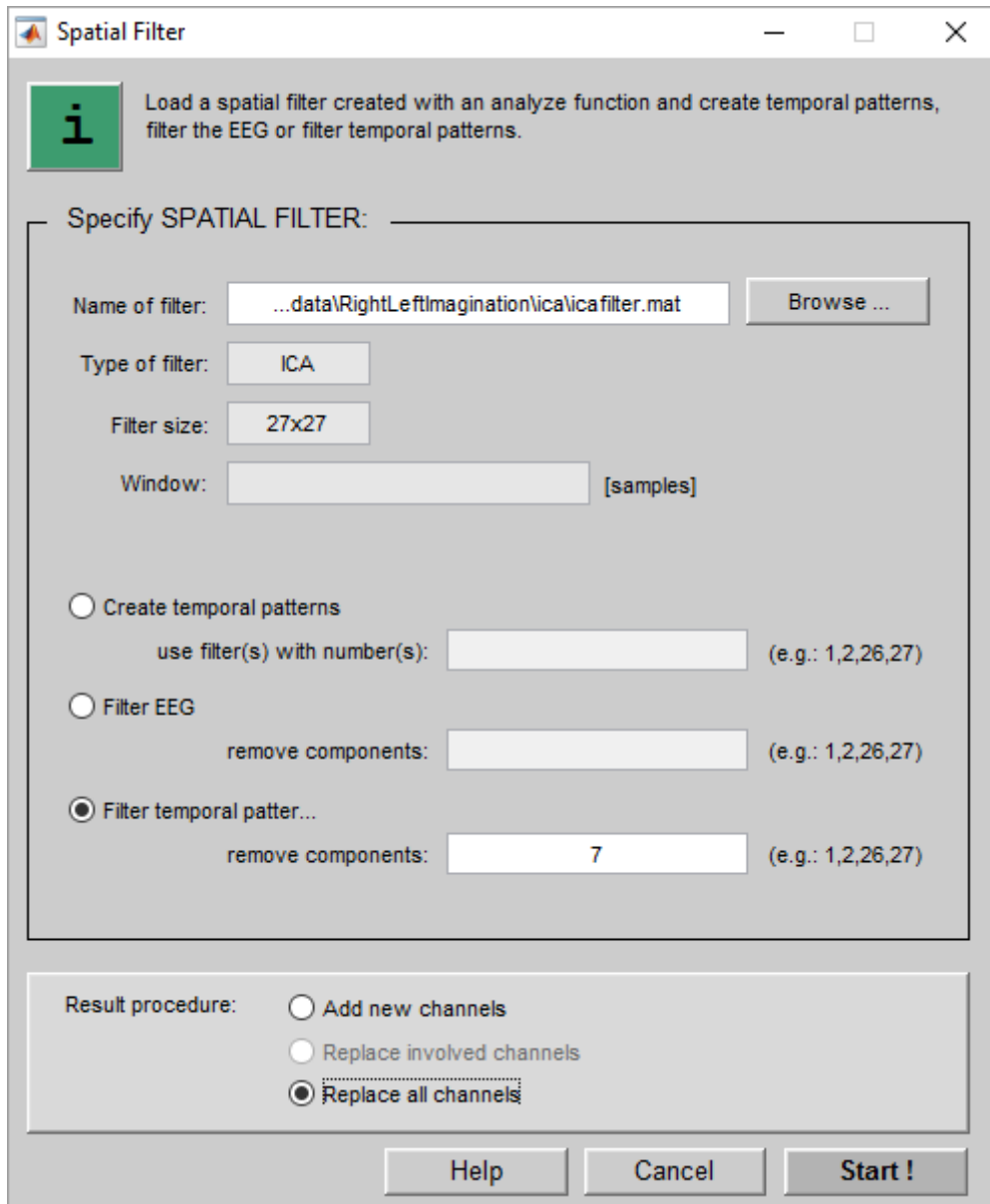
4. Click on **Spatial Filter** under **Pre-Processing** and load with the **Browse** button the created ICA filter



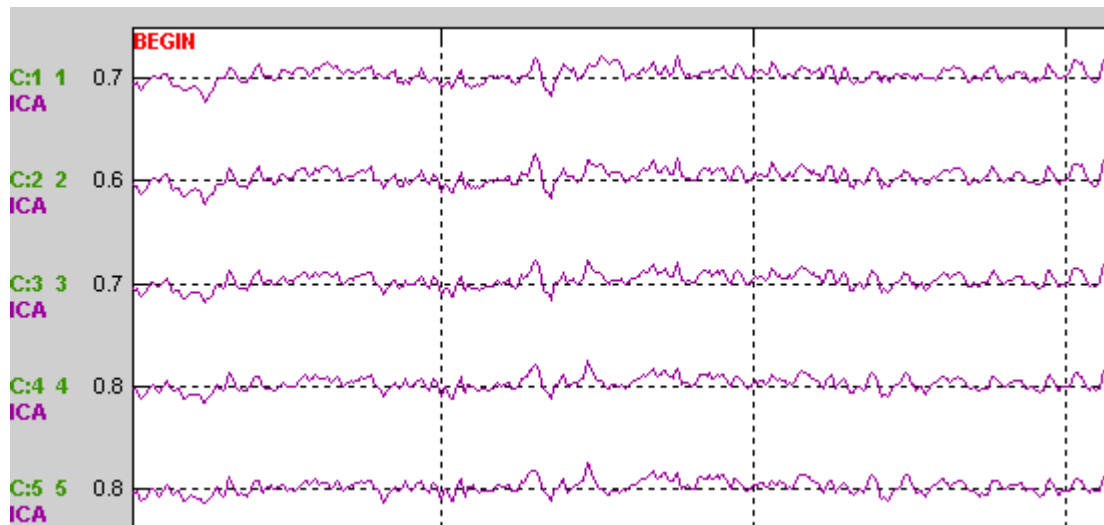
5. Select **Create temporal patterns** with filter number 1 to 27 to transform the EEG data into the ICA space. Click on **Start** to perform the operation. Each channel transformed to the ICA space is marked with the channel attribute ICA. The eye movement artifacts at second 0.2 and second 1.3 are isolated in ICA component 7.



- Open the **Spatial Filter** window from the **Pre-Processing** menu and select **Filter temporal patterns**. Enter **remove component 7** to obtain the corrected EEG signal.



- Click **Start** to transform the ICA components without component 7 back to the EEG space. After eliminating this artifactual component, by zeroing out the corresponding rows of the activation matrix U , the corrected EEG data are free of EOG artifacts.



Assume that the EEG data is loaded in the Data Editor and ICA component 7 should be removed. Open **Spatial Filter** from the **Pre-Processing** menu and load the ICA-filter. Check **Filter EEG** and enter **remove component 7**. This will eliminate ICA component 7 (the EOG artifact) from your EEG data directly.

The following code shows how to perform the example demonstrated above from the MATLAB command line.

```
%Load the data
P_C=data;
File= ['C:\Users\' getenv('USERNAME')
'\Documents\gtec\gBSanalyze\testdata\RightLeftImagination\run1234_geom.mat'
];
P_C=load(P_C,File);

%Calculate the ICA filter
clear T
T.lags=[0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16
17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35
36 37 38 39 40];
T.hos=0;
T.hosc=1;
T.rmmean=1;
T.sort=0;
TrialExclude=[];
ChannelExclude=[];
FileName= ['C:\Users\' getenv('USERNAME')
'\Documents\gtec\gBSanalyze\testdata\RightLeftImagination\ica\icafilter.mat'
'];
I_O=gBSica(P_C,T,TrialExclude,ChannelExclude,FileName,1);

%Transform to ICA space
SPF=spf;
Filter=load(SPF,['C:\Users\' getenv('USERNAME')
User Manual g.BSanalyze 5.16.02 136
```



```

'\Documents\gtec\gBSanalyze\testdata\RightLeftImagination\ica\icafilter.mat
');
FilterNumber=[1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16
17 18 19 20 21 22 23 24 25 26 27];
Replace='replace all channels';
Transformation='Create temporal pattern';
P_C=gBSspatialfilter(P_C,Filter,FilterNumber,Replace,Transformation);

%Remove ICA component 7
SPF=spf;
Filter=load(SPF, ['C:\Users\' getenv('USERNAME')
'\Documents\gtec\gBSanalyze\testdata\RightLeftImagination\ica\icafilter.mat
']);
FilterNumber=[7];
Replace='replace all channels';
Transformation='Filter temporal pattern';
P_C=gBSspatialfilter(P_C,Filter,FilterNumber,Replace,Transformation);

```

Filter

The **Filter** window allows to select a predefined filter and to highpass, lowpass, bandpass and bandstop-filter the data.

The method uses the `filtfilt` function with is a zero-phase forward and reverse digital filter.

The filter is described by the difference equation:

$$y(n) = b(1)*x(n) + b(2)*x(n-1) + \dots + b(nb+1)*x(n-nb) \\ - a(2)*y(n-1) - \dots - a(na+1)*y(n-na)$$

After filtering in the forward direction, the filtered sequence is then reversed and run back through the filter; Y is the time reverse of the output of the second filtering operation. The result has precisely zero phase distortion and magnitude modified by the square of the filter's magnitude response (see MATLAB documentation).

If the data-set is longer than 2^{15} samples the filter function is used.

The filter is a "Direct Form II Transposed" implementation of the standard difference equation (see MATLAB documentation):

$$a(1)*y(n) = b(1)*x(n) + b(2)*x(n-1) + \dots + b(nb+1)*x(n-nb) \\ - a(2)*y(n-1) - \dots - a(na+1)*y(n-na)$$

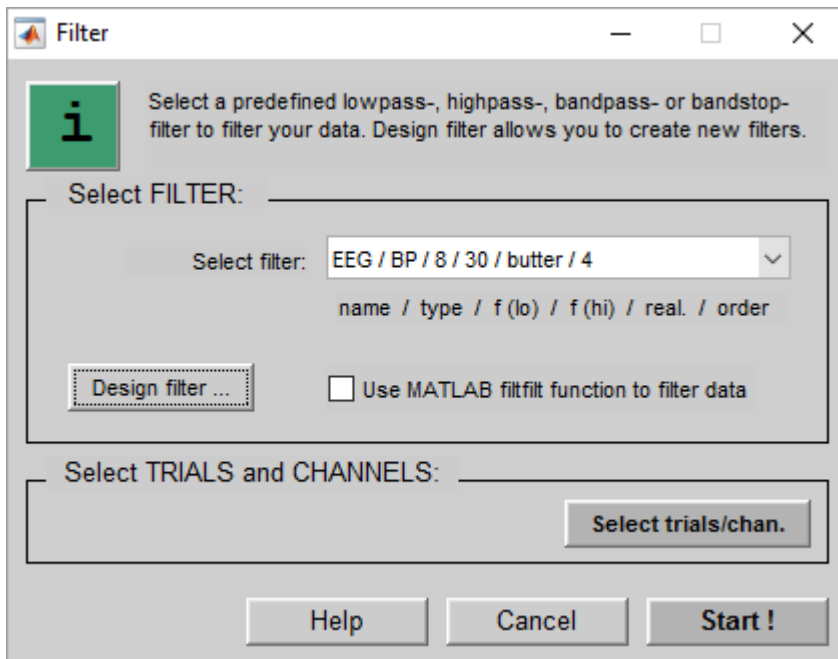
The window allows the following inputs:

- **Select filter** - select the predefined filter from the pulldown-menu
- **Design filter** - design a new filter
- **Select trials / chan.** – apply the filter only to specific data channels and trials

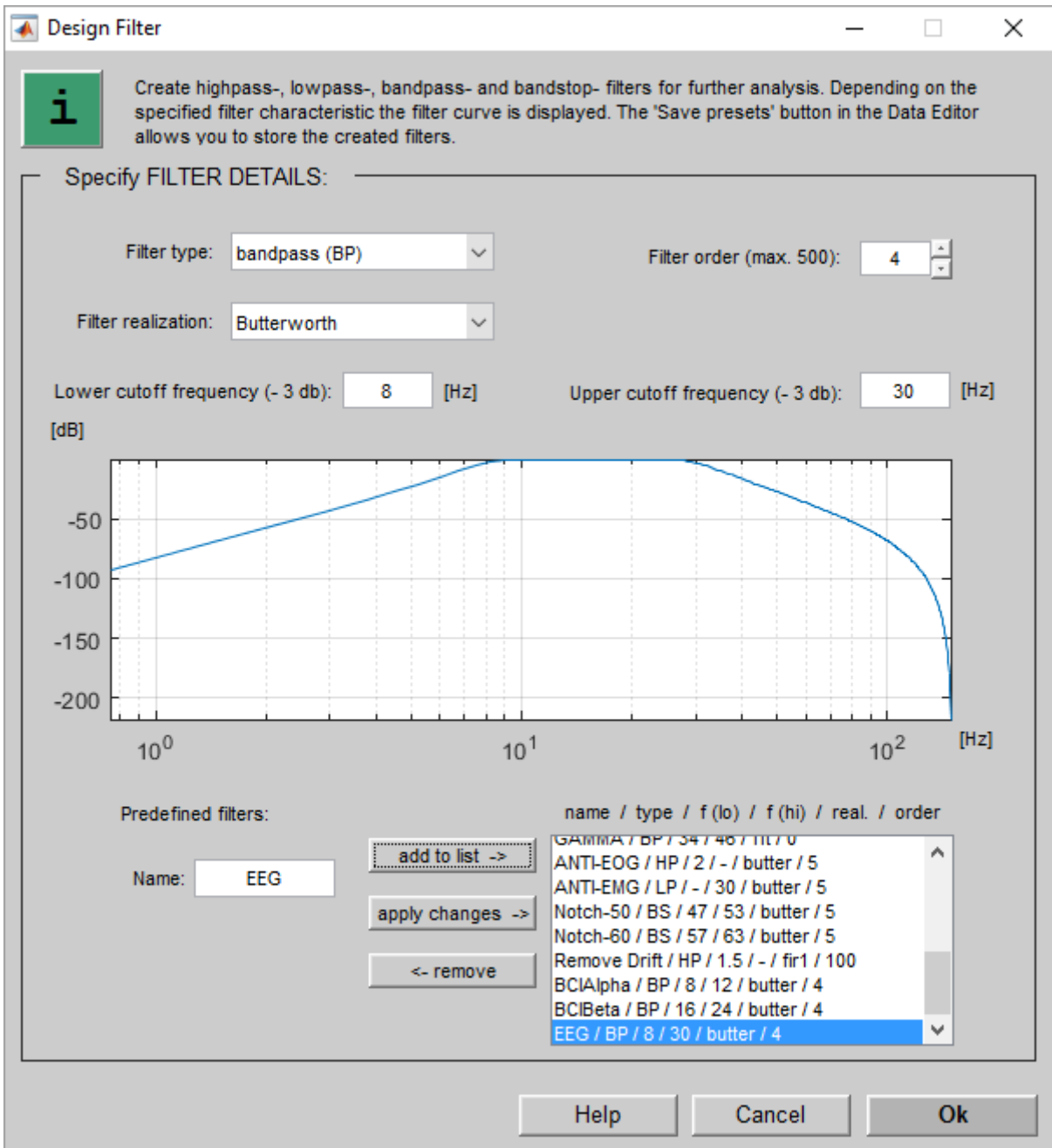
1. Load file `run1234_geom.mat` from

`Documents\gtec\gBSanalyze\testdata\RightLeftImagination`

2. Click on **Filter** under the **Pre-Processing** menu



3. Click on the **Design filter** button to open the following window



Select as **Filter type** bandpass, as **Filter realization** Butterworth, enter a **Filter order** of 4, a **Lower cutoff frequency** of 8 Hz and a **Upper cutoff frequency** of 30 Hz. The transfer function of the designed filter is shown in the window. Assign the name EEG to the filter and press **add to list**. Close the Window with the **OK** button. To store the designed filter press the **Save presets** button in the Data Editor.

4. Select the EEG filter from the pulldown menu
5. Click **Start** to perform the operation

The following code shows how to perform the example demonstrated above from the MATLAB command line.

```
%Load the 27 channel data set
P_C=data;
File=['C:\Users\' getenv('USERNAME')
'\Documents\gtec\gBSanalyze\testdata\RightLeftImagination\run1234_geom.mat'
];
P_C=load(P_C,File);

%Filter
Filter.Realization='butter';
Filter.Type='BP';
Filter.Order=4;
Filter.f_high=30;
Filter.f_low=8;
FiltFiltFlag=0;
TrialExclude=[];
ChannelExclude=[];
P_C=gBSfilter(P_C,Filter,FiltFiltFlag,ChannelExclude,TrialExclude);
```

Tools

This chapter describes the following tools:

[Reaction Time](#) – calculate reaction times between 2 events

[Single Trial Analysis](#) – perform horizontal averaging

[Trigger Finder](#) – find on-sets and off-sets of triggers created with g.TRIGbox

Reaction Time

This function searches for a starting event and a response event and calculates the time between them. As criterion a threshold voltage in μV , a threshold level or a slew-rate can be defined.

Specify a SEARCH INTERVAL – allows to define an interval before and after the starting event. Within this interval the method searches for the response.

Specify a STIMULUS – define to search for a starting-event in a physical channel or to search for a time marker. For the physical channel define also the edge and the threshold criteria.

Calculate intervals between stimuli – check this box to search for responses within one channel or from one marker to the next marker with the same name.

Accept first stimuli within trial only – check to accept only the first event per trial

Specify a RESPONSE – define to search for a physical channel with the threshold criteria or to search for a marker

Accept first response within search interval only – accept only the first response

Ignore multiple responses within intervals of – accept only responses which are separated by the specified time window

Assign trial ATTRIBUTES – can be used to assign trial attributes to the data

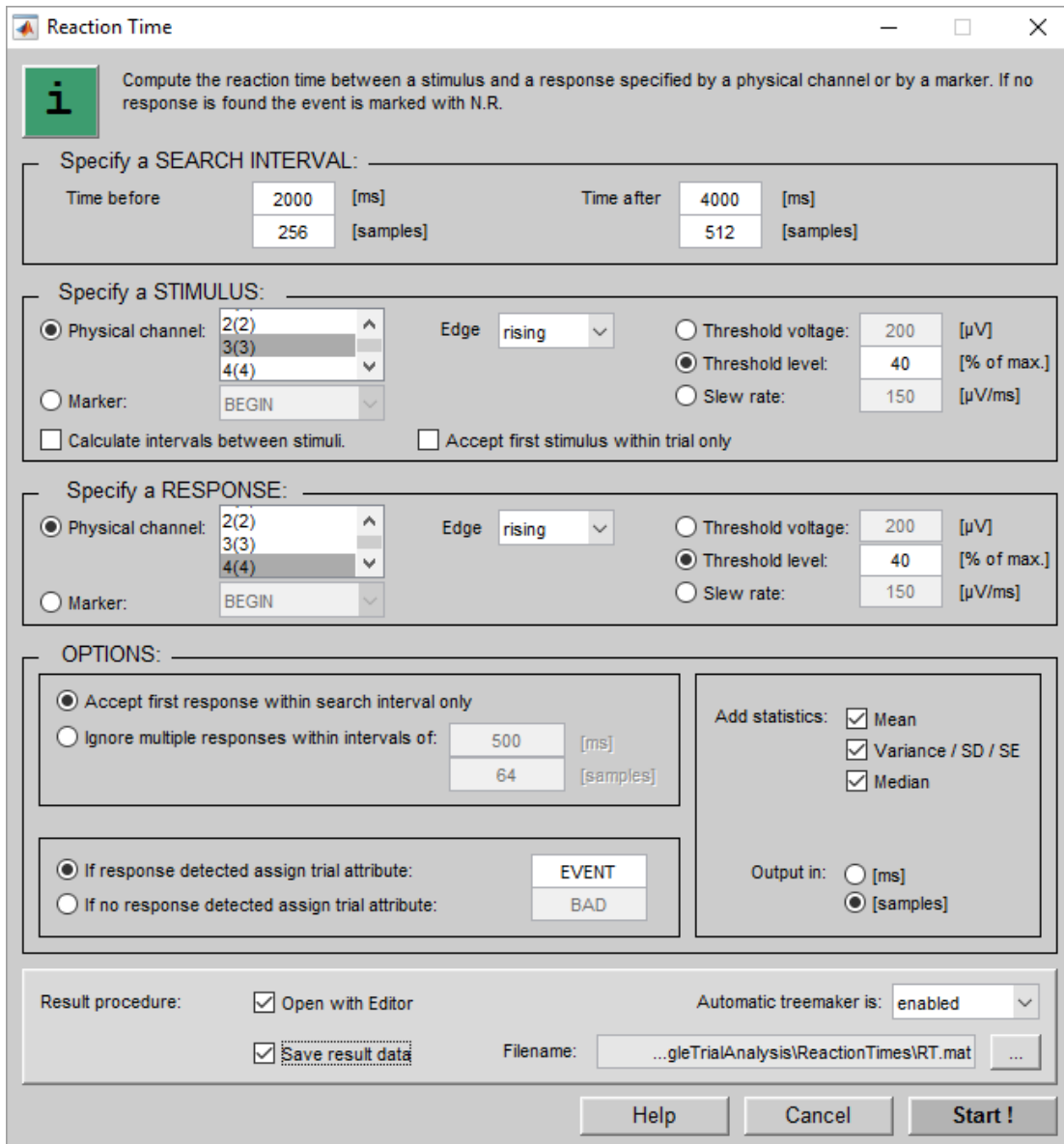
Add statistics – add some statistic parameters about the reaction times to the report

Output – set the output unit to **ms** or **samples**

Result procedure – check the **Open with Editor** box to investigate the resulting reaction time matrix with the MATLAB editor. Check **Save result data** and specify a filename to store the matrix as MATLAB file for further processing. If the **Automatic treemaker** is enabled a subdirectory called `ReactionTimes` is created and the file is saved into this directory.

1. Load the file `test_data.mat` with a sampling rate of 128 Hz from

Documents\gtec\gBSanalyze\testdata\SingleTrialAnalysis



2. Click on **Reaction Time** under the **Tools** menu

3. Select **Physical Channel 3** as **STIMULUS** and set the **Threshold level** to 40 % of max

4. Select **Physical channel 4** as **RESPONSE** and set also the **Threshold level** to 40 % of max

5. Check the **Save result data** box and set the filename to `rt.mat` under the suggested directory

6. Press **START** to perform the calculation

7. Go to the directory where `rt.mat` was saved and type `load rt` to load the matrix into MATLAB. Type `Mat_gReactionTime` to investigate the result. The first and second

column specify the stimulus number and the trial number of the event, the third column the starting time point within the trial of the STIMULUS, the fourth column the inter-stimulus interval followed by the time point of the response. The last column represents the time difference between the STIMULUS and the RESPONSE in samples (reaction time).

*** Reaction Times *** ASCII Result File

Input File: C:\Users\\gtec\gBSanalyze\testdata\SingleTrialAnalysis\test_data.mat
Generated: 27-Jan-2003 15:45:12
Search Interval: -2000.00 ms [-256 samples] to 4000.00 ms [512 samples]
Stimulus: Ch3 (3) raising edge 40.00% of max threshold level
Response: Ch4 (4) raising edge 40.00% of max threshold level

[samples]

Stim#:	Tr#:	StimTP:	ISI:	RTP:	RT:
1	1	129.0000	0.0000	384.0000	255.0000
2	2	129.0000	0.0000	416.0000	287.0000
3	3	129.0000	0.0000	448.0000	319.0000

Statistics of the reaction times (column RT):

Mean:	287.0000
Variance:	1024.0000
Standard deviation SD:	32.0000
Standard error SE:	18.4752
Median:	287.0000

*)ISI...interstimuli interval for stimuli of the same trial

8. Load the results into MATLAB by typing

```
load RT.mat
```

and enter

```
Mat_gReactionTime
```

to view the reaction time matrix and enter

```
Statistics_gReactionTime
```

to view the statistic.

The following code shows how to perform the example demonstrated above from the MATLAB command line.

```
%Load Data
P_C=data;
File=['C:\Users\' getenv('USERNAME')
'\Documents\gtec\gBSanalyze\testdata\SingleTrialAnalysis\test_data.mat'];
P_C=load(P_C,File);
P_C.SamplingFrequency=128;

% Reaction Time
rt.SamplesBefore = 256;
rt.SamplesAfter = 512;
rt.Source_stim = 'channel';
rt.ChannelNum_stim = 3;
rt.Source_stim = 'channel';
rt.Edge_stim = 'raising';
rt.EventBy_stim = 'level';
rt.Value_stim = 40;
rt.MarkerName_stim = 'BEGIN';
rt.setIntervals_stim = 0;
rt.setOneStimulusPerTrial = 0;
rt.ChannelNum_resp = 4;
rt.Source_resp = 'channel';
rt.Edge_resp = 'raising';
rt.EventBy_resp = 'level';
rt.Value_resp = 40;
rt.MarkerName_resp = 'BEGIN';
rt.OnlyOneResp = 1;
rt.Delay_resp = 64;
rt.add_mean = 1;
rt.add_variance = 1;
rt.add_median = 1;
rt.set_values = 'samples';
rt.setAttribute_resp = 1;
rt.Attribute_resp = 'EVENT';
rt.setAttribute_noresp = 0;
rt.Attribute_noresp = 'BAD';
FileName = ['C:\Users\' getenv('USERNAME')
'\Documents\gtec\gBSanalyze\testdata\SingleTrialAnalysis\ReactionTimes\RT.m
at'];
ProgressBarFlag = 0;

[P_C, Mat_gReactionTime, Statistics_gReactionTime] = gBSreactiontime(P_C,
rt, FileName, ProgressBarFlag);
```

Single Trial Analysis

The function allows to average horizontally over samples for a reference period (Re) and for an active period (Act) and to calculate the following equation for each trial and channel:

$$ERD = \frac{Act - Re}{Re} * 100 [\%]$$

The window has the following entries:

Specify REFERENCE PERIOD – enter the start and end point used as reference period

Specify ACTIVE PERIOD – allows to enter either a

Fixed period – enter the start point and end point of the active interval

or to load a reaction time file created with the **Reaction Time** function in the **Tools** menu.

Entire reaction period - uses the interval from a fixed time point to the occurrence of the reaction. Therefore, the active interval is of variable length depending on the user-response.

In relation to response – uses an interval starting before the reaction and ending after the reaction. Therefore, the active interval is of the same size for all trials.

Horizontal averaging method – select mean or median

Exclude marked epochs – allows to exclude multichannel epochs from the calculation. Use the **Eventfinder** to assign multichannel epochs automatically to the data or assign the epochs yourself with the **EPOCHING** tool in the Data Editor.

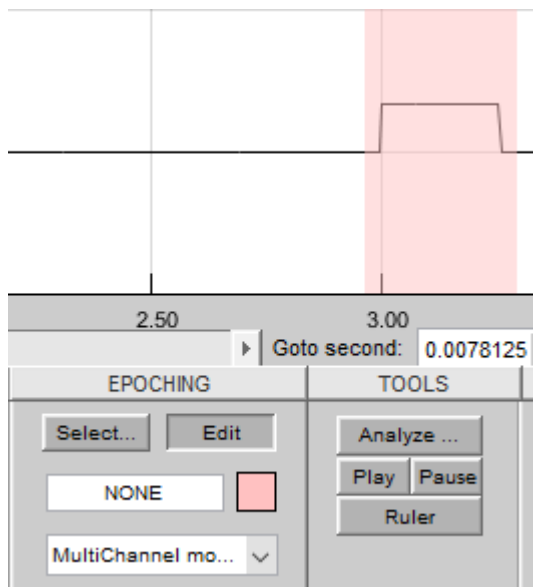
Example:

1. Load the file `test_data.mat` with a sampling rate of 128 Hz from

`Documents\gtec\gBSanalyze\testdata\SingleTrialAnalysis`

2. Perform the **Reaction Time** calculation described in Section **Reaction Time**

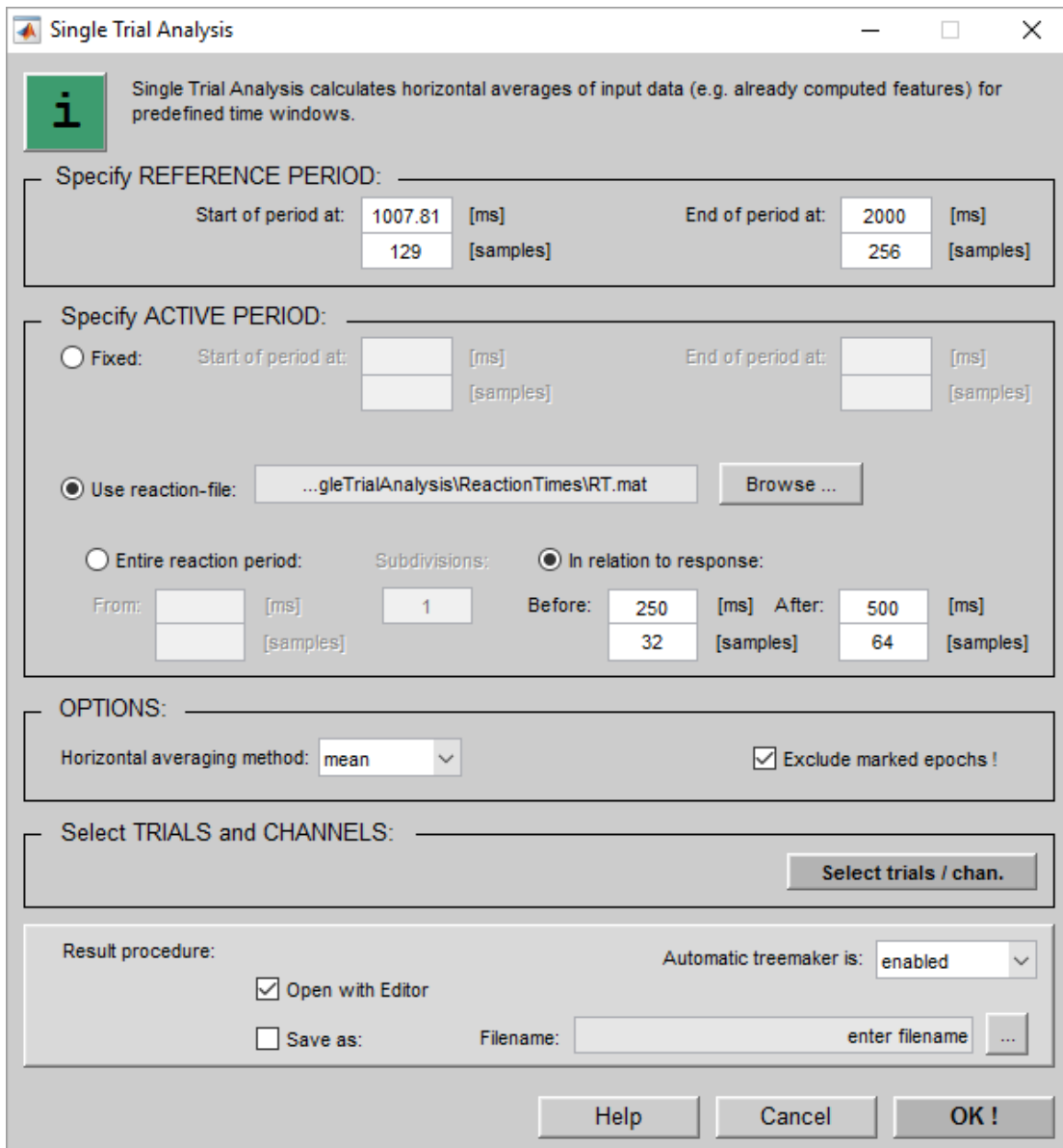
3. Select the **Multi chan. EPOCHING** mode and mark all data segments that should be excluded from the calculation. In this example the user-response impulse in channel 4 of the first trial is marked.



4. Click on **Single Trial Analysis** under the **Tools** menu and select the **REFERENCE PERIOD** from 129 to 256 samples
5. Select use **reaction-file** and **Browse** for the file `rt.mat` under

```
Documents\gtec\gBSanalyze\testdata\SingleTrialAnalysis\...
ReactionTimes
```
6. Select **In relation to response** and enter an interval of 32 samples before the user response and 64 samples after the user response
7. Check **Exclude marked epochs**

8. Press **OK** to perform the calculation and to open the MATLAB Editor with the results



The averages for the reference period, the averages for the active period and the ERD values are presented for each trial and channel. The total number of discarded samples, samples used for the calculation of the reference period and samples used for the calculation of the active period are given for each trial. Note that the mean of trial 1 in the active period for channel 4 is zero because of the removal of the multi-channel epoch.

```

1 | Single Trial Analysis: Horizontal Mean of Reference and Active Period
2 |
3 | Reference Interval:      129      256
4 | Reference Period Mean: [trials x channels]
5 | 1.000000 1.000000 1.000000 0.000000
6 | 0.500000 2.000000 1.000000 0.000000
7 | 1.500000 3.000000 1.000000 0.000000
8 |
9 | Mean: [1 x channels]
10 | 1.000000 2.000000 1.000000 0.000000
11 |
12 | Active Period Mean: [trials x channels]
13 | 0.550000 0.550000 0.000000 0.000000
14 | 0.136082 1.020619 0.000000 0.340206
15 | 0.544330 0.680412 0.000000 0.340206
16 |
17 | Mean: [1 x channels]
18 | 0.544330 0.680412 0.000000 0.340206
19 |
20 | ERD: [trials x channels]
21 | -45.000000 -45.000000 -100.000000      NaN
22 | -72.800000 -49.000000 -100.000000      Inf
23 | -63.700000 -77.300000 -100.000000      Inf
24 |
25 | Mean: [1 x channels]
26 | -63.700000 -49.000000 -100.000000      NaN
27 |
28 | Total number of discarded samples [trials x 1]:
29 | 43.000000
30 | 0.000000
31 | 0.000000
32 |
33 | Samples used to calculate Reference Period [trials x 1]:
34 | 128.000000
35 | 128.000000
36 | 128.000000
37 |
38 | Samples used to calculate Active Period [trials x 1]:
39 | 54.000000
40 | 97.000000
41 | 97.000000
42 |

```

The following code shows how to perform the example demonstrated above from the MATLAB command line.

```
%Load Data
P_C=data;
File=['C:\Users\' getenv('USERNAME')
'\Documents\gtec\gBSanalyze\testdata\SingleTrialAnalysis\test_data.mat'];
P_C=load(P_C,File);
P_C.SamplingFrequency=128;

%Single Trial Analysis
Ref_Start=[129];
Ref_End=[256];
Active_Definition='user_reaction_response_related';
Reaction_Filename= ['C:\Users\' getenv('USERNAME')
'\Documents\gtec\gBSanalyze\testdata\SingleTrialAnalysis\ReactionTimes\RT.m
at'];
Act_Start=[32];
Act_End=[64];
SubDivisions=[];
Method='mean';
ExcludeEpochs=1;
ChannelExclude=[];
TrialExclude=[];
FileName='';
out=gBSSingletrialanalysis( ...
    P_C,Ref_Start,Ref_End,Active_Definition,...
    Reaction_Filename,Act_Start,Act_End,SubDivisions,Method,...
    ExcludeEpochs,ChannelExclude,TrialExclude,FileName,0);
```

Trigger Finder

This function searches for trigger on-sets and trigger off-sets created with g.TRIGbox. Therefore the channel with the trigger signal must be specified and the function assigns markers with specific names to the on-sets and off-sets. Markers with the same name that already exist will be deleted. **Trigger Finder** has different modes depending on the amplifier that was used for the data acquisition. Therefore set the **Amplifier name** either to g.USBamp or g.MOBILab in the **Amplifier** window.

The window has the following settings:

Set start marker – activate the check-box to search for trigger on-sets and specify a name for the marker

Set end marker – activate the check-box to search for trigger off-sets and specify a name for the marker

Ignore multiple responses within intervals of – markers which are too close together will be ignored

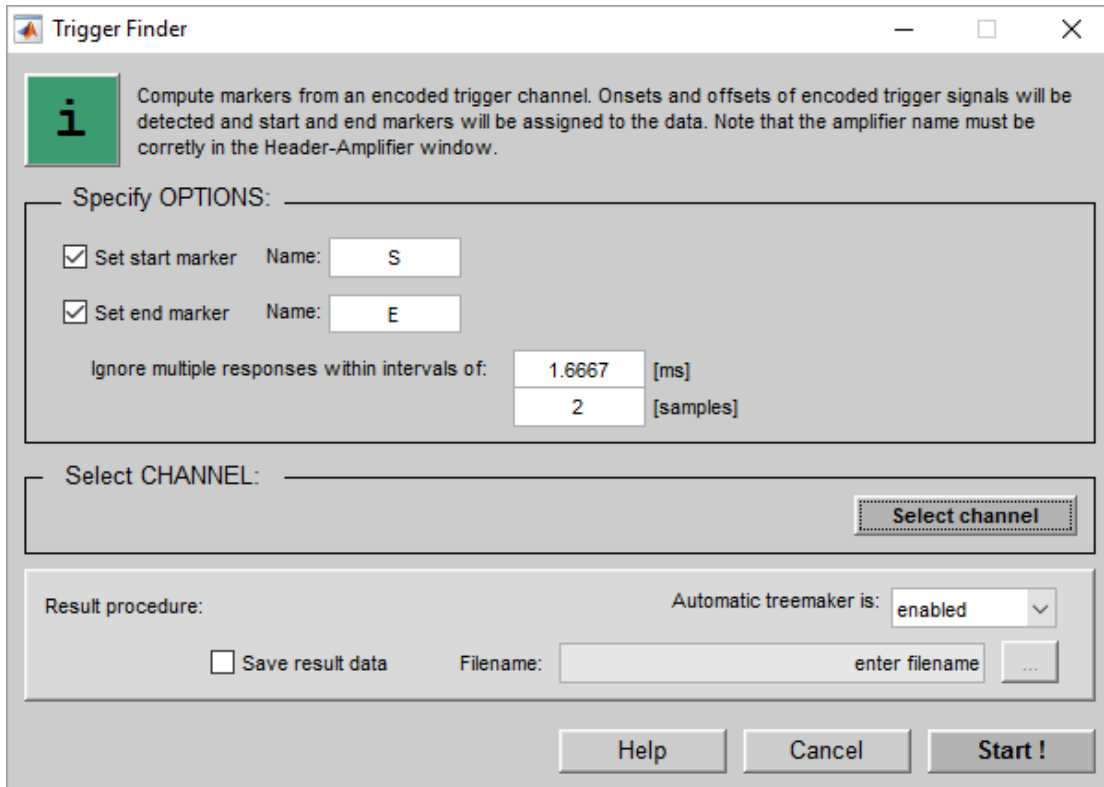
Select channel – specify the channel that should be used to search for the triggers. The function works only for a single channel.

Example:

1. Load the file 1200Hz.mat from

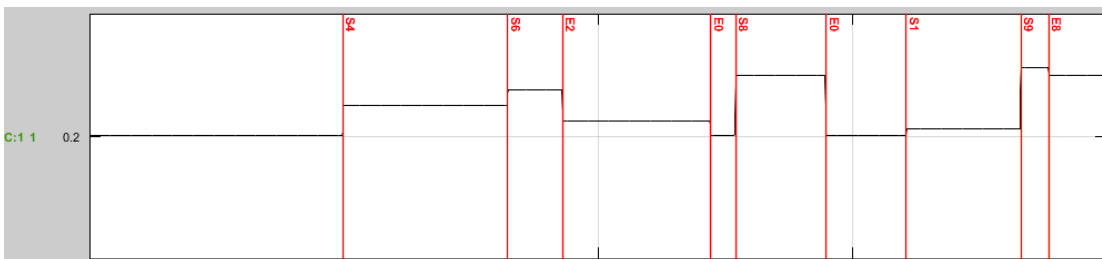
Documents\gtec\gBSanalyze\testdata\TrigBox

2. Open the **Trigger Finder** window from the **Tools** menu



3. Activate the **Set start marker** and **Set end marker** check-boxes to assign the marker names S to on-sets and E to off-sets.
4. Click on **Select channel** and select channel 1
5. Press the **Start** button to search for the triggers.

The assigned markers are shown in the Data Editor:



All markers with the name beginning with S indicate on-sets and markers with E indicate off-sets. The numbers indicate the level of the trigger impulse. g.TRIGbox can generate from 4 inputs a binary coded output signal. Therefore 16 trigger levels exist. The first marker S4 is therefore the on-set marker with level 4 (binary 0 1 0 0).

The following code shows how to perform the example demonstrated above from the MATLAB command line.

```
%Load Data
P_C=data;
File=['C:\Users\' getenv('USERNAME')
```

```
'\Documents\gtec\gBSanalyze\testdata\TrigBox\1200Hz.mat'];
P_C=load(P_C,File);

%Trigger Finder
SetStartMarker=[1];
SetEndMarker=[1];
StartMarkerName=['S'];
EndMarkerName=['E'];
IgnoreInterval=[2];
ChannelExclude=[2 3 4 5];
FileName='';
P_C=gBStriggerfinder(P_C,SetStartMarker,SetEndMarker,...
StartMarkerName,EndMarkerName,IgnoreInterval,ChannelExclude,...
FileName,0);
```

Artifact

The **Artifact** window allows the following operations

[Eventfinder](#) – detect overflows and zero-lines in the data-set

[Signal Quality Check](#) – detect bad signal quality based on standard deviation and amplitude criteria

[TMS Discharge Artifact Removal](#) – detect and remove typical TMS discharge artifacts from EEG data.

Eventfinder

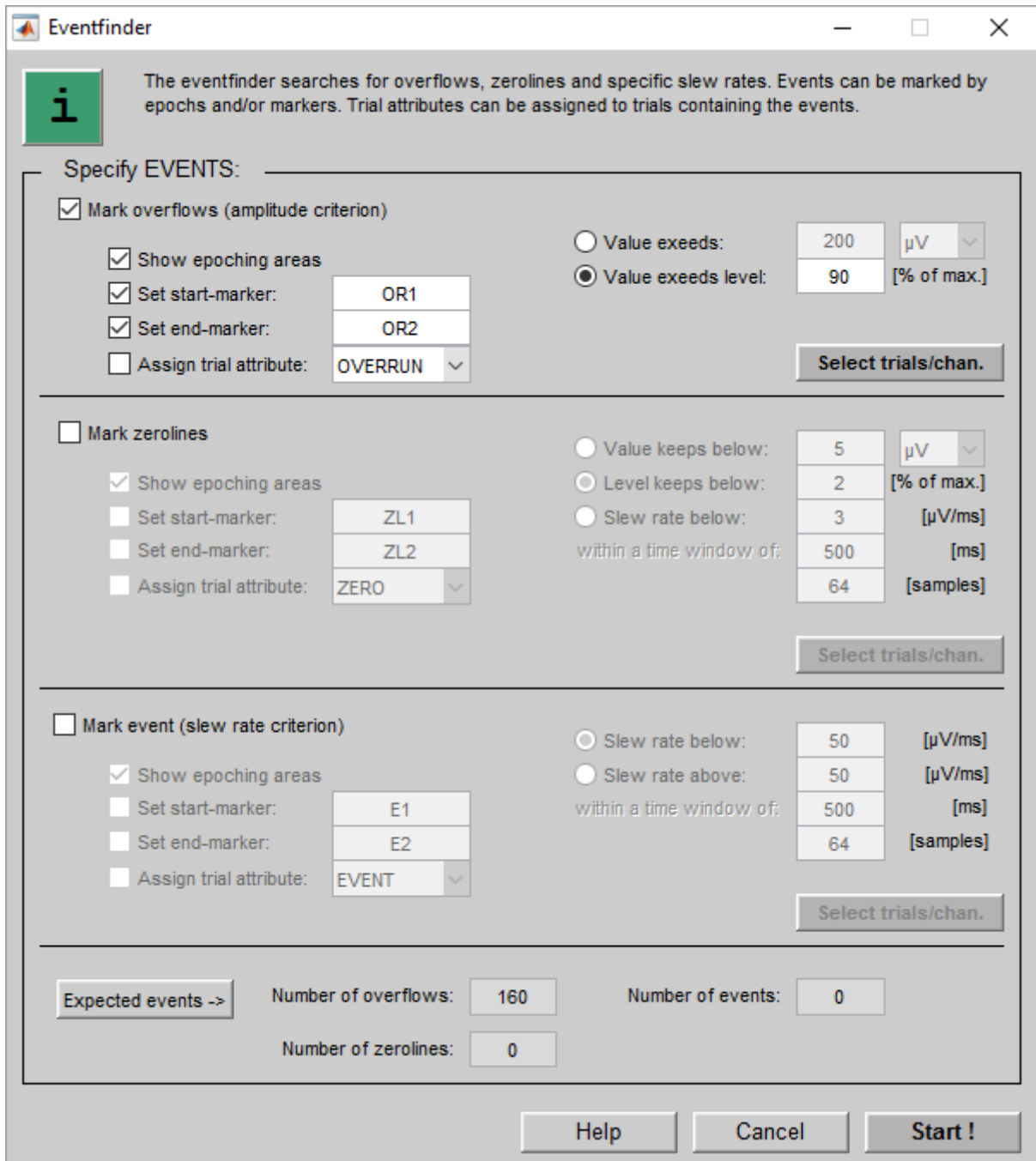
Eventfinder can be used to find the following events:

- **overflows** – search for amplifier overflows, EOG artifacts, EMG artifacts or trigger pulses by defining a specific threshold in μV or percentage of the maximum of each channel. Note that the absolute value of the input signal is compared to the threshold.
- **zerolines** – search for flat-lines causes for example by amplifier failure. To detect the zeroline the signal must be below a certain threshold in μV , percentage of the maximum or below a certain slew-rate within a specific time window.
- **events** – search for distortions causes e.g. by cross-talk of electrical signals, movement artifacts in the data,... To be accepted as event the slew-rate of the signal must be below or above a certain value within a specific time window. Note that positive and negative slew-rates are compared to the criterion.

1. Load the data-set `session1234triggered.mat` from

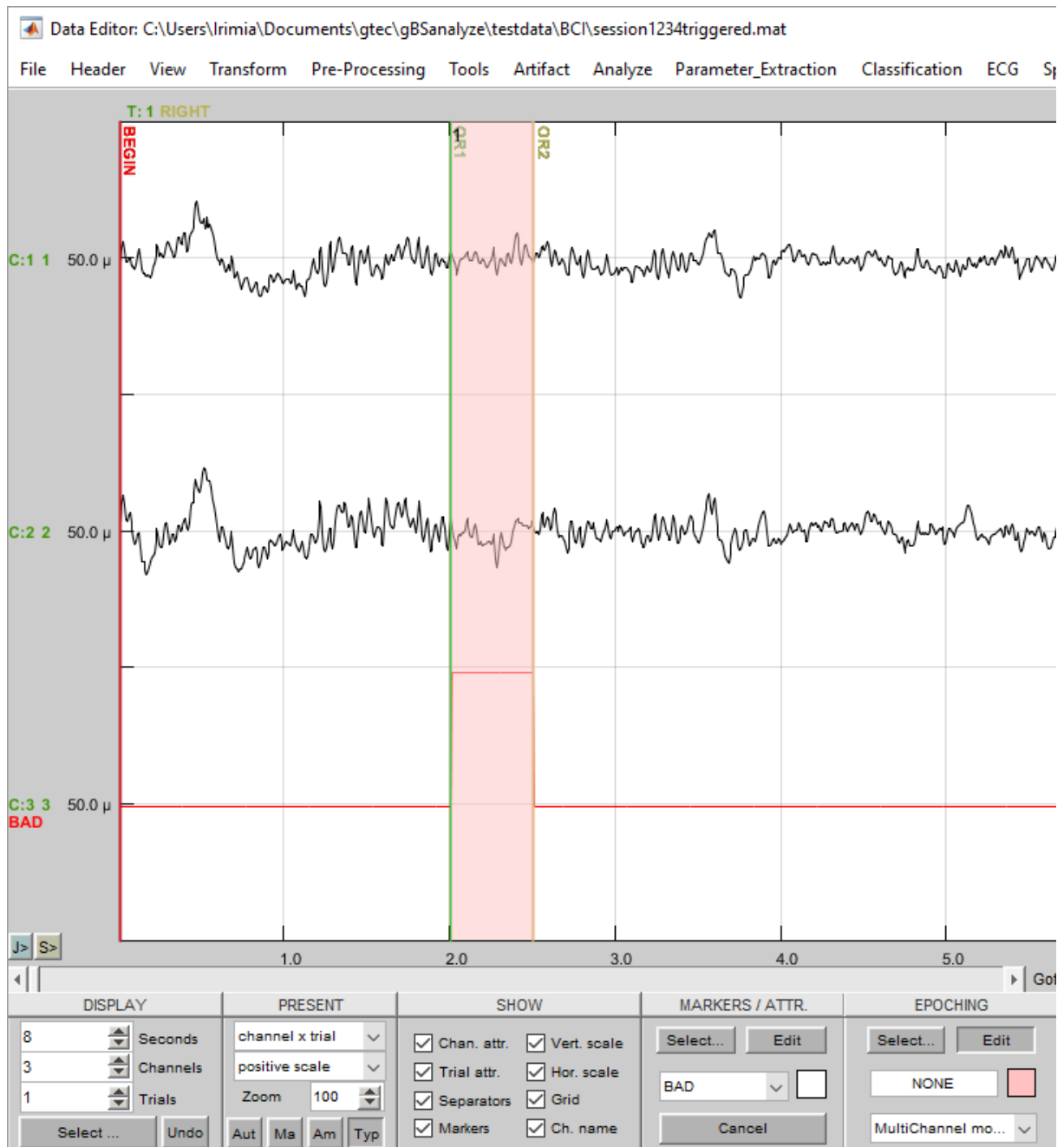
`Documents\gtec\gBSanalyze\testdata\bci`

and click on **Eventfinder** under the **Artifact** menu. Check **Mark overflows**.



2. Check **Show epoching areas** to mark the detected events as multi-channel epochs

3. Check the **Set start-marker** and **Set end-marker** to automatically set a time-marker at the beginning and ending of the event
4. Select as search criteria **Value exceeds level** 90 % of max. Press **Select trials/chan.** and include only channels with the attribute **BAD** (trigger channel).
5. Press the **Expected Events** button to investigate your criterion. Increase the threshold if too many events are found or decrease the threshold if no events are found.



6. Press the **Start** button to perform the operation
7. Select the **Multi chan.** mode under **EPOCHING** to investigate the result

Note: An event is determined by its start marker and stop marker. One pair of markers is counted as one event. Events might be found in different channels. The markers, resulting from these events are valid for all channels in the Data Editor! Overlapping intervals of start and stop markers are merged.

The following code shows how to perform the example demonstrated above from the MATLAB command line.

```
%Load Data
P_C=data;
File=['C:\Users\' getenv('USERNAME')
'\Documents\gtec\gBSanalyze\testdata\BCI\session1234triggered.mat'];
P_C=load(P_C,File);

%Select Trials and Channels
trial_id=[];
channel_id=[];
type_id=[];
channelnr_id=[3];
flag_tr='tr_exc';
flag_ch='ch_exc';
flag_type='type_exc';
flag_nr='nr_inc';
[TrialExclude,
ChannelExclude]=gBSselect(P_C,trial_id,flag_tr,channel_id,flag_ch,type_id,flag_type,channelnr_id,flag_nr);

%Eventfinder (overflow)
MarkOverflow = 1;
showEpochingAreas_over = 1;
setStartMarker_over = 1;
setStopMarker_over = 1;
AssignAttribute_over = 0;
StartMarker_over = 'OR1';
StopMarker_over = 'OR2';
TrialAttribute_over = 'OVERRUN';
Threshold_over = 90;
getUnit_over = '% of max';
TrialExclude_over = [];
ChannelExclude_over = [1 2];
ProgressBarFlag = 0;
[P_C, PreviewOverflow, VecThreshold] = gBSoverflow...
(P_C, MarkOverflow, showEpochingAreas_over,...
setStartMarker_over, setStopMarker_over,...
AssignAttribute_over, StartMarker_over, StopMarker_over,...
TrialAttribute_over, Threshold_over, getUnit_over,...
TrialExclude_over, ChannelExclude_over, ProgressBarFlag);
```

Signal Quality Check

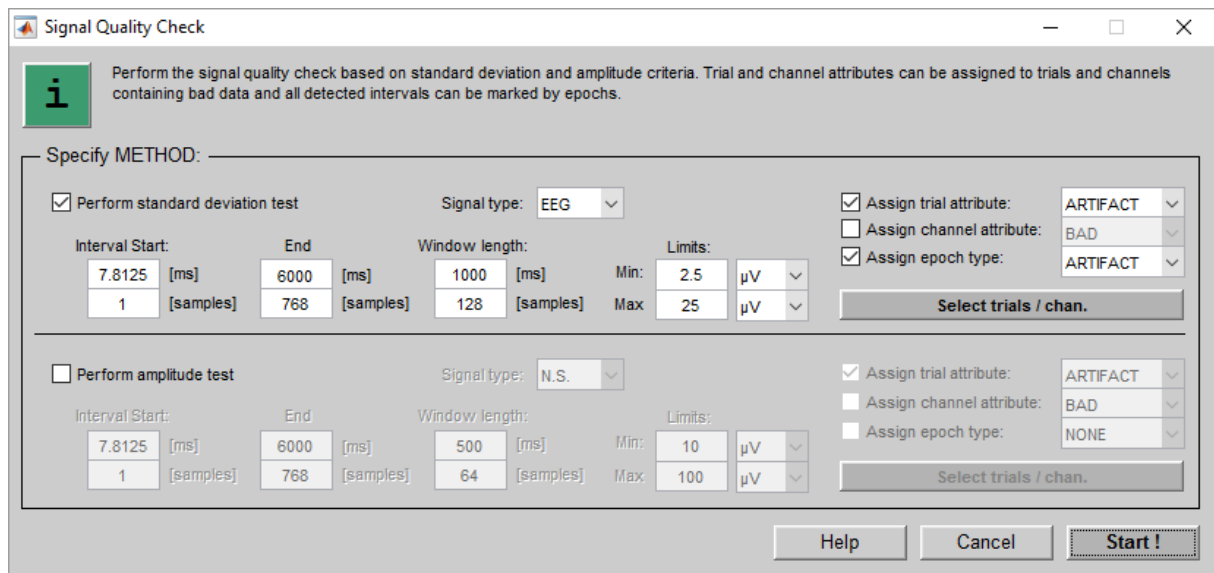
Signal Quality Check can be used to find and mark epochs, trials or channels containing bad signal quality. The check is done within a moving window shifted forward sample by sample and based on the following criteria:

- **Amplitude** – an artefact is detected if the absolute value of the signal is within the whole window below the lower threshold or above the upper threshold.
- **Standard deviation** – the samples within a window are considered to be artefacts if the standard deviation value computed for that window is out of the pre-defined limits.

1. Load the data-set `run1234.mat` from

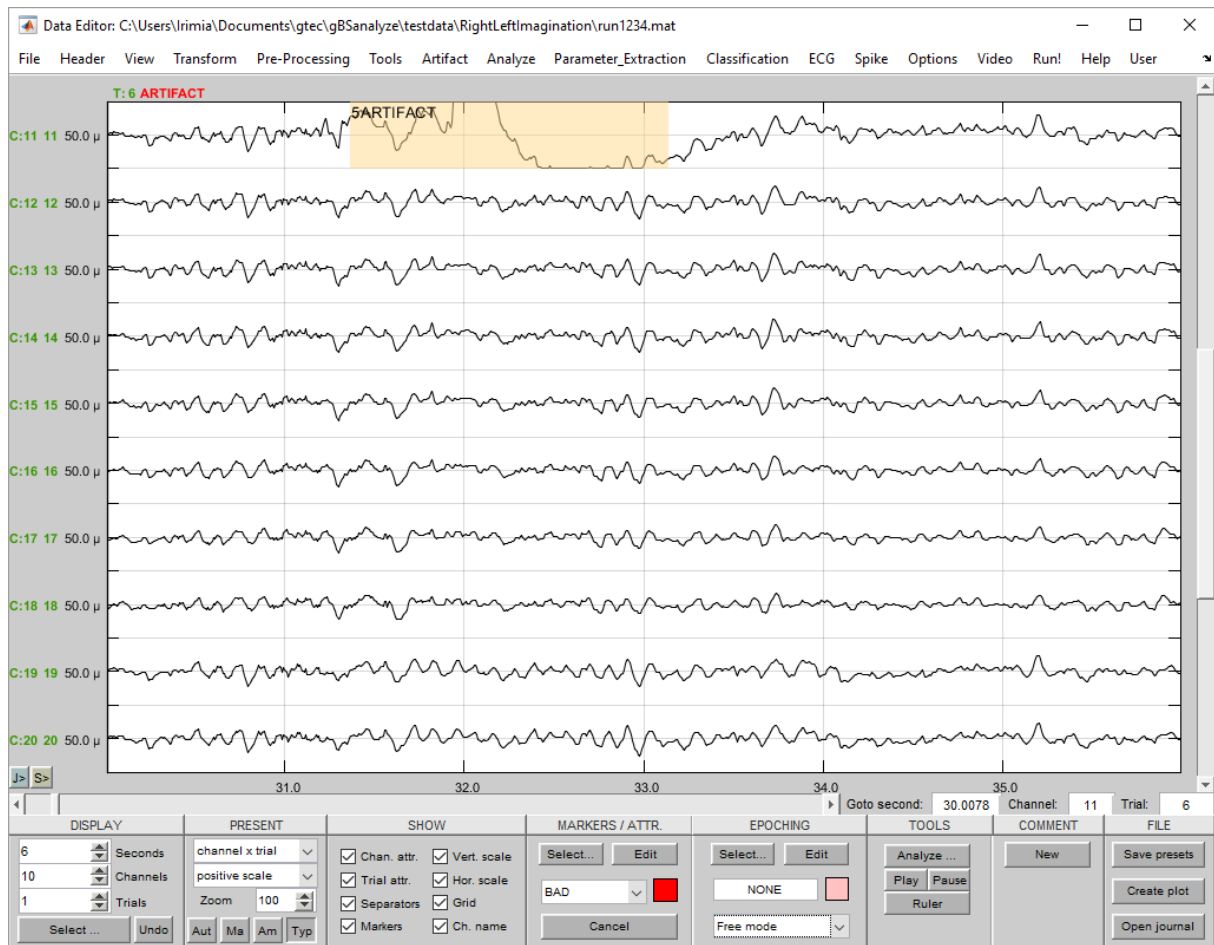
`Documents\gtec\gBSanalyze\testdata\RightLeftImagination`

and click on **Signal Quality Check** under the **Artifact** menu. Check **Perform standard deviation**.



2. Click on the **Signal type** popup menu and select **EEG** to automatically load the pre-defined parameters for EEG signal (the **Window length** for calculating the standard deviation and the **Min** and **Max** standard deviation limits);
3. Check the **Assign trial attribute** checkbox and select the **ARTIFACT** attribute to be assigned to the trials where the standard deviation of the signal is out of the pre-defined limits;
4. Check the **Assign epoch type** checkbox and select the **ARTIFACT** epoch name which will be assigned to the epochs containing bad data.
5. Press the **Start !** button to perform the operation.

After the operation is done, you can scroll through the data and investigate the marked trials and epochs.



The following code shows how to perform the example demonstrated above from the MATLAB command line.

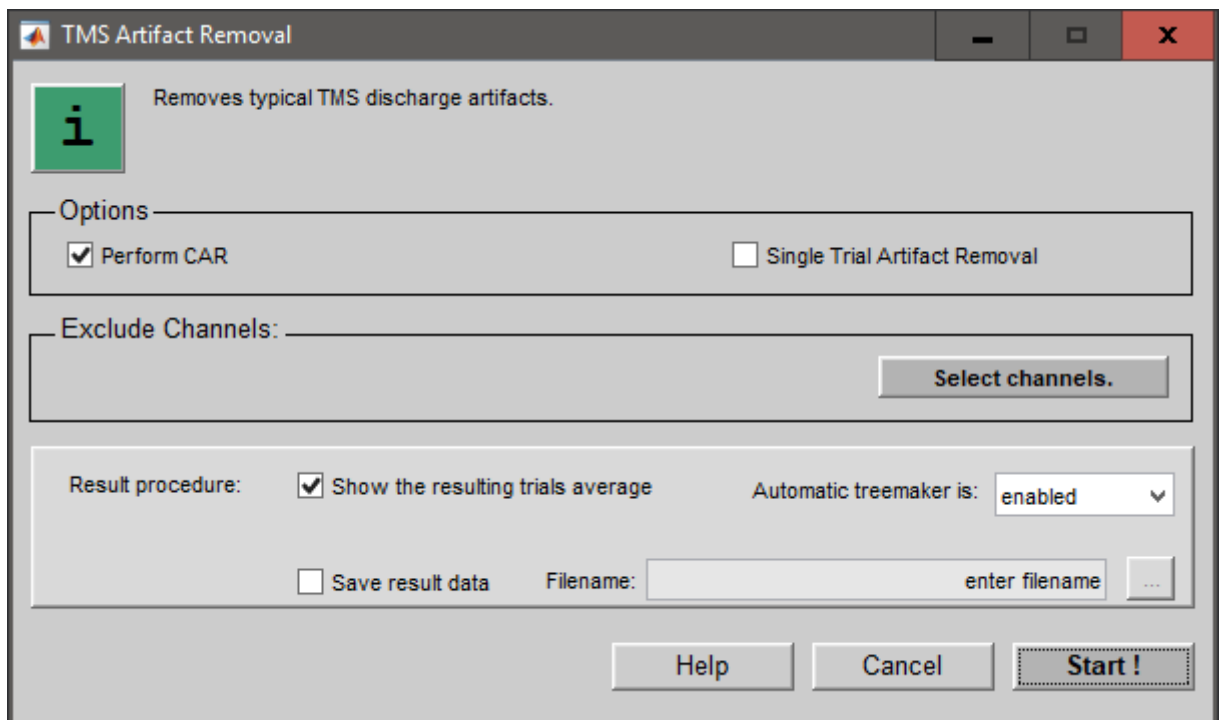
```
%Load Data
P_C=data;
File= ['C:\Users\' getenv('USERNAME')
'\Documents\gtec\gBSanalyze\testdata\RightLeftImagination\run1234.mat'];
P_C=load(P_C,File);

% Signal quality check based on standard deviation
IntervalStart = 1;
IntervalEnd = 768;
WindowLength = 128;
MinLimit = 2.5;
MaxLimit = 25;
Unit = 'μV';
TrialAttribute = 'ARTIFACT';
ChannelAttribute = '';
EpochName = 'ARTIFACT';
TrialExclude = [];
ChannelExclude = [];
ProgressBarFlag = 1;
P_C = gBSsqstandarddeviationcheck(P_C, IntervalStart, IntervalEnd, ...
WindowLength, MinLimit, MaxLimit, Unit, TrialAttribute, ...
ChannelAttribute, EpochName, TrialExclude, ChannelExclude, ...
ProgressBarFlag);
```

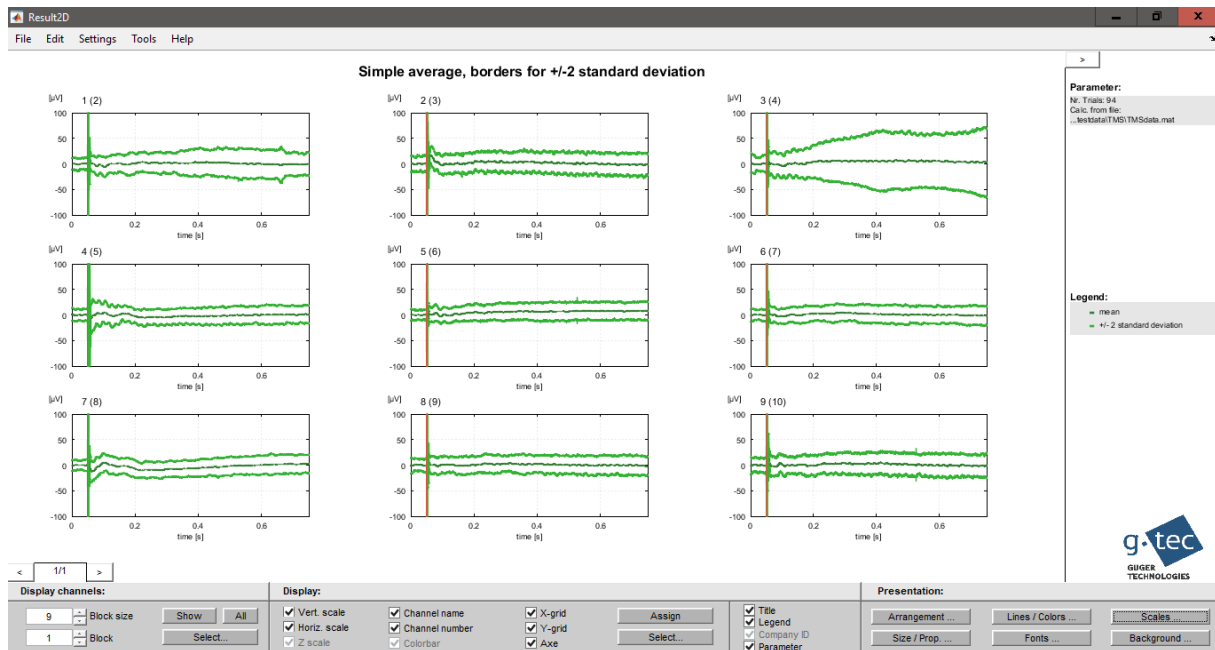
TMS Discharge Artifact Removal

TMS Discharge Artifact Removal Algorithm eliminates typical discharge artifacts occurring in TMS experiments. To this end, a model of superimposed exponentially decaying functions is robustly fit to the data. To diminish line noise interference, common-average referencing (CAR) is advised. To speed up computation whilst obtaining identical results on average, the artifact can be removed directly from the trial mean. This avoids fitting the exponential model to each trial individually.

1. Load the data-set `TMSdata.mat` from:
`Documents\gtec\gBSanalyze\testdata\TMS`
2. Click on **TMS Discharge Artifact Removal** under the **Artifact** menu.
3. Check the **Perform CAR** checkbox in order to re-reference the data with the Common Average Reference method.
4. Under **Result Procedure** check the **Show the resulting trials average** checkbox and then click on the **Start!** button.



After the operation is done, **gResult2D** will open automatically displaying the mean and standard deviation of the trials for each channel.



The following code shows how to perform the example demonstrated above from the MATLAB command line.

```
%Load Data
P_C=data;
File= ['C:\Users\' getenv('USERNAME')
'\Documents\gtec\gBSanalyze\testdata\TMS\TMSdata.mat'];
P_C=load(P_C,File);

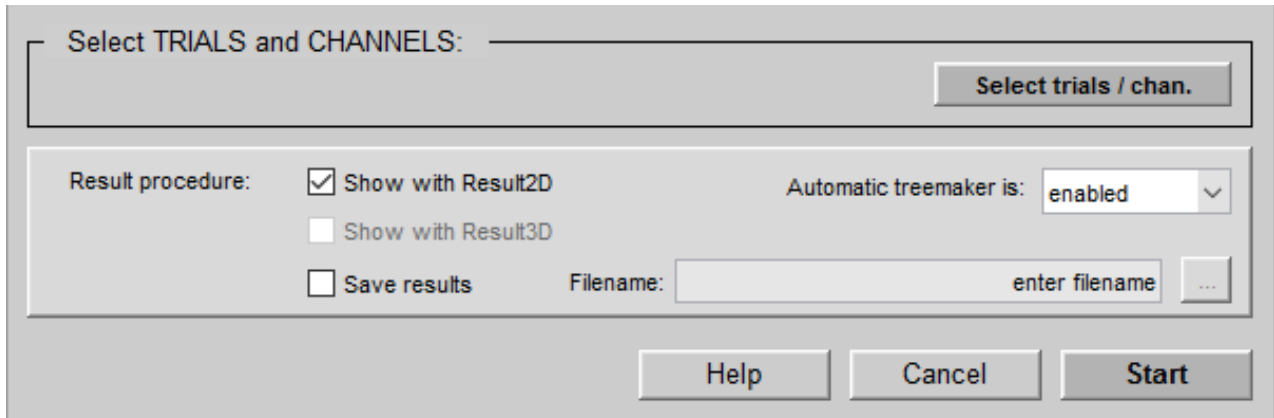
P_C.SamplingFrequency=19200;

% TMS Artifact Removal
PerformCAR = 1;
SingleTrialArtifactRemoval = 0;
TrialExclude = [];
ChannelExclude = [];
FileName = '';
ProgressBarFlag = 1;
P_C = gBStmsartifactremoval(P_C,PerformCAR,...
SingleTrialArtifactRemoval,TrialExclude,ChannelExclude,FileName,...
ProgressBarFlag);
```

Analyze

The **Analyze** menu contains functions which produce plots that are viewed with Result2D.

To create histograms, to average the data, to perform spectral analysis and to make time-frequency plots use



[Data Quality](#) - to analyze the quality of the acquired biosignals

[Average](#) – to average across trials

[Spectrum](#) – to calculate the power spectrum

[Mean Frequency](#) – find the mean frequency and center of gravity of the EEG signal

[ERD](#) – to calculate event-related desynchronization/synchronization (ERD/ERS)

[ERD - Maps](#) – to calculate ERD plots for multiple frequency bands

[Wavelet Transformation](#) – create Scalograms

Spatial patterns can be created with

[CSP](#) – common spatial patterns

[PCA](#) – principal component analysis

[ICA](#) – independent component analysis

[CCA](#) – canonical correlation analysis

To investigate the coupling of channels use

[Coherence](#) – coherence analysis between 2 channels

[Event Related Coherence](#) – event-related coherence analysis between 2 channels

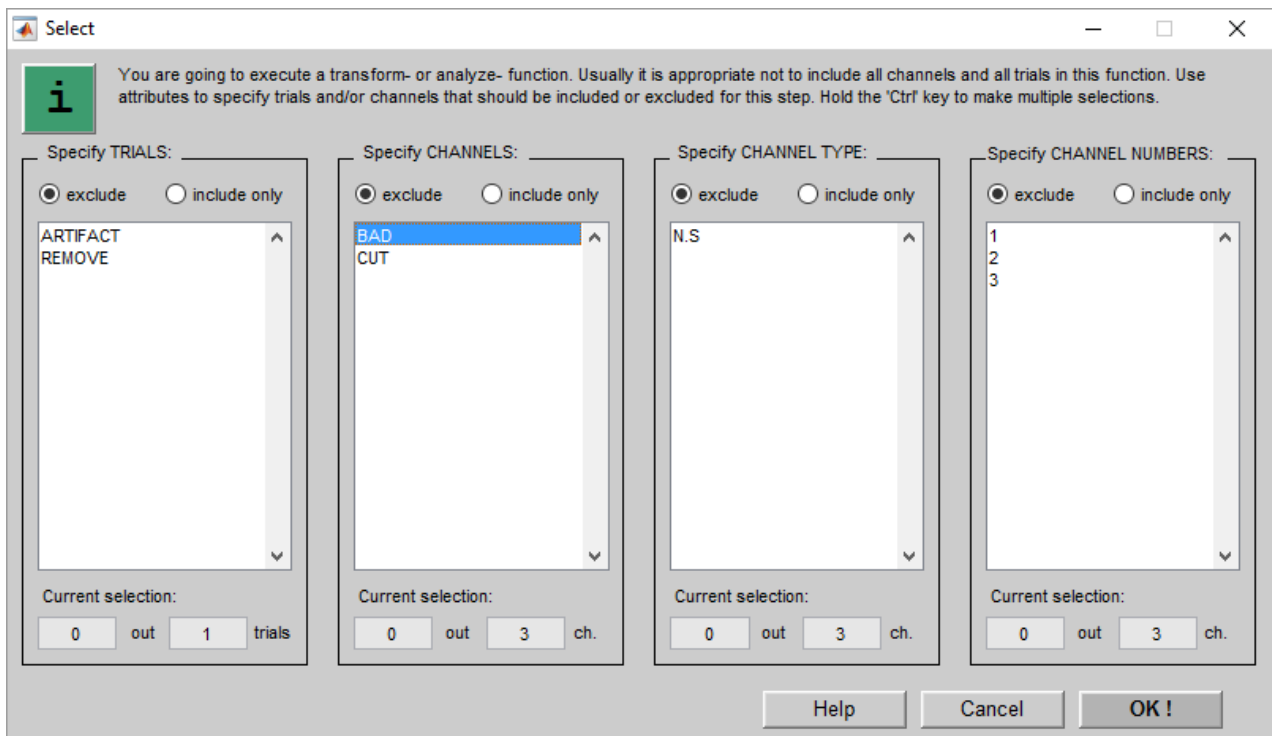
Each analyze window has one section called **Select TRIALS and CHANNELS** and another section called **Result procedure**.

In the case of **Coherence** and **Event Related Coherence** the **Select TRIALS and CHANNELS** field is replaced by a field that allows to define channel-pairs.



Select TRIALS and CHANNELS

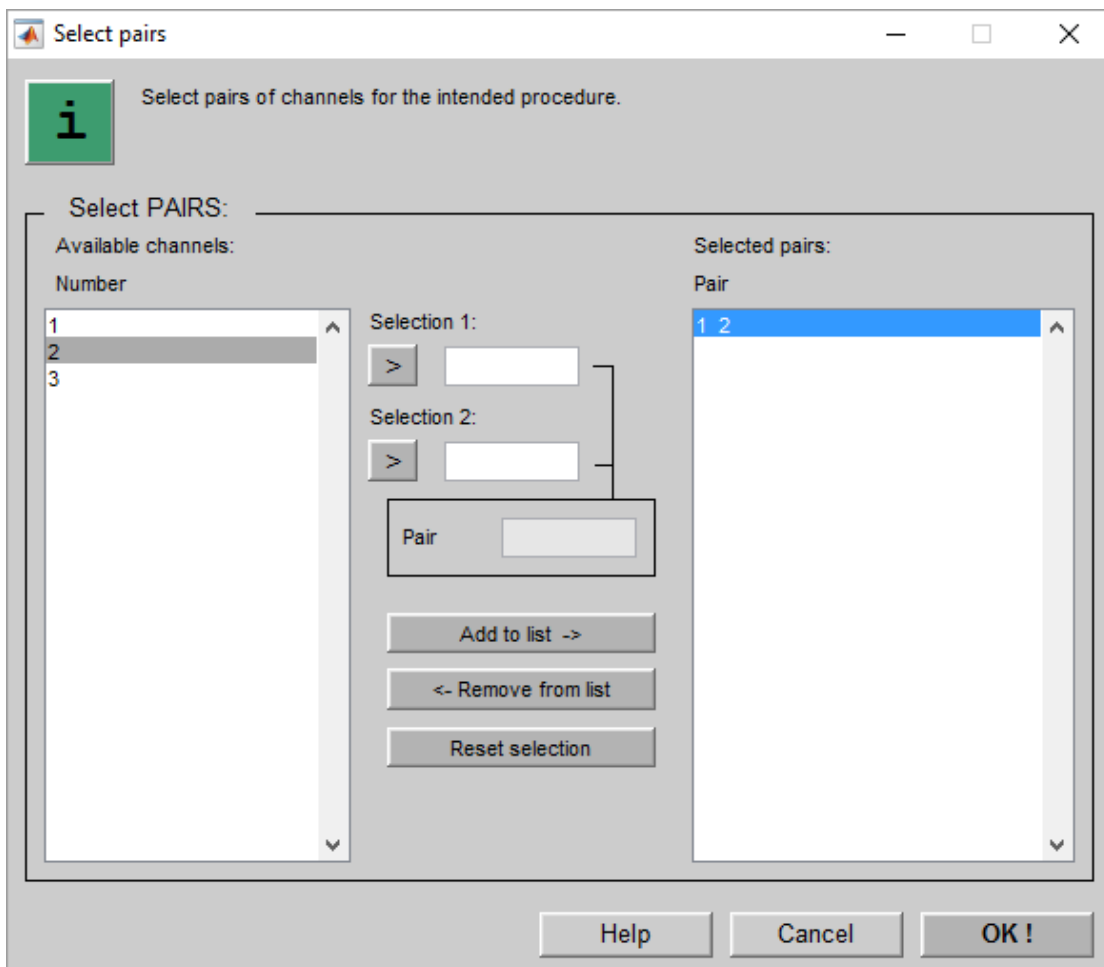
1. Press the **Select trials / chan.** button to exclude/include specific trials and channels for calculation



2. Use the **exclude** radiobutton to mark the attributes that should be excluded from further calculation and the **include only** radiobutton to include specific attributes. The **Current selection** gives a preview of how many channels/trials are affected by your assignment. The example excludes all channels which have the attribute BAD (n=12 out of 16 channels) from further analysis.
3. Press the **OK** button to perform the selection

Select PAIRS

1. Press the **Select pairs** button to define channels pairs for the calculation
2. Select in the **Available channels** listbox the first channel and click on the **Selection 1** button
3. Select channel 2 and click on the **Selection 2** button and confirm the pair with **Add to list**. The new created channel-pair appears in the **Selected pairs** listbox.
4. Close the window with the **OK** button



Result procedure

1. Check the box **Show with Result2D** to plot the results

The screenshot shows a dialog box titled "Result procedure". At the top, there is a text field labeled "Select TRIALS and CHANNELS:" followed by a button labeled "Select trials / chan.". Below this, the "Result procedure:" section contains three checkboxes: "Show with Result2D" (checked), "Show with Result3D" (unchecked), and "Save results" (unchecked). To the right of these checkboxes is a dropdown menu labeled "Automatic treemaker is:" with "enabled" selected. Below the checkboxes is a "Filename:" label followed by a text input field containing "enter filename" and a browse button "...". At the bottom of the dialog are three buttons: "Help", "Cancel", and "Start".

2. If the **Automatic treemaker** is **enabled** the results will be stored in a subdirectory of your data with the name of the analyze function that is performed. If the treemaker is **disabled** the file will be stored to the data directory.
3. By clicking **Save results** a window opens that allows to specify the location for calculation result storage. Use the button beside to browse for a file location.

Data Quality

Artefacts in biosignal data can be caused by saturation effects of the amplifier system. For example subject movement can contaminate the EEG activity on the head and can produce an amplifier range overflow. Also electrode movement, movement of the electrode cable, polarization of the electrodes or a defective electrode can produce overflows.

To reduce the probability of overflows the sensitivity of the amplifier input must be decreased (which means that the input range is increased). In this case the amplitude histogram can be used for the quantification of the dynamic range. The change of the sensitivity changes also the signal-to-noise ratio (SNR). It is important to know that the SNR gets worse by decreasing the sensitivity of the amplifier.

In an European sleep research project the following parameters were used to proof the data quality: minimum and maximum value, variance, skewness, kurtosis, entropy and histogram plots [Schlögl 1999].

The histogram plots are a compressed data representation that can easily be investigated for data quality control. The histogram shows how many samples of a data series have a certain voltage level. An amplifier overflow would cause a high number of samples with the maximum or minimum voltage level of the amplifier input sensitivity.

Assume a time series Y with N samples and a data acquisition board with 16 Bit. Each sample is assigned to a value out of 2^{16} possible ones. The histogram $H(i)$ gives the number of samples with value i . Therefore, a certain value of the time series Y has the probability

$$p(i) = H(i) / N \text{ for a large amount of samples } N.$$

The histogram can be used to calculate the mean

$$\mu = \sum_i (iH(i)) / N$$

and to calculate the variance

$$\sigma^2 = \sum_i ((i - \mu)^2 H(i)) / N$$

Further the skewness is defined as

$$\gamma^3 = \sum_i ((i - \mu)^3 H(i)) / N$$

and the kurtosis as

$$\gamma^4 = \sum_i ((i - \mu)^4 H(i)) / N - 3(\sigma^2)^2$$

as discussed in [Nikias 1993]

The entropy [bits] is given by

$$I = \sum_i p(i) \log_2(p(i))$$

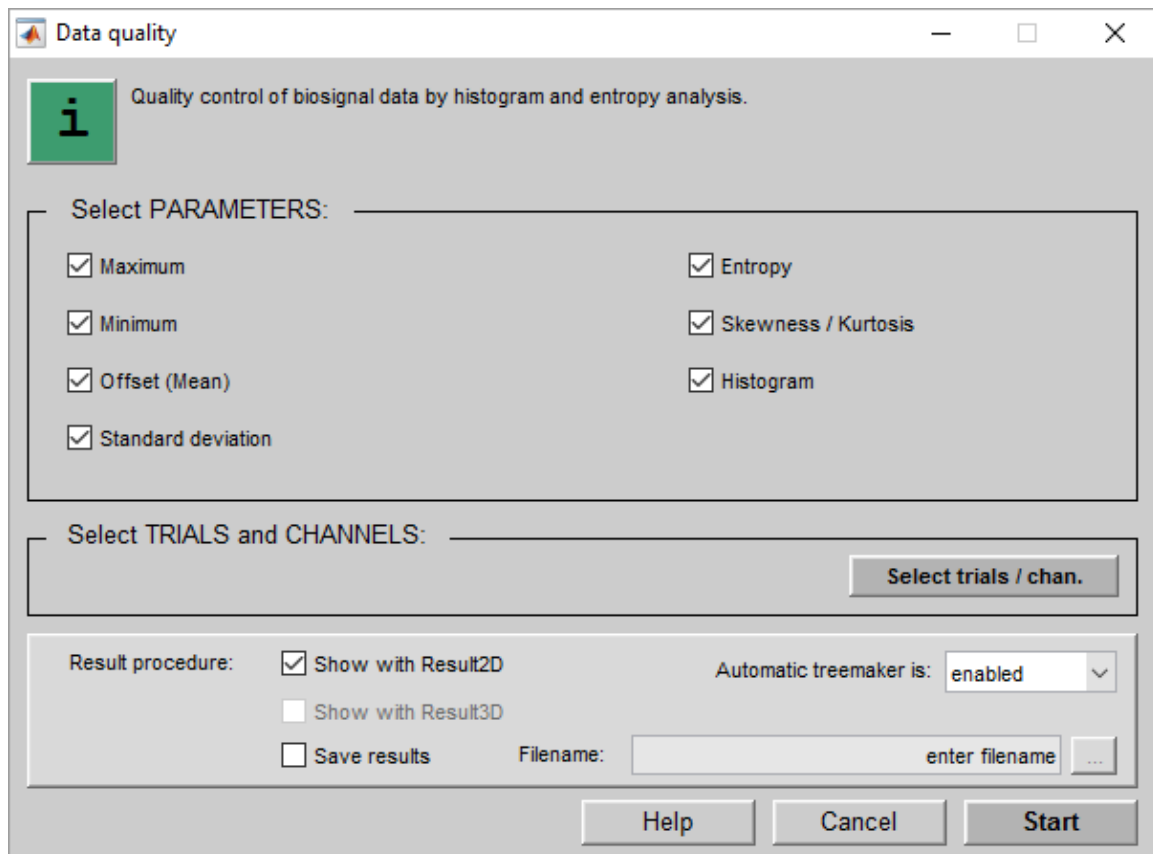
The entropy is maximum for a signal where the histogram is flat. It is e.g. 16 bit if each $p(i)$ is $1/2^{16}$.

[Schlögl 1999] Schlögl, A., Kemp, B., Penzel, T., Kunz, D., Himanen, S.L., Värri, A., Dorffner, G., Pfurtscheller, G., "Quality control of polysomnographic sleep data by histogram and entropy analysis," *Clinical Neurophysiology*, vol. 110, pp. 2165-2170, 1999.

[Nikias 1993] Nikias, C.L., Petropulu, A.P., "Higher-order spectra analysis", Englewood Cliffs, NJ: Prentice Hall, 1993.

Select the following parameters to perform the calculation for each channel:

- **Maximum** - calculate the maximum of each data channel
- **Minimum** - calculate the minimum
- **Offset (Mean)** - calculate the mean
- **Standard deviation** - calculate the standard deviation
- **Entropy** - calculate the entropy [bit]
- **Skewness / Kurtosis** - calculate skewness and kurtosis
- **Histogram / Boxplot** - calculate and plot the histogram



Example

1. Load the data-set `session1.mat` under

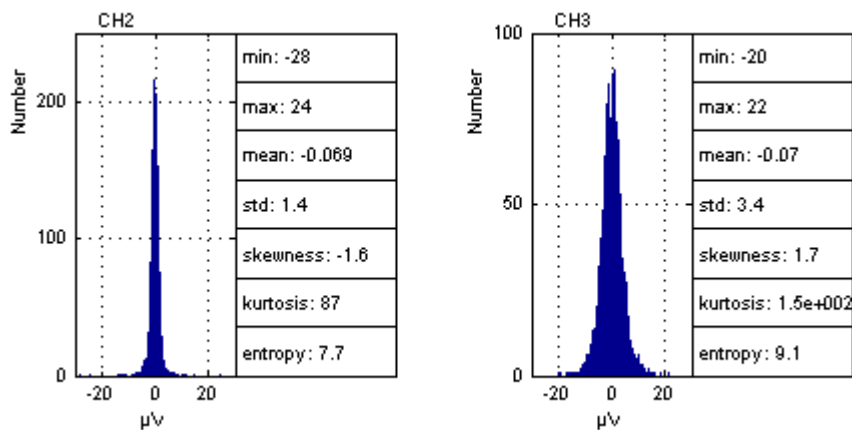
`Documents\gtec\gBSanalyze\testdata\aircraft_simulator`

The data consists of 4 EEG-, 2 EOG-, 1 ECG- and 2 trigger-channels. The data was recorded with a resolution of 12 bit.

2. Click on **Data Quality** under the **Analyze** menu and check all parameters for calculation. Check the **Show with Result2D** box and press the **Start** button to perform the analysis.

Note that the data acquisition board resolution is required for the calculation of the data quality! (12 in this example)

3. Result2D opens automatically with the histograms and data quality parameters. Channels 2 and 3 are presented as examples. The y-axis of the histogram shows the number of samples with a certain quantization value. The x-axis shows the quantization values recalculated to the amplifier input sensitivity.



The histogram analysis allows to calculate the mean, standard deviation, skewness, kurtosis and entropy with very low computing effort. The skewness and kurtosis show the deviation from Gaussian data distribution. The entropy shows how much information of brain processes can be observed by the EEG signal.

The following code shows how to perform the example demonstrated above from MATLAB command line.

%Load Data

```
P_C=data;  
File=['C:\Users\' getenv('USERNAME')  
\Documents\gtec\gBSanalyze\testdata\Aircraft_Simulator\' ...  
'Session1.MAT'];  
P_C=load(P_C,File);
```

%Data Quality

```
P.with_min=1;  
P.with_max=1;  
P.with_mean=1;  
P.with_std=1;  
P.with_skewness=1;  
P.with_kurtosis=1;  
P.with_entropy=1;  
P.with_histogram=1;  
TrialExclude=[];  
ChannelExclude=[9];  
P_C.DAQResolution=12;  
FileName="";  
D_O=gBSdataquality(P_C,P,TrialExclude,ChannelExclude,FileName,0);
```

Average

Evoked potential (EP) tests are used to investigate the condition of the nerve pathways. The brain's electrical response to the signals sent by the nerves can be checked. EP tests help to diagnose nervous system abnormalities, hearing loss, and assess neurological functions.

Major types of evoked potentials:

Brainstem auditory evoked potential (BAEP). Used to check the pathway from the ear to the brain. BAEP tests are used to uncover the cause of hearing and balance problems, and other symptoms.

Visual evoked potential (VEP). Checks the pathway from the eyes to the brain. VEP's help to find the cause of certain vision problems and other conditions.

Somatosensory evoked potential (SSEP). Used to check the pathway from the nerves in the limbs to the brain. It is a way to study the function of the nerves, the spinal cord and brain.

A well known characteristic of the quality of a measurement is the signal-to-noise ratio (SNR), given by the ratio of the signal power over the power of noise [Graben 2001]. The residual noise in evoked potentials is estimated by subtracting separate averages: one consisting of the odd-numbered trials (\bar{x}_1) and the other consisting of the even-numbered trials (\bar{x}_2). The i^{th} trial can be represented as

$$x_i = s_i + n_i$$

consisting of the response s_i and the noise n_i . The noise is uncorrelated to the signal. Calculate

$$\bar{x}_1 = \frac{1}{M} \sum_{i=1, \text{ odd}}^N (s_i + n_i) \qquad \bar{x}_2 = \frac{1}{O} \sum_{i=1, \text{ even}}^N (s_i + n_i)$$

where N is the total number of trials. M is the number of odd trials and O is the number of even trials.

The response terms s_i are identical for odd and even trials and therefore are eliminated by subtracting

$$\bar{x}'_N = \bar{x}_1 - \bar{x}_2$$

This results in an estimation of the noise with zero mean and variance σ_n^2 [van de Velde 2000]. The true signal-to-noise power ratio can be calculated by [Schimmel 1974]

$$SNR = \frac{\text{var}(\bar{x}_{(t)})}{\text{var}(\bar{x}'_{(t)})}$$

The SNR should be computed for evoked potentials for intervals with most consistent response. E.g. for middle latency auditory evoked potentials over 5-15 ms, for long latencies from 5-200 ms.

[Graben 2001] Peter beim Graben, "Estimating and improving the signal-to-noise ratio of time series by symbolic dynamics," *Physical Review*, vol. 64, 2001.

[Schimmel 1974] H., Schimmel, I., Rapin, and M.M., Cohen, „Improving evoked response audiometry with special reference to the use of machine scoring," *Audiology*, vol. 13, pp. 33-65, 1974.

[van de Velde 2000] M., van de Velde, „Signal validation in electroencephalography research, " *PhD thesis. Eindhoven University of Technology, Eindhoven, The Netherlands, 2000.*

Select the following parameters to perform the averaging over trials for each channel:

- **Baseline correction** - enable the baseline correction

Correct either for the **entire trial** or select a specific **reference interval** for the correction.

- **Compute simple average** - compute average over all trials

Add standard deviation time course - add a standard deviation time course

Compute borders – add specific borders to the averaged signal

Compute signal-to-noise ratio – add SNR to each channel

- **Compute split-half averages** – compute average and/or correlation of the first 50 % of trials and compare it to the last 50 % of trials

Compute simple average and split-half correlation – computes the average over all trials. But the correlation is calculated between the first 50 % and the last 50 % of trials.

Compute separate split-half averages and correlation – computes the average and the correlation of the first 50 % and compares it to the last 50 %

- **Compute separate average of the first ...** – compute the average over the first number of trials and compare it with the average of the rest or with all
- **Compute separate averages for odd and even trials** – compute the average of all trials with odd trial-numbers and compare it to the average of all trials with even trial-numbers

- **Compute separate averages for different classes** - compute the average over trials assigned to the selected classes
- **Use statistical analysis** – select to perform a Mann-Whitney U-test test to compare the samples of two selected classes.

Statistical significance threshold: The critical p-value to determine whether the result is statistical significant (default 0.05).

- **Smoothing** - select to smooth the results with the selected algorithm and time window
- **Horizontal averaging** - select to average over consecutive samples and to reduce the number of samples by a specific factor

Average

Average across trials for the selected trials and channels. Averaging results can be immediately viewed with Result2D or saved to disk.

Specify BASELINE CORRECTION:

Baseline correction Correct for entire trial Correct for reference interval from 1 [samples] to 1002 [samples]

0.5 [ms] 501 [ms]

Specify AVERAGING OPTIONS:

Compute simple average Nothing else Compute borders for: +/- 1 standard deviation

Add standard deviation time course Compute signal-to-noise ratio

Compute split-half averages Compute simple average and split-half correlation

Compute separate split-half averages and correlation

Compute separate average of the first 400 trials and compare to the rest ... all

Compute separate averages for odd and even trials

Compute separate averages for different classes

Use statistical analysis Mark areas with p < 0.05

ARTIFACT REMOVE

SMOOTHING / DOWNSAMPLING:

Smoothing average Window: 4 [samples] Horizontal averaging: Factor= 4

Select TRIALS and CHANNELS:

Select trials / chan.

Result procedure: Show with Result2D Show with Result3D Automatic treemaker is: enabled

Save results Filename: enter filename

Help Cancel Start

Example 1

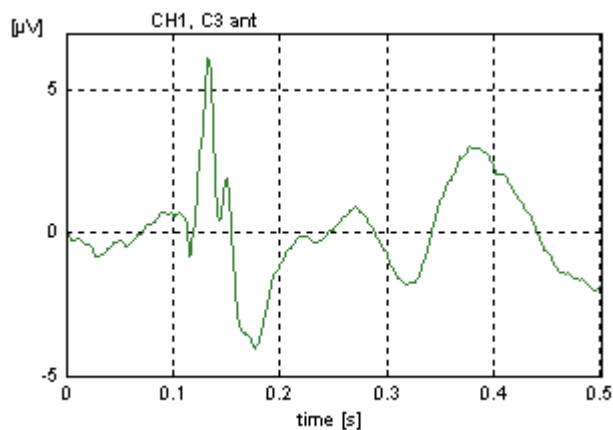
1. Import the data-set `session1.bkr` under

`Documents\gtec\gBSanalyze\testdata\ep`

The data consists of 4 EEG channels recorded anterior to C3, anterior to Cz, posterior to C3 and posterior to Cz.

2. Click on **Average** under the **Analyze** menu and check the **Baseline correction** for the **entire trial** to remove offsets
3. Select **Compute simple average** to perform an averaging over trials for each channel
4. Check the **Smoothing** function and select an **exponential** window. Check the **Show with Result2D** box, enter a filename for result storage and press the **Start** button.

Result2D opens automatically with the evoked potentials. Channel 1 recorded anterior to C3 is presented as example. The y-axis shows the amplitude in μV , the x-axis the time in seconds of the EP.



The following code shows how to perform the example demonstrated above from the MATLAB command line.

```
%Load the demo file
P_C=data;
File=['C:\Users\' getenv('USERNAME')
'\Documents\gtec\gBSanalyze\testdata\EP\session1.bkr'];
P_C=import(P_C, 'BKR', File);

%Perform the EP analysis
Baseline=[0];
Smoothing={'exponential' 4};
DownSampling=0;
TrialExclude=[];
ChannelExclude=[];
FileName=['C:\Users\' getenv('USERNAME')
'\Documents\gtec\gBSanalyze\testdata\ep\average.mat'];
Averaging='simple';
```

```

var1=0;
var2=0;
var3=0;
A_O = gBSaverage(P_C,Baseline,Smoothing,DownSampling,...
TrialExclude,ChannelExclude,FileName,Averaging,0,var1,var2,var3);
result2d = CreateResult2D(A_O);
gResult2d(result2d);

```

Example 2

1. Import the data-set `ep8h.mat` under

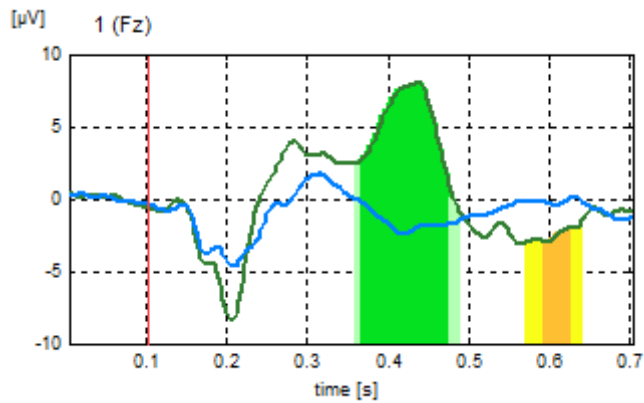
`Documents\gtec\gBSanalyze\testdata\ep`

The data consists of 8 EEG channels recorded from the following positions: Fz, Cz, P3, Pz, P4, PO7, Oz and PO8.

2. Click on **Average** under the **Analyze** menu and check under **Baseline correction** the option to **Correct for reference interval from** 1 to 26 samples.
3. Select **Compute separate averages for different classes**.
4. Select **TARGET** and **NONTARGET** classes in the displayed listbox to perform averaging over trials assigned to these classes.
5. Check **Use statistical analysis** to perform a Mann-Whitney U-test. Type the 0.05 value in the **Mark areas with p <** field.
6. Check the **Show with Result2D** box, enter a filename for result storage and press the **Start** button.

Please note: when triggering and assigning attributes to the data, the **TARGET** attribute must be assigned to the corresponding trials before **NONTARGET**; otherwise, the statistical analysis will not be performed.

Result2D opens automatically with the evoked potentials. Channel 1 recorded from Fz position is presented as example. The y-axis shows the amplitude in μV , the x-axis the time in seconds of the EP. The average over the TARGET trials is plotted with solid dark-green line, and the average over the NONTARGET trials is plotted with solid blue line. The light-green and yellow colors are marking the timepoints when the p-values generated by the Mann-Whitney U-test are lower than the significance value (p), and green and orange colors are marking the timepoints when the results are highly significant, p-values under 0.01.



The following code shows how to perform the example demonstrated above from the MATLAB command line.

```
%Perform the EP analysis with significance test
P_C=data;
File=['C:\Users\' getenv('USERNAME')
'\Documents\gtec\gBSanalyze\testdata\ep\ep8ch.mat'];
P_C=load(P_C,File);

%Perform the EP analysis with statistical significance test
Baseline=[1 26];
Smoothing={'none'};
DownSampling=0;
TrialExclude=[];
ChannelExclude=[];
FileName=['C:\Users\' getenv('USERNAME')
'\Documents\gtec\gBSanalyze\testdata\ep\averages.mat'];
Averaging='different';
Threshold=0.05;
ClassIndex=[3 4];
A_O = gBSaverage(P_C,Baseline,Smoothing,DownSampling,TrialExclude,...
ChannelExclude,FileName,Averaging,0,ClassIndex,Threshold);
result2d = CreateResult2D(A_O);
gResult2d(result2d);
```

Spectrum

A classical way of describing the EEG signal is in terms of frequency by common EEG frequency bands. The most appropriate method is Fourier analysis of specific data segments.

The Power Spectral Density (PSD) of signal vector X is estimated by de-trending and windowing the input signal for each segment that should be investigated. Then the absolute value of the Fast Fourier Transformation (FFT) is squared. The dimension of the power spectra is $\mu V^2/Hz$.

The PSD is computed in a frequency range of 0 to half of the sampling frequency of the data. To decrease the computation time it is possible to down-sample the data beforehand. Note that the frequency range decreases by the down-sampling factor.

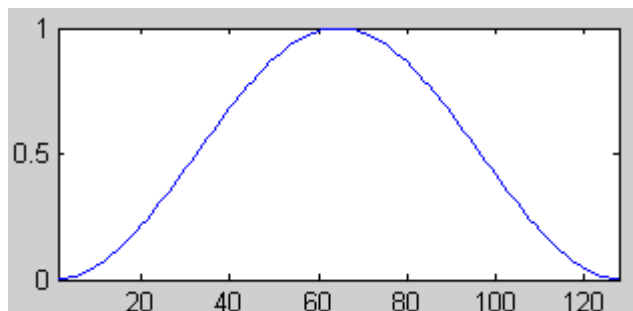
The length of the segment T which should be analyzed must be short enough to avoid non-stationarity and long enough to achieve the desired frequency resolution according to

$$\Delta f = \frac{1}{T}$$

The estimated PSD of one frequency point of one EEG epoch has a chi-square distribution. In order to reduce the variance of the PSD it is necessary to average over a number of equivalent segments or to perform a smoothing over adjacent frequency components. The estimate of PSD of one frequency point and one EEG epoch has only 2 degrees of freedom. Generally, the spectral estimation should have at least 60 degrees of freedom (Vos, 1975) to achieve an acceptable spectral estimation. Therefore, at least 30 trials should be used for the analysis.

The Hanning window decreases triggering effects caused by cutting out specific reference and action intervals of the data and performing the PSD analysis. Each sample in the time window is then multiplied by the corresponding weight from the Hanning window.

To investigate the windowing function use e.g.: $W=HANNING(N)$ to return the N -point symmetric Hanning window in a column vector and $PLOT(W)$ to show the windowing function. N corresponds to the interval in samples.



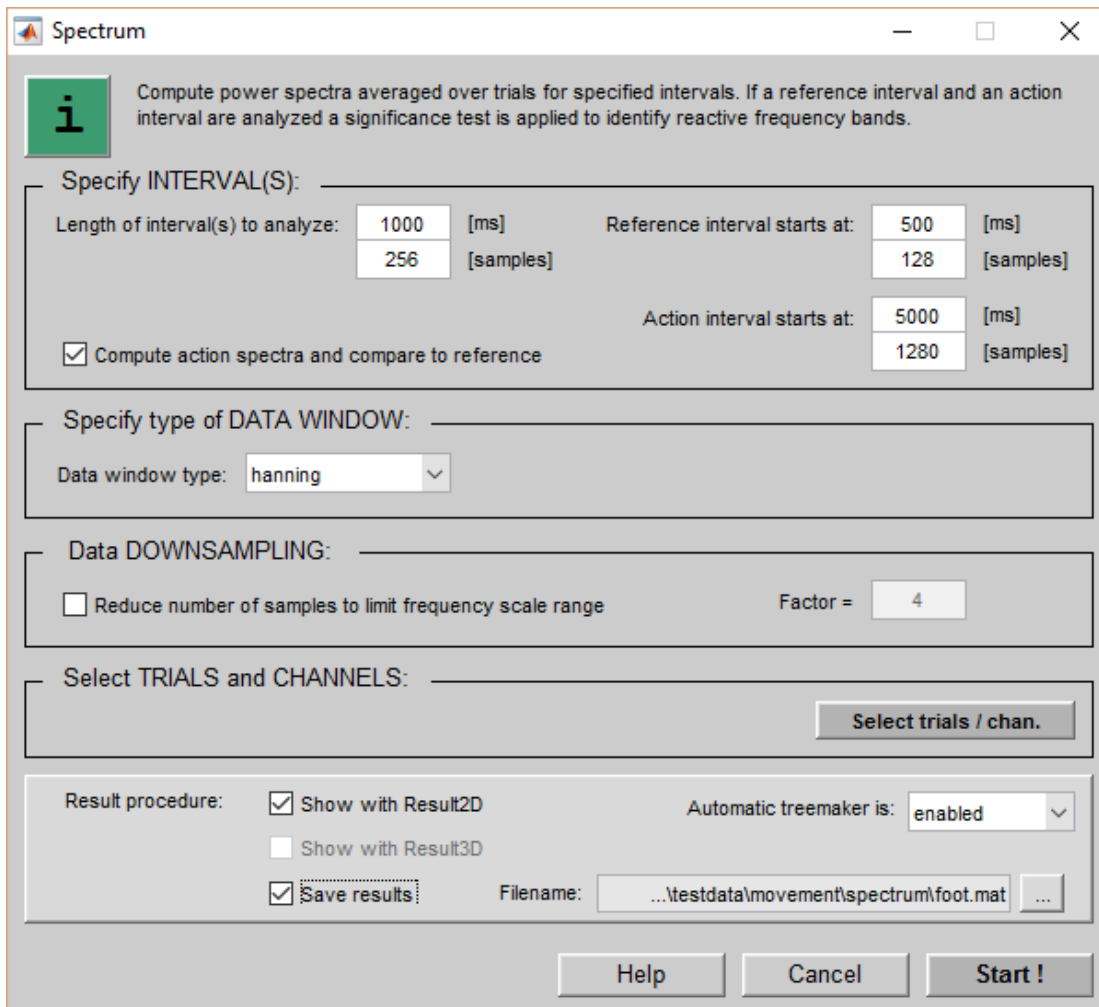
References:

G. Pfurtscheller, "Quantification of ERD and ERS in the time domain". *Handbook of Electroencephalography and Clinical Neurophysiology*, vol. 6, G. Pfurtscheller and F.H. Lopes da Silva, Elsevier, 1999.

Vos, J.E., "Representation in the frequency domain of nonstationary EEGs. In *CEAN – Computerized EEG Analysis*, Ed., G. Dolce and H. Küinkel, pp. 41-50. Stuttgart: Fischer, 1975.

Select the following parameters to perform the calculations for each channel:

- **Length of interval(s) to analyze** – specify the length of the reference and action interval in ms or samples
- **Reference interval starts at** - insert the start point of the reference interval in ms or samples
- **Action interval starts at** - insert the start point of the action interval in ms or samples
- **Compute action spectra and compare to reference** – check the box to calculate the spectrum of the reference interval and of the action interval and compare the results
- **Data window type** - select the data window type for action and reference interval. Window type can be *boxcar*, *hamming* or *hanning*.
- **Reduce number of samples to limit frequency scale range** - select to reduce the number of samples and to reduce the frequency scale range by the specified factor.



Example

To determine the reactive frequency bands a comparison of the power spectra of two 1 second intervals is performed. One spectrum is calculated from a reference interval which lies some seconds before an event and an active interval which is e.g. an interval where a movement imagination took place. The difference between these two power spectra can be used to find reactive frequency bands which show a power increase or decrease related to the reference period.

1. Import the data-set `foot.bkr` under

`Documents\gtec\gBSanalyze\testdata\movement`

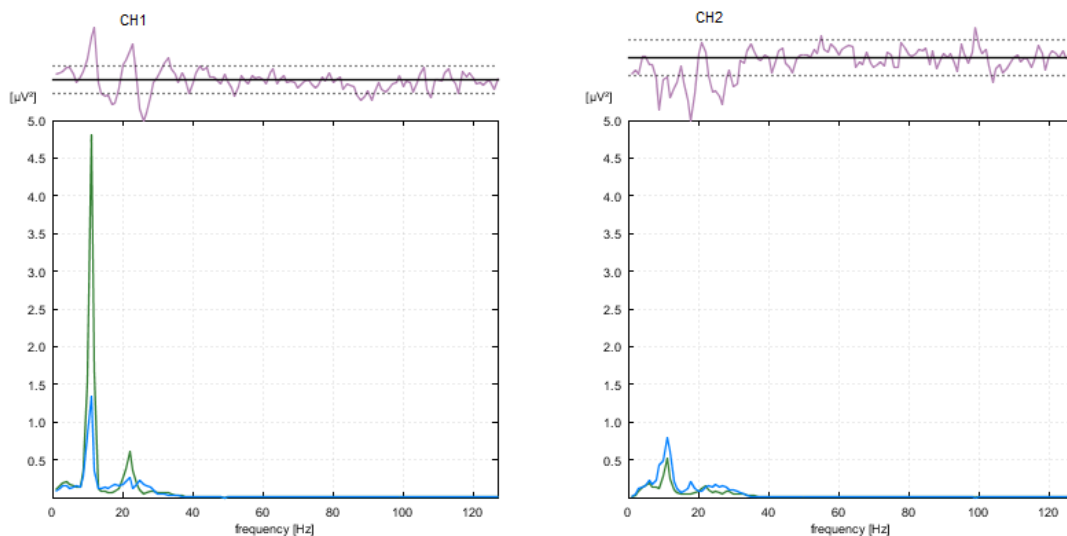
The data consists of 2 local average reference (LAR) derivations recorded over C3, and Cz during foot movement imagination.

2. Click on **Spectrum** under the **Analyze** menu to calculate the spectrum for all trials and to average the spectral information over trials
3. Set the **Length of interval(s) to analyze** to 1000 ms
4. Insert **Reference interval starts** at 500 ms and **Action interval starts** at 5000 ms. This means that the FFT for the reference interval will be computed from 500 ms to 1500 ms

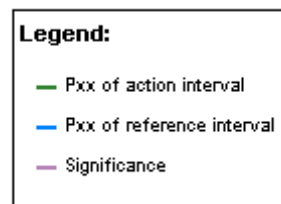
and for the action interval from 5000 ms to 6000 ms.

5. Select a `hanning` window in the **Data window type** pull-down menu
6. Check the **Show with Result2D** box and specify a filename `foot.mat`. The **Automatic treemaker** stores the results into the spectrum subdirectory.
7. Press the **Start** button

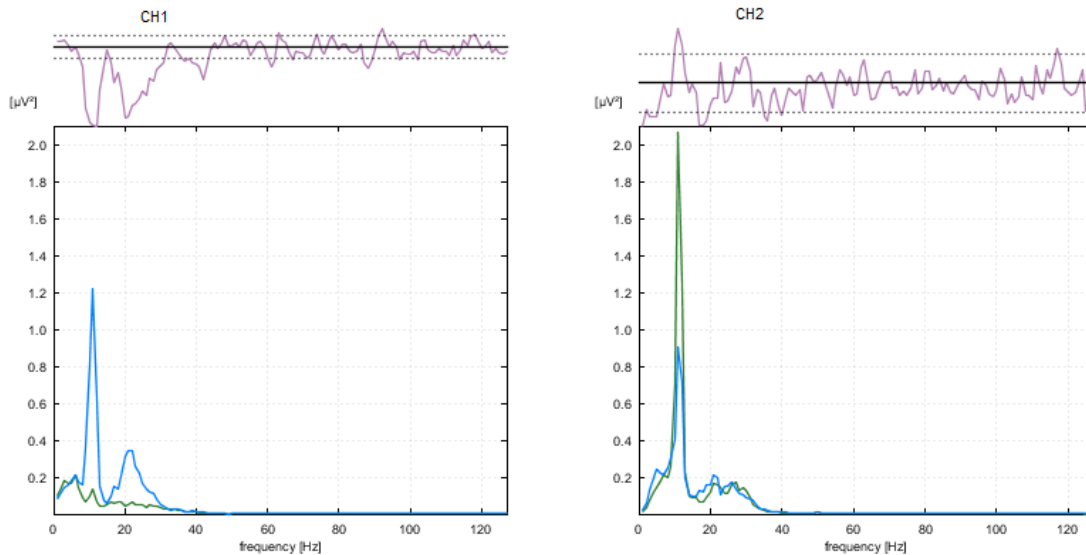
Result2D opens automatically with the spectral information for reference and active intervals. The y-axis shows the power spectrum in μV^2 , the x-axis the frequency in Hz. On top of each axis the significance of the difference of the 2 spectra is shown. If the difference of the two spectra is significant (95% significance level) then the solid line crosses the dashed lines.



During foot movement significant differences can be found over C3 and Cz in the alpha and beta frequency ranges.



8. Import the data-set `right.bkr` under `Documents\gtec\gBSanalyze\testdata\movement`. The data consists of 2 local average reference (LAR) derivations recorded over C3, and Cz during right hand movement imagination.
9. Perform steps 2 to 7 to get the following power spectrum for right hand movement.



The power spectrum over C3 (right hand representation area) shows a clear amplitude attenuation during right hand movement in the alpha and beta frequency ranges. Over Cz the amplitude in the alpha range increases.

The following code shows how to perform the example demonstrated above from the MATLAB command line.

```
%Load the demo file
P_C=data;
File=['C:\Users\' getenv('USERNAME')
'\Documents\gtec\gBSanalyze\testdata\movement\foot.bkr'];
P_C=import(P_C, 'BKR', File);

%Perform the spectral analysis
ActionBegin=1280;
RefBegin=128;
IntervalLength=256;
Window='hanning';
DownSampling=0;
TrialExclude=[];
ChannelExclude=[];
FileName= ['C:\Users\' getenv('USERNAME')
'\Documents\gtec\gBSanalyze\testdata\movement\spectrum\foot.mat'];
S_O=gBSspectrum(P_C, ActionBegin, RefBegin, IntervalLength, Window, ...
DownSampling, TrialExclude, ChannelExclude, FileName, 0);
```

Mean Frequency

EEG oscillations in specific frequency bands reflect e.g. motor tasks, cognitive tasks, memory performance,... The frequency windows for alpha, beta or theta oscillations can be adjusted for each subject by calculating the individual mean frequency.

The alpha oscillation is the dominant frequency in the human EEG and can be described by the peak or gravity frequency in a specific frequency range (f1 and f2). Therefore the power spectrum is calculated for each subject between f1 and f2. Then the peak alpha frequency (PAF) measures the frequency with the highest amplitude. In contrast, the individual alpha frequency (IAF) measures the center of gravity in the range f1 to f2.

The PAF and IAF measures are estimated by de-trending and windowing the input signal for each segment that should be investigated. Then the Fast Fourier Transformation (FFT) is calculated and averaged over the segments with a specific overlap. This signal is again averaged over all trials. Then, the maximum frequency and the center of gravity in the range f1 and f2 are calculated.

g.BSanalyze uses the terms peak frequency (PF) and gravity frequency (GF) instead of PAF and IAF to allow also the calculation of the dominant frequencies in the theta or beta range.

Select the following parameters to perform the calculations for each channel:

Specify DATA INTERVAL

Start interval at - specify the start time point for the analysis

End at - specify the end time point

Specify type of DATA WINDOW

Data window type - the windowing function can be *boxcar*, *hamming* or *hanning*

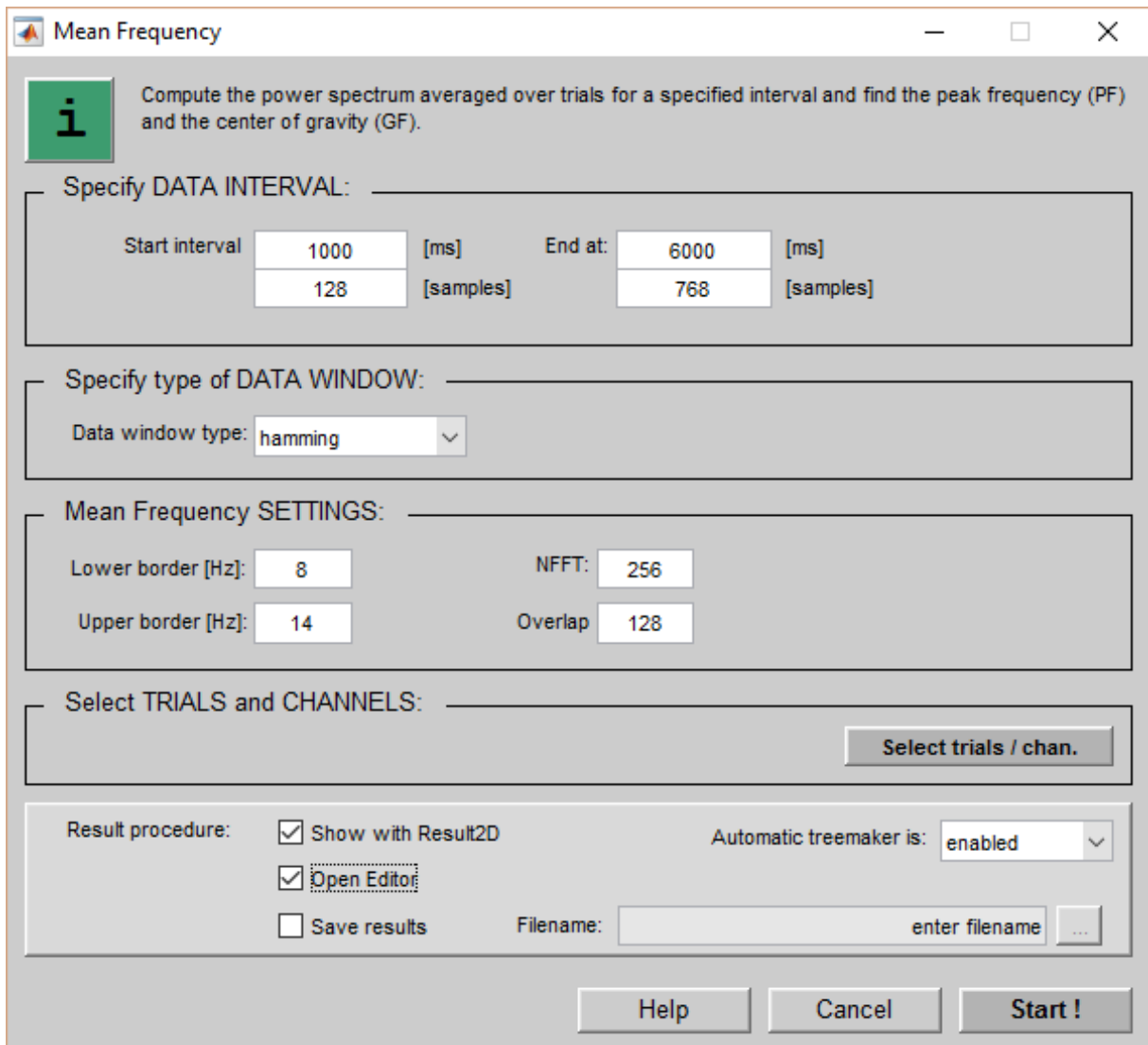
Mean Frequency SETTINGS

Lower border [Hz] - define the lower border for the PF and GF calculation

Upper border [Hz] - define the upper border for the calculation

NFFT - number of FFT points for the power spectrum estimation

Overlap - define the segment overlap of the segments used for the FFT



Example

To calculate the peak frequency and the center of gravity frequency of a triggered data-set perform the following steps:

1. Load the data-set `trig1.mat` under

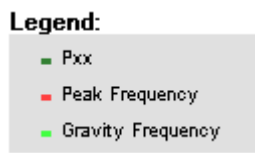
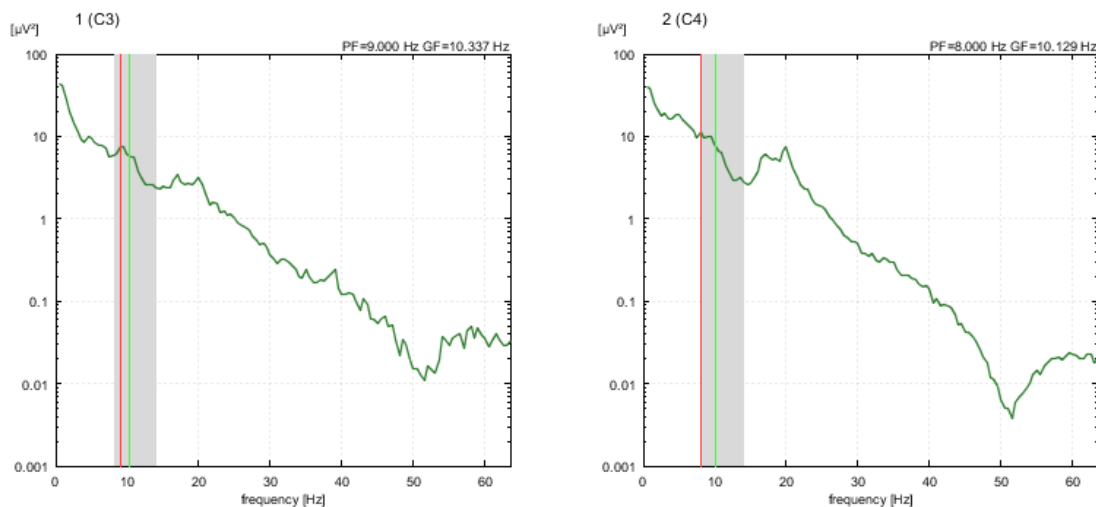
`Documents\gtec\gBSanalyze\testdata\bci`

The data consists of 2 EEG derivations recorded over C3 and C4 during left and right hand movement imagination.

2. Click on **Mean Frequency** under the **Analyze** menu to calculate the spectrum for all trials, to average the spectral information over trials and to find the PF and GF.
3. Set **Start interval at** to 1000 ms and **End at** to 6000 ms

4. Select a `hamming` window in the **Data window type** pull-down menu
5. Set the **Lower** and **Upper border** to a range of 8 to 14 Hz
6. Enter under **NFFT** 256 to perform a 256 point FFT. This gives a frequency resolution of 0.5 Hz.
7. Enter a segment **Overlap** of 128 samples
8. Select channels 1 and 2 for the operation by clicking on the **Select trials / chan.** button
9. Check the **Show with Result2D** and **Open Editor** box
10. Press the **Start** button

gResult2D opens automatically with the PSD plot and the PF and GF. The red vertical line indicates the PF and the green line the GF. Channel 1 has a peak frequency of 9 Hz and a GF of 10.337 Hz. Channel 2 has a PF of 8 Hz which is exactly the lower border used for the maximum calculation. The GF is 10.129 Hz. Note that PF can only have values with steps of 0.5 Hz, but GF has a resolution below the theoretical frequency resolution of the FFT method. The gray shaded segment indicates the frequency range used to find PF and GF.



The MATLAB Editor opens with the PF and GF values for each channel. Additionally, some statistical values calculated of all channels are given: mean over all channels, median over all channels and the standard deviation over all channels. If the PF or GF value lies below or above the frequency range the channel number is also given.

```
Mean Frequency
Interval: 128 - 768 samples
NFFT: 256
Overlap: 128
Window: hamming
Border: 8.000 - 14.000 Hz
Frequency Resolution: 0.500 Hz
Number trials: 40
Number channels: 2

Peak Frequency [Ch1 Ch2 ... ChN] in Hz
9.000 8.000

Gravity Frequency [Ch1 Ch2 ... ChN] in Hz
10.337 10.129

Statistic of all channels:

Peak Frequency Mode
Mean [Hz]:8.500
Median [Hz]: 8.500
Std:0.707
Channels below or equal to lower border:
2.000
Channels above or equal to upper border:

Gravity Frequency Mode
Mean [Hz]: 10.233
Median [Hz]:10.233
Std:0.147
Channels below or equal to lower border:

Channels above or equal to upper border:
```

The following code shows how to perform the example demonstrated above from the MATLAB command line

%Load Data

```
P_C=data;
File= ['C:\Users\' getenv('USERNAME')
'\Documents\gtec\gBSanalyze\testdata\BCI\trig1.mat'];
P_C=load(P_C,File);
```

%Select Trials and Channels

```
trial_id=[];
channel_id=[];
type_id=[];
channelnr_id=[1 2];
flag_tr='tr_exc';
flag_ch='ch_exc';
flag_type='type_exc';
flag_nr='nr_inc';
[TrialExclude,
ChannelExclude]=gBSselect(P_C,trial_id,flag_tr,channel_id,flag_ch,...
type_id,flag_type,channelnr_id,flag_nr);
```

%Mean Frequency

```
Interval=[128 768];
Window='hamming';
Border=[8 14];
NFFT=[256];
Overlap=[128];
TrialExclude=[];
ChannelExclude=[3];
FileName='';
M_O=gBSmeanfrequency(P_C,Interval,NFFT,Overlap,Window,Border,...
TrialExclude,ChannelExclude,FileName,0);
```

ERD

Event-related synchronizations and event-related desynchronization (ERS/ERD) in the classical sense are calculated by first digitally bandpass filtering (with a selected lower and upper cutoff frequency) the data resulting in x_f , squaring each sample and averaging over several trials M (method: mean) to give a time course of band-averaged power as shown in the figure below [Pfurtscheller and Aranibar 1979, Pfurtscheller 1999].

$$\bar{P}_{(i)} = \frac{1}{M} \sum_{m=1}^M x_{f(m,i)}^2 \quad (1)$$

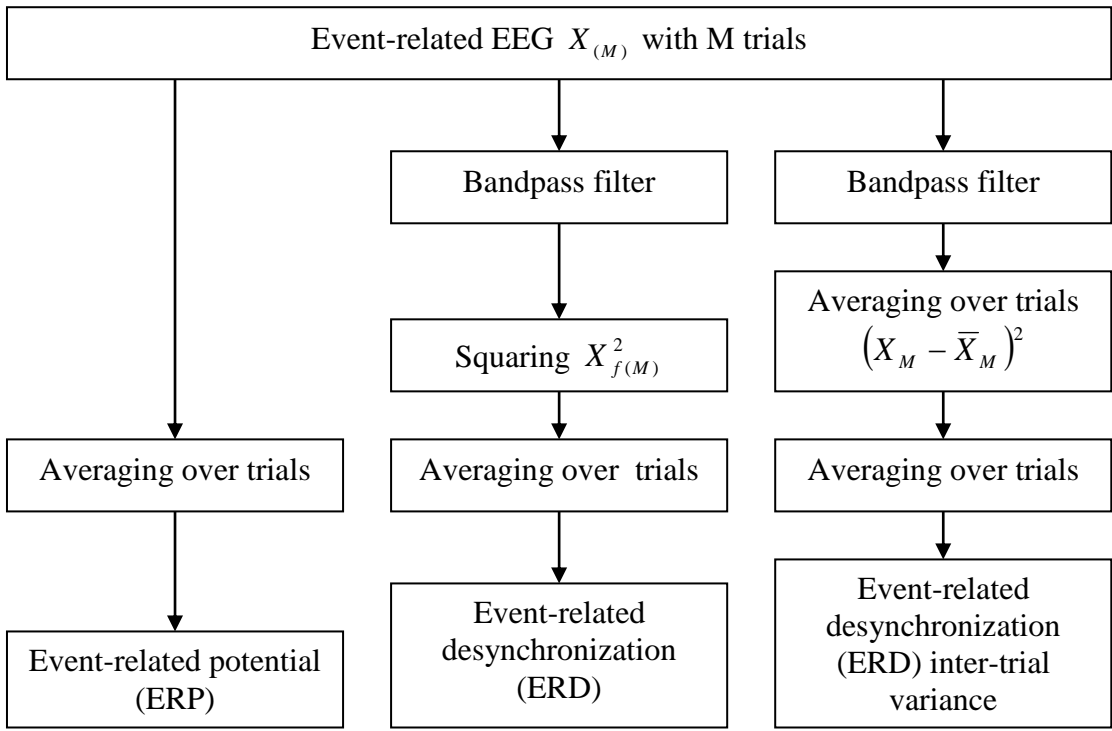
$\bar{P}_{(i)}$ is the band-averaged power (absolute power) for sample i of the data-set. To obtain absolute amplitudes the square root of the power is taken.

In contrast event-related potentials (ERPs) are obtained by averaging over M trials of the raw signal.

In a slightly modified version referred as inter-trial variance by Pfurtscheller [Kalcher and Pfurtscheller 1995] the average across trials is subtracted from the bandpass filtered signal before squaring according to the following equation

$$PIV_{(i)} = \frac{1}{M-1} \sum_{m=1}^M \{x_{f(m,i)} - \bar{x}_{f(i)}\}^2 \quad (2)$$

Therefore, the classical method considers induced and evoked components, while the inter-trial variance method considers only induced components.



To reduce the variance of the power estimates, consecutive power samples are averaged, resulting in a temporal sequence with reduced temporal spacing, e.g. at a sampling frequency of 128 Hz, often 16 consecutive values are averaged to give a temporal spacing of 125 ms. This 125 ms spacing does not correspond to the time resolution. The time resolution is determined by the frequency band of the filter, there being an inverse relationship between the bandwidth of the filter and the effective time resolution.

The mean band power in the reference interval (R) at the beginning of the trial is determined and the percentage power change of the active time point Act(i), relative to the reference power is calculated as

$$ERD_{(i)} = \frac{Act_{(i)} - R}{R} * 100 \text{ in \%} \tag{3}$$

The significance of bandpower decrease (increase) for sample i is calculated with a sign test according to the following equation

$$P_{(i)} = (0.5)^M \sum_{m=1}^{k_{(i)}} \binom{M}{m} \tag{4}$$

At each sample i the sign (positive or negative) of the power change in each single trial relative to the average power in the reference period is detected. When there is no significant change in power then on average the number of positive and negative signs across the M trials should be equal. But if there is a significant ERD, the majority of signs will be positive.

z-Transformation of the band power values

For z-transformed ERD quantification spectral power in specific frequency bands x at sample i are transformed to z-values by the following equation

$$z_i = \frac{x_i - \bar{x}}{\text{std}(x)} \quad (5)$$

where \bar{x} is the mean of the current trial and $\text{std}(x)$ is the standard deviation. Then these normalized values are averaged over trials. T-tests are used to find significant differences between conditions [Klimesch 1998]. A z-value of +1.6 means that the measured value is 1.6 times higher as the mean value.

Kalcher, J., Pfurtscheller, G., „Discrimination between phase-locked and non-phase-locked event-related EEG activity,” Electroencephalogr. Clin. Neurophysiol., vol. 94, pp. 381-384, 1995.

Klimesch, W., Russegger, H., Doppelmayr, M., Pachinger, T., “A method for the calculation of induced band power: implications for the significance of brain oscillations. Electroencephalogr. Clin. Neurophysiol., vol. 108 (2), pp. 123-130, 1998.

Pfurtscheller, G., Aranibar, A., “Evaluation of event-related desynchronization (ERD) preceding and following voluntary self-paced movements,” Electroencephalogr. Clin. Neurophysiol., vol. 46, pp. 138-146, 1979.

Pfurtscheller, G., “Quantification of ERD and ERS in the time domain”. Handbook of Electroencephalography and Clinical Neurophysiology, vol. 6, G. Pfurtscheller and F.H. Lopes da Silva, Elsevier, 1999.

Select the following parameters to perform the calculation for each channel:

- **Start of reference period at** - insert the start point of the reference interval in ms or samples
- **End of reference period at** - insert the end point of the reference interval in ms or samples
- **No filter** – do not filter the data
- **Use filter** - select a predefined filter
Design new filter – click to create a new lowpass, highpass, bandpass or bandstop filter
- **Induced components only** - only non-phase locked components. Phase locked components will be removed (see Equation 2).
- **Raw signal** – non-phase locked and phase locked components will be analyzed (see Equation 1).
- **Result scaling**
 Relative power change in % (Equation 3)
 Absolute power
 Absolute amplitude
 z-Values (Equation 5)
- **Use the 'Envelope method'** – calculates the Hilbert envelope after filtering and squaring the data
- **Horizontal averaging** - average over consecutive samples
averaging method – can be mean or median
factor – number of samples to average
- **Significance test** – perform the significance test
- **Smoothing** – smooth the result
 average – average over the specified **window** length in ms
 exponential - exponential window with **window** length in ms
 cosine - cosine window with a window length of 3 samples

ERD / ERS

ERD / ERS (event-related desynchronization / synchronization) is characterized by changes of signal power over time (relative to values in a reference period). To design filters use the "Design filter" function.

Specify REFERENCE PERIOD:

Start of reference period at: 500 [ms] / 128 [samples] End of reference period at: 1500 [ms] / 384 [samples]

Select FILTER:

No filter Use filter: ALPHA1 / BP / 10 / 12 / fft / 0 Design filter ...

name / type / f (lo) / f (hi) / real. / order

Specify COMPONENTS to be analyzed:

Induced components only (none phase locked) Raw signal (induced and evoked components)

SCALING of result:

Result scaling: Relative power change [%] Use the "Envelope method"

AVERAGING, SMOOTHING and STATISTICS:

Horizontal averaging Method: mean Factor: 16

Significance test: sign test Smoothing: average Window: 125 [ms]

Select TRIALS and CHANNELS:

Select trials / chan.

Result procedure: Show with Result2D Automatic treemaker is: enabled

Show with Result3D

Save results Filename: ...nalyze\testdata\movement\erd\foot.mat

Help Cancel Start

Example

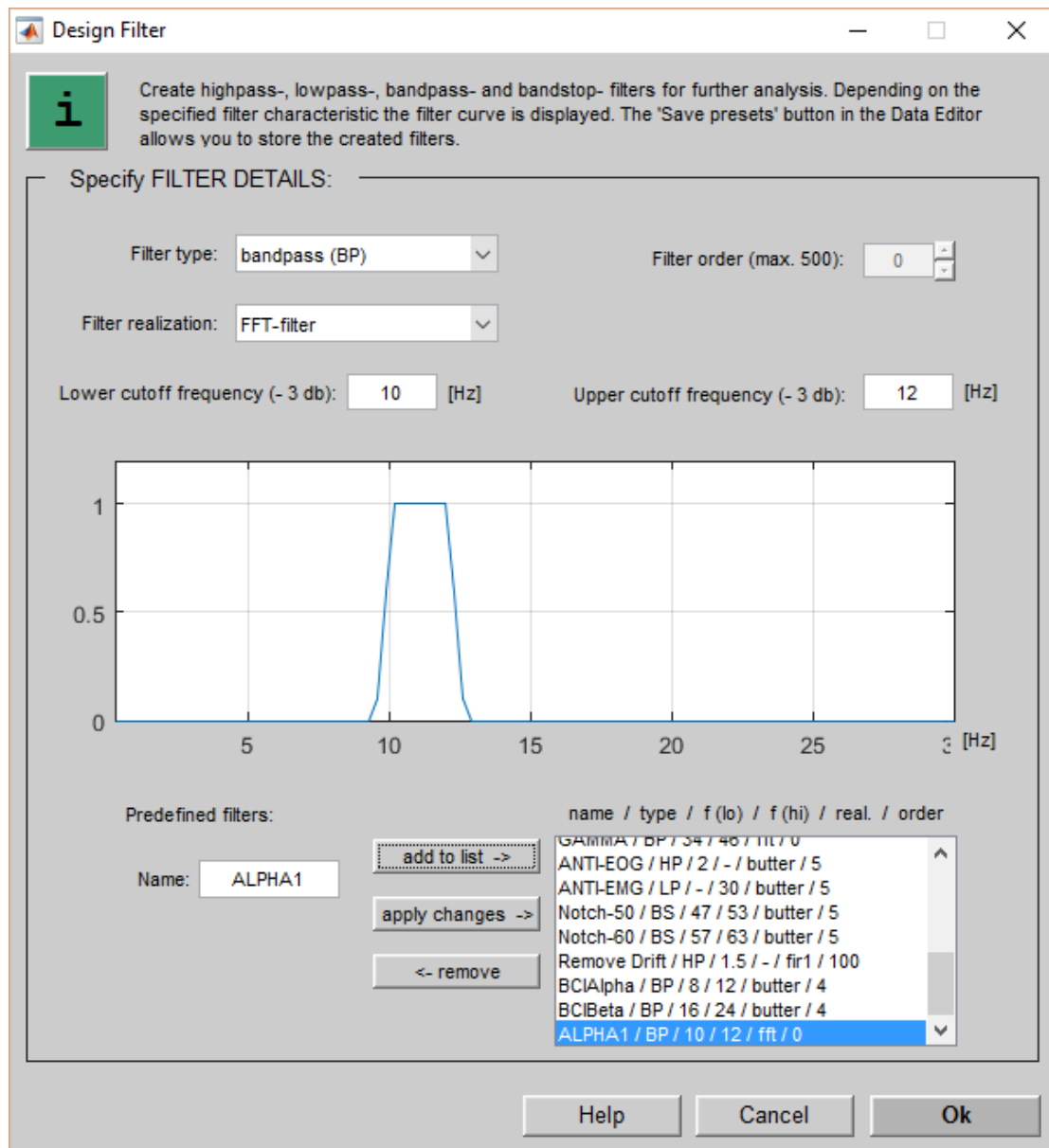
1. Import the data-set `foot.bkr` under

`Documents\gtec\gBSanalyze\testdata\movement`

The data consists of 2 local average reference (LAR) derivations recorded over C3, and Cz during foot movement imagination.

2. Click on **ERD** under the **Analyze** menu to calculate the event-related desynchronization

3. Set **Start of the reference period** to 500 ms and **End of reference period** to 1500 ms
4. Click on **Design filter** to open the following window:

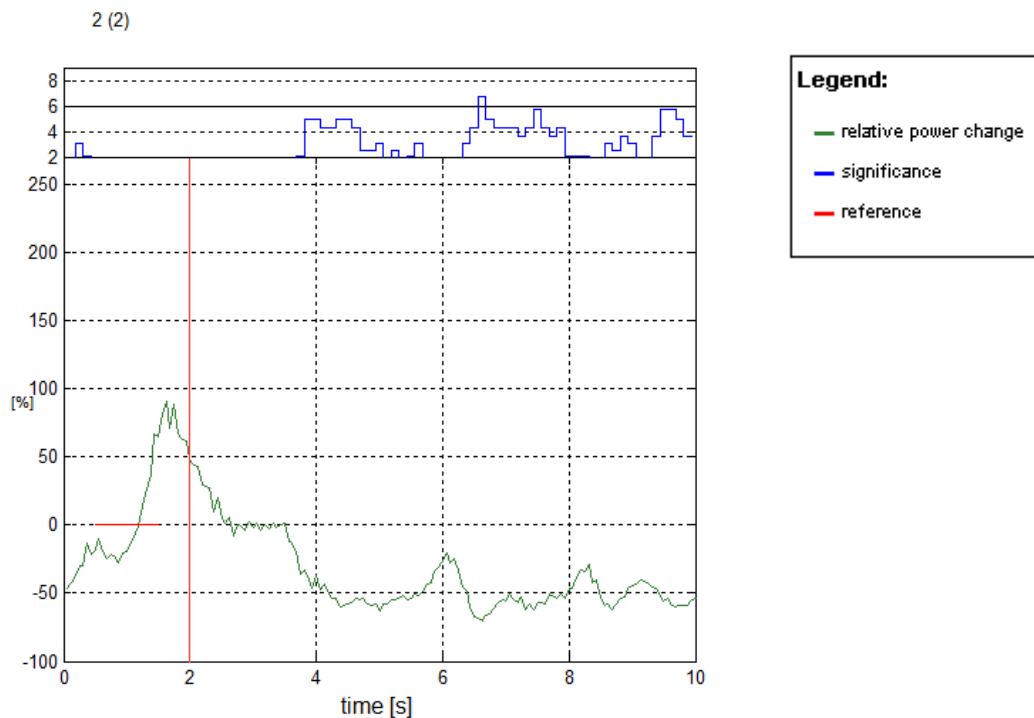


Select the **Filter type** `bandpass` and **Filter realization** `FFT`. Specify a **Lower cutoff frequency** of 10 Hz and a **Upper cutoff frequency** of 12 Hz. Enter the **Name** `ALPHA` and press **add to list** to create a new filter for the alpha frequency range. Press **OK** to close the window.

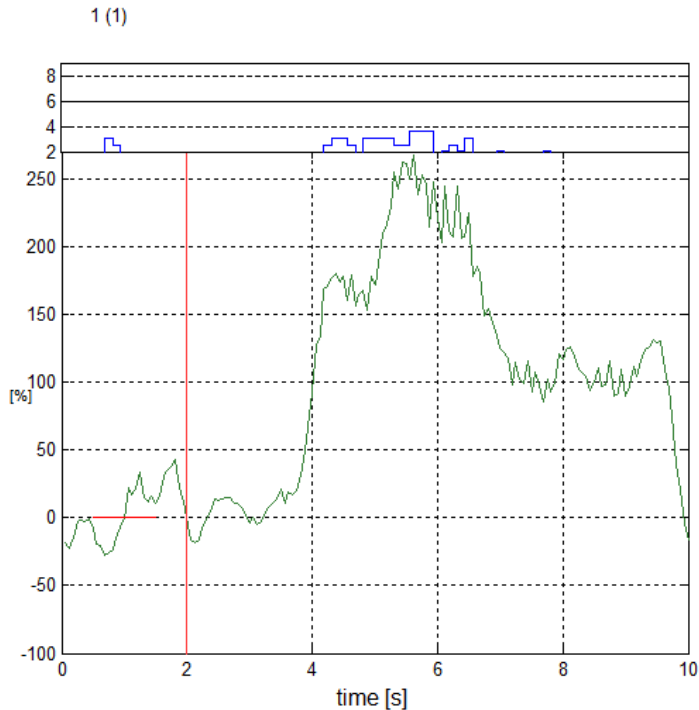
5. Check **Induced components only** to remove phase locked components
6. Select **horizontal averaging** to calculate the mean over 16 consecutive samples
7. Enable the **sign test**

8. Check the **Show with Result2D** box and specify a filename `foot.mat`. The **Automatic treemaker** stores the results into the `erd` subdirectory.
9. Click on **Start**

Result2D opens automatically with the ERD time courses. The y-axis shows the power change in % referred to the reference interval indicated by the red line. The x-axis shows the time in seconds. The significance of the ERD is shown on the top of each plot. A value of 2 corresponds to an error probability of 10^{-2} , a value of 8 to 10^{-8} (highly significant).



ERD time course over Cz during foot movement imagination in the alpha frequency range from 10 to 12 Hz, averaged over 35 trials. The cue is presented to the subject at second 3 and the subject had the task to imagine the movement until second 10. A highly significant ERD can be observed from second 3.5 until second 10.



ERD time course over C3 during foot movement imagination. The amplitude attenuation over Cz is accompanied by an amplitude enhancement (ERS) over the right hand representation area.

%Import the demo file

```
P_C=data;
File= ['C:\Users\' getenv('USERNAME')
'\Documents\gtec\gBSanalyze\testdata\movement\foot.bkr'];
P_C=import(P_C, 'BKR', File);
```

%Perform the ERD calculation

```
FileName=['C:\Users\' getenv('USERNAME')
'\Documents\gtec\gBSanalyze\testdata\movement\erd\foot.mat'];
Reference=[128 384];
UsedComponents=1;
Filter.Realization='fft';
Filter.Type='BP';
Filter.Order=1;
Filter.f_high=12;
Filter.f_low=10;
ResultScaling=1;
Average={'mean' 16};
Smoothing={'none'};
Significance='sign';
UseEnveloped=0;
TrialExclude=[];
ChannelExclude=[];
E_O=gBSerd(P_C, UsedComponents, Filter, Reference, ResultScaling, ...
Average, Smoothing, Significance, UseEnveloped, TrialExclude, ...
ChannelExclude, 0);
```

ERD - Maps

ERDmaps allows to calculate ERDs and ERSs for multiple frequency bands and to present the results in a single colormap per channel [Graumann 2002]. This is a very effective method for finding event-related changes of oscillatory brain activity. The method is of special importance for data-sets with a high amount of channels to find easily reactive channels and reactive frequency bands.

ERDmaps can be calculated without significance statistic (fast) or with a bootstrap algorithm. When using the bootstrap algorithm it is possible to define a significance level. Then, ERDs and ERSs are only displayed if they are within the specific confidence level.

B. Graumann, J.E. Huggins, S.P. Levine, G. Pfurtscheller, "Visualization of significant ERD/ERS patterns in multichannel EEG and ECoG data," Clinical Neurophysiology, vol. 113, pp. 43-47, 2002.

The window allows the following settings:

Specify REFERENCE PERIOD:

- **Start of reference period at** - insert the start point of the reference interval in ms or samples
- **End of reference period at** - insert the end point of the reference interval in ms or samples

Specify FREQUENCY RANGE and RESOLUTION:

- **Range**

Lower frequency border - calculate ERD maps starting at this frequency border

Upper frequency border - calculate ERD maps up to this frequency border

Bandwidth of single bands - bandwidth for each ERD calculation

Step to shift single band - step size to the next frequency band

- **Fixed bands** - define user-specific frequency bands

Add - add a frequency band

Remove - remove the frequency band

Upper cutoff - upper frequency for user-specific frequency band

Lower cutoff - lower frequency for user-specific frequency band

Specify METHOD:

- **Subtract phase locked signals** - select this to analyze only induced components
- **Raw signal** - include induced and evoked components
- **Complex demodulation** - perform the bandpass filtering with complex demodulation
- **Hilbert transformation** - use Hilbert transformation after bandpass filtering
- **Wavelet transformation** - use a Morlet wavelet for the bandpass filtering

AVERAGING, SMOOTHING and STATISTICS:

- **Averaging method** - select the horizontal averaging method
- **Horizontal averaging factor** – number of samples to average
- **Overlap** – number of samples overlapping
- **Significance test** - select to determine the significance by a bootstrap algorithm
- **Advanced settings** – define bootstrap settings for significance test. The settings have no effect if no significance test is performed.

ERD - Maps

Compute time- frequency plots of ERD/ERS. ERD values are shown in red and ERS values in blue colors.

Specify REFERENCE PERIOD:

Start of reference period at: 1000 [ms] / 256 [samples] End of reference period at: 2500 [ms] / 640 [samples]

Specify FREQUENCY RANGE and RESOLUTION:

Range: Lower frequency 6 [Hz] Upper frequency border: 40 [Hz] Bandwidth of single bands: 2 [Hz] Step to shift single band: 2 [Hz]

Fixed bands: Add -> 4 7, 8 10, 11 13, 14 20, 21 32, 33 42 <- Remove

Specify METHOD:

Subtract phase locked signals (only induced components)

AVERAGING, SMOOTHING and STATISTICS:

Averaging: mean Significance test: bootstrap

Horizontal averaging: Average 16 Overlap 8 samples

Select TRIALS and CHANNELS:

Select trials / chan.

Result procedure: Show with Result2D Show with Result3D Save results

Automatic treemaker is: enabled

Filename: ...indexfingermovement\erdmap\finger.mat

Help Cancel Start

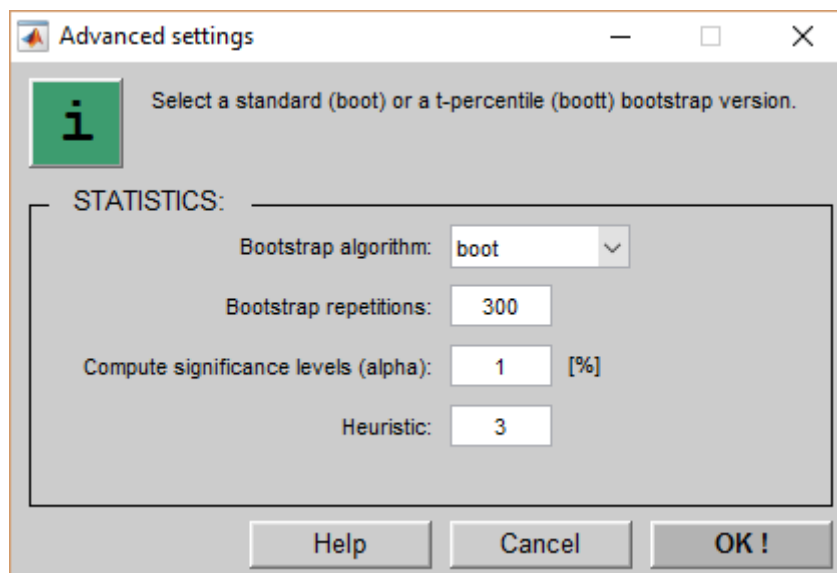
Perform the following steps to calculate ERDmaps from a data-set with self paced right index-finger movement.

1. Import the file `finger.bkr` from

`Documents\gtec\gBSanalyze\testdata\indexfingermovement`

The data-set consists of 34 EEG channels. Channel 11 represents C3, channel 14 Cz and Channel 17 C4. The subject had the task to move the index finger every 10 seconds self-paced. The data-set is movement off-set triggered with a pre-trigger interval of 4000 ms and a post-trigger interval of 4000 ms.

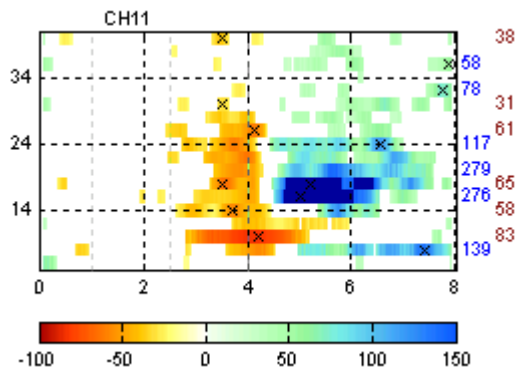
2. Assign the attribute `analyze` to channel 11, 14 and 17 in order to analyze only 3 channels
3. Click on **ERDmaps** under the **Analyze** menu entry to open the window
4. Set the **reference period** from 1000 ms to 2500 ms
5. Enter a **Lower frequency border** of 6 Hz and a **Upper frequency border** of 40 Hz. Set the **Bandwidth of single bands** to 2 Hz and the **Step to shift single band** to 2 Hz.
6. Analyze only induced components by checking **Subtract phase locked signals**
7. Check **Horizontal averaging** and set the **Average** window to 16 with an **Overlap** of 8
8. Enable the **Significance test** and press the **Advanced settings** button to open the following window:



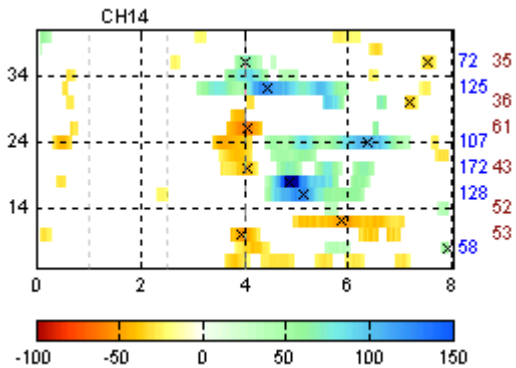
The **Advanced settings** window is used to define the bootstrap algorithm used for the significance test. The standard bootstrap algorithm is `boot`, a t-percentile bootstrap version is `boott`. Default setting for **Bootstrap repetitions** is 300 and for the **Compute significance level** 1 %. The **Heuristic** value defines how many pixels in the ERD calculation must show an ERD value to be shown in the ERD map. Heuristic 3 means that 3 neighbouring pixels must show an ERD to be shown.

9. Press **Select trials/chan.** and select include only channels with the `analyze` attribute
10. Press **Start** to perform the calculation. Result2D opens automatically with the results for all 3 channels.

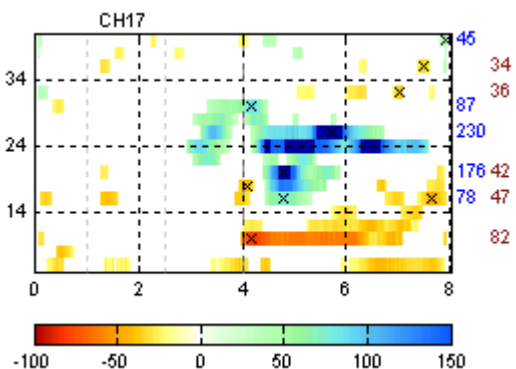
The y-axis shows the frequency ranges used for the ERD calculation. The x-axis shows the time in seconds. The scaling of the percentage of power decrease (red) and power increase (blue) is shown by the colorbar. Blue colors on the right border represent the maxima (power increase) found in the ERD bands. Red colors represent the minima (power decrease).



Channel C3 (right hand representation area) shows a contra-lateral significant ERD and ERS



Channel Cz (foot representation area)



Channel C4 (left hand representation area)

The following code shows how to perform the example demonstrated above from the MATLAB command line.

```
%Import Data
P_C=data;
File= ['C:\Users\' getenv('USERNAME')
'\Documents\gtec\gBSanalyze\testdata\indexfingermovement\finger.bkr'];
P_C=import(P_C, 'BKR', File);

%Select Trials and Channels
trial_id=[];
channel_id=[];
type_id=[];
channelnr_id=[11 14 17];
flag_tr='tr_exc';
flag_ch='ch_exc';
flag_type='type_exc';
flag_nr='nr_inc';
[TrialExclude, ChannelExclude]=gBSselect(P_C,trial_id,flag_tr,...
channel_id,flag_ch,type_id,flag_type,channelnr_id,flag_nr);

%ERD - Maps
clear T
T.ref=[256 640];
T.borders=[6 40];
T.bandwidths=[2];
T.steps=[2];
T.lc=[];
T.hc=[];
T.mode=0;
T.med=0;
T.stats = 'boot';
T.statsopt.B=300;
T.statsopt.alpha=0.01;
T.heuristic=3;
T.statsopt.med=0;
T.smooth=[16 8];
TrialExclude=[];
ChannelExclude=[1 2 3 4 5 6 7 8 9 10 12 13 15 16 18
19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34];
FileName='';
E_M=gBSerdmaps(P_C,T,TrialExclude,ChannelExclude,FileName,0);
```

Wavelet Transformation

The function performs a continuous wavelet transformation for each trial and averages the result over all selected trials.

The window allows the following settings:

Lowest frequency, **Highest frequency**, and **Frequency step** specify the frequency range used for the analysis.

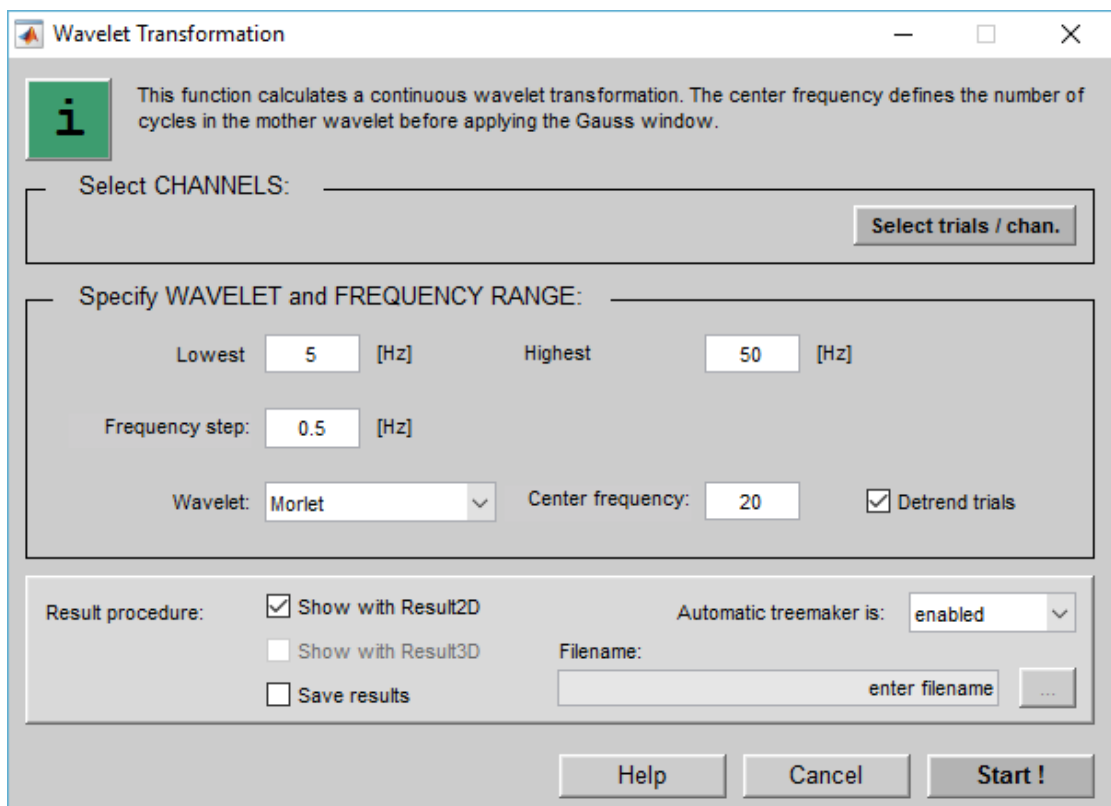
Wavelet – can be Morlet or Mexican

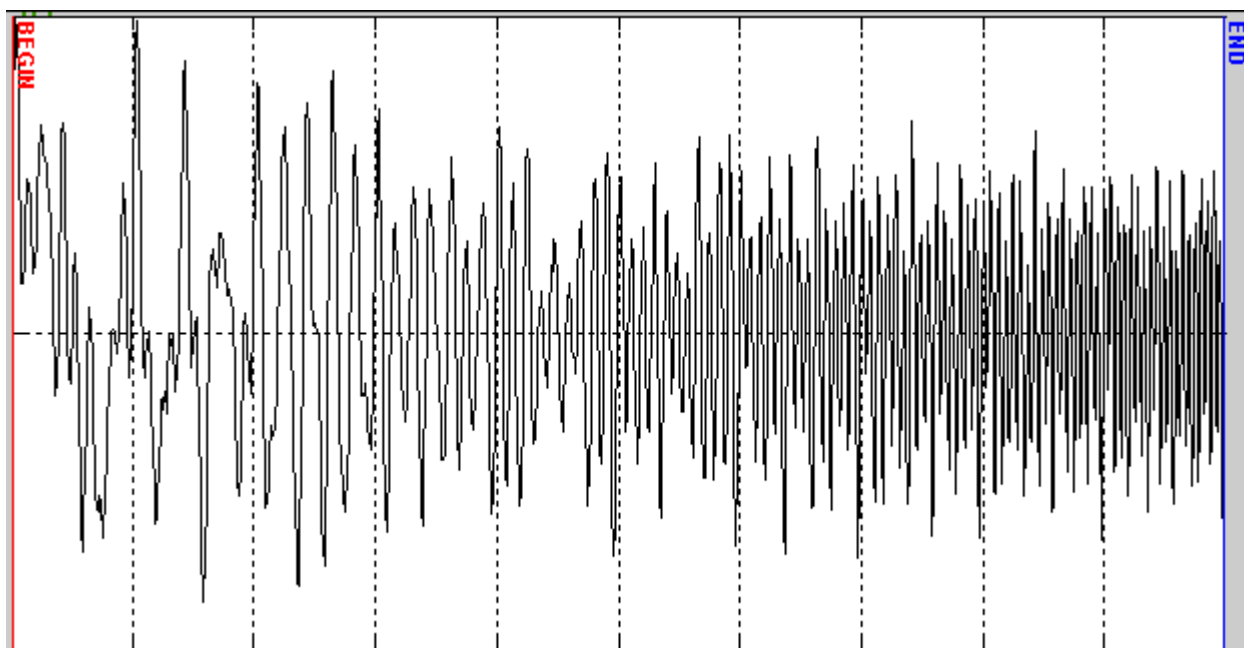
Cycles – parameter of the mother wavelet (cycle), lower values give a sharper time resolution

Perform the following steps to calculate the wavelet transformation from a noisy sweep signal.

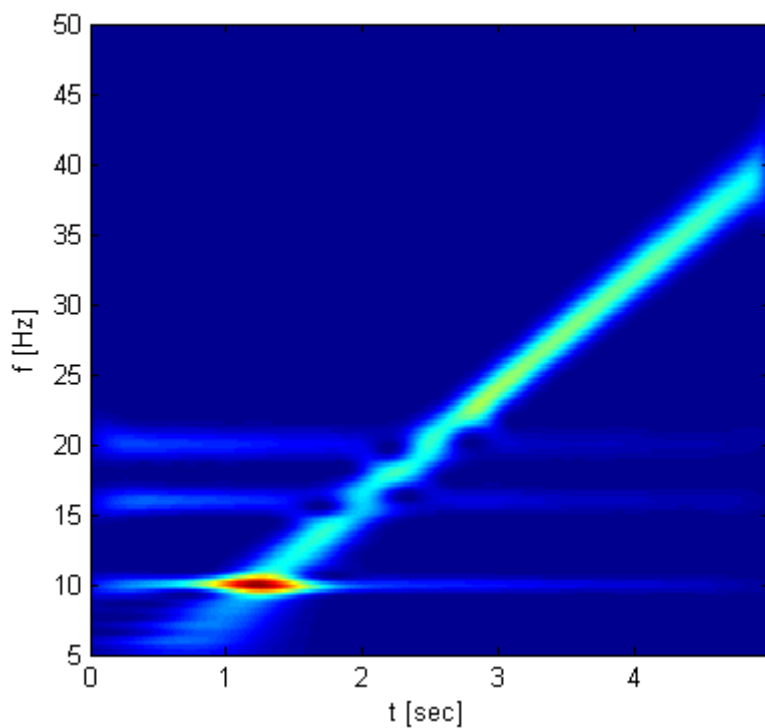
1. Open the file `sweep.mat` from

`Documents\gtec\gBSanalyze\testdata\sweep`





2. Select **Wavelet Transformation** from the **Analyze** menu
3. Enter as **Lowest frequency** 5, as **Highest Frequency** 50 and as **Frequency step** 0.5. Select the Morlet wavelet and set **Cycles** to 20
4. Press **Start** to perform the calculation. Result2d opens automatically with the following time-frequency map:



The image shows the increase of the frequency in the test signal from 10 Hz to 50 Hz.

The following code shows how to perform the example demonstrated above from the MATLAB command line.

```
%Load Data
P_C=data;
File=['C:\Users\' getenv('USERNAME')
'\Documents\gtec\gBSanalyze\testdata\Sweep\sweep.mat'];
P_C=load(P_C,File);

% Wavelet transformation
ChannelExclude=[];
TrialExclude=[];
FrequencyRange.fmin=5;
FrequencyRange.fstep=0.5;
FrequencyRange.fmax=50;
Wavelet='MORLET';
Cycles=20;
Detrend=1;
FileName='';
ProgressBarFlag=1;
W_O = gBSwavelettransformation(P_C,ChannelExclude,...
TrialExclude,Wavelet,Cycles,Detrend,FrequencyRange,...
FileName,ProgressBarFlag);
```

Common Spatial Patterns

The method uses the covariance to design common spatial patterns and is based on the simultaneous diagonalization of two covariance matrices [Fukonaga 1972]. The decomposition (or filtering) of the EEG leads to new time series, which are optimal for the discrimination of two populations. The patterns are designed such that the signal that results from the EEG filtering with the CSP has maximum variance for the first class and minimum for the second class and vice versa. In this way, the difference between class 1 and class 2 is maximized, and the only information contained in these patterns is where the variance of the EEG varies most when comparing two conditions.

Given N channels of EEG for each trial \mathbf{X} of class 1 and class 2, the CSP method gives an $N \times N$ projection matrix \mathbf{W} according to [Koles 1991, Müller-Gerking 1999, Ramoser 2000, Guger 2000]. This matrix is a set of N subject-specific spatial patterns which reflect the specific activation of cortical areas. With the projection matrix \mathbf{W} the decomposition of a trial \mathbf{X} is described by

$$\mathbf{Z} = \mathbf{W}\mathbf{X} \quad (1)$$

This transformation projects the variance of \mathbf{X} onto the rows of \mathbf{Z} and results in N new time series. The columns of \mathbf{W} are a set of CSPs and can be considered as time-invariant EEG source distributions. After interpolation the patterns can be displayed as topographical maps. By construction, the variance for class 1 is largest in the first row of \mathbf{Z} and decreases with the increasing number of the subsequent rows. The opposite is the case for a trial with class 2.

References:

K. Fukonaga, "Introduction to Statistical Pattern Recognition," Oxford, U.K., Clarendon, 1972.

C. Guger, H. Ramoser, G. Pfurtscheller, "Real-time EEG Analysis with Subject-Specific Spatial Patterns for a Brain-Computer Interface (BCI)", IEEE Trans. Rehab. Engng., vol. 8, pp. 447-456, 2000.

Z.J. Koles, "The quantitative extraction and topographic mapping of the abnormal components in the clinical EEG," Electroenceph. Clin. Neurophysiol., vol. 79, pp. 440-447, 1991.

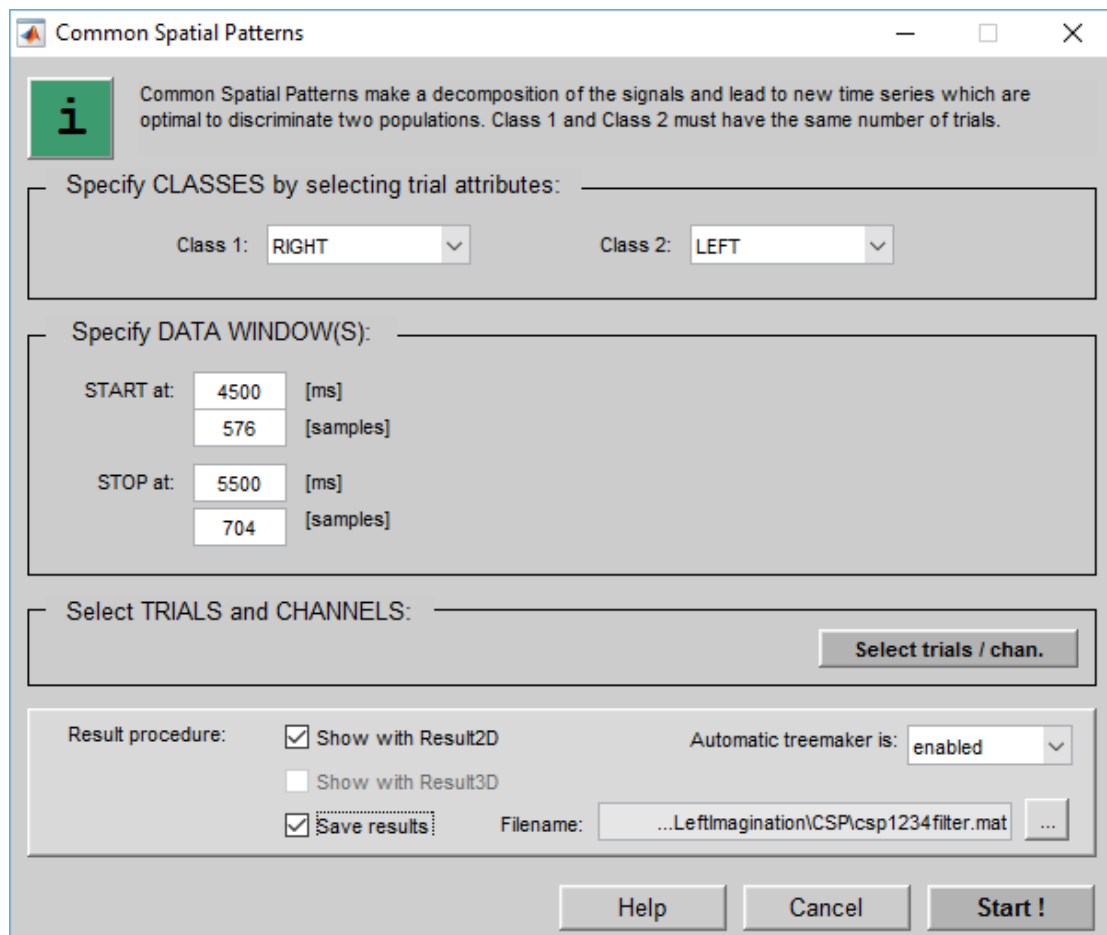
J. Müller-Gerking, G. Pfurtscheller and H. Flyvbjerg, "Designing optimal spatial filters for single-trial EEG classification in a movement task," Clin. Neurophysiol., vol. 110, pp. 787-798, 1999.

H. Ramoser, J. Müller-Gerking and G. Pfurtscheller, "Optimal spatial filtering of single-trial EEG during imagined hand movement," IEEE Trans. Rehab. Engng., vol. 8, pp. 441-446, 2000.

The method allows the following settings:

- **Class 1** - select the first class attribute
 - **Class 2** - select the second class attribute
- Note that the number of trials for class 1 and class 2 must be equal.

- **Specify DATA WINDOW(S)**
START at: starting of data window used for the calculation of the CSP
STOP at: stop time of the data window



1. Load the data-set `run1234.mat` from the directory

`Documents\gtec\gBSanalyze\testdata\RightLeftImagination`

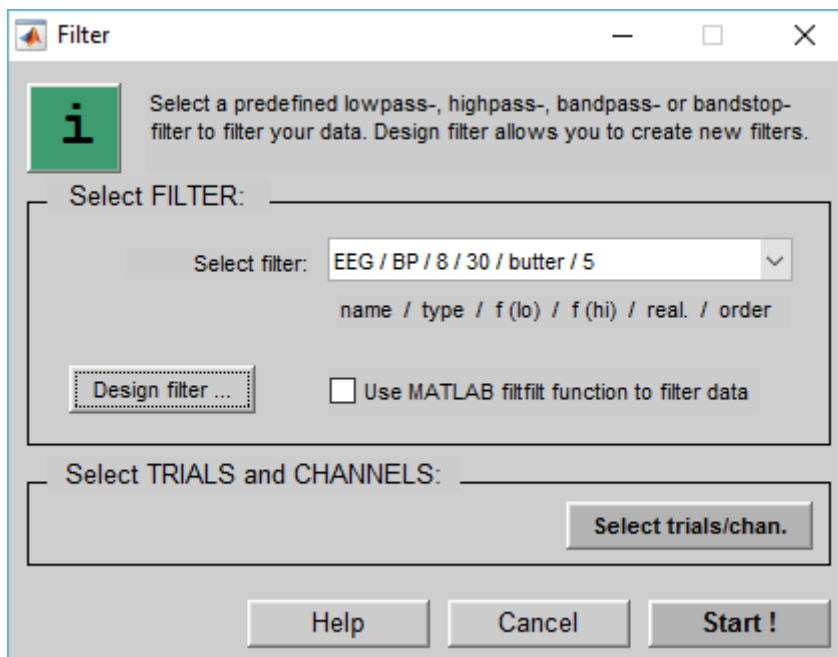
This is a data-set with 160 trials consisting of 6 seconds and 27 channels. The subject imagined 80 times a right hand movement and 80 times a left hand movement after a visual cue at second 2 indicating the direction on a computer monitor.

2. Select **Load Class Information** from the **File** menu to assign the class attributes using the **Import Wizard** stored in `class160.mat`. Assign the name `RIGHT` and `LEFT` to the trials.
3. Open the **Geometry** window under **Header** and **Browse** for the file `27ch1.mat` under

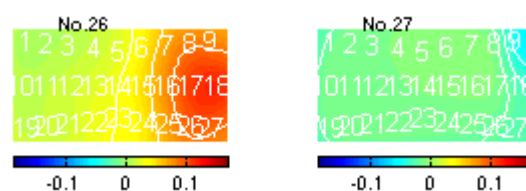
`Documents\gtec\gBSanalyze\testdata\montage`

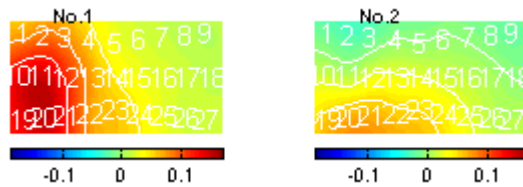
Click **OK** to import the x-, y- and z-positions of the electrodes. To inspect the montage open the Montage Creator.

4. Open **Filter** under **Pre-Processing** and select or define a bandpass filter with Butterworth characteristic, order 5. The lower cutoff frequency should be 8, the upper cutoff frequency should be 30 Hz.

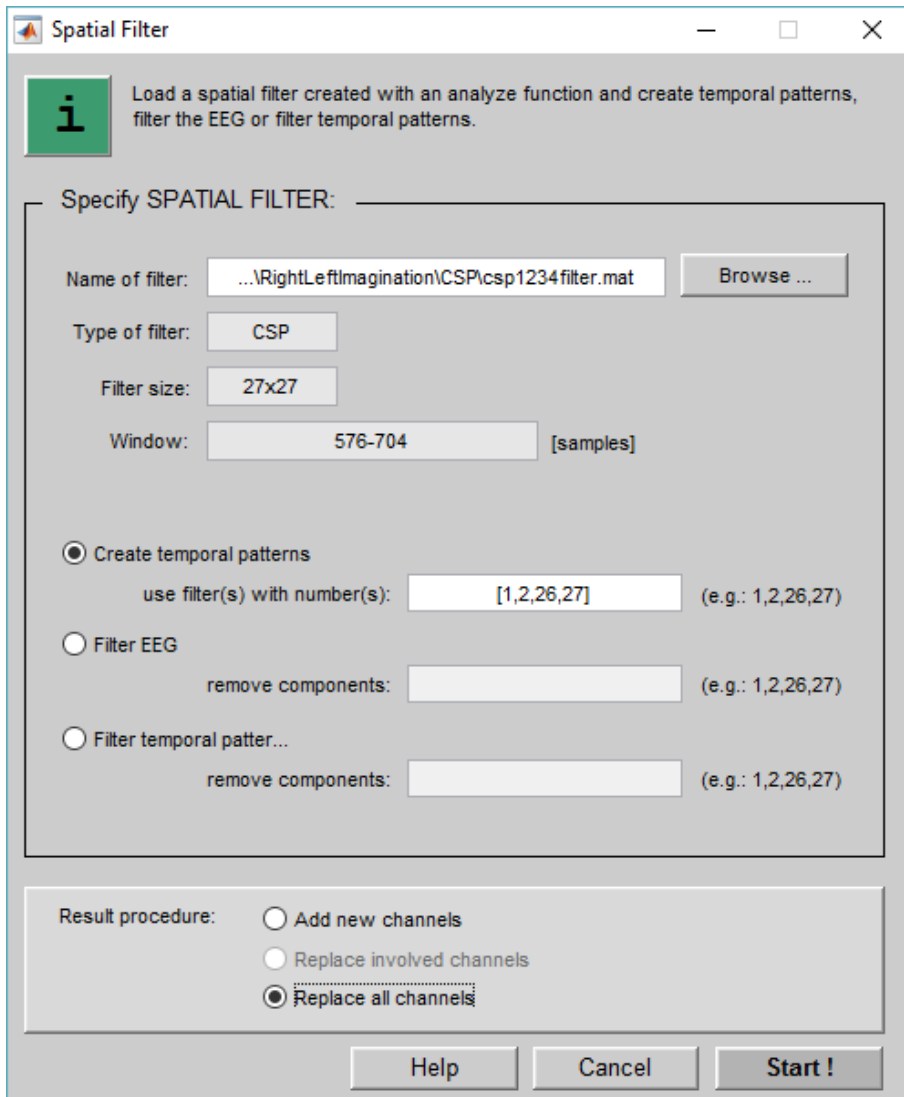


5. Click **Start** to filter your EEG data
6. Open **CSP** under the **Analyze** menu and select **RIGHT** as class 1 and **LEFT** as class 2. Define the **DATA WINDOW** from 4500 ms to 5500 ms.
7. Check **Show with Result2D** to view the common spatial patterns
8. Click **Save results** and enter a path and filename to save the maps. The **Automatic treemaker** creates the CSP subdirectory.
9. Result2D opens automatically with 27 CSP filters. The numbers within the maps indicate the 27 electrode positions. The CSPs with index 1 and 27 are the most and 2 and 26 are the second most discriminating filters. Electrodes surrounded by yellow colors are of lower importance to the discrimination task than electrodes surrounded by red or blue colors.

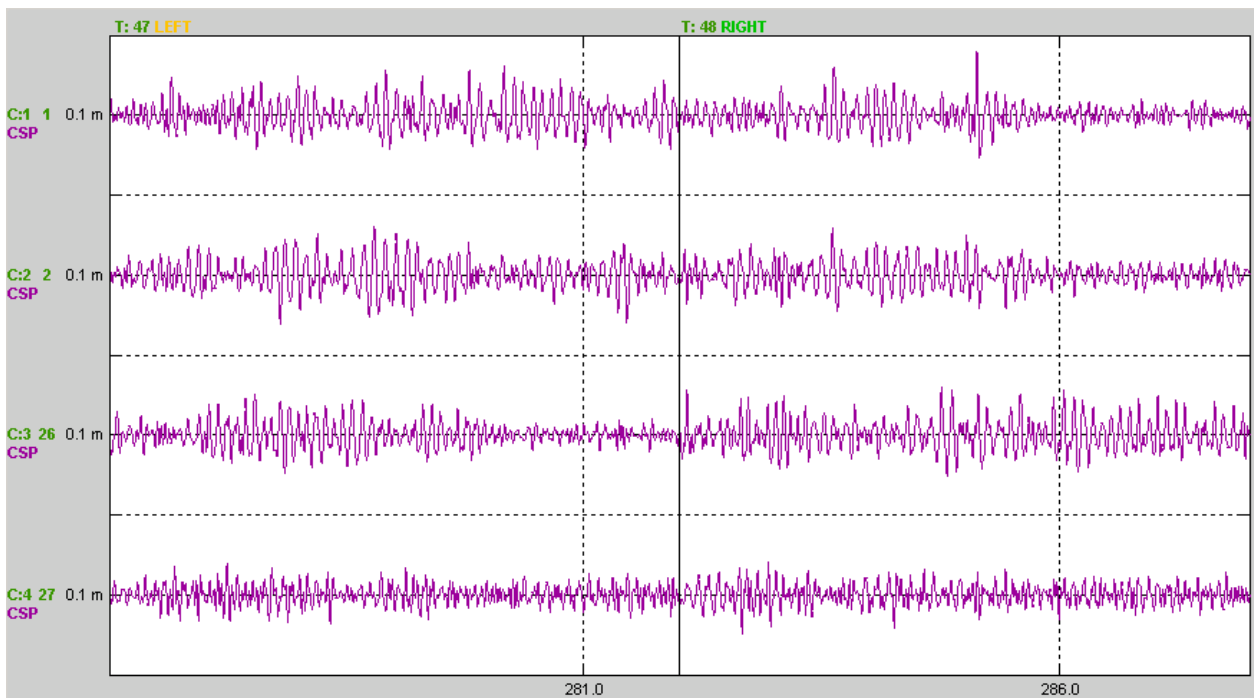




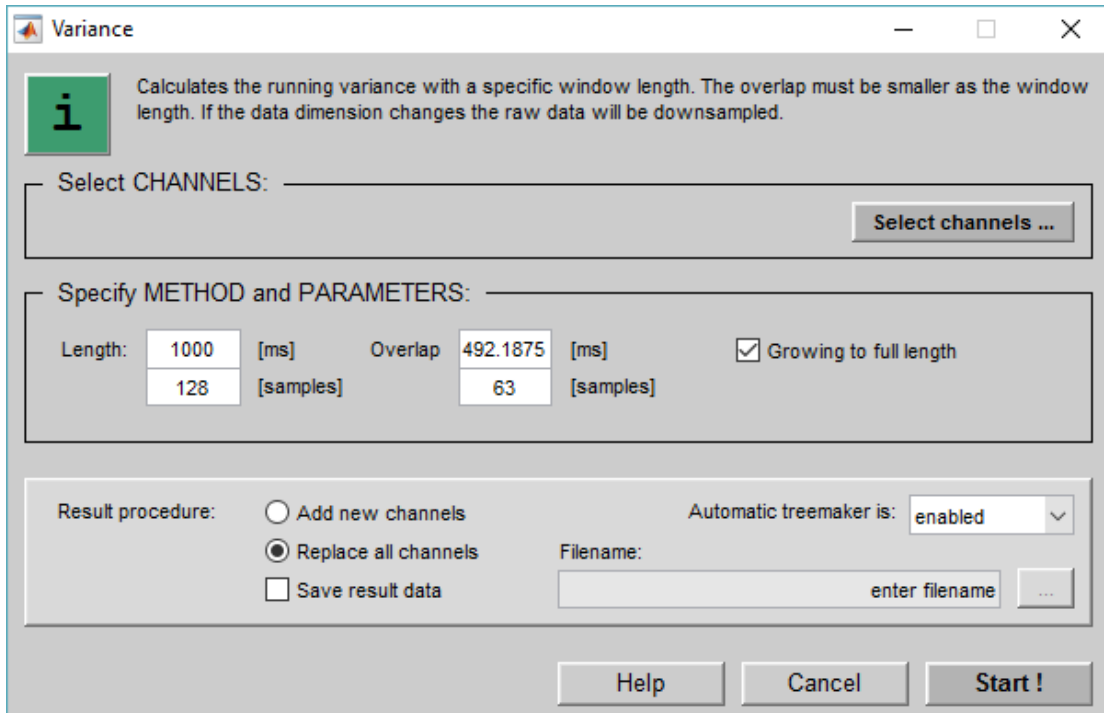
10. Open **Spatial Filter** under the menu **Pre-Processing** and load the created CSP filter by using the **Browse** button
11. Check **Create temporal patterns** and enter the 4 most important filters: 1, 2, 26 and 27



12. Press **Start** to convert the EEG data in temporal patterns. The figure below shows the resulting time series after filtering with the two most important (1, 27) and two second most important (2, 26) common spatial patterns according to (1).

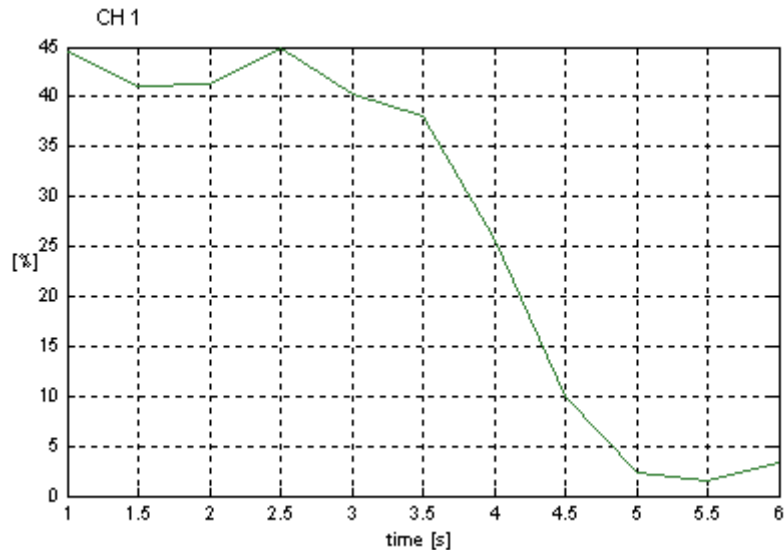


The filter was constructed in such a way that the variance in filter 1 and 2 will be maximized during left-hand hand movement imagination and minimized in filter 26 and 27. The left column shows the new time series of a left trial, the right column of a right trial. By construction in the most discriminating time series (1 and 27), a high amplitude difference can be observed. When comparing the second most important time series (2 and 26), still a difference can be seen, albeit a smaller one. The opposite is the case for the right trial. The variance in times series 1 and 2 is smaller, than in 26 and 27.



13. Then the variance of the time series is calculated. Therefore, open **Variance** under **Parameter Extraction** and select a time window of 1 second. **Start** the calculation.

14. The classification accuracy is calculated with a 10 times 10 fold cross-validation of a linear discriminant for 500 ms steps as shown in Chapter **Linear Discriminant Analysis**. The MATLAB Editor opens automatically with the classification errors. The first column shows the time points used for the cross-validation. And the next 10 columns correspond to the 10 runs of the cross-validation. A minimum of the classification error can be found at second 5.5. The right and the left trials are separable with about 1-2 % error, which corresponds to an accuracy of about 98-99%.



The following code shows how to perform the example demonstrated above from the MATLAB command line.

```
%Load Data
P_C=data;
File=['C:\Users\' getenv('USERNAME')
'\Documents\gtec\gBSanalyze\testdata\RightLeftImagination\run1234.mat'];
P_C=load(P_C,File);

%Load Class Information
class_info=[
0 0 0 1 0 0 1 1 1 0 1 1 1 0 0 1 0 0 1 0 0 0 1
1 0 1 1 0 1 1 1 0 0 1 0 0 1 1 0 0 1 0 0 1 0 1 0 0
0 1 1 0 1 1 0 1 1 1 0 0 1 0 0 1 1 0 0 0 1 0 0 1 1
1 0 1 1 1 1 0 0 1 0 0 1 0 1 0 0 0 0 1 0 0 1 1 1 1
0 1 1 0 1 1 0 1 1 0 1 1 1 0 0 1 0 0 1 1 0 1 0 0 1
0 1 0 0 0 1 0 0 1 0 0 1 0 0 1 1 1 0 1 1 0 0 1 1 0
1 1 1 0 0 1 0 0 1 1; 1 1 1 0 1 1 0 0 1 0 0 0 1 1 0
0 1 1 0 1 0 1 1 1 0 0 1 0 0 1 0 0 0 1 1 0 1 1 0 0 1
1 0 1 1 0 1 0 1 1 1 0 0 1 0 0 1 0 0 0 1 1 0 1 1 0
0 1 1 1 0 1 1 0 0 0 1 0 0 0 0 1 1 0 1 1 0 1 0 1 1
1 1 1 0 1 1 0 0 0 1 0 0 1 0 0 1 0 0 1 0 0 0 1 1 0
1 1 0 0 1 0 1 1 0 1 0 1 1 1 0 1 1 0 1 1 0 0 0 1 0
0 0 1 1 0 1 0 0 1 0 0 0 1 1 0 1 1 0 1 1 0 0];
name_classes={'RIGHT'; 'LEFT'};
use_rows=[1 2];
P_C=gBSloadclass(P_C,class_info,name_classes,use_rows);

%Filter
Filter.Realization='butter';
Filter.Type='BP';
Filter.Order=5;
Filter.f_high=30;
Filter.f_low=8;
TrialExclude=[];
ChannelExclude=[];
P_C=gBSfilter(P_C,Filter,ChannelExclude,TrialExclude);

%CSP
clear T
```

```

Class1_nr=3;
Class2_nr=4;
T=[576 704];
TrialExclude=[];
ChannelExclude=[];
FileName=['C:\Users\' getenv('USERNAME')
'\Documents\gtec\gBSanalyze\testdata\RightLeftImagination\CSP\csp1234filter
.mat'];
C_O=gBSscsp(P_C,T,Class1_nr,Class2_nr,TrialExclude,ChannelExclude,FileName,0
);

```

%Spatial Filter

```

SPF=spf;
Filter=load(SPF, ['C:\Users\' getenv('USERNAME')
'\Documents\gtec\gBSanalyze\testdata\RightLeftImagination\CSP\csp1234filter
.mat']);
FilterNumber=[1 2 26 27];
Replace='replace all channels';
Transformation='Create temporal pattern';
P_C=gBSspatialfilter(P_C,Filter,FilterNumber,Replace,Transformation);

```

% Variance

```

ChannelExclude = [];
IntervalLength = 128;
GrowingWindow = 1;
Overlap = 127;
Replace = 'replace all channels';
FileName = '';
ProgressBarFlag = 0;
P_C = gBSvariance(P_C, ChannelExclude, IntervalLength, GrowingWindow,
Overlap, Replace, FileName, ProgressBarFlag);

```

Principal Component Analysis

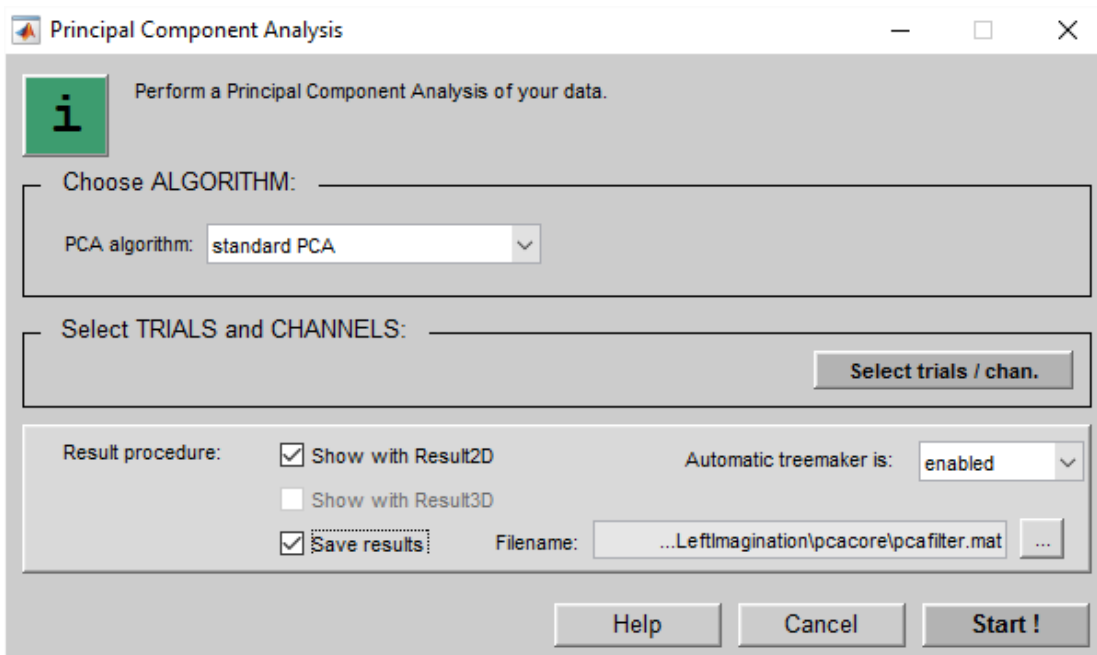
The method uses Singular Value Decomposition (SVD) to derive principal components of the signals. Multichannel signals are expressed by a P (time points) \times N (channels) matrix \mathbf{E} , and decomposed as a product of

$$\mathbf{E}=\mathbf{U}\mathbf{S}\mathbf{V}'$$

where \mathbf{U} is a $P \times N$ matrix such that $\mathbf{U}'\mathbf{U}=\mathbf{I}$. \mathbf{S} is an $N \times N$ diagonal matrix and \mathbf{V} is an $N \times N$ matrix. $\mathbf{V}'\mathbf{V} = \mathbf{V}\mathbf{V}' = \mathbf{I}$. If \mathbf{E} is a biosignal epoch of N channels and P time points, \mathbf{U} contains its N normalized principal components. These components are linearly decorrelated.

References:

Tzyy-Ping Jung, Colin Humphries, Te-Won Lee, Scott Makeig, Martin J. McKeown, Vicente Iragui, Terrence J. Sejnowski, "Removing Electroencephalographic Artifact: Comparison between ICA and PCA", Neural Networks for Signal Processing.



1. Load the data-set `run1234.mat` from the directory

```
Documents\gtec\gBSanalyze\testdata\RightLeftImagination
```

This is a data-set with 160 trials consisting of 6 seconds per trial and 27 channels. The subject imagined 80 times a right hand movement and 80 times a left hand movement after a visual cue at second 2 indicating the direction on a computer monitor.

2. Open the **Geometry** window under **Header** and **Browse** for the file `27ch1.mat` under

```
Documents\gtec\gBSanalyze\testdata\montage
```

Click **OK** to import the x-, y- and z-positions of the electrodes. To inspect the montage open the Montage Creator.

3. Click on **PCA** under the **Analyze** menu to perform the standard PCA
4. Check **Show with Result2D** and define a path and filename to store the PCA filters

The following code shows how to perform the example demonstrated above from the MATLAB command line.

%Load Data

```
P_C=data;  
File=['C:\Users\' getenv('USERNAME')  
'\Documents\gtec\gBSanalyze\testdata\RightLeftImagination\run1234.mat'];  
P_C=load(P_C,File);
```

%PCA

```
clear T  
T=[];  
TrialExclude=[];  
ChannelExclude=[];  
FileName= ['C:\Users\' getenv('USERNAME')  
'\Documents\gtec\gBSanalyze\testdata\RightLeftImagination\pca\pcafilter.mat  
';  
P_O=gBSpca(P_C,T,TrialExclude,ChannelExclude,FileName,0);
```

Independent Component Analysis

Independent Component Analysis (ICA) is an emerging technique aiming to recover unobserved signals or sources from observed mixtures, exploiting only the assumption of mutual independence between the signals. Specially the weakness of the assumptions make ICA a powerful approach.

The observed signals are obtained at the output of multiple sensors and each of these sensors receives a different combination of the source signals. Assume the source signal **S** with N rows represents the different source signals. Then **X** collects the N observed signals

$$\mathbf{X}=\mathbf{AS} \quad (1)$$

where **A** is the “mixing matrix” with NxN mixture coefficients. But no information about the source signals and of the mixture is available. The problem is to recover the source signal using only the observed data **X** with the assumption of independence between the entries of **S**. With the “separation matrix” **B** an estimate **Y** of the source signals **X** can be formulated as

$$\mathbf{Y}=\mathbf{BX} \quad (2)$$

Several implementations for ICA exist. The implemented ICA algorithm is based on the simultaneous diagonalization of cumulant and autocovariance matrices [Cardoso 1999].

References:

J.F., Cardoso, “High-order contrasts for independent component analysis,” Neural Computation, vol. 11(1), pp. 157-192, 1999.

J.F., Cardoso, “Blind signal separation: statistical principles,” IEEE Proc., vol. 9, nr. 10, pp. 2009-2025, 1998.

T.P., Jung, C., Humphries, T.W., Lee, S., Makeig, M., McKeown, V., Iragui, T., Sejnowski, “Removing Electroencephalographic Artifacts: Comparison between ICA and PCA,” Neural Networks for Signal Processing.

The **ICA** window allows the following settings:

ICA algorithm - use 2nd order statistic or higher order statistic

Lag – time lags used for time-delayed correlations

min - minimum time lag

step - lag step size

max - maximum time lag

Sort according to

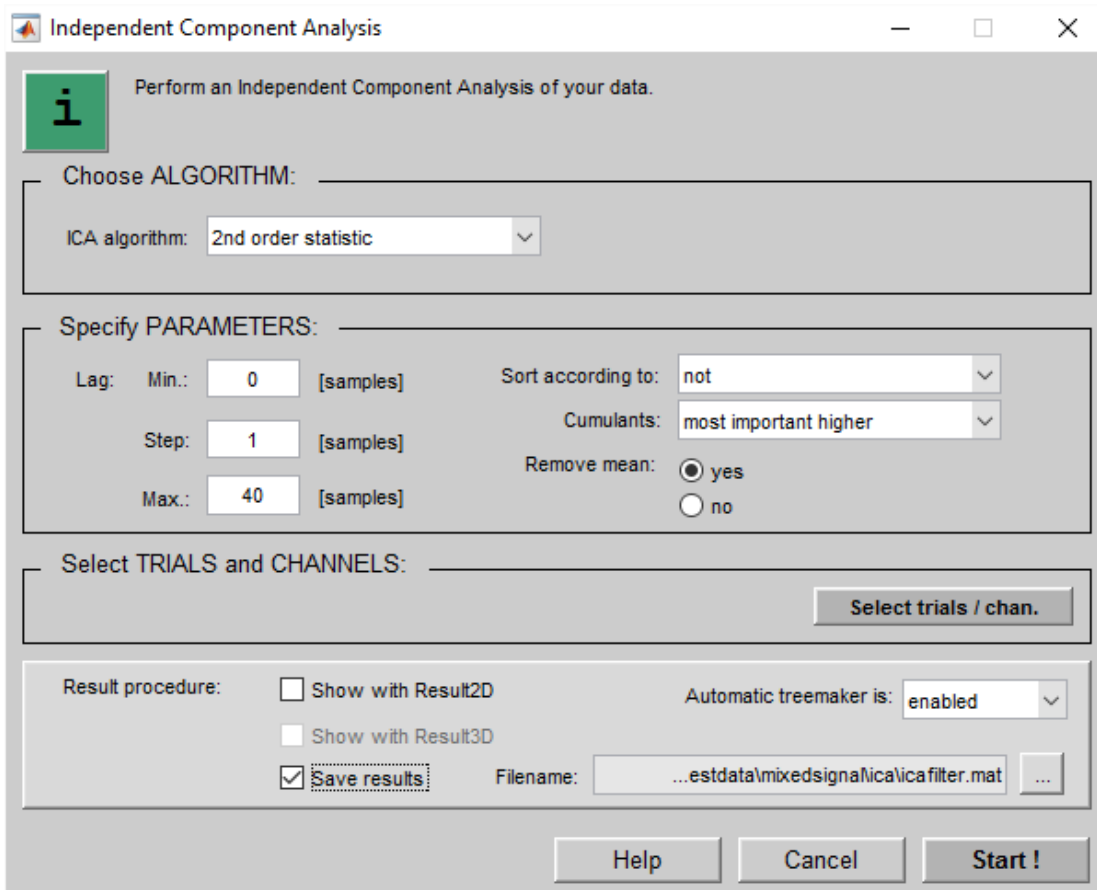
not - do not sort the components

column norm of mixing matrix

Cumulants

most important higher – use only the most important cumulants
 $n(n+1)/2$ – use $n(n+1)/2$ cumulants with n is the number of channels

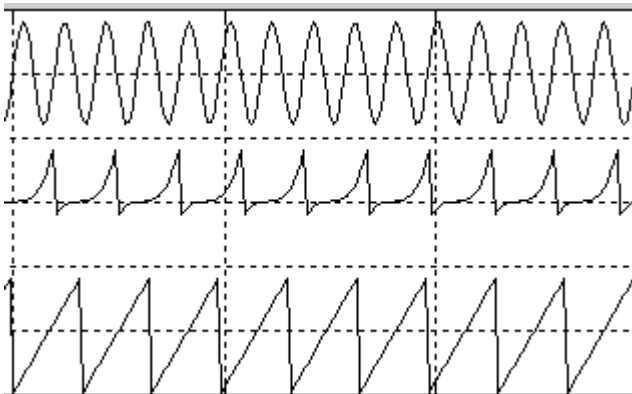
Remove mean - remove mean of each channel before calculation



1. Load the data-set `signal1.mat` from the directory

`Documents\gtec\gBSanalyze\testdata\mixedsignals`

Sampling frequency of this data set is 128 Hz.

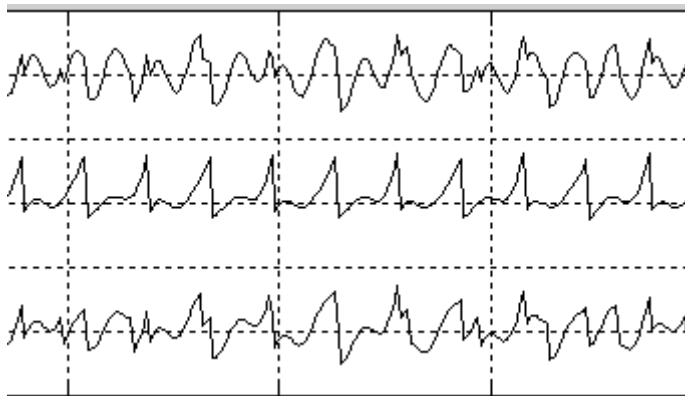


Channel 1, 2 and 3 are artificial signals created with

```
T=450;  
sig(1,:) = sin(t/2); % sinusoid  
sig(2,:) = ((rem(t,19)-7)/7).^3; % a curve  
sig(3,:) = ((rem(t,21)-11)/7); % saw-tooth
```

2. To investigate a mixture of the 3 signals open `signal2.mat`

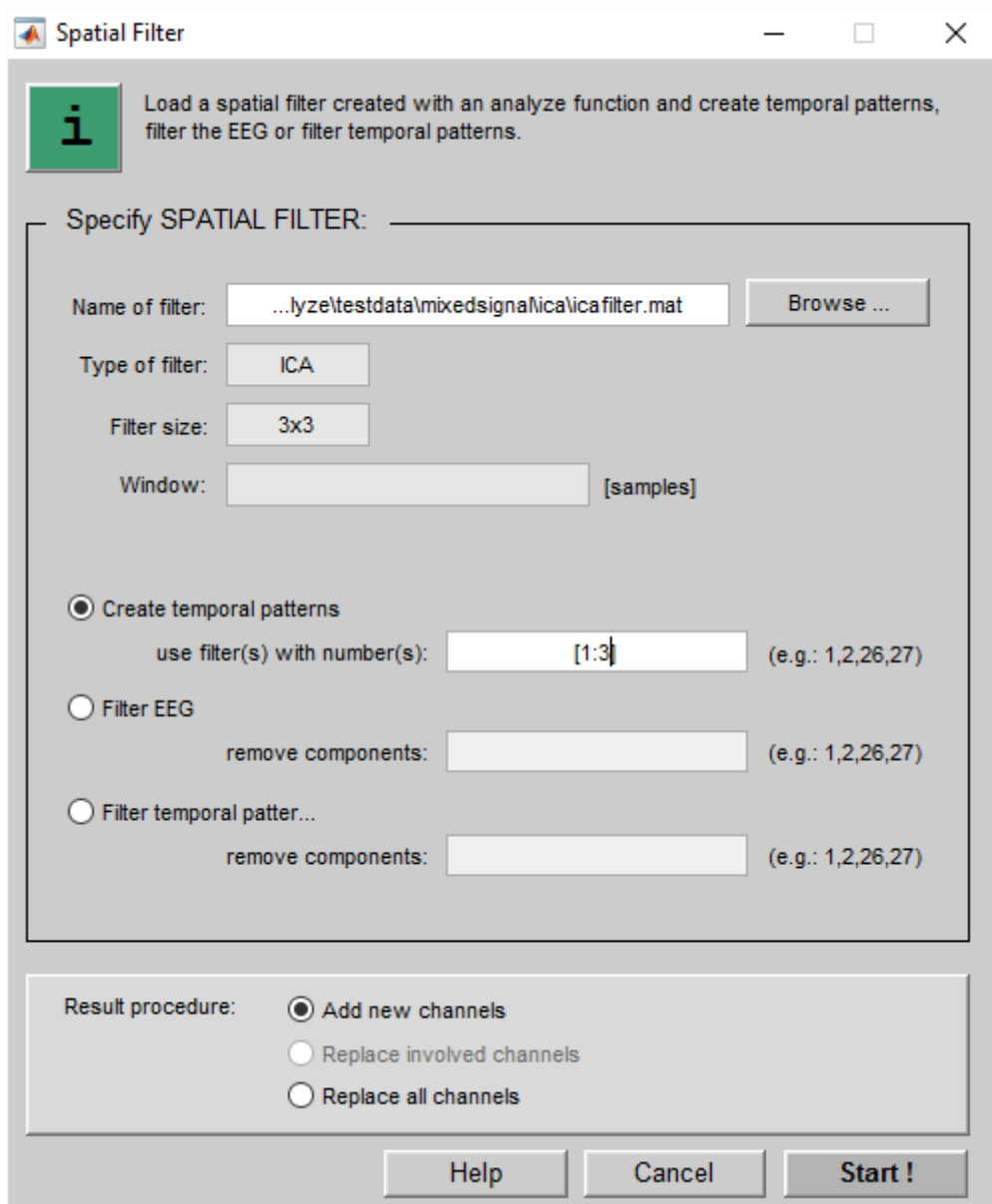
Sampling frequency for this dataset is again 128 Hz.



3. Click on **ICA** under the **Analyze** menu to perform a blind source separation of the 3 channels. Uncheck **Show with Result2D** and define a path and filename to store the ICA filter.

4. Click on **Spatial Filter** under **Pre-Processing** and load the file `icafilter.mat`

Select **Create temporal patterns** and enter `[1:3]`



5. Click **Start** to perform the blind source separation. The Data Editor shows the separated signals.

The following code shows how to perform the example demonstrated above from the MATLAB command line.

%Load Data

```
P_C=data;
File= ['C:\Users\' getenv('USERNAME')
'\Documents\gtec\gBSanalyze\testdata\mixedsignal\signal2.mat'];
P_C=load(P_C,File);
P_C.SamplingFrequency=128;
```

%ICA

```
clear T
T.lags=[0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16
17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35
36 37 38 39 40];
T.hos=0;
T.hosc=1;
T.rmmean=1;
T.sort=0;
TrialExclude=[];
ChannelExclude=[];
FileName= ['C:\Users\' getenv('USERNAME')
'\Documents\gtec\gBSanalyze\testdata\mixedsignal\ica\icafilter.mat'];
I_O=gBSica(P_C,T,TrialExclude,ChannelExclude,FileName,1);
```

%Spatial Filter

```
SPF=spf;
Filter=load(SPF, ['C:\Users\' getenv('USERNAME')
'\Documents\gtec\gBSanalyze\testdata\mixedsignal\ica\icafilter.mat']);
FilterNumber=[1 2 3];
Replace='add channels';
Transformation='Create temporal pattern';
P_C=gBSspatialfilter(P_C,Filter,FilterNumber,Replace,Transformation);
```

Canonical Correlation Analysis (CCA)

The Canonical Correlation Analysis (CCA), developed by H. Hotelling [1], can be used for measuring the linear relationship between two stochastic vectors (of not necessary same length). Compared to other correlation methods, the CCA is not dependent on the original basis, the input variables are described in. For BCI the CCA is often used for the frequency recognition in Steady-State Visual Evoked Potentials (SSVEP) Paradigms.

Method

For two stochastic vectors \mathbf{x} and \mathbf{y} the CCA consists of finding a pair of basis vectors $\hat{\mathbf{w}}_x$ and $\hat{\mathbf{w}}_y$ so that the transformed vectors $x = \mathbf{x}^T \hat{\mathbf{w}}_x$ and $y = \mathbf{y}^T \hat{\mathbf{w}}_y$ have maximum correlation ρ

$$\max_{\hat{\mathbf{w}}_x, \hat{\mathbf{w}}_y} \rho = \max_{\hat{\mathbf{w}}_x, \hat{\mathbf{w}}_y} \frac{E[xy]}{\sqrt{E[x^2] E[y^2]}}$$

By using the definition of x and y and applying the linearity of the expectation E , this equation transforms to

$$\begin{aligned} \max_{\hat{\mathbf{w}}_x, \hat{\mathbf{w}}_y} \rho &= \max_{\hat{\mathbf{w}}_x, \hat{\mathbf{w}}_y} \frac{E[xy]}{\sqrt{E[x^2] E[y^2]}} = \max_{\hat{\mathbf{w}}_x, \hat{\mathbf{w}}_y} \frac{E[\hat{\mathbf{w}}_x^T \mathbf{x} \mathbf{y}^T \hat{\mathbf{w}}_y]}{\sqrt{E[\hat{\mathbf{w}}_x^T \mathbf{x} \mathbf{x}^T \hat{\mathbf{w}}_x] E[\hat{\mathbf{w}}_y^T \mathbf{y} \mathbf{y}^T \hat{\mathbf{w}}_y]}} = \\ &= \max_{\hat{\mathbf{w}}_x, \hat{\mathbf{w}}_y} \frac{E[\hat{\mathbf{w}}_x^T] E[\mathbf{x} \mathbf{y}^T] E[\hat{\mathbf{w}}_y]}{\sqrt{E[\hat{\mathbf{w}}_x^T] E[\mathbf{x} \mathbf{x}^T] E[\hat{\mathbf{w}}_x] E[\hat{\mathbf{w}}_y^T] E[\mathbf{y} \mathbf{y}^T] E[\hat{\mathbf{w}}_y]}} = \max_{\mathbf{w}_x, \mathbf{w}_y} \frac{\mathbf{w}_x^T \mathbf{C}_{xy} \mathbf{w}_y}{\sqrt{\mathbf{w}_x^T \mathbf{C}_{xx} \mathbf{w}_x \mathbf{w}_y^T \mathbf{C}_{yy} \mathbf{w}_y}} \end{aligned}$$

Given two stochastic variables x and y with zero mean, the canonical correlations can be obtained as the eigenvalues ρ in the following equations

$$\begin{cases} \mathbf{C}_{xx}^{-1} \mathbf{C}_{xy} \mathbf{C}_{yy}^{-1} \mathbf{C}_{yx} \hat{\mathbf{w}}_x = \rho^2 \hat{\mathbf{w}}_x \\ \mathbf{C}_{yy}^{-1} \mathbf{C}_{yx} \mathbf{C}_{xx}^{-1} \mathbf{C}_{xy} \hat{\mathbf{w}}_y = \rho^2 \hat{\mathbf{w}}_y \end{cases}$$

Where \mathbf{C}_{xx} and \mathbf{C}_{yy} are the within-set covariance matrices of x and y and $\mathbf{C}_{xy} = \mathbf{C}_{yx}^T$ is the

between-sets covariance matrix. It only necessary to solve one of the equations, since they are related by

$$\begin{cases} \mathbf{C}_{xy} \hat{\mathbf{w}}_y = \rho \lambda_x \mathbf{C}_{xx} \hat{\mathbf{w}}_x \\ \mathbf{C}_{yx} \hat{\mathbf{w}}_x = \rho \lambda_y \mathbf{C}_{yy} \hat{\mathbf{w}}_y \end{cases}$$

where

$$\lambda_x = \lambda_y^{-1} = \sqrt{\frac{\mathbf{w}_y^T \mathbf{C}_{yy} \mathbf{w}_y}{\mathbf{w}_x^T \mathbf{C}_{xx} \mathbf{w}_x}}$$

For further information on that topic please consult the extensive tutorial on CCA of Magnus Borga [4].

References:

- [1] Hotelling, H. (1933). Analysis of a complex of statistical variables into principal components. *Journal of educational psychology*, 24(6), 417.
- [2] Lin, Z., Zhang, C., Wu, W., & Gao, X. (2006). Frequency recognition based on canonical correlation analysis for SSVEP-based BCIs. *Biomedical Engineering, IEEE Transactions on*, 53(12), 2610-2614.
- [3] Li, Y., Bin, G., Gao, X., Hong, B., & Gao, S. (2011, April). Analysis of phase coding SSVEP based on canonical correlation analysis (CCA). In *Neural Engineering (NER), 2011 5th International IEEE/EMBS Conference on* (pp. 368-371). IEEE.
- [4] Borga, M. (2001). Canonical correlation: a tutorial. *On line tutorial <http://people.imt.liu.se/magnus/cca>*, 4.

Calculate the CCA – Canonical Correlation Analysis

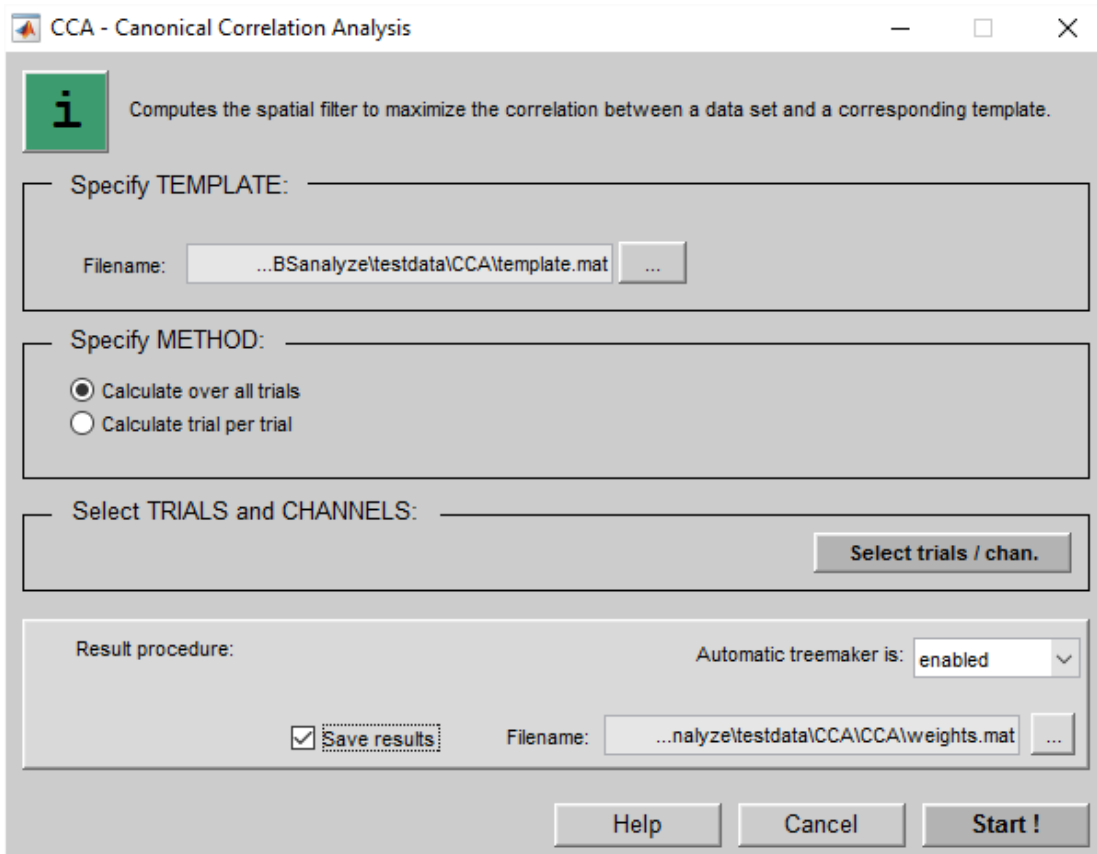
This function from the menu **Analyze** takes two input data sets – the triggered data set loaded in gBSanalyze and the template file - and calculates the weight vectors for the maximum canonical correlation coefficient.

Note: data set and template vector must have zero mean! If this is not the case, please perform **Pre-Processing/Baseline Correction** before applying **CCA – Canonical Correlation Analysis**.

The input mask allows the following settings:

- **Specify Template** – enter the name (and if necessary path) of the .mat file containing the template the currently loaded dataset is compared to by using the file browser by clicking on the ‘...’-symbol next to the textbox.
Note, that in most cases the template is the idealized version of one trial, approximated by taking the signal average over all trials. It has to be made sure that the template sample length matches the length of one trial of the triggered data, and that the number of channels in both data sets is equal.
- **Specify Method** – choose whether to calculate the canonical correlation weights for the whole data set (concatenation of trials) or to calculate specific optimized weights for each trial in the triggered data set. When **Calculate over all trials** is selected, the result is one weight vector of dimension channels × 1. **Calculate trial per trial** outputs one such weight vector for every trial which results in a channels × trials matrix.

- **Select Trials and Channels** – chose which trials and channels of the currently loaded data set to include into the calculation
- **Result Procedure:** - When the results will be used for further processing (e.g. with ‘Parameter Extraction, Canonical Correlation Analysis’), click on the checkbox **Save results**. This opens a file browser window that allows choosing the desired file. Otherwise, the results will be stored in the MATLAB workspace under the variable name ‘W_V’.



Example:

Perform the following steps to create a CCA filter (weight vector for maximal correlation):

1. Load data-set `cvep_triggered.mat` under
`Documents\gtec\gBSanalyze\testdata\CCA`
and specify a Sampling frequency of 256 Hz if asked.
2. Click on **CCA – Canonical Correlation Analysis** under the **Analyze** menu.
3. Load template file `template.mat` under
`Documents\gtec\gBSanalyze\testdata\CCA`
4. Select the option **Calculate over all trials**.

5. Click on **Select trials / chan.** and **include only** channels 2-9. Confirm the settings with the **OK** button.
6. Check the **Save results** box and type `weights.mat` to name the results file. The file should be placed automatically in the folder

`Documents\gtec\gBSanalyze\testdata\CCA\CCA`

7. Press **Start** to compute the canonical correlation weight vector.

The following code shows how to implement the example demonstrated above from the MATLAB command line.

```
%Load Data
P_C=data;
File=['C:\Users\' getenv('USERNAME')
'\Documents\gtec\gBSanalyze\testdata\CCA\cvcp_triggered.mat'];
P_C=load(P_C,File);
P_C.SamplingFrequency=256;

%CCA - Canonical Correlation Analysis
MergeTrials=[1];
TemplateFilename= ['C:\Users\' getenv('USERNAME')
'\Documents\gtec\gBSanalyze\testdata\CCA\template.mat'];
TrialExclude=[];
ChannelExclude=[1 10 11 12];
FileName= ['C:\Users\' getenv('USERNAME')
'\Documents\gtec\gBSanalyze\testdata\CCA\CCA\weights.mat'];

W_V=gBSccafilter(P_C,MergeTrials,TemplateFilename,TrialExclude,ChannelExclude,FileName,1);
```


Coherence

The **Coherence** window allows the following settings:

Specify INTERVAL(S):

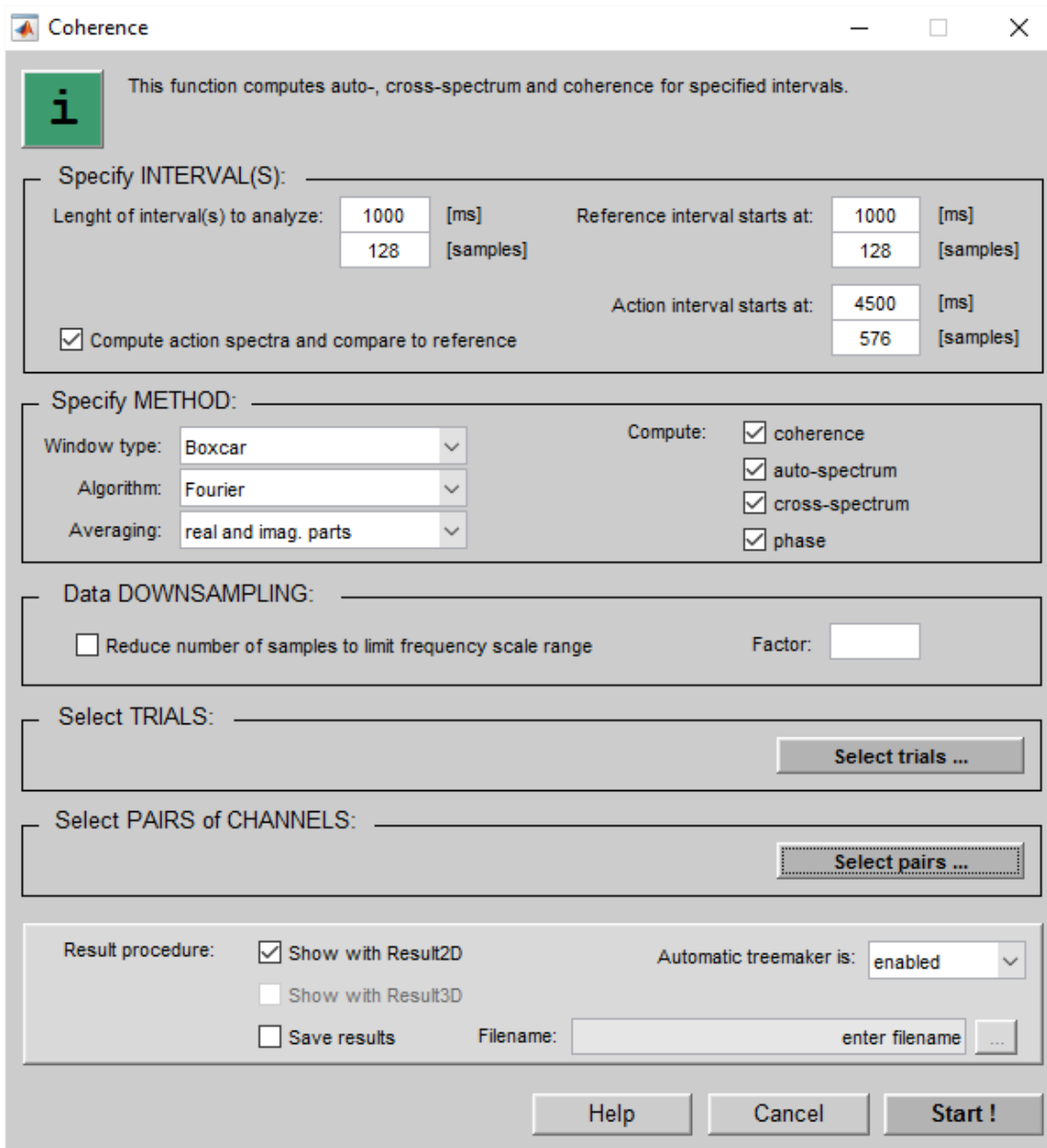
1. **Length of interval(s) to analyze** – specify the length of the reference and action interval in ms or samples
2. **Reference interval starts at** - insert the start point of the reference interval in ms or samples
3. **Action interval starts at** - insert the start point of the action interval in ms or samples
4. **Compute action spectra and compare to reference** – check the box to calculate the spectrum of the reference interval and of the action interval and compare the results

Specify METHOD:

5. **Window type** – specify the windowing function used before applying the FFT algorithm to the action and reference interval. Window type can be `Boxcar`, `Hamming`, `Hanning`, `Bartlett` or `Blackman`
6. **Algorithm** – the method performs a Fourier Transformation
7. **Averaging** – select `real` and `imag. parts` to average the real and imaginary parts of the calculated auto- and cross-spectrum over all trials. Select `magnitude` and `phase` to calculate the magnitude and the phase of the cross-spectrum before averaging over trials.
8. **Compute** – check to compute and visualize the coherence, the cross-spectrum and the phase between two channels and check `auto-spectrum` to calculate it from every channel

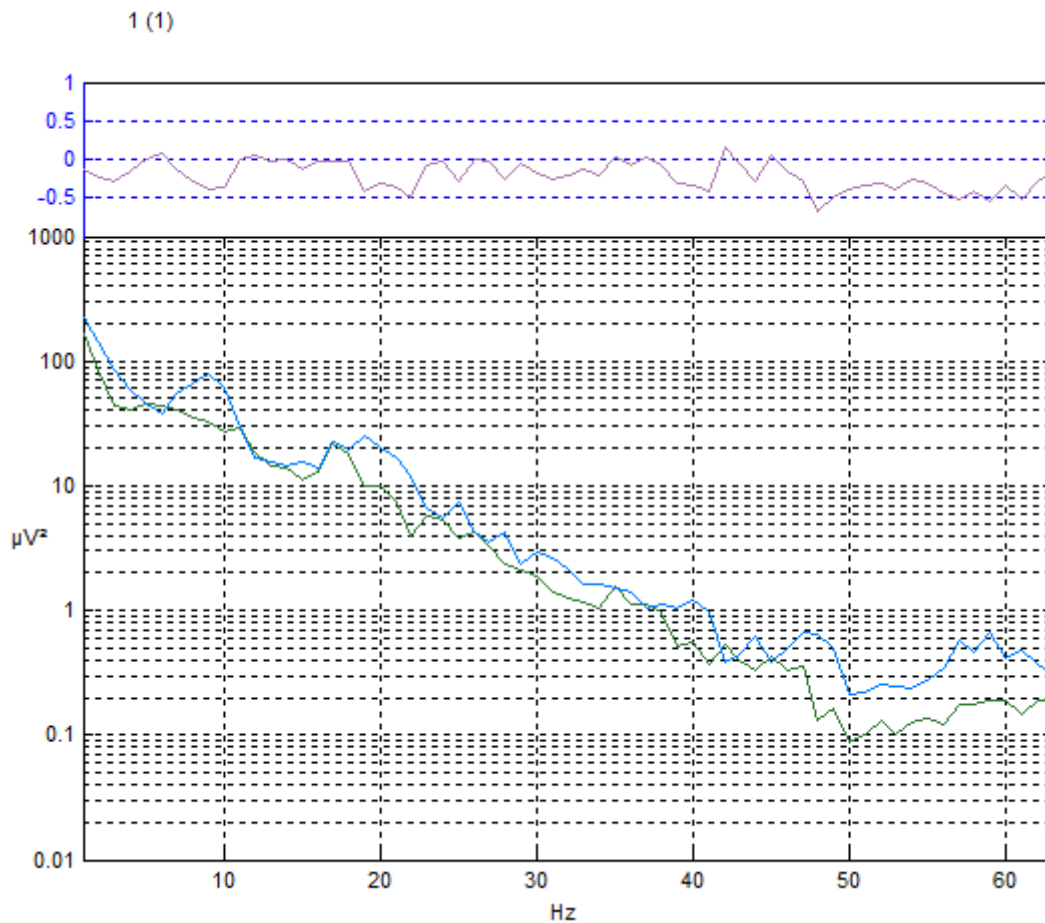
Data DOWNSAMPLING

9. **Reduce number of samples to limit frequency scale range** - select to reduce the number of samples and to reduce the frequency scale range by the specified factor.



1. Load the data-set `trig1.mat` from the directory
`Documents\gtec\gBSanalyze\testdata\BCI`
2. Select **Coherence** from the **Analyze** menu
3. Set the **Length of interval(s) to analyze** to 1000 ms and check **Compute action spectra and compare to reference**
4. Insert **Reference interval starts** at 1000 ms and **Action interval starts** at 4500 ms. This means that the FFT for the reference interval will be computed from 1000 ms to 2000 ms and for the action interval from 4500 ms to 5500 ms.
5. Select a `Boxcar` window in the **Window type** pull-down menu
6. Open **Select pairs** and define a channel-pair with channel 1 and channel 2

7. Check the **Show with Result2D** box.
8. Press the **Start** button



Result2D opens automatically with the auto-spectrum for each channel and the coherence, cross-spectrum and phase for the defined channel pair. The following figure shows a coherence plot of the reference (blue line) and active interval (green line). On top of the plot a significance measure of the differences between reference interval and action interval is given.

The following code shows how to perform the example demonstrated above from the MATLAB command line.

```
%Load Data
P_C=data;
File = ['C:\Users\' getenv('USERNAME')
'\Documents\gtec\gBSanalyze\testdata\BCI\trig1.mat'];
P_C=load(P_C,File);

%Coherence
ActionStart=576;
RefStart=128;
IntervalLength=128;
Window='Boxcar';
Averaging='real and imag. parts';
Compute.coh=1;
```

```
Compute.pxx=1;
Compute.pxy=1;
Compute.phase=1;
ChannelPairs=[1 2];
TrialExclude=[];
DownSampling=[0];
FileName='';
C_O=gBScoherence(P_C,ActionStart,RefStart,IntervalLength,Window,Averaging,Com
pute,ChannelPairs,TrialExclude,DownSampling,FileName,0);
```

Event Related Coherence

The **Event Related Coherence** window allows the following settings:

Specify REFERENCE PERIOD:

The reference period is used for the calculation of the relative coherence in % and is not active for the absolute coherence mode.

- **Start of reference period at** - insert the start point of the reference interval in ms or samples
- **End of reference period at** - insert the end point of the reference interval in ms or samples

Specify FREQUENCY (BAND):

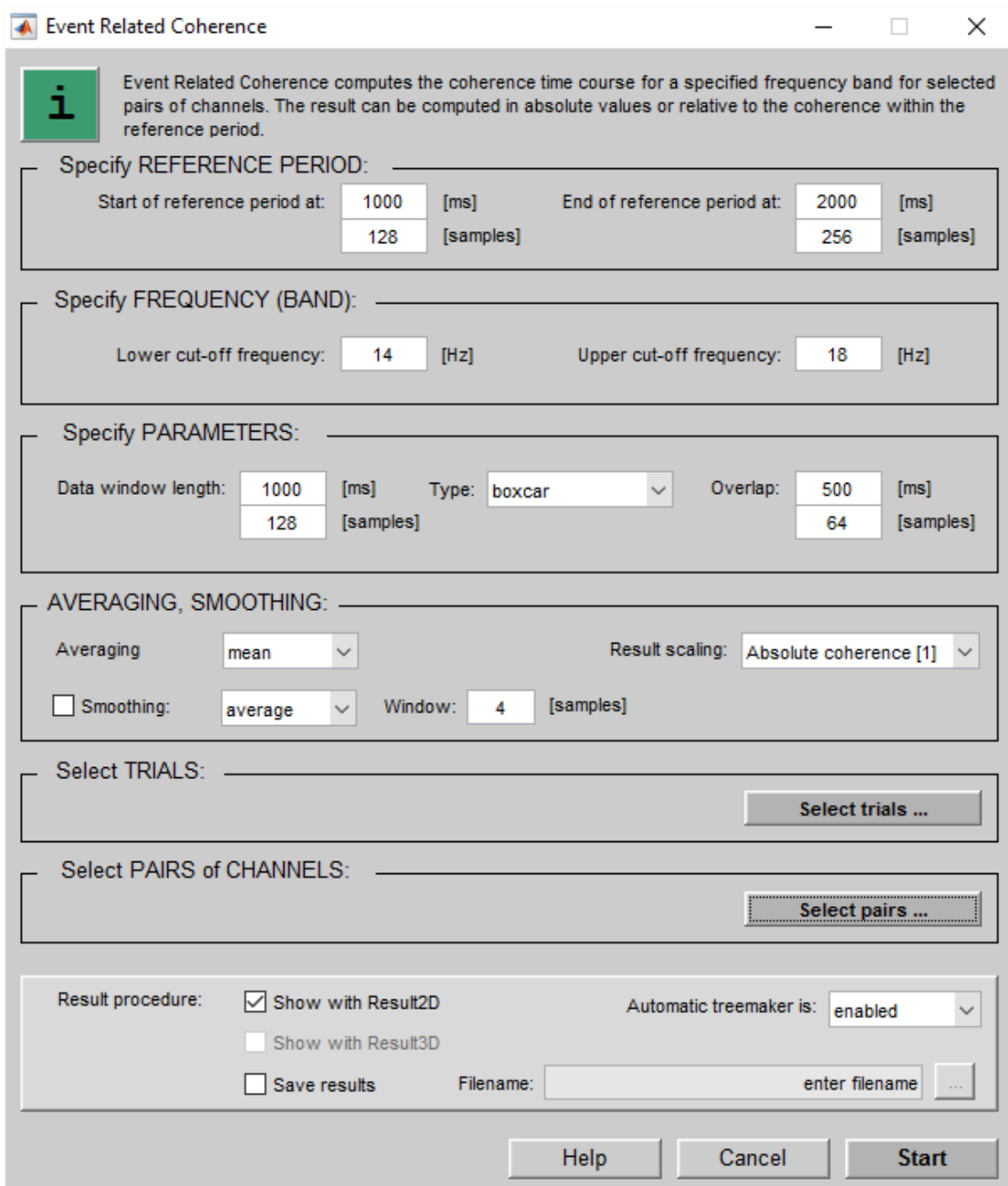
- **Lower cut-off frequency** - calculate ERC starting at this frequency border
- **Upper cut-off frequency** - calculate ERC up to this frequency border

Specify PARAMETERS:

- **Data window length** – specify the length of the interval in ms or samples
- **Type** - specify the windowing function used before applying the FFT algorithm to the interval. Window type can be `boxcar`, `hamming`, `hanning`, `bartlett` or `blackman`
- **Overlap** – overlap of the intervals

AVERAGING, SMOOTHING:

- **Averaging method** – `mean` or `median` can be used to average the power spectrum from the lower cut-off frequency to the upper cut-off frequency
- **Result scaling**
Relative coherence in %
Absolute coherence
- **Smoothing** – smooth the result
`average` – average over the specified **window** length in samples
`exponential` - exponential window with **window** length in samples
`cosinus` - cosine window with a window length of 3 samples



1. Load the data-set `trig1.mat` that is stored under
`Documents\gtec\gBSanalyze\testdata\BCI`
2. Select **Event Related Coherence** from the **Analyze** menu;
3. Insert **Start of reference period at** 1000 ms and **End of reference period at** 2000 ms.;
4. Set the **Lower cut-off frequency** to 14 Hz and the **Upper cut-off frequency** to 18 Hz;
5. Set the **Data window length** to 1000 ms, select a `Boxcar` window in the **Type** pull-down menu and set the **Overlap** to 500 ms;

6. In the **AVERAGING, SMOOTHING** section select `mean` for **Averaging method** and `Absolute Coherence[1]` for **Result scaling**;
7. Open **Select pairs** and define a channel-pair with channel 1 and channel 2;
8. Check the **Show with Result2D** box;
9. Press the **Start** button.

Result2D opens automatically with the coherence time course for the specified frequency band for the selected pair of channels.

The following code shows how to perform the example demonstrated above from the MATLAB command line.

```
%Load Data
P_C=data;
File= ['C:\Users\' getenv('USERNAME')
'\Documents\gtec\gBSanalyze\testdata\BCI\trig1.mat'];
P_C=load(P_C,File);

%Event Related Coherence
RefStart=128;
RefEnd=256;
Lcf=14;
Ucf=18;
IntervalLength=128;
Window='Boxcar';
Overlap=64;
Averaging='mean';
ResultScaling=2;
Smoothing={'average',0};
TrialExclude=[];
ChannelPairs=[1 2];
FileName='';
ProgressBarFlag=0;
E_O=gBSerc(P_C,RefStart,RefEnd,Lcf,Ucf,IntervalLength,Window,...
Overlap,Averaging,ResultScaling,Smoothing,ChannelPairs,TrialExclude,...
FileName,ProgressBarFlag);
```

Peri-Stimulus Time Histogram

This function plots the reactivity of triggered multichannel (spike) activity data. The PSTH is calculated in the following way: first the mean amplitude of the rectified baseline periods preceding the trigger onsets is calculated. This average multiplied by the **Threshold multiplication factor** is taken as threshold. The scatterplot shows the data points where the signal reaches the threshold. The single trials are depicted on the y-axis. These crossings are called events, each black dot in the scatterplot marks an event. The histogram depicts the number of events falling into a specified bin which is defined by the **Bin size**.

Specify BASELINE INTERVAL:

Start at and **Stop at** - specify the interval to analyze

Specify PARAMETERS:

Bin size - define the bins for the histogram

Threshold multiplication factor - threshold to show data points

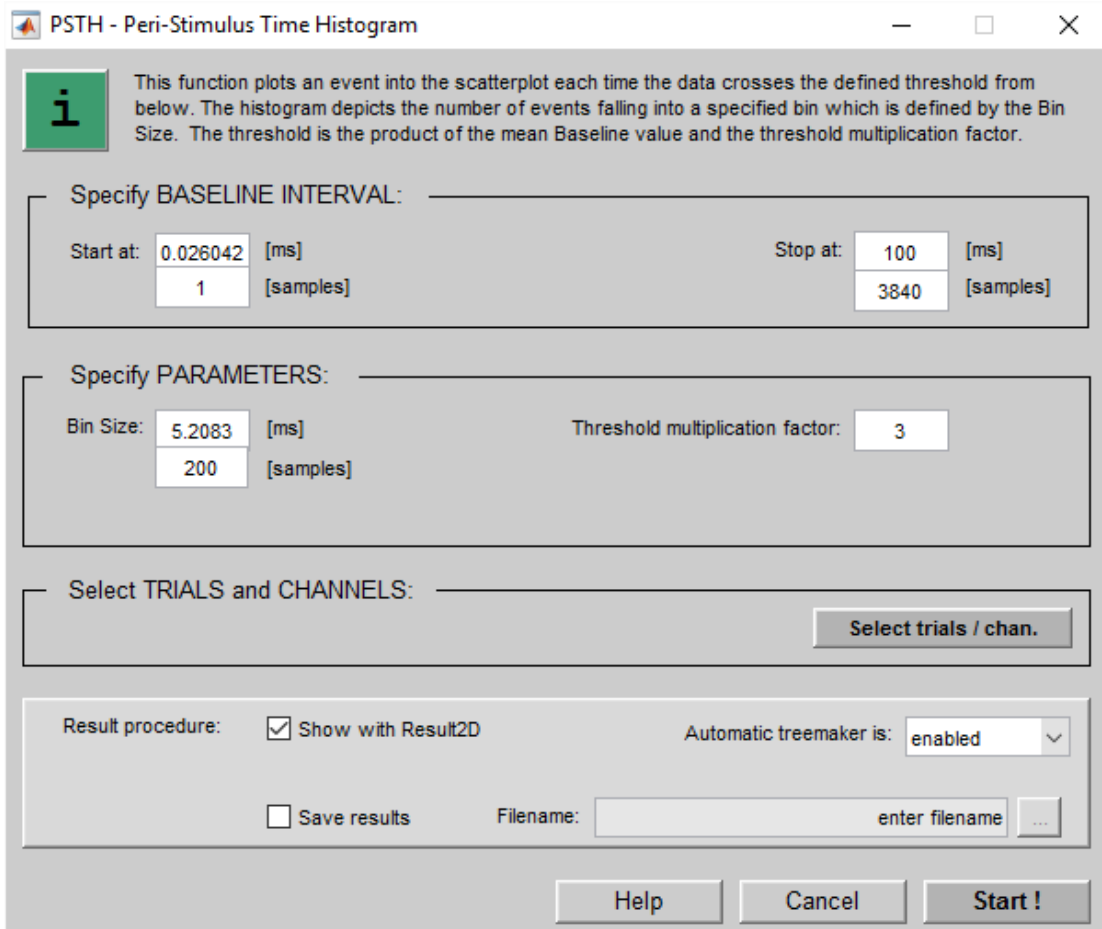
Example:

Perform the following steps to apply the PSTH:

1. Load data-set `psth1.mat` under

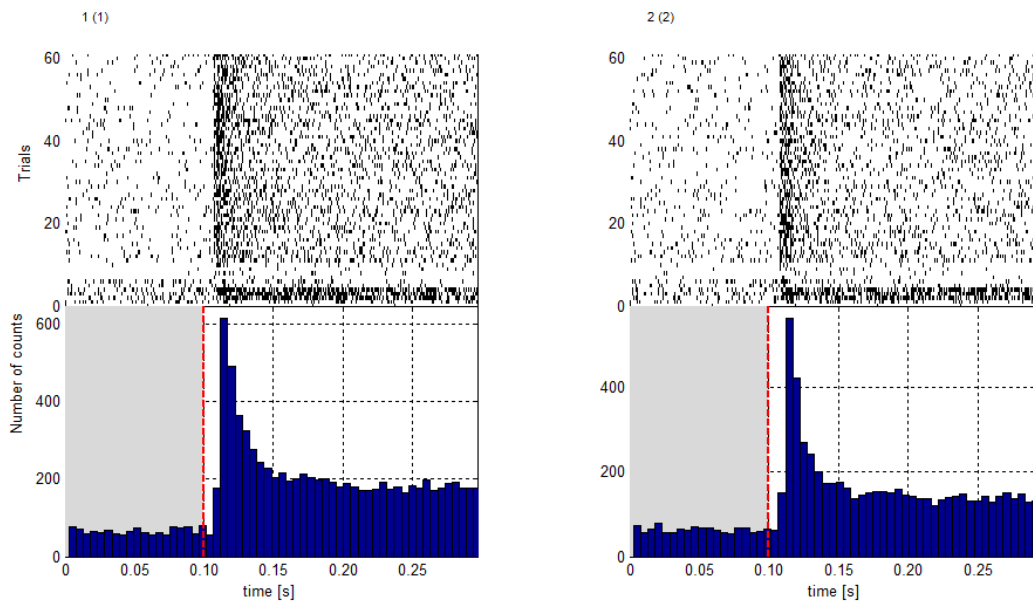
```
Documents\gtec\gBSanalyze\testdata\psth\psth1.mat;
```

2. Click on **PSTH - Peri-Stimulus Time Histogram** under the **Analyze** menu to open the following window:



3. Select for **Start at** 1 **samples** and for **Stop at** 100 **ms**, to specify the **BASELINE INTERVAL**.
4. Set the **Bin Size** to 200 **samples**.
5. Set the **Threshold multiplication factor** to 3
6. Press **Start** to apply the function.

The following result will appear:



This is the reactivity of two channels taken from the Pontine Nucleus of a rat. The reactivity was elicited via stimulation with a tone (white noise). The red vertical line depicts the trigger onset, the time where the tone was given. Data was recorded at 38400 Hz.

The following code show how to perform the example demonstrated above from the MATLAB command line.

```
%Load Data
P_C=data;
File= ['C:\Users\' getenv('USERNAME')
'\Documents\gtec\gBSanalyze\testdata\psth\psth1.mat'];
P_C=load(P_C,File);

% PSTH - Peri Stimulus Time Histogram
BLInterval = [1 3840];
BinSize = [200];
MThreshold = [3];
ChannelExclude = [];
TrialExclude = [];
FileName = '';
ProgressBarFlag = 1;
P_O = gBSpsth(P_C,BLInterval,BinSize,MThreshold,ChannelExclude,...
              TrialExclude,FileName,ProgressBarFlag);
R_O = CreateResult2D(P_O);
gResult2d(R_O);
```

P300 Accuracy

This function calculates the accuracy of a P300 classifier dependent on the number of flashes. The number of EEG channels must fit to the number of channels used to calculate the weight vector. The order of channels must be:

- EEG channels
- Flash-ID channel
- Target channel

Any other channels (e.g. the timestamp) must be removed before proceeding, by using **Cut Trials & Channels**.

The first step before calculating the P300 Accuracy is to create a classifier from the recorded data. To do that, perform the following steps:

1. Start gBSanalyze and load the recorded P300 data file. If no dataset is available you can load P300data.mat stored under:

```
Documents\gtec\gBSanalyze\testdata\P300Accuracy
```

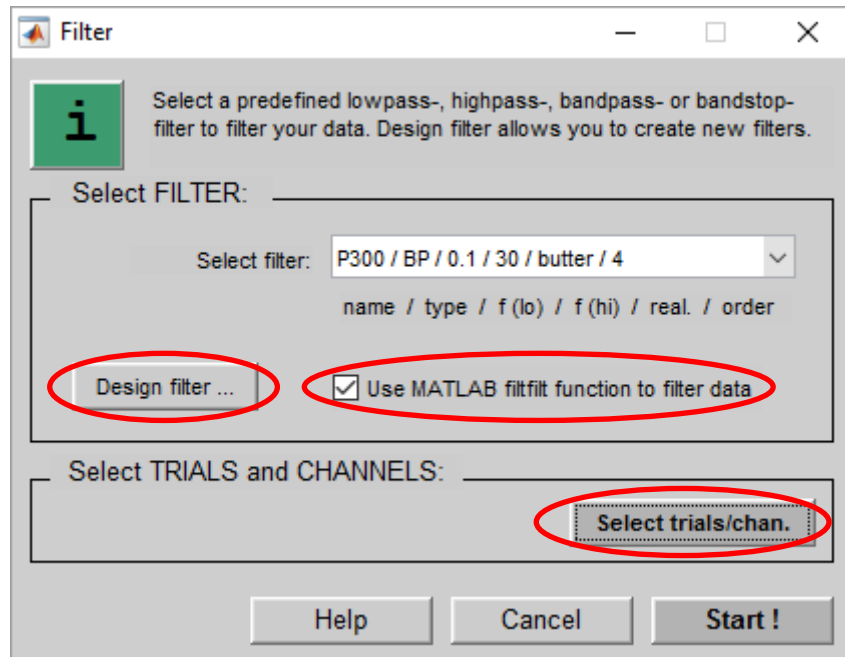
This testdata is already prepared for use and contains no timestamp. If your available raw-data file was recorded with intendix or g.P300 it contains the timestamp on the first channel. Remove it by using the **Cut Trials & Channels** function.

2. Read out the classinfo of the dataset – the classinfo describes which of the recorded trials are target-trials and which ones are nontarget-trials. To read out the classinfo copy and paste the following code into the Matlab command window (Later on, you can save the code in a file):

```
global P_C;
dat = P_C.Data;
[NTrials,NSamples,NChannels] = size(dat);
triggerData = dat(:, :, NChannels - 1);
targetData = dat(:, :, NChannels);
triggerTime = find(diff(triggerData) > 0);
targetTime = find(diff(targetData) > 0);
classInfo = zeros(2,length(triggerTime));
for NT = 1:length(triggerTime)
    if isempty(intersect(triggerTime(NT),targetTime))
        classInfo(:,NT) = [1;0];
    else
        classInfo(:,NT) = [0;1];
    end
end
clear dat;
save('P300classinfo','classInfo');
```

This code saves the classinfo in the P300classinfo.mat file.

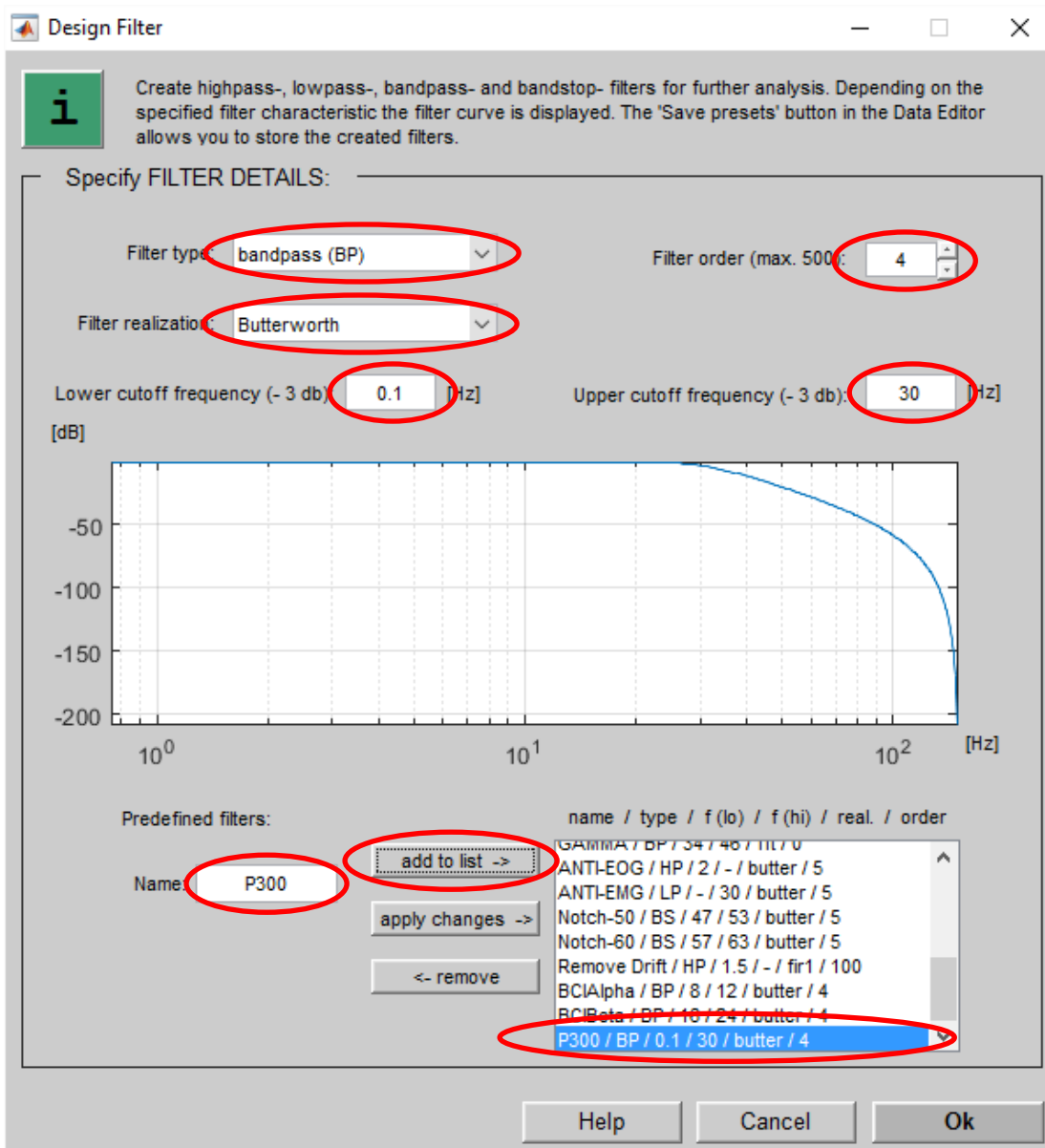
3. This step consists of filtering the data with a bandpass filter (lower cutoff frequency: 0.1 Hz; upper cutoff frequency: 30 Hz). Click in the menu on **Pre-Processing** and then on **Filter**. Check the **Use MATLAB filfilt function to filter data** box.



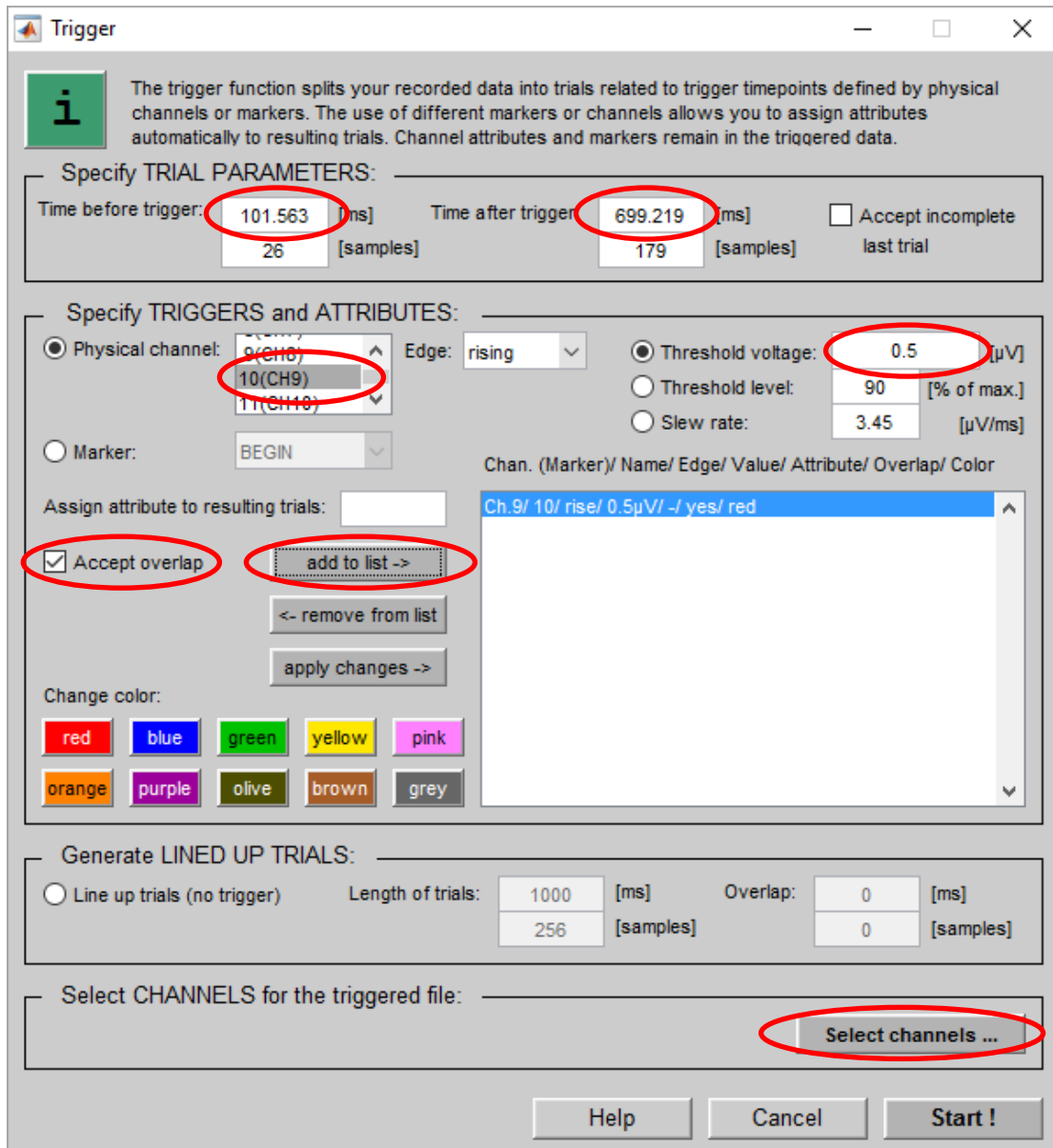
When performing this calculation for the first time, you need to define your filter. Click on **Design filter...** and enter the following settings:

- **Filter type:** bandpass (BP)
- **Filter realization:** Butterworth
- **Lower cutoff frequency (- 3 db):** 0.1 [Hz]
- **Filter order (max 500):** 4
- **Upper cutoff frequency (- 3 db):** 30 [Hz]
- **Name:** P300

Click on **add to list ->** to confirm your settings and click on **Ok** to close this window. Now you can select the P300 filter in the **Select filter:** window. Next, click on **Select trials/chan.** and exclude channels 9 and 10 to apply the filter only on the EEG data. When the filter is selected start filtering by pressing **Start !**.



4. By using the **Moving Window Filter** function from the **Pre-Processing** menu, apply a moving average filter (window length: 12 [samples]) to EEG channels 1 to 8.
5. Open the **Trigger** dialog in the **Transform** menu and set the following parameters:
 - Time before trigger: 100 ms.
 - Time after trigger: 700 ms.
 - In the pane **Specify TRIGGERS and ATTRIBUTES**, select **Physical channel** and choose 10 (CH9) from the list. As **Threshold voltage** choose 0.5 μ V.
 - Click on **Accept overlap** and **add to list->** to acknowledge your selections.
 - Click on **Select channels...** and exclude the channels 9 and 10. Now, only the triggered EEG data remain now for further processing.
 - Click on **Start!** to trigger the data.



6. Click on **Load Class Information** in the **File** menu. Press **Import Wizard**, select the file `P300classinfo.mat` that was created at step 2, and press **Finish** in the window that appears. Enter the name `NONTARGET` in the **Change name to** field and click on the button **Assign Attribute** to assign the name to the class 3. Now select class 4 with the mouse and enter `TARGET` into the **Change name to** field. Again click on **Assign Attribute** to change the name of class 4 into `TARGET`. Finally click on **OK** to assign the new classinfo to all trials.
7. Under the **Pre-Processing** menu click on **Baseline Correction**. Select 1 [samples] in the **Start at** field and 26 [samples] (~100 ms) in the **Stop at** field. Press **Start** to apply the correction onto every single trial.
8. Click on **Feature matrix** under **Classification** menu and choose the following parameters:
 - **Start at:** 27 [samples]
 - **Step:** 12 [samples]

- **Stop at:** 205 [samples]
 - **Merge time points**
 - Select the classes `NONTARGET` and `TARGET` and click on **Start** to create the Feature Matrix.
9. After calculating the Feature Matrix, the Linear Classifier window will appear automatically. Select `Train 100 % - Test 0 %` in the popup list for **Training/test-sets**. Check the **Save results** checkbox and enter the filename `P300Classifier` to save the file. The Classifier is generated by clicking **Start**.

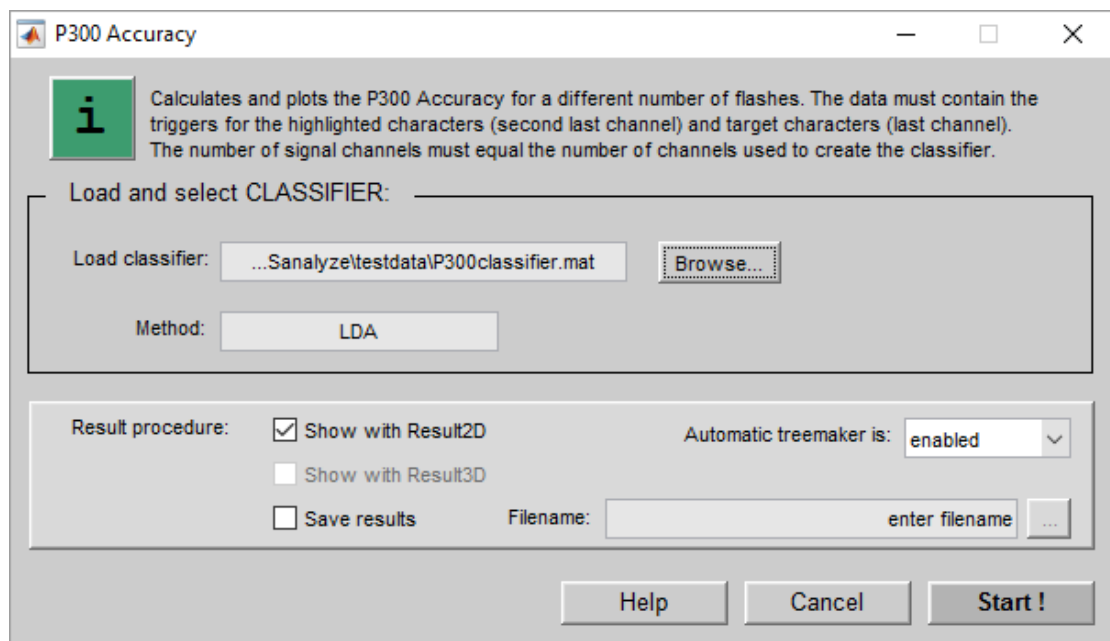
For calculating P300 accuracy, perform the following steps:

1. In `g.BSanalyze`, load your P300 dataset that you want to test, e.g.

`Documents\gtec\gBSanalyze\testdata\P300Accuracy\P300data.mat`

The file needs to contain the same number of EEG channels that were used for creating the classifier. It also must contain the triggers for the highlighted characters on the second last channel and the information regarding the characters that were selected as target characters on the last channel. For a raw datafile, the timestamp therefore needs to be removed. (Again this step is not necessary if you have loaded the file above).

2. Perform the same filtering process described at step 3 for generating the P300 classifier.
3. Click on **P300 Accuracy** under the **Analyze** menu to open the following window:

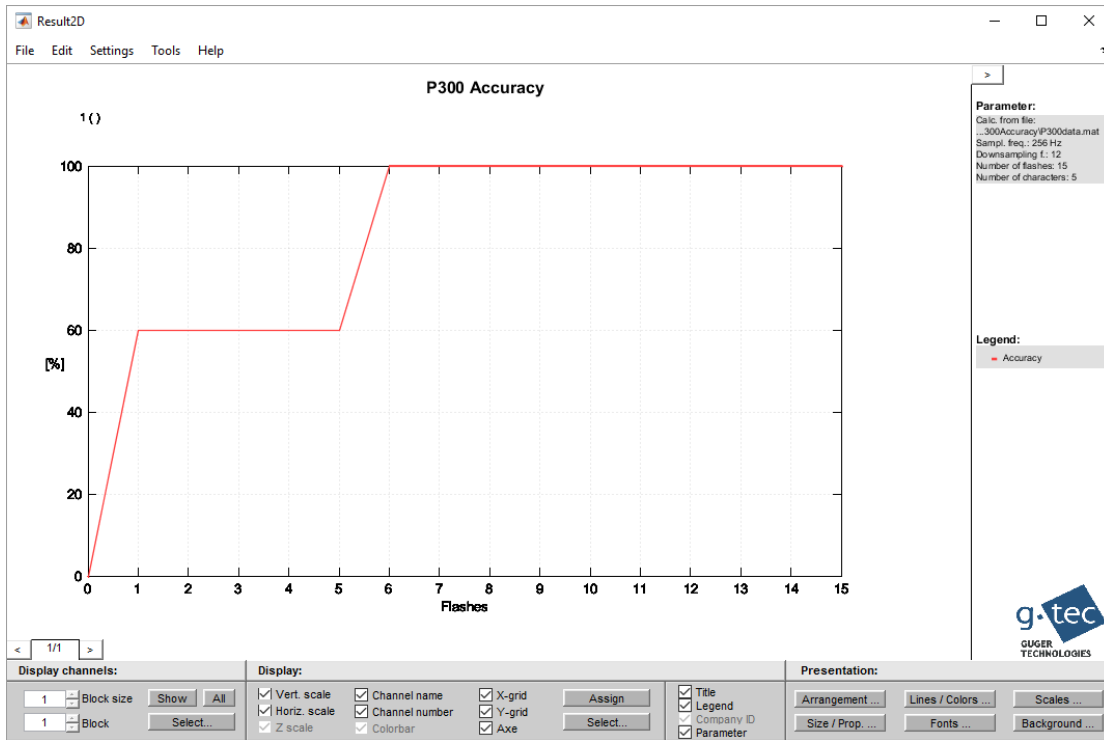


4. Press the **Browse** button and select the classifier file you created. Or, if you used the dataset presented above, select `P300classifier.mat` under:

`Documents\gtec\gBSanalyze\testdata\P300Accuracy`

5. Select **Show with Result2D** and press **Start!**

gResult2D will automatically open showing the accuracy level of the applied classifier as a function of used number of flashes used for discrimination. The number of flashes equals the number of times that each row and each column (or each character for a SC speller) is highlighted before each classification.



The following code show how to perform the example demonstrated above from the MATLAB command line.

```
%Load Data
P_C=data;
File= ['C:\Users\' getenv('USERNAME')
'\Documents\gtec\gBSanalyze\testdata\P300Accuracy\P300data.mat'];
P_C=load(P_C,File);
P_C.SamplingFrequency=256;

%load P300 Classifier
FileName= ['C:\Users\' getenv('USERNAME')
'\Documents\gtec\gBSanalyze\testdata\P300Accuracy\P300classifier.mat'];
C_O_S=classifierobj;
C_O_S=load(C_O_S,FileName);
FileName='';
ProgressBarFlag=[1];
P_O = gBSP300Accuracy(P_C, C_O_S, FileName, ProgressBarFlag);
R_O = CreateResult2D(P_O);
gResult2d(R_O);
```


Parameter Extraction

This section includes the description of the following parameter extraction algorithms:

[Hjorth](#) - calculate the Activity, Mobility and Complexity

[Barlow](#) - estimate Mean Amplitude, Mean Frequency and Spectral Purity Index

[AAR](#) - estimate adaptive autoregressive parameters with the recursive least squares (RLS), least mean squares (LMS) or Kalman algorithm

[Variance](#) - calculate the running variance of your data-set

[Bandpower](#) - extract bandpower values of a predefined frequency interval

[Minimum Energy](#) - extract the average SNR values for specific frequencies

[Exponential Window](#) - smooth the data over a specific interval

[Running Fractal Dimension](#) - calculate diagnostic information

[Temporal and Spatial Complexity](#) - calculate complexity measures

[Cross Correlation](#) - search for correlations between channels

[Cross Correlation Template Matching](#) - search for correlations between a template and a channel

[Canonical Correlation](#) - calculates the canonical correlation of the input-signal with a template signal

[Phase Locking Value](#) - calculate the phase synchrony of brain signals

[Calculate Speed](#) - calculate speed considering a specific threshold

Each **Parameter extraction** window has one section called **Select CHANNELS** or **Select**

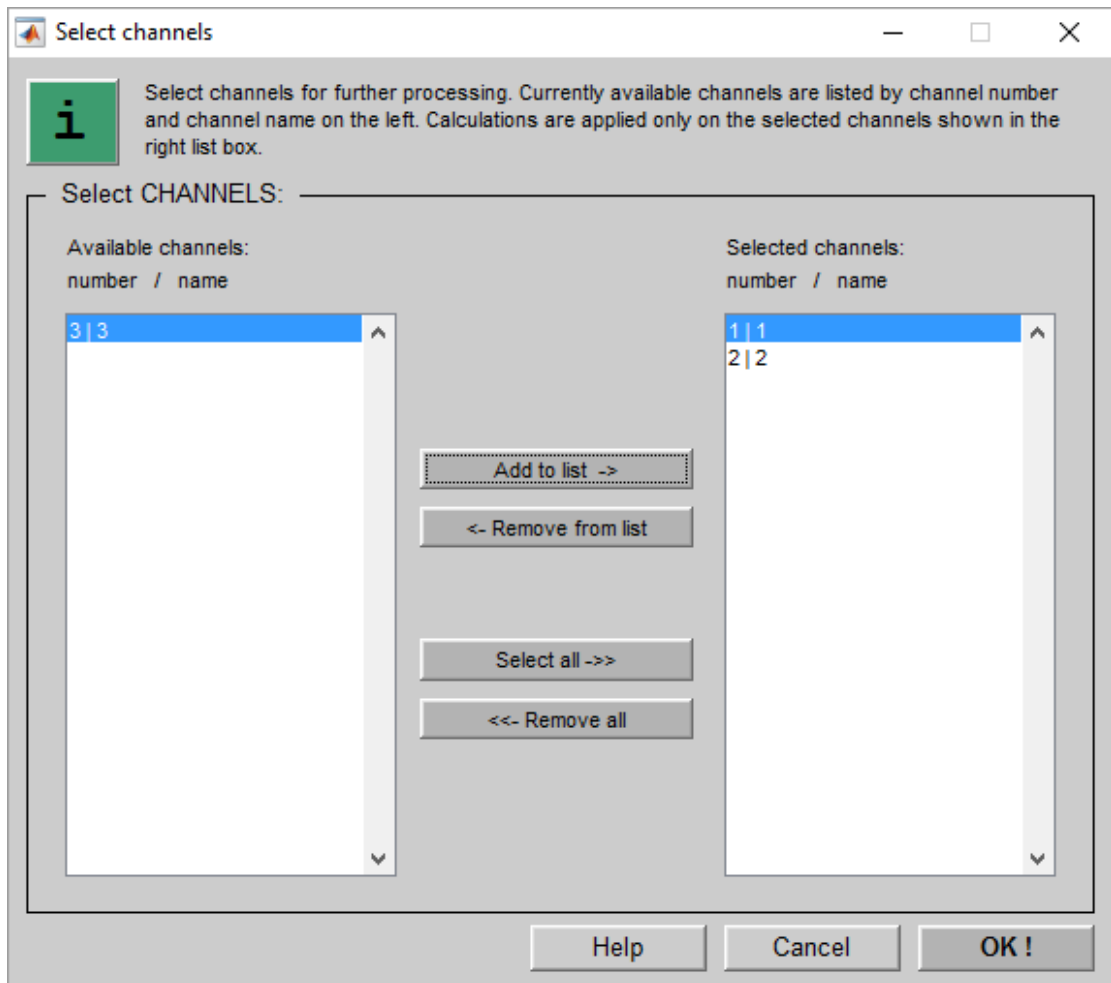
The image shows two sections of a software window. The top section is titled 'Select CHANNELS:' and contains a large empty text box on the left and a button labeled 'Select channels' on the right. The bottom section is titled 'Select PAIRS OF CHANNELS:' and contains a large empty text box on the left and a button labeled 'Select pairs ...' on the right.

PAIRS OF CHANNELS and another section called **Result procedure**.

The image shows the 'Result procedure' section of a software window. It contains three radio buttons: 'Add new channels' (unselected), 'Replace all channels' (selected), and 'Save result data' (unselected). To the right, there is a dropdown menu for 'Automatic treemaker is:' with 'enabled' selected. Below these is a 'Filename:' label followed by a text input field containing 'enter filename' and a button with three dots.

Select CHANNELS

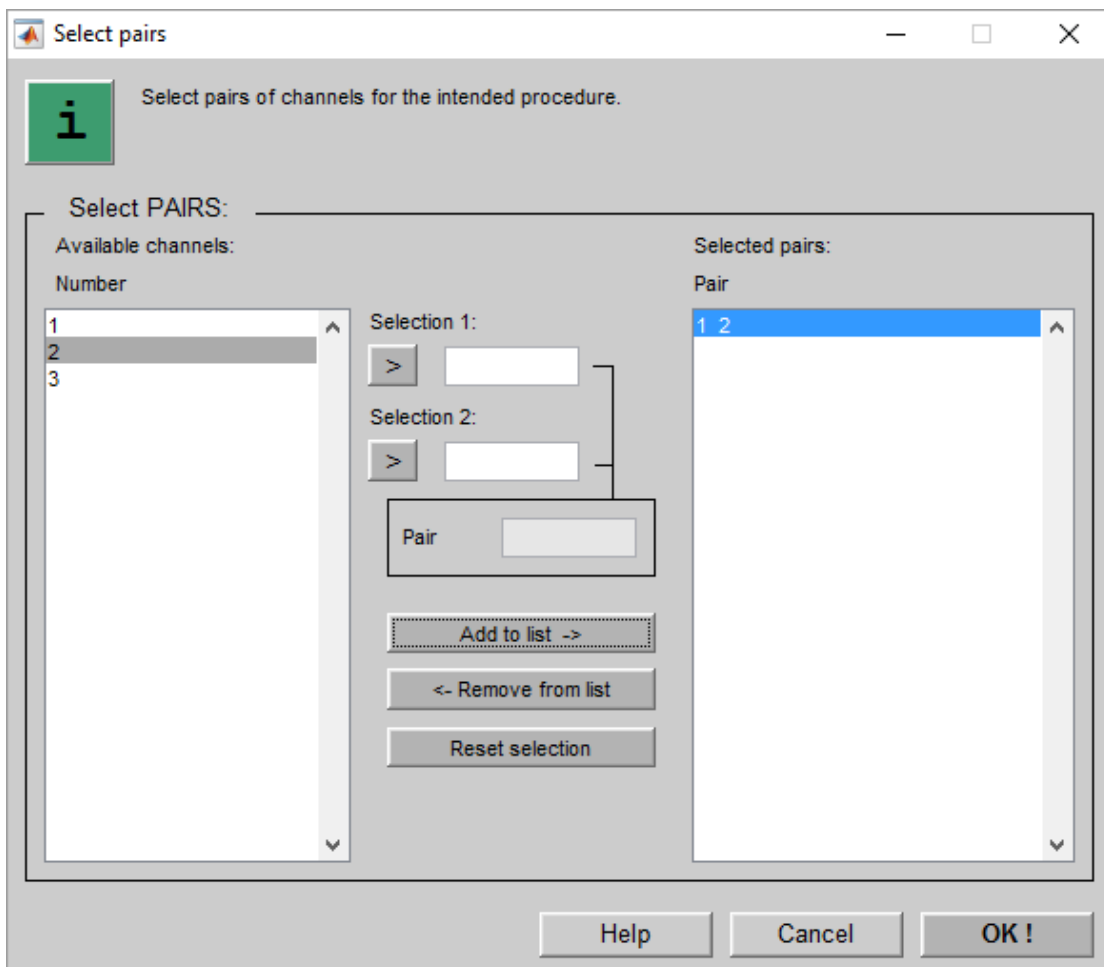
1. Press the **Select channels** button to exclude/include specific channels for calculation



2. Use the **Add to list** button and the **Remove from list** button to select channels for further calculation
3. Press the **OK** button to confirm the selection and to close the window

Select PAIRS

5. Press the **Select pairs ...** button to define channel pairs for the calculation
6. Select in the **Available channels** listbox the first channel and click on the **Selection 1** button
7. Select channel 2 and click on the **Selection 2** button and confirm the pair with **Add to list**. The new created channel pair appears in the **Selected pairs** listbox.
8. Close the window with the **OK** button



Result procedure

1. Select **Add new channels** to append the calculated features to the input data in the Data Editor or select **Replace all channels** to show only the new features

Result procedure: Add new channels Replace all channels Save result data

Automatic treemaker is:

Filename:

2. If the **Automatic treemaker** is **enabled** the results will be stored in a subdirectory of your data with the name of the parameter extraction function that is performed. If the treemaker is **disabled** the file will be stored in the data directory.
3. By clicking **Save result data** a window opens that allows to specify the location. Use the button beside to browse for a file location.

Hjorth

Hjorth introduced a method for clinical EEG analysis in 1970 [Hjorth 1970]. The method is based on an amplitude/time analysis and uses 3 parameters:

1. **ACTIVITY**, which is the variance of the EEG signal or the mean power
2. **MOBILITY**, gives a measure of the mean frequency
3. **COMPLEXITY**, shows the deviation from a sine shape

Method

ACTIVITY [μV^2]

is the mean power in time domain

$$Activity = A^2 = \frac{1}{T} \int_{t-T}^t x(t)^2 dt \quad (1)$$

which is the squared standard deviation of the EEG signal $x(t)$ or the variance (VAR) by statistical considerations as discussed from Hjorth [Hjorth 1970] and [Niedermeyer 1993].

$$Activity = VAR(x(t)) \quad (2)$$

MOBILITY [$\frac{1}{s}$]

First calculate the squared standard deviation of the slope

$$D^2 = \frac{1}{T} \int_{t-T}^t \left(\frac{dx(t)}{dt} \right)^2 dt \quad (3)$$

and with (1) we obtain

$$Mobility = \sqrt{\frac{D^2}{Activity}} = \sqrt{\frac{D^2}{A^2}} \quad (4)$$

D and A are both dependent on the mean amplitude. Therefore, the Mobility is just dependent on the curve shape of the input signal. It shows the relative average slope.

The Mobility is the standard deviation of $\frac{dx(t)}{dt}$ over standard deviation of $x(t)$.

$$Mobility = \sqrt{\frac{VAR\left(\frac{dx(t)}{dt}\right)}{VAR(x(t))}} \quad (5)$$

The dimension is cycles per second.

If the unit is $\left[\frac{1}{2\pi s}\right]$ than the Mobility is the mean frequency (gyrating frequency).

COMPLEXITY [1]

is the Mobility of $\frac{dx(t)}{dt}$

$$Mobility\left(\frac{dx(t)}{dt}\right) = \frac{\sqrt{\frac{1}{T} \int_{t-T}^t \left(\frac{d^2x(t)}{dt^2}\right)^2 dt}}{\sqrt{\frac{1}{T} \int_{t-T}^t \left(\frac{dx(t)}{dt}\right)^2 dt}} = \frac{\sqrt{VAR\left(\frac{d^2x(t)}{dt^2}\right)}}{\sqrt{VAR\left(\frac{dx(t)}{dt}\right)}} \quad (6)$$

divided by Mobility of x(t).

$$Complexity = \frac{Mobility\left(\frac{dx(t)}{dt}\right)}{Mobility(x(t))} \quad (7)$$

It shows the deviation of the slope. You can see the Complexity also as a measure of change in frequency of the input signal.

The Complexity is unity if x(t) is a sine function and it becomes bigger if the signal form deviates more from sinusoidal form. This parameter is dimensionless.

For each sample we obtain 3 characteristic values, which are describing the EEG. The main advantages of Hjorth's method are that there are only 3 values which represent the EEG for each time step and it is done without conventional frequency domain description. But there is a lack of clearness if the input signal has more than one peak in the power spectrum. This problem can be suppressed if the EEG is bandpass filtered before it is passed to the algorithm [Niedermeyer 1993].

References:

C., Guger, "Implementation of different EEG processing algorithms with S-Functions", Master thesis, University of Technology Graz, 1997.

C., Guger, "Real-time data processing under Windows for an EEG based brain-computer interface," Thesis, University of Technology Graz, 1999.

B. Hjorth, "EEG analysis based on time domain properties," *Electroencephalography and Clinical Neurophysiology*, vol. 29, pp. 306-310, 1970.

B. Hjorth, "The physical significance of time domain descriptors in EEG analysis," *Electroencephalography and Clinical Neurophysiology*, vol. 34, pp. 321-325, 1973.

E. Niedermeyer, F., Lopez da Silva, "Electroencephalography", 3rd edition, Chapter 61, Williams and Wilkins, Baltimore, 1993.

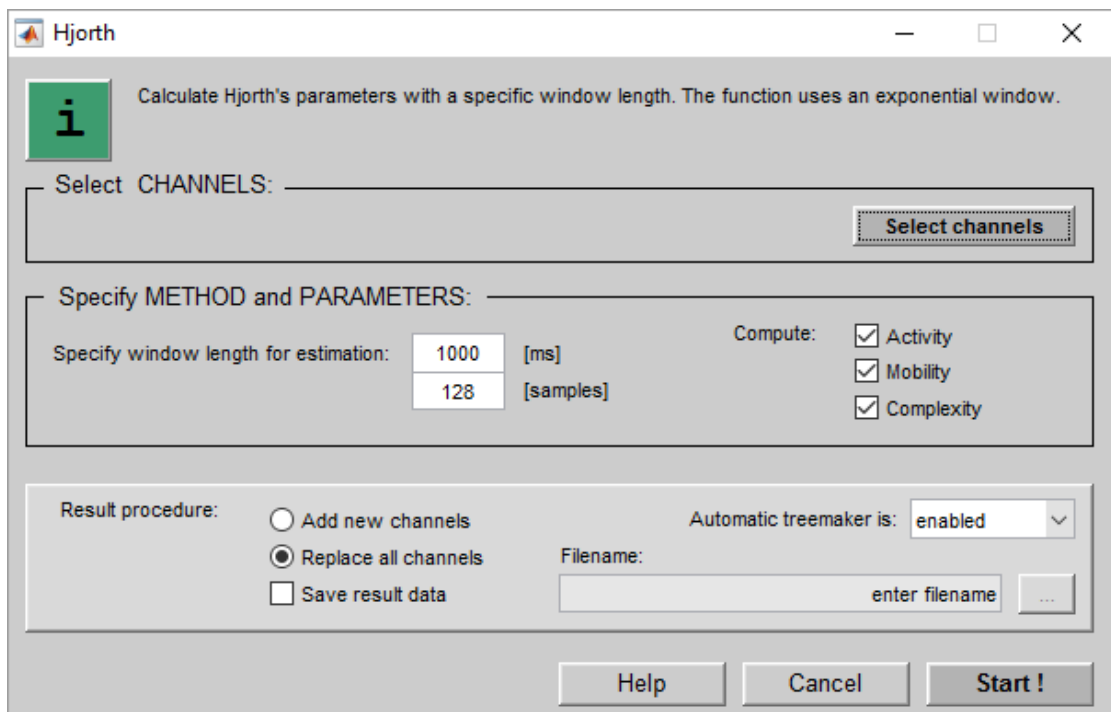
Example:

Perform the following steps to calculate the Hjorth parameters:

1. Load data-set `trig1.mat` under

`Documents\gtec\gBSanalyze\testdata\bci`

2. Click on **Hjorth** under the **Parameter Extraction** menu to open the following window:



3. Specify the width of the estimation interval as 1000 ms. The method uses an exponential windowing function.

4. Click on **Select channels** and add the appropriate channels to the list. Confirm the settings with the **OK** button.
5. Press **Start** to calculate the parameters

The following code show how to perform the example demonstrated above from the MATLAB command line.

```
% Load Data
P_C=data;
File=['C:\Users\' getenv('USERNAME')
'\Documents\gtec\gBSanalyze\testdata\BCI\trig1.mat'];
P_C=load(P_C,File);

% Hjorth
ChannelExclude = [3];
IntervalLength = 128;
Activity = 1;
Mobility = 1;
Complexity = 1;
Replace = 'replace all channels';
FileName = '';
ProgressBarFlag = 0;
P_C = gBSHjorth(P_C, ChannelExclude, IntervalLength,...
               Activity, Mobility, Complexity, Replace,...
               FileName, ProgressBarFlag);
```


Barlow

This function of the **Parameter Extraction** menu computes three Barlow parameters:

1. Mean Amplitude
2. Mean Frequency
3. Spectral Width

It is a simplified method for EEG-analysis in comparison to the normal frequency techniques [Goncharova 1990].

Method

MEAN AMPLITUDE (MA) [μV]

$$MA = \frac{1}{T} \int_{t-T}^t |x(t)| dt \quad (1)$$

which is the expected value of the input signal $x(t)$

$$MA = E|x(t)| \quad (2)$$

MEAN FREQUENCY (MF) $\left[\frac{1}{s} \right]$

shows the mean of all frequencies which are included in the input signal.

Let

$$B = \frac{1}{T} \int_{t-T}^t \left| \frac{dx(t)}{dt} \right| dt = E \left| \frac{dx(t)}{dt} \right| \quad (3)$$

and

$$MA = E|x(t)| \quad (4)$$

to obtain

$$MF = \frac{B}{MA} = \frac{E \left| \frac{dx(t)}{dt} \right|}{E|x(t)|} \quad (5)$$

which is the ratio of the running mean absolute slope to the running mean absolute amplitude (mean amplitude). The running mean absolute slope depends linearly on the frequency of the input signal if the amplitude is constant.

Assume a sinusoidal input signal, then the ratio shows the frequency of the sine wave in Hertz.

SPECTRAL PURITY INDEX (SPI) [1]

is a measure of the irregularity of a signal. It is dimensionless.

Define

$$C = \left(E \left| \frac{dx(t)}{dt} \right| \right)^2 \quad (6)$$

which is the squared running mean absolute slope (steepness). SPI is the ratio of C to the product of the mean absolute amplitude $E|x(t)|$ and the running mean absolute curvature (sharpness)

$$SPI = \frac{\left(E \left| \frac{dx(t)}{dt} \right| \right)^2}{E \left| \frac{d^2x(t)}{dt^2} \right| E|x(t)|} \quad (7)$$

The maximum of the SPI is 1 for a sine wave. It will have values less than 1, if a band of frequencies is present. Because the running mean absolute curvature extends more (caused by higher frequencies as the mean frequency) than the frequency components lower than the mean frequency.

Goncharova and Barlow showed the important role of the SPI in EEG analysis [Goncharova 1990]. If human subjects close their eyes, the mean frequency stays almost unchanged, but the SPI decreases significantly. The reliability of the result is still dependent on the amount of peaks in the power spectrum. A bandpass filter could provide help in this case.

References:

I.I., Goncharova, J.S., Barlow, "Changes in EEG mean frequency and spectral purity during spontaneous alpha blocking," *Electroencephalography and Clinical Neurophysiology*, vol. 76, pp. 197-204, 1990.

C., Guger, "Implementation of different EEG processing algorithms with S-Functions", Master thesis, University of Technology Graz, 1997.

C., Guger, "Real-time data processing under Windows for an EEG based brain-computer interface," Thesis, University of Technology Graz, 1999.

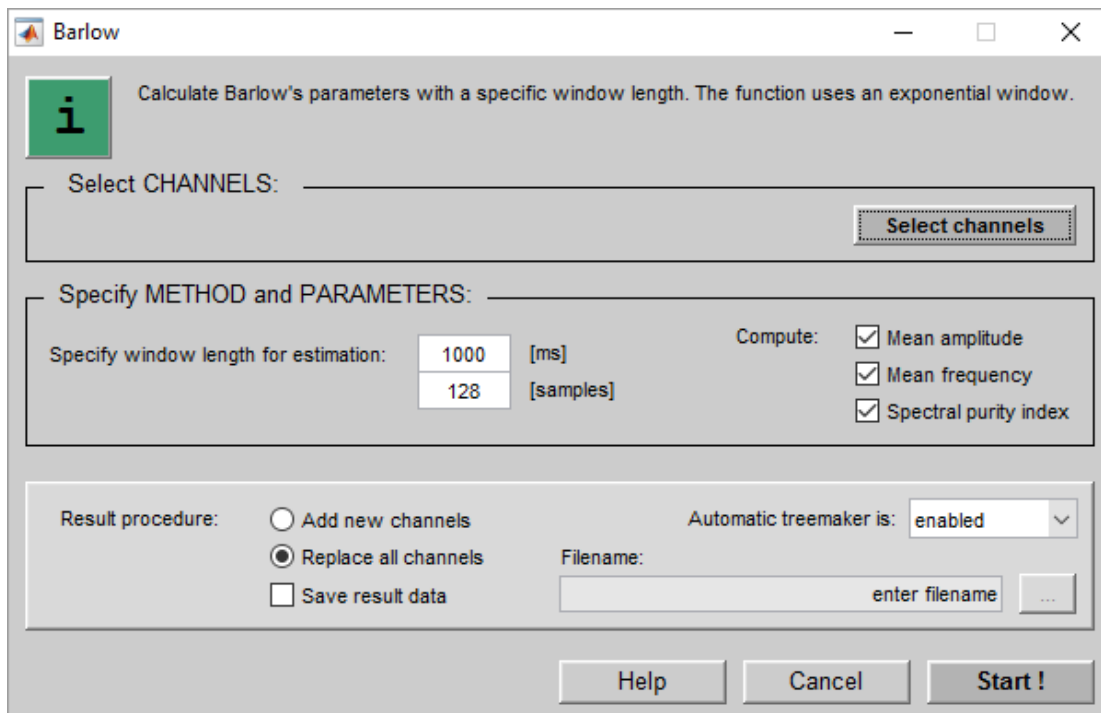
Example:

Perform the following steps to calculate the **Barlow** parameters:

1. Load data-set `trig1.mat` under

`Documents\gtec\gBSanalyze\testdata\bci`

2. Click on **Barlow** under the **Parameter Extraction** menu to open the following window:



3. Specify the width of the estimation interval as 1000 ms. The method uses an exponential windowing function.
4. Click on **Select channels** and add the appropriate channels to the list. Confirm the settings with the **OK** button.
5. Press **Start** to calculate the parameters

The following code show how to perform the example demonstrated above from the MATLAB command line.

```
%Load Data
P_C=data;
File= ['C:\Users\' getenv('USERNAME')
'\Documents\gtec\gBSanalyze\testdata\BCI\trig1.mat'];
P_C=load(P_C,File);

%Barlow
ChannelExclude = [3];
IntervalLength = 128;
MeanAmplitude = 1;
MeanFrequency = 1;
SPI = 1;
Replace = 'replace all channels';
SaveData = 0;
FileName = '';
ProgressBarFlag = 0;
P_C = gBSbarlow(P_C, ChannelExclude, IntervalLength,MeanAmplitude,...
                MeanFrequency, SPI, Replace, FileName,...
                ProgressBarFlag);
```

AAR

This function of the **Parameter Extraction** menu calculates adaptive autoregressive (AAR) parameters. In the first part of this section an adaptive autoregressive model (AAR) for analyzing event-related EEG is discussed and in the second part the least mean squares (LMS) algorithm and the recursive least squares (RLS) algorithm are introduced.

The ARMA, AR and AAR model

A stationary EEG signal can be described with an AR model using a few coefficients. The values of these coefficients show the signal's time-varying properties [Niedermeyer 1993]. An autoregressive moving average (ARMA) process of order p,q is described by the input signal ε_k which is a white noise signal with zero mean that is passed to the filter. The output of the filter operation is the EEG signal x_k and the filter corresponding to the autoregressive moving average model as described by the following linear difference equation [Niedermeyer 1993]:

$$a_0x_k - a_1x_{k-1} - \dots - a_px_{k-p} = b_0\varepsilon_k + b_1\varepsilon_{k-1} + \dots + b_q\varepsilon_{k-q} \quad (1)$$

where $q \leq p$.

The coefficients $a_0=1$, $a_1 \dots a_p$ and $b_1 \dots b_q$ give the relation between x_k and ε_k . For $b_1 \dots b_q=0$ we obtain the autoregressive model (AR) [Niedermeyer 1993]

$$x_k - a_1x_{k-1} - \dots - a_px_{k-p} = \varepsilon_k \quad (2)$$

Therefore the computation problem is the estimation of the coefficients. Such an AR-model is used to describe a stationary EEG, but for event-related EEG an adaptive autoregressive model is suitable. An AAR model is used to show variations of the signal. The term adaptive shows that the AR-parameters are time-varying. The AAR model has the following structure [Niedermeyer 1993]:

$$x_k - a_{1,k}x_{k-1} - \dots - a_{p,k}x_{k-p} = \varepsilon_k \quad (3)$$

With this equation we can predict x_k with a weighted linear combination of our past values with an resulting error ε_k .

The model is of order p and $a_{1,k} \dots a_{p,k}$ are the time varying coefficients. The parameters of the AAR model are estimated at each iteration. The estimation is done with the least mean squares (LMS), the recursive least squares (RLS) or Kalman algorithms.

Least Mean Squares - LMS

In Equation (3) the weights are selected to cause an output which is as similar as possible to the sample, x_k . The prediction error depends on the past coefficients $a_1 \dots a_p$ and the EEG signal. We can use the fact that the EEG is better described with a smaller error ε_k , therefore we define the Mean Square Error $MSE(\varepsilon)$

$$MSE(\varepsilon) = \xi = \frac{1}{N} \sum_{k=1}^N \varepsilon_k^2 \quad (4)$$

Minimizing the $MSE(\varepsilon)$ has many advantages which are discussed in [Haykin 1986, Solo and Widrow 1985].

For the following considerations the p AR-parameters are written in vector notation

$$\mathbf{a}_k = \begin{bmatrix} a_{1,k} \\ a_{2,k} \\ \cdot \\ \cdot \\ a_{p,k} \end{bmatrix} \quad (5)$$

and the past p samples of the time series are represented as

$$\mathbf{x}_{k-1} = \begin{bmatrix} x_{k-1} \\ x_{k-2} \\ \cdot \\ \cdot \\ x_{k-p} \end{bmatrix} \quad (6)$$

And now the LMS-algorithm is characterized by the following two equations

$$\varepsilon_k = x_k - x_{k-1}a_{1,k-1} - x_{k-2}a_{2,k-1} \dots x_{k-p}a_{p,k-1} \quad (7)$$

$$\mathbf{a}_k = \mathbf{a}_{k-1} + \mu \varepsilon_k \mathbf{x}_{k-1} \quad (8)$$

Where

\mathbf{a}_k is the new weight vector

\mathbf{a}_{k-1} is the old weight vector

μ controls the speed of adaptation and stability

ε_k is the error signal

x_k is the new sample of the time series

$x_{k-1} \dots x_{k-p}$ are the past samples of the time series

The weights are changed at each iteration step by the correction term $\mu \varepsilon_k \mathbf{x}_{k-1}$. A good initial guess for the weight vector is $\mathbf{a}=0$ [Haykin 1986].

The advantages of the LMS algorithm are:

- no differentiation
- no averaging
- no squaring
- the estimates of the gradients are made on-line
- the calculation is easy (just multiplication and addition)
- no matrix inversion

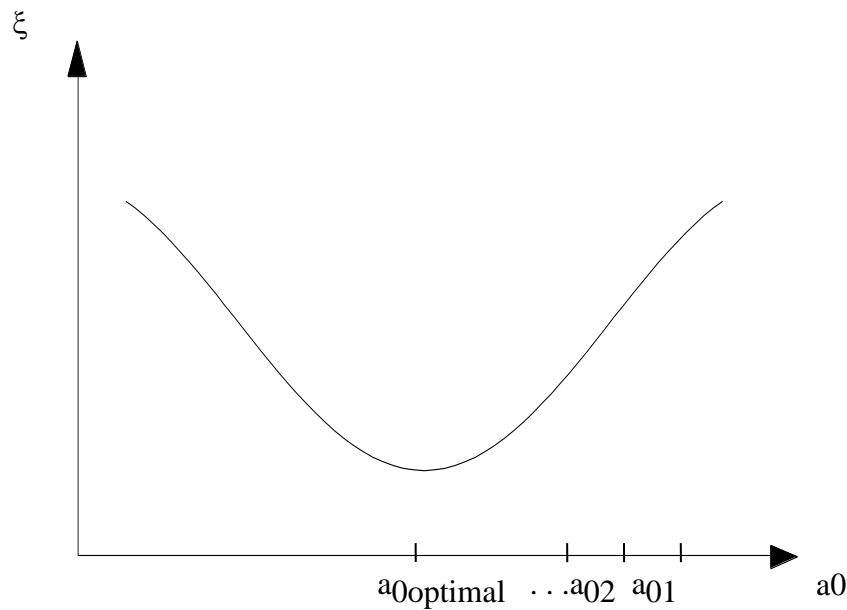
When the algorithm is started it contains noise, but it is suppressed by adaptation.

Practical ways to adjust the weight vector

To show how the LMS algorithm operates a simple example is shown in the following section with a stationary signal assumed.

In many applications no analytic description is available. But it is possible to calculate the ξ , which is an estimate of the points on the surface. Therefore procedures for estimating the optimal weight vector are necessary. These algorithms are able to find an almost optimal solution.

For simplicity a system with just one weight is assumed. The performance surface of this system is shown in the following figure:



One weight performance surface.

The first step is to initialize the weight vector, a_{00} . The slope of the performance surface is estimated at this point and subtracted from a_{00} , this yields a_{01} . After that a_{02} , a_{03} ,...until $a_{0optimal}$ are calculated. At point $a_{0optimal}$ the slope is zero and we have found the optimal weight for a stationary signal. In an not stationary environment the performance surface would change its shape. Obviously the problem is the estimation of the slope. These methods are thus called ‘Gradient search algorithms’.

Formally written

$$a_{k+1} = a_k + \mu(-\nabla_k) \tag{9}$$

Where

a_{k+1} is the new weight

a_k is the old weight

μ is the step size and controls the speed of adaptation

∇_k is the estimated gradient at a_k

Implementation of the LMS algorithm

Event-related EEG can be analyzed with the LMS-algorithm, by just looking at the actual value of the weights at a specific time point, because the weights represent the actual mental state. The order is determined from the number of peaks in the spectrum.

The LMS algorithm is implemented according to

$i=1\dots p$

$$\varepsilon_k = x_k - x_k - 1a_{1,k-1} - x_k - 2a_{2,k-1} \dots x_k - pa_{p,k-1} \quad (10)$$

$$a_{i,k} = a_{i,k-1} + \mu \varepsilon_k x_{k-i} \quad (11)$$

Where ε_k is the prediction error and μ is the step size parameter. Schack et al [Schack 1993] proposed to chose μ in such a way that

$$\mu = \frac{\text{UpdateCoefficient}}{\text{totalpower}} = \frac{\text{UpdateCoefficient}}{\text{VAR}(\mathbf{x}_k)} \quad (12)$$

In this way every input signal is reverred to the variance and has the advantage that the update coefficient can be chosen independently of the signal power.

Initialization of the LMS

was done according to

$$a_{i,k}=0 \quad i=1\dots p \quad k \leq i$$

$$\varepsilon_0=0$$

Recursive Least Squares - RLS

The weights of a transversal filter are adjusted for each set of input data, starting from known initial conditions at time k, therefore for each set there exists a different filter.

The RLS-algorithm uses all the information that is contained in the input data extended back to the initial conditions.

It converges faster than the LMS but the computational complexity is higher.

The initialization of the RLS needs a starting value that ensures the nonsingularity of the correlation matrix Φ_k .

Let

$$\Phi_{(0)} = \delta \mathbf{I} \quad (14)$$

where the choice of δ is of minor importance for long data length k. \mathbf{I} is $p \times p$.

The weight vector $\hat{\mathbf{a}}_k$ ($p \times 1$) is customary set to

$$\hat{\mathbf{a}}(0) = \mathbf{0} \quad (15)$$

Define Φ_k ($p \times p$) as the correlation matrix

$$\Phi_k = \sum_{i=1}^k \lambda^{k-i} \mathbf{x}_i \mathbf{x}_i^H \quad (16)$$

and the cross correlation matrix Θ_k ($p \times 1$) by

$$\Theta_k = \sum_{i=1}^k \lambda^{k-i} \mathbf{x}_i \mathbf{x}_i^* \quad (17)$$

where \mathbf{x}_i is the input vector of the past samples and \mathbf{x}_k is the new sample.

Both, Φ_k and Θ_k are weighted with λ^{k-i} , which can be seen as a forgetting factor. λ is close to, but less than one and it controls the adaptation.

Φ_k and Θ_k can be written as recursion

$$\Phi_k = \lambda \Phi_{k-1} + \mathbf{x}_k \mathbf{x}_k^H \quad (18)$$

and

$$\Theta_k = \lambda \Theta_{k-1} + \mathbf{x}_k \mathbf{x}_k^* \quad (19)$$

In the method of exponentially weighted least squares we minimize the index of performance [Haykin 1986]

$$\Gamma_k = \sum_{i=1}^k \lambda^{k-i} |\varepsilon_i|^2 \quad (20)$$

where ε_i is defined in Equation (28).

The optimum tap-weight vector, $\hat{\mathbf{a}}_k$, for which Γ_k reaches its minimum may be achieved from

$$\Phi_k \hat{\mathbf{a}}_k = \Theta_k \quad (21)$$

by inversion of Φ_k , which can be very time consuming if the number of weights, p , is high.

Also $\hat{\mathbf{a}}_k$ is computed recursively at each time step. Therefore we use the basic matrix inversion lemma [Haykin 1986] to rewrite equation (25) and (26), define

$$\mathbf{P}_k = \Phi_k^{-1} \quad (22)$$

and let the gain vector \mathbf{k}_k ($p \times 1$) be

$$\mathbf{k}_k = \frac{\lambda^{-1} \mathbf{P}_{k-1} \mathbf{x}_k}{1 + \lambda^{-1} \mathbf{x}_k^H \mathbf{P}_{k-1} \mathbf{x}_k} \quad (23)$$

Let

$$\mathbf{r}_k = \lambda^{-1} \mathbf{P}_{k-1} \mathbf{x}_k \quad (24)$$

of dimension $(p \times 1)$ and insert into equation 23 to obtain

$$\mathbf{k}_k = \frac{\mathbf{r}_k}{1 + \mathbf{x}_k^T \mathbf{r}_k} \quad (25)$$

Further we obtain the matrix \mathbf{P}_k that must be adapted and stored in each iteration step for computing the gain vector

$$\mathbf{P}_k = \lambda^{-1} \mathbf{P}_{k-1} - \mathbf{k}_k \mathbf{r}_k^T \quad (26)$$

with dimension $(p \times p)$.

These last two equations update the gain vector and the matrix inversion of Φ_k is replaced by a scalar division for faster computation.

The recursive equation for updating the least squares estimate for the tap-weight vector $\hat{\mathbf{a}}_k$ is

$$\hat{\mathbf{a}}_k = \hat{\mathbf{a}}_{k-1} + \mathbf{k}_k \varepsilon_k \quad (27)$$

that describes the adaptive operation of the algorithm. It is updated by adding a priori estimation error ε_k times the gain vector to the old weight vector.

The priori estimation error is defined by

$$\varepsilon_k = x_k - \hat{\mathbf{a}}_{k-1}^H \mathbf{x}_k \quad (28)$$

which is the difference of the new sample and the estimated sample, calculated with the old tap vector.

Implementation of the RLS algorithm

Compute in this order for each time step:

$$\mathbf{r}_k = \lambda^{-1} \mathbf{P}_{k-1} \mathbf{x}_k \quad (29)$$

$$\mathbf{k}_k = \frac{\mathbf{r}_k}{1 + \mathbf{x}_k^T \mathbf{r}_k} \quad (30)$$

$$\varepsilon_k = x_k - \hat{\mathbf{a}}_{k-1}^H \mathbf{x}_k \quad (31)$$

$$\hat{\mathbf{a}}_k = \hat{\mathbf{a}}_{k-1} + \mathbf{k}_k \varepsilon_k \quad (32)$$

$$\mathbf{P}_k = \lambda^{-1} \mathbf{P}_{k-1} - \mathbf{k}_k \mathbf{k}_k^T \quad (33)$$

Initialization of the RLS

Initialize the algorithm by

$$\hat{\mathbf{a}}(0) = 0 \quad (34)$$

$$\mathbf{P}_{(0)} = \delta^{-1} \mathbf{I} \quad (35)$$

Digital Implementation of the RLS

There are problems with round off errors mentioned, that can cause the algorithm to diverge [Haykin 1986]. This problem occurs when long data-sets are analyzed and depends on the size of λ^{-1} . If λ^{-1} becomes bigger than the algorithms becomes earlier unstable. Also artefacts might cause that the algorithm becomes unstable. To solve the problem the update coefficient must be lower or the algorithm must be initialized periodically [Guger 1997, Guger 1999]. Therefore, trigger your data-set and apply the algorithm to each trial.

Main advantages of RLS:

- it converges faster than the LMS
- the estimate is more accurate
- no matrix inversion

References:

C., Guger, "Implementation of different EEG processing algorithms with S-Functions", Master thesis, University of Technology Graz, 1997.

C., Guger, "Real-time data processing under Windows for an EEG based brain-computer interface," Thesis, University of Technology Graz, 1999.

S., Haykin, "Adaptive Filter Theory," Prentice Hall, Englewood Cliffs, New Jersey, 1986.

E., Niedermeyer, F., Lopez da Silva, "Electroencephalography," 3rd edition, Chapter 61, Williams and Wilkins, Baltimore, 1993.

B., Schack, H., Witte, G., Griebbach, „Parametrische Methoden der dynamischen Spektralanalyse und ihre Anwendung in der Biosignalanalyse," Biomedizinische Technik, vol. 38, pp. 79-80, 1993.

A. Schloegl, "The electroencephalogram and the adaptive autoregressive model: theory and applications," Shaker Verlag, Aachen, Germany, 2000.

V., Solo, X., Kong, "Adaptive Signal Processing Algorithms," Prentice Hall. Englewood Cliffs, New Jersey.

B., Widrow, "Adaptive signal processing," Prentice Hall, Englewood Cliffs, New Jersey, 1985.

Example:

Perform the following steps to calculate the AAR parameters:

1. Load data-set `trig1.mat` under

`Documents\gtec\gBSanalyze\testdata\bci`

2. Click on **AAR** under the **Parameter Extraction** menu to open the following window:

AAR

Calculate adaptive autoregressive (AAR) parameters with the following algorithms: Kalman estimation of AAR parameters, recursive least squares (RLS), variance normalized least mean squares (LMS1) or least mean squares with adaptive variance normalization (LMS2).

Select CHANNELS: _____ Select channels

Specify METHOD and PARAMETERS: _____

Method:

Order:

Constant update coefficient:

Load UC from file: View

Search optimal UPDATE COEFFICIENT: _____ Create UC

Result procedure: Add new channels Replace all channels Save result data

Automatic treemaker is:

Filename

Help Cancel Start !

3. **Specify METHOD and PARAMETERS** allows to select the estimation algorithm (RLS, LMS1, LMS2, Kalman), the model order and an update coefficient. Select the method `RLS`, with order `6` and an update coefficient of `0.006` for all channels.
RLS – recursive least squares [Guger 1997, 1999]
LMS1 – variance normalized least mean squares [Guger 1997, 1999]
LMS2 – least mean squares with adaptive variance normalization
Kalman – Kalman estimation of AAR parameters [Schlögl 2000]
4. Click on **Select channels** and add the appropriate channels to the list. Confirm the settings with the **OK** button.
5. Press **Start** to calculate the parameters

The following code show how to perform the example demonstrated above from the MATLAB command line.

%Load Data

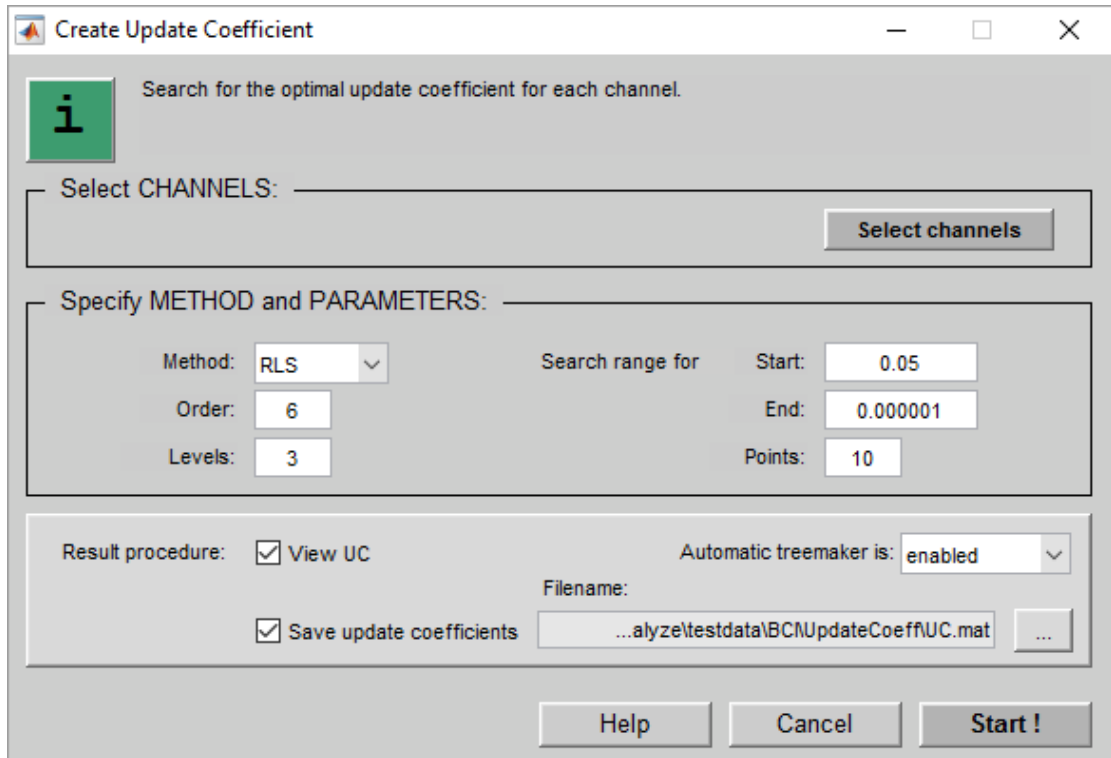
```
P_C=data;  
File= ['C:\Users\' getenv('USERNAME')  
'\Documents\gtec\gBSanalyze\testdata\BCI\trig1.mat'];  
P_C=load(P_C,File);
```

%AAR

```
ChannelExclude = [3];  
Method = 'RLS';  
Order = 6;  
UC(1) = [0.006];  
UC(2) = [0.006];  
Replace = 'replace all channels';  
FileName = '';  
ProgressBarFlag = 0;  
P_C = gBSaar(P_C, ChannelExclude, Method, Order, UC,...  
Replace, FileName, ProgressBarFlag);
```

Search for optimal UPDATE COEFFICIENT allows to calculate the optimal update coefficients for each channel and to store the results for later analysis.

1. Click on **Create Update Coefficient** to open the following window:



2. Select the **Method** `RLS` with **Order** `6` and specify the **Start** and **End** value for searching for the update coefficients.

Start: specifies the largest possible value of an update coefficient

End: specifies the smallest possible value of an update coefficient

Levels: specifies the searching iterations. In the first iteration the **Start** and **End** borders are used for the calculation, in the next iteration the borders are taken from the best UC found during the first iteration.

Points: The search interval is divided into equally spaced intervals. The point with the smallest prediction error is taken as update coefficient.

3. Check **View ...** to open the MATLAB Editor with the results and click on **Save update coefficients** to store the result
4. Click **Start** to calculate the update coefficient

Variance

This function of the **Parameter Extraction** menu computes the variance of the selected channels for each channel.

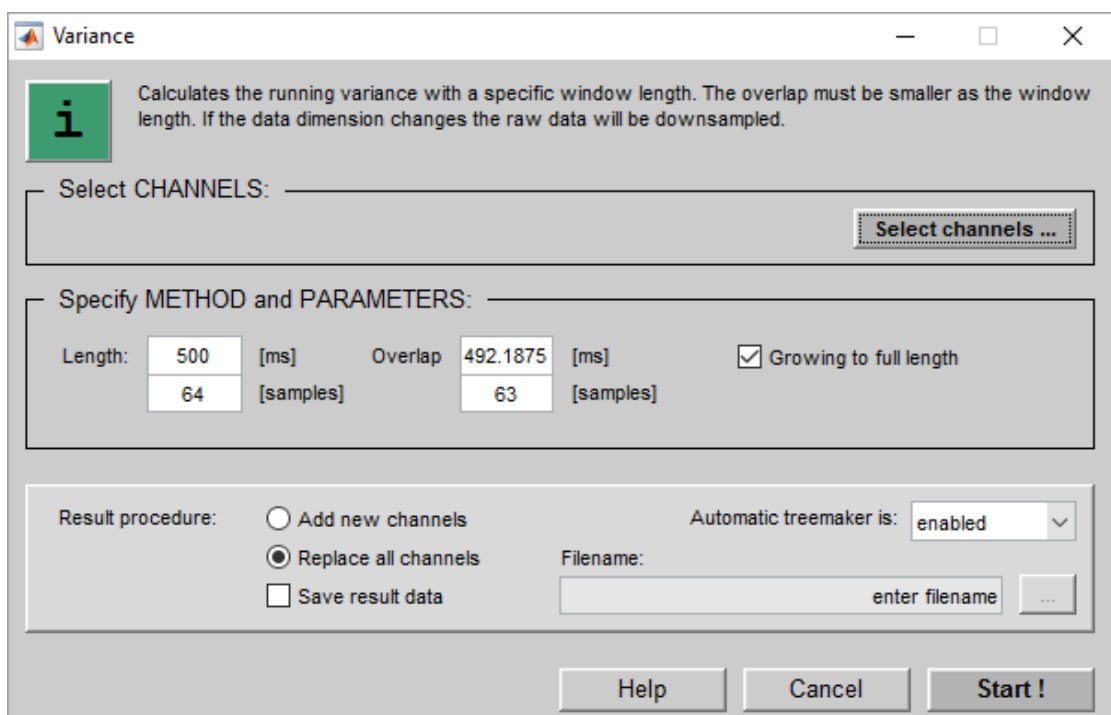
References:

C. Guger, H. Ramoser, and G. Pfurtscheller, "Real-time EEG analysis for a brain-computer interface (BCI) with subject-specific spatial patterns," IEEE Trans. Rehab. Engng., vol. 8, 2000.

Example:

Perform the following steps to calculate the Variance:

1. Load data-set `trig1.mat` under
`Documents\gtec\gBSanalyze\testdata\bci`
2. Click on **Variance** under the **Parameter Extraction** menu to open the following window:



3. Specify the length of the estimation interval as 500 ms. The method uses an rectangular windowing function. Check **Growing to full length** to adjust the window length at the beginning of the data-set from 1 sample up to the specified window length. This backwards orientated window is moved through the data-set. The variance is calculated within this window and assigned to the last element of it.
4. Click on **Select channels** and add the appropriate channels to the list. Confirm the settings with the **OK** button.
5. Press **Start** to calculate the parameters

The following code show how to perform the example demonstrated above from the MATLAB command line.

```
% Load Data
P_C=data;
File= ['C:\Users\' getenv('USERNAME')
'\Documents\gtec\gBSanalyze\testdata\BCI\trig1.mat'];
P_C=load(P_C,File);

% Variance
ChannelExclude = [];
IntervalLength = 64;
GrowingWindow = 1;
Overlap = 63;
Replace = 'replace all channels';
FileName = '';
ProgressBarFlag = 0;
P_C = gBSvariance(P_C, ChannelExclude, IntervalLength,...
GrowingWindow, Overlap, Replace, FileName, ProgressBarFlag);
```

Bandpower

This function of the **Parameter Extraction** menu computes the bandpower in a certain frequency range of the selected channels. The Bandpower is estimated by digitally bandpass filtering the data, squaring and averaging over consecutive samples according to the window length.

The window allows the following settings:

Design new filter - define lowpass, highpass, bandpass and bandstop filters

Choose a filter for the estimation – select a predefined filter for bandpower calculation

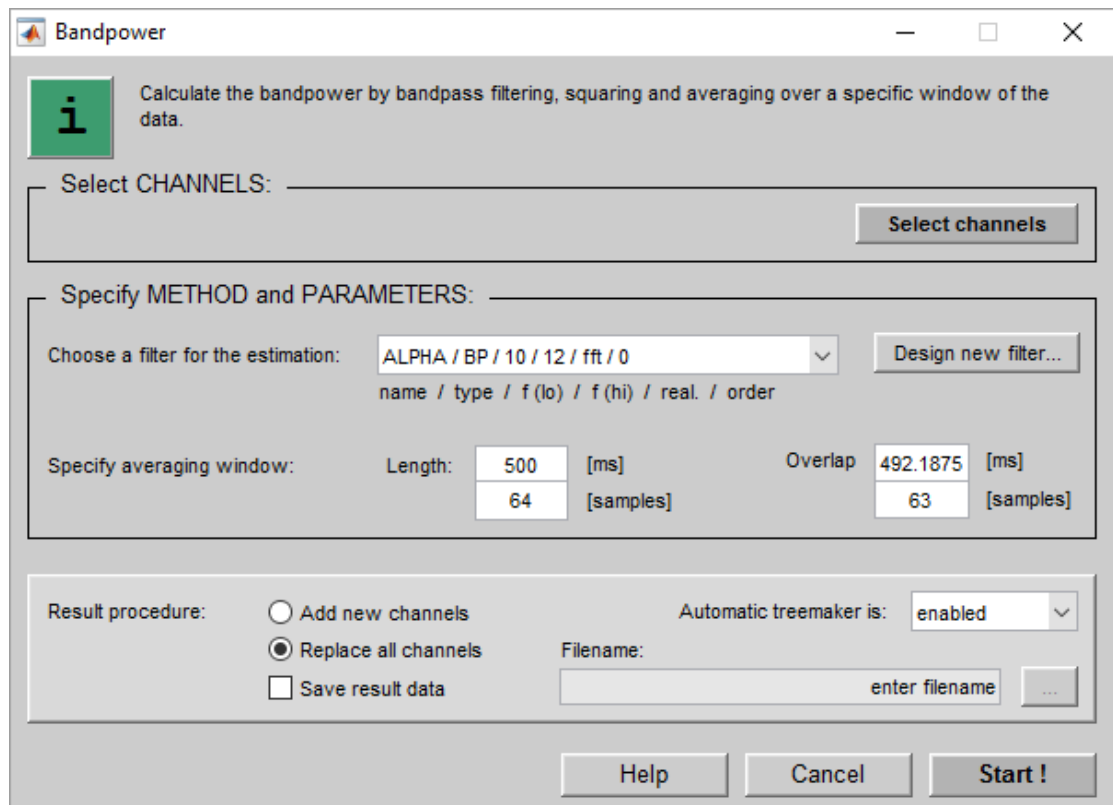
Specify averaging window - specify the width of the averaging interval. This backwards-oriented window is moved through the data-set. The Bandpower is calculated within this window and assigned to the last element of it.

Perform the following steps to calculate the Bandpower:

1. Load data-set `trigl.mat` under

`Documents\gtec\gBSanalyze\testdata\bci`

2. Click on **Bandpower** under the **Parameter Extraction** menu to open the following window:



The screenshot shows the 'Bandpower' dialog box with the following settings:

- Select CHANNELS:** (Empty text field)
- Choose a filter for the estimation:** ALPHA / BP / 10 / 12 / fft / 0
- Specify averaging window:**
 - Length: 500 [ms], 64 [samples]
 - Overlap: 492.1875 [ms], 63 [samples]
- Result procedure:**
 - Add new channels
 - Replace all channels
 - Save result data
- Automatic treemaker is:** enabled
- Filename:** enter filename

Buttons: Help, Cancel, Start !

3. Select an FFT filter from 10 to 12 Hz
4. Specify the length of the estimation interval as 500 ms.
5. Click on **Select channels ...** and add the appropriate channels to the list. Confirm the settings with the **OK** button.
6. Press **Start** to calculate the parameters

The following code show how to perform the example demonstrated above from the MATLAB command line.

%Load Data

```
P_C=data;
File= ['C:\Users\' getenv('USERNAME')
'\Documents\gtec\gBSanalyze\testdata\BCI\trig1.mat'];
P_C=load(P_C,File);
```

%Bandpower

```
ChannelExclude = [3];
Filter.Name = 'ALPHA';
Filter.Type = 'BP';
Filter.f_low = [10];
Filter.f_high = [12];
Filter.Realization = 'fft';
Filter.Order = [1];
IntervalLength = 64;
Overlap = 63;
Replace = 'replace all channels';
FileName = '';
ProgressBarFlag = 0;
P_C = gBSbandpower(P_C, ChannelExclude, Filter, IntervalLength,...
    Overlap, Replace, FileName, ProgressBarFlag);
```

Minimum Energy

This function of the **Parameter Extraction** menu computes the average signal to noise ratios for sinusoid signals of certain frequencies across the selected channels. The signal to noise ratios are computed using the Minimum Energy approach developed by Friman et Al.

The window allows the following settings:

- **Frequencies** - Specify a comma separated list of frequencies for which the average signal to noise ratios shall be computed
- **Number of harmonics** - Define the number of higher order harmonics which shall be considered. If set to 0 no higher order harmonics will be considered
- **Order of AR model** - The order of the autoregressive model which is used to estimate the noise contained in each channel
- **Specify window** - Specify the **Length** of the interval for which the SNR value shall be computed. This backwards-oriented window is moved through the data-set and reevaluated every **Evaluation step**. The average SNR is calculated within this window and assigned to the last element of it

Minimum Energy

Calculates the minimum energy of a given data set. The output signals represent the signal-to-noise ratio (SNR) for the specified frequency with respect to the base EEG-signal.

Select CHANNELS:

Specify METHOD and PARAMETERS:

Frequencies [Hz]: (e.g.: 10,11,12,13)

Number of harmonics: Order of AR model:

Specify window: Length: [ms] Evaluation step: [ms]
 [samples] [samples]

Result procedure: Add new channels Replace all channels Save result data

Automatic treemaker is:

Filename:

References: O. Friman, I. Volosyak and A. Gräser, Multiple Channel Detection of Steady-State Visual Evoked Potentials for Brain-Computer Interfaces, IEEE Transactions on Biomedical Engineering, 54 (4), pp 743 - 750, 2007.

Example:

Perform the following steps to apply the minimum energy:

8. Load data-set `training_testsubj1.mat` under

```
Documents\gtec\gBSanalyze\testdata\SSVEP_MinimumEnergy
```

and specify a Sampling frequency of 256 Hz.

9. Click on **Minimum Energy** under the **Parameter Extraction** menu.
10. Select **Frequencies** by specifying 10, 11, 12, 13 Hz for the list of **Frequencies**.
11. Include the first order harmonics (20Hz, 22Hz, 24Hz and 26Hz) by setting **Num of harmonics** to 1.
12. Set the **Order of AR model** to 7.
13. Set the window length to 3000 ms and the **Evaluation step** size to 51.
14. Click on **Select Channels** and add the appropriate channels 2 - 9 to the list. Confirm the settings with the **OK** button.
15. Press **Start** to compute the SNR values for each frequency.

The following code shows how to perform the example demonstrated above from the MATLAB command line.

```
%Load Data
P_C=data;
File= ['C:\Users\' getenv('USERNAME')
'\Documents\gtec\gBSanalyze\testdata\SSVEP_MinimumEnergy\training_testsubj1
.mat'];
P_C=load(P_C,File);

P_C.SamplingFrequency=256;

% Minimum energy
ChannelExclude = [1 10 11];
Frequencies = [10 11 12 13];
NumberOfHarmonics = 1;
ModelOrder = 7;
IntervalLength = 768;
EvaluationStep = 51;
Replace = 'add channels';
FileName = '';
ProgressBarFlag = 0;
P_C=gBSminimumenergy(P_C,ChannelExclude,Frequencies,...
NumberOfHarmonics,ModelOrder, IntervalLength, EvaluationStep,...
Replace, FileName, ProgressBarFlag);
```

Exponential Window

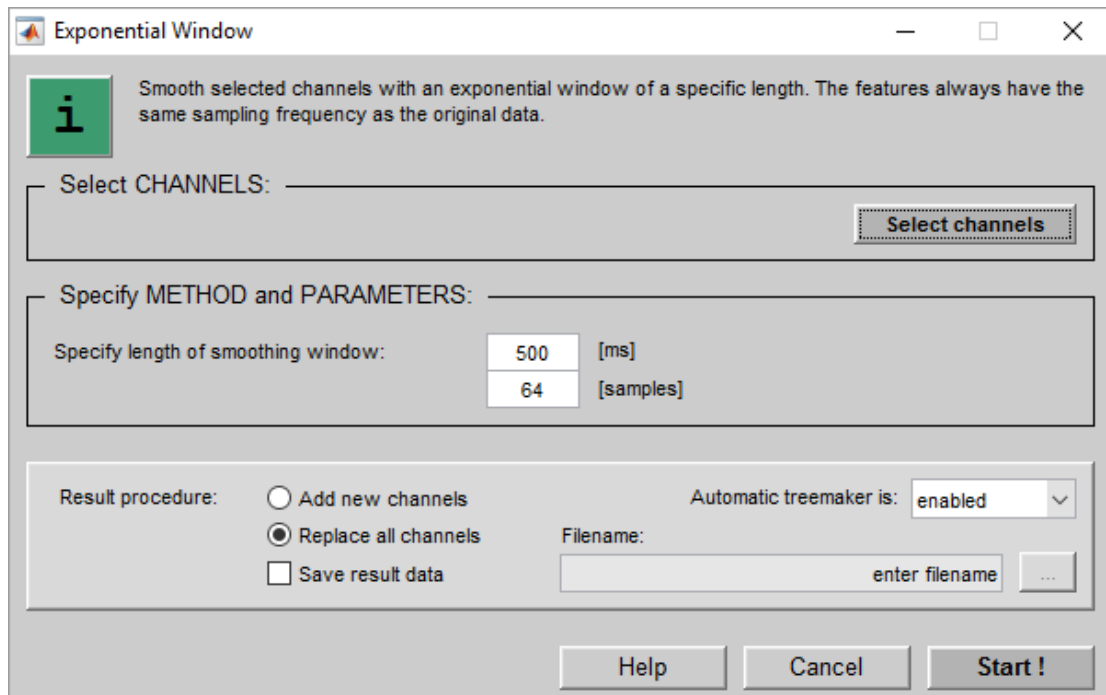
This function of the **Parameter Estimation** menu smoothes the data of the selected channels.

References:

A.Schlögl, *Time-varying autoregressive models with application to EEG*, Master thesis, TU-Graz, 1995.

Perform the following steps:

1. Load data-set `trig1.mat` under
Documents\gtec\gBSanalyze\testdata\bci
2. Click on **Exponential Window** under the **Parameter Extraction** menu to open the following window:



3. **Specify length of smoothing window as 500 ms.** This backwards orientated window is moved through the data-set. The result is assigned to the last element of it.
4. Click on **Select channels** and add the appropriate channels to the list. Confirm the settings with the OK button.
5. Press **Start** to apply the windowing function.

The following code show how to perform the example demonstrated above from the MATLAB command line.

%Load Data

```
P_C=data;  
File=['C:\Users\' getenv('USERNAME')  
'\Documents\gtec\gBSanalyze\testdata\BCI\trig1.mat'];  
P_C=load(P_C,File);
```

%Exponential Window

```
ChannelExclude = [3];  
IntervalLength = 64;  
Replace = 'replace all channels';  
FileName = '';  
ProgressBarFlag = 0;  
P_C = gBSexponentialwindow(P_C, ChannelExclude, IntervalLength,...  
                           Replace, FileName, ProgressBarFlag);
```

Running Fractal Dimension

The Fractal dimension of biosignals is computed using Higuchi's algorithm. The method provides diagnostically important information for e.g. epilepsy onset identification and provides a data compression depending on the selected overlap.

The algorithm analyzes a specific interval at once by splitting this interval into k subdivisions, calculating the fractal dimension and taking the next interval specified by the overlap.

References:

J., Virkkala, A., Värri, S.L., Himanen, J., Hasan, "Fractal dimension of EEG in sleep onset", Proceedings of 3rd European Interdisciplinary School on Nonlinear Dynamics for System and Signal Analysis, Warsaw, June 18 to June 27, 2002

W., Klonowski, J. Ciszewski, W. Jernajzyk, K. Niedzielska, "Application of chaos theory and fractal analysis for EEG signal processing in patients with seasonal affective disorder", Proceedings of 1999 International Symposium on Nonlinear Theory and its Applications (NOLTA'99), 1999, Waikoloa, Hawaii, U.S.A., 339-342.

The window allows the following settings:

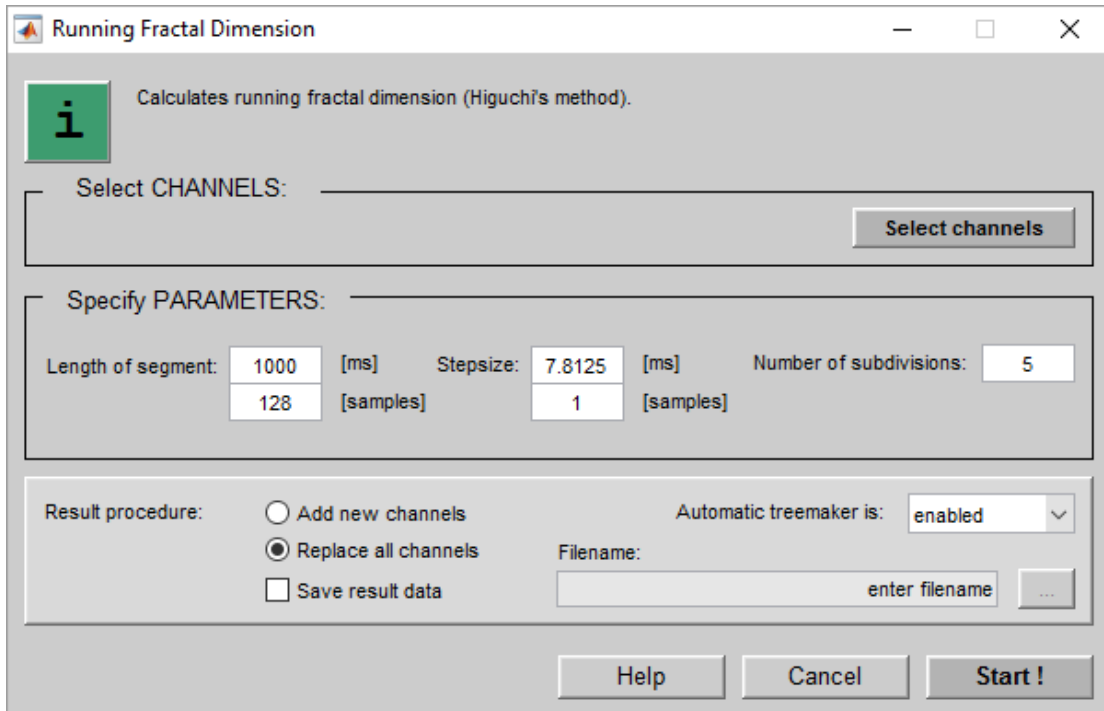
Length of segment - specify the interval to analyze in ms or samples

Stepsize – overlap between the segments

Number of subdivisions – the interval is split into subdivisions for the calculation of the fractal dimension for each interval

Perform the following steps:

1. Load data-set `trig1.mat` under
`Documents\gtec\gBSanalyze\testdata\bci`
2. Click on **Running Fractal Dimension** under the **Parameter Extraction** menu to open the window
3. Select as **Length of segment** 128 samples, as **Stepsize** 1 sample and as **Number of subdivisions** 5
4. Click on **Select Channels** and add the appropriate channels to the list. Confirm the settings with the **OK** button.
5. Press **Start** to perform the calculation



The following code show how to perform the example demonstrated above from the MATLAB command line.

%Load Data

```
P_C=data;
File=['C:\Users\' getenv('USERNAME')
'\Documents\gtec\gBSanalyze\testdata\BCI\trig1.mat'];
P_C=load(P_C,File);
```

%Running Fractal Dimension

```
ChannelExclude = [3];
FDParam.length = [128];
FDParam.step = [1];
FDParam.subd = [5];
Replace = 'add channels';
FileName = '';
ProgressBarFlag = 1;
P_C = gBSfractaldimension(P_C, ChannelExclude, FDParam, Replace,...
FileName, ProgressBarFlag);
```

Temporal and Spatial Complexity

Normally the EEG is described by amplitude or frequency measures. Temporal and spatial complexity can be used to measure the data sequence complexity. They are independent of a precise frequency content and therefore are well suited as a robust measure across individuals.

References:

S.J. Roberts, W. Penny, I. Rezek, "Temporal and spatial complexity measures for EEG-based brain-computer interfacing" *Medical & Biological Engineering and Computing*, vol 37, No. 1, pp. 93-99, 1998.

The window allows the following settings:

Method - select one of the following methods:

Spatial Complexity

Temporal Complexity

Spatial-Temporal Complexity

Length of segment - specify the interval to analyze in ms or samples

Shift – overlap between the segments

Averaging – can be mean or median

Embedding Dimension – the interval is repeatedly windowed. The **Embedding Dimension** determines the repetitions.

Lag – lag between the samples in the interval

Temporal Spatial Complexity

Estimate the temporal complexity separately for each channel or estimate the spatial and spatial-temporal complexity for all channels (results only in one feature vector).

Select CHANNELS:

Specify METHOD/ PARAMETERS:

Method: Temporal Complexity

Length of segment: 2000 [ms] / 256 [samples] Shift: 7.8125 [ms] / 1 [samples] Averaging: Mean / Median

Embedding Dimension: 10 Lag: 7.8125 [ms] / 1 [samples]

Result procedure: Add new channels Automatic treemaker is: enabled

Replace all channels Filename: enter filename

Save result data

Example:

Perform the following steps to calculate the temporal complexity:

1. Load data-set `trig1.mat` under
`Documents\gtec\gBSanalyze\testdata\bci`
2. Click on **Temporal Spatial Complexity** under the **Parameter Extraction** menu
3. Select the **Method** `Temporal Complexity` with default settings
4. Click on **Select Channels** and add the appropriate channels to the list. Confirm the settings with the **OK** button.
5. Perform the calculation with **Start**

The following code show how to perform the example demonstrated above from the MATLAB command line.

```
%Load Data
P_C=data;
File=['C:\Users\' getenv('USERNAME')
'\Documents\gtec\gBSanalyze\testdata\BCI\trig1.mat'];
P_C=load(P_C,File);

%Temporal Spatial Complexity
ChannelExclude = [3];
TSCParam.mode = [1];
TSCParam.segsize = [256];
TSCParam.shift = [1];
TSCParam.med = [0];
TSCParam.embdim = [10];
TSCParam.lag = [1];
Parameter = 'replace input channels';
SaveData = 0;
FileName = '';
ProgressBarFlag = 1;
P_C = gBStscomplexity(P_C, ChannelExclude, TSCParam, Parameter,...
                    SaveData, FileName, ProgressBarFlag);
```

Cross Correlation

The cross correlation function can be used to find correlations between 2 channels. The cross-correlation is performed between a moving segment with a specific length between channel X and channel Y.

The window allows the following settings:

Specify CROSS-CORRELATION INTERVAL:

Length of interval - specify the interval to analyze in ms or samples

Overlap – overlap between the segments

Select CROSS-CORRELATION LAG and SCALE:

Lag maximum - computes the cross-correlation over the range of lags:

MAXLAG to MAXLAG, which gives $2*MAXLAG+1$ cross-correlation results.

Scale - *biased* - scales the raw cross-correlation by $1/M$. M is the length of the data-set.

unbiased - scales the raw correlation by $1/(M-abs(lags))$.

coeff - normalizes the sequence so that the auto-correlations at zero lag are identically 1.0.

none - no scaling (this is the default).

The screenshot shows the 'Cross Correlation' dialog box with the following settings:

- Specify CROSS-CORRELATION INTERVAL:**
 - Length of interval: 1000 [ms] (with a secondary input of 128 [samples])
 - Overlap: 992.1875 [ms] (with a secondary input of 127 [samples])
- Select CROSS-CORRELATION LAG and SCALE:**
 - Lag maximum: 0
 - Scale: none
- Select PAIRS OF CHANNELS:** A 'Select pairs ...' button is visible.
- Result procedure:**
 - Add new channels
 - Replace all channels
 - Save result data
- Automatic treemaker is:** enabled
- Filename:** enter filename

Buttons at the bottom: Help, Cancel, Start

Example:

Perform the following steps to calculate the cross-correlation between channel 1 and 2.:

1. Load data-set `trig1.mat` under
`Documents\gtec\gBSanalyze\testdata\bci`
2. Click on **Cross Correlation** under the **Parameter Extraction** menu
3. Use **Select pairs ...** to define channel 1 and channel 2 for the operation
4. Perform the operation with **Start**

The following code show how to perform the example demonstrated above from the MATLAB command line.

```
%Load Data
P_C=data;
File=['C:\Users\' getenv('USERNAME')
'\Documents\gtec\gBSanalyze\testdata\BCI\trig1.mat'];
P_C=load(P_C,File);

%Cross Correlation
T.IntervalLength=128;
T.Overlap=127;
T.Lag=0;
T.Scale='none';
ChannelPairs=[1 2];
Replace=['replace all channels'];
FileName='';
ProgressBarFlag=0;
P_C=gBScrosscorrelation(P_C,T,ChannelPairs,Replace,FileName,...
ProgressBarFlag);
```

Cross Correlation Template Matching

The function allows to calculate the cross correlation between a template and a feature channel loaded in g.BSanalyze. The template can be created with the **Average** function in the **Analyze** menu and can be a vector of a MATLAB file.

The window allows the following settings:

Load or import TEMPLATE:

Browse – search for a *.mat file that was created with the **Analyze** function or a vector stored in a MATLAB file

Template waveshape – gives a preview of the template

Specify OPTIONS:

Normalization – check to normalize the cross correlation

Scaling - *biased* - scales the raw cross-correlation by $1/M$. M is the length of the data-set.

unbiased - scales the raw correlation by $1/(M-\text{abs}(\text{lags}))$.

coeff - normalizes the sequence so that the auto-correlations at zero lag are identically 1.0.

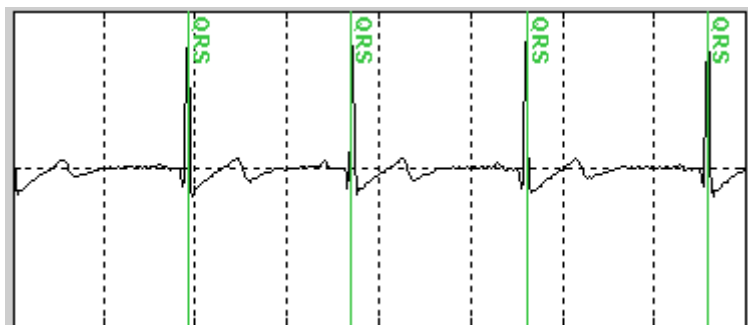
none - no scaling (this is the default).

Perform the following steps to calculate a template and to perform a cross-correlation with an ECG signal:

1. Open the file `session1.mat` from

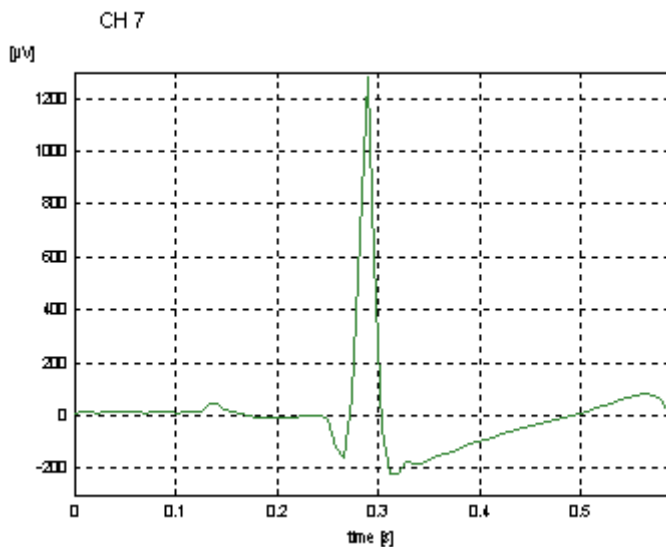
```
Documents\gtec\gBSanalyze\testdata\Aircraft_Simulator
```

2. Set QRS markers at the R-peaks in the Data Editor to mark about 10 complexes for setting up the template

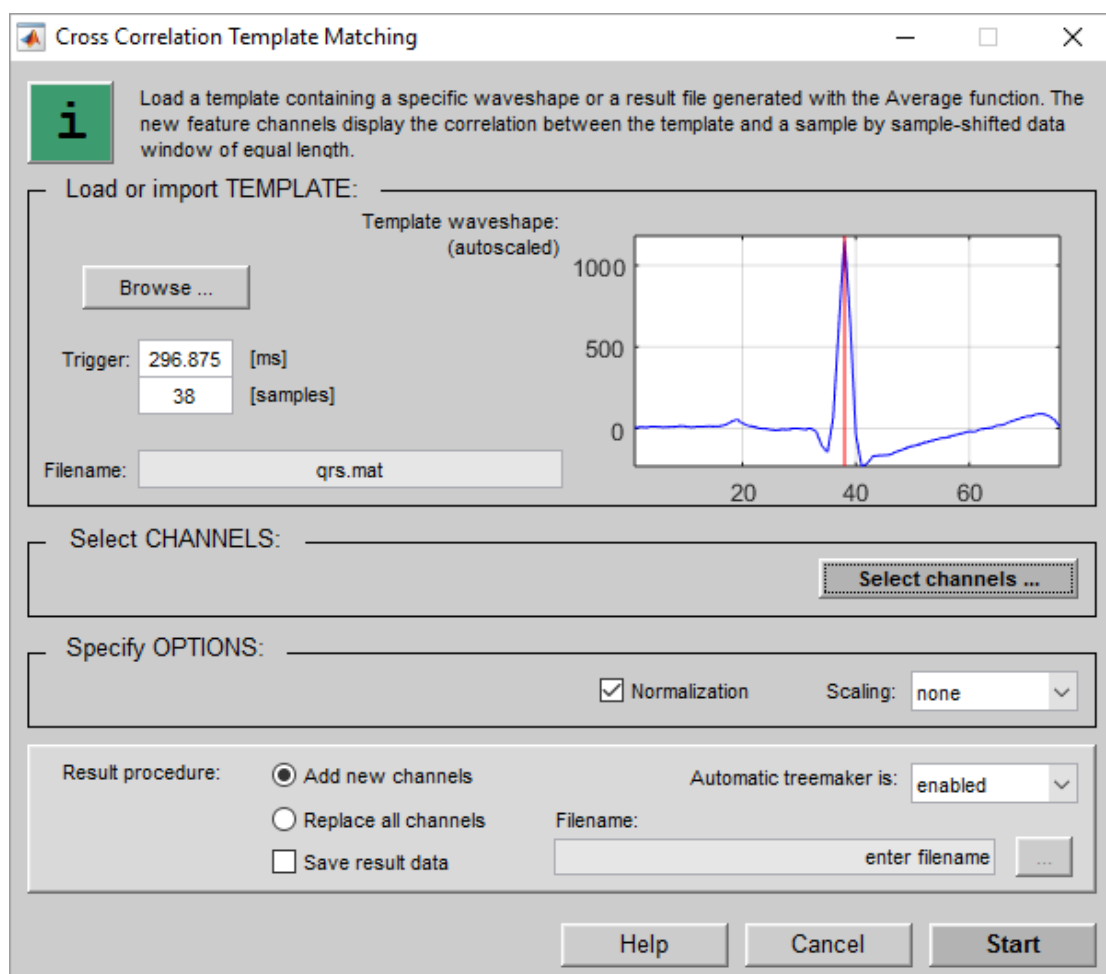


3. Open **Trigger** under the **Transform** menu and create trials of the marked QRS complexes of length 600 ms

4. Select **Average** from the **Analyze** menu to average over the QRS complexes and to view/store the template



5. Reload the data-set and open **Cross Correlation Template Matching** from the **Parameter Extraction** menu and load the created template. Select channel 7 (ECG signal) and press **Start** to perform the cross-correlation.



6. The new cross-correlation feature signal is displayed in the Data Editor and can be used to find QRS complexes.

The code below shows how to perform the example from the MATLAB command line.

%Load Data

```
P_C=data;
File= ['C:\Users\' getenv('USERNAME')
'\Documents\gtec\gBSanalyze\testdata\Aircraft_Simulator\Session1.MAT'];
P_C=load(P_C,File);
```

%Complex Detector

```
SelectInput = ['channel'];
ChannelExclude = [1 2 3 4 5 6 8 9];
Method = ['QRS detector BP'];
Parameters.BP = [10 60];
Parameters.Threshold = [0.5];
Parameters.IntervalMin = [60];
Parameters.IntervalMax = [140];
Parameters.MaxBPM = [180];
MarkerName.Accepted = ['QRS'];
Feature.Generate = [0];
Feature.Scaling = ['ms'];
Feature.Replace = ['add channel'];
Feature.FileName = [''];
ProgressBarFlag = 0;
[P_C] =gBScomplexdetector(P_C,SelectInput,ChannelExclude,Method,...
Parameters,MarkerName,Feature,ProgressBarFlag);
```

%Trigger

```
New_tm{1}={3 0};
SamplesBefore=38;
SamplesAfter=38;
Uncomplete=0;
ChannelExclude=[];
P_C=gBStrigger(P_C,New_tm,SamplesBefore,SamplesAfter,...
Uncomplete,ChannelExclude);
```

%Select Trials and Channels

```
trial_id=[];
channel_id=[];
type_id=[];
channelnr_id=[7];
flag_tr='tr_exc';
flag_ch='ch_exc';
flag_type='type_exc';
flag_nr='nr_inc';
[TrialExclude, ChannelExclude]=gBSselect(P_C,trial_id,flag_tr,...
channel_id,flag_ch,type_id,flag_type,channelnr_id,flag_nr);
```

%Average

```
Baseline=[0];
Smoothing={'none'};
DownSampling=0;
TrialExclude=[];
ChannelExclude=[1 2 3 4 5 6 8 9];
FileName= ['C:\Users\' getenv('USERNAME')
'\Documents\gtec\gBSanalyze\testdata\Aircraft_Simulator\qrs.mat'];
Averaging='simple';
var1=0;var2=0;var3=0;
A_O = gBSaverage(P_C,Baseline,Smoothing,DownSampling,TrialExclude,...
ChannelExclude,FileName,Averaging,0,var1,var2,var3);
```

%Load Data

```
P_C=data;
```



```
File=['C:\Users\' getenv('USERNAME')
'\Documents\gtec\gBSanalyze\testdata\Aircraft_Simulator\Session1.MAT'];
P_C=load(P_C,File);
```

%Cross Correlation Template Matching

```
Template=['C:\Users\' getenv('USERNAME')
'\Documents\gtec\gBSanalyze\testdata\Aircraft_Simulator\qrs.mat'];
ChannelExclude=[1 2 3 4 5 6 8 9];
Normalization=1;
ScaleOption='none';
Trigger=38;
Replace='add channels';
FileName='';
ProgressBarFlag=1;
P_C=gBSctm(P_C,ChannelExclude,Template,Normalization,ScaleOption,...
Trigger,Replace,FileName,ProgressBarFlag);
```

Canonical Correlation

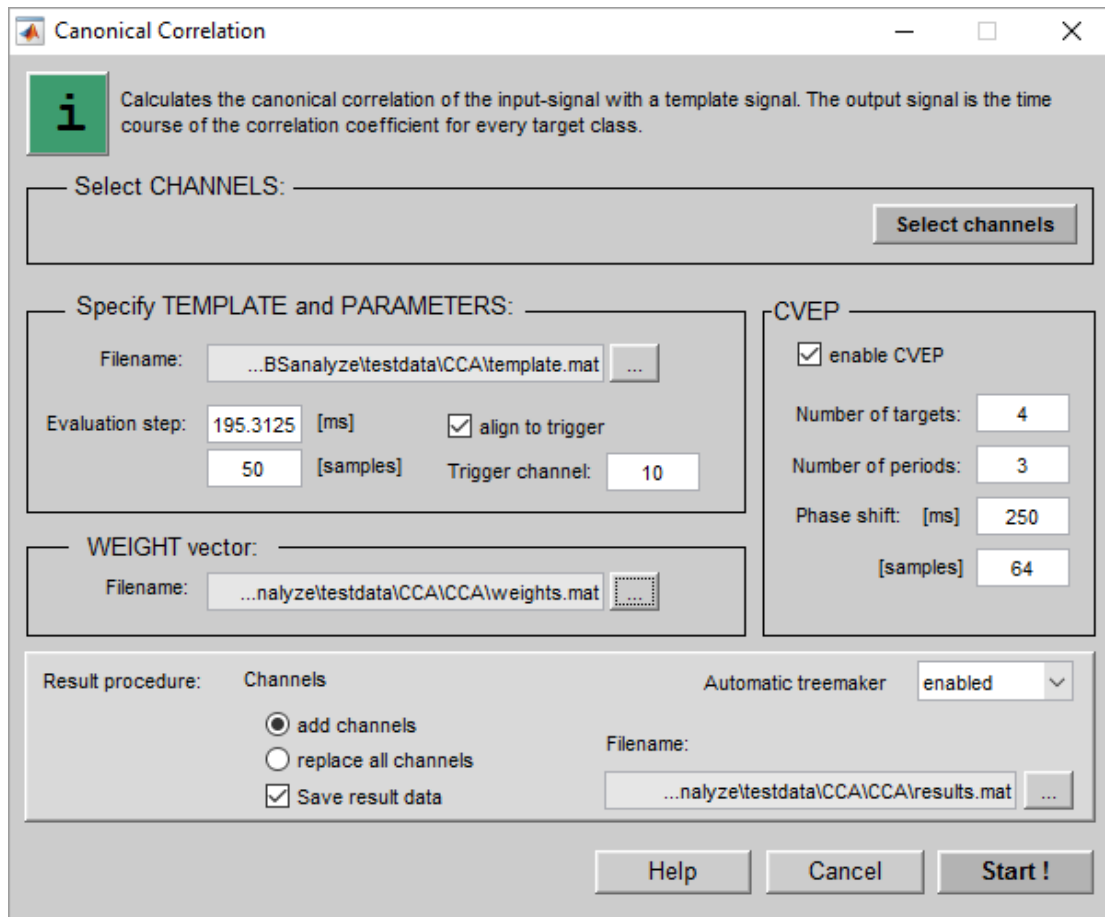
This function of the **Parameter Extraction** applies the weight vector defined by **Analyze/CCA – Canonical Correlation Analysis** as a spatial filter and calculates the time course correlation coefficient by comparing the template with a sliding window of the data set. The result is a new channel which contains the canonical correlation coefficient. In the event of a dataset from Code-based Visual Evoked Potentials (CVEP), the number of targets can be specified, which results in one channel per target class.

The window allows the following settings:

- **Select Channels** – allows to pick the set of channels of the current data set which should be used for the application of the CCA-based spatial filter. Number of channels must match the length of the weight vector!
- **Template Filename** – the name (and if necessary path) of the .mat file containing the template to match by using the file browser by clicking on the ‘...’-symbol next to the textbox.
- **Evaluation Step** – the time in milliseconds or number of samples that the moving window over the data set is shifted forward in time per step.
- **Align to Trigger** – check if the template file should be shifted so that its starting sample matches a trigger from a trigger channel in the moving window time span. Enter the channel number of the trigger channel of the current data set into the field **Trigger channel**. Note that the trigger spikes all have to be of the same amplitude!
- **Weight Vectors** - the name (and if necessary path) of the .mat file containing the weight vectors calculated by **CCA Filter**. Alternatively indicate the file by using the file browser by clicking on the ‘...’-symbol next to the textbox.
- **Result Procedure:** - under **Channels** one can decide to append the CC-channels to the existing dataset or to replace the data set by the new features. If the results should be saved click on the checkbox **Save results**. This opens a file browser window which allows choosing the desired filename.

The GUI window also contains an additional block named **CVEP**, which is an extension for Code-based Visual Evoked Potentials, which work with circular shifts of a periodic template data set. When this box is checked, the following parameters can be specified.

- **Number of Classes** – number of different classes in the paradigm (default: 1)
- **Number of Periods** – if the template file is periodic set this to the number of complete cycles; if the template file is NOT periodic, set to 1 (default: 1)
- **Phase Shift** – phase shift of the template in milliseconds or samples between classes (default: 0)



Example:

Perform the following steps to apply the minimum energy:

1. Load data-set `cvep_online.mat` under
`Documents\gtec\gBSanalyze\testdata\CCA`
and specify a Sampling frequency of 256 Hz if asked.
2. Click on the **Parameter Extraction** menu and then on **Canonical Correlation**.
3. Select channels 2-9 under the **Select Channels** menu.
4. Under **Specify Template** load the template file `template.mat` stored in
`Documents\gtec\gBSanalyze\testdata\CCA\CCA`
5. Set the **Evaluation step** to 50 samples
6. Check the box **align to trigger** and indicate the **Trigger channel** 10.
7. Check the box **enable CVEP**

8. Set **Number of targets** to 4

9. Set **Number of periods** to 3

10. Set **Phase shift** to 64 samples

11. Under **Weight vector** load the template file `weights.mat` stored in

```
Documents\gtec\gBSanalyze\testdata\CCA\CCA
```

12. Chose **add channels** in the **Result procedure** panel to append the correlation channels to the current data set

13. Check the **Save results** box and type in `results.mat` to name the results file. The file should be placed automatically in the folder

```
Documents\gtec\gBSanalyze\testdata\CCA\CCA
```

14. Press **Start** to compute the canonical correlation values.

The following code shows how to perform the example demonstrated above from the MATLAB command line.

```
%Load Data
P_C=data;
File=['C:\Users\' getenv('USERNAME')
'\Documents\gtec\gBSanalyze\testdata\CCA\cvep_online.mat'];
P_C=load(P_C,File);
P_C.SamplingFrequency=256;
% Canonical Correlation Analysis

% Canonical Correlation Analysis
ChannelExclude = [1 10 11 12];
NumberOfClasses = [4];
TriggerChannel = 10;
Synchronization = 'trigger';
TemplateFilename = ['C:\Users\' getenv('USERNAME')
'\Documents\gtec\gBSanalyze\testdata\CCA\template.mat'];
TemplatePeriods = 3;
PhaseShift = 64;
WeightsFilename = ['C:\Users\' getenv('USERNAME')
'\Documents\gtec\gBSanalyze\testdata\CCA\CCA\weights.mat'];
EvaluationStep = 50;
Replace = 'add channels';
FileName = ['C:\Users\' getenv('USERNAME')
'\Documents\gtec\gBSanalyze\testdata\CCA\CCA\results.mat'];
ProgressBarFlag = 0;
P_C = gBScca(P_C,ChannelExclude,NumberOfClasses,EvaluationStep,...
TemplateFilename,TemplatePeriods,WeightsFilename,TriggerChannel,...
Synchronization,PhaseShift,Replace,FileName,ProgressBarFlag);
```

Phase Locking Value

The Phase Locking Value (PLV) can be used to detect synchrony in a precise frequency range between two recording channels. If the phase difference varies little, the PLV is close to 1; it is close to 0 otherwise. This PLV can be computed for several frequencies in order to study a broader frequency range.

The algorithm first bandpass filters the EEG channels and then computes the convolution with a wavelet centred at a specific center frequency. The phase of the convolution is extracted and is used to measure the PLV. Then PLV values are averaged according to the **Integration length**.

The window allows the following settings:

Specify PLV SETTINGS:

Mode – specify the method used for the calculation

Bandwidth – bandwidth in Hz used for the butterworth bandpass filter (center frequency \pm Bandwidth/2)

Center frequency – center frequency of the butterworth bandpass filter and of the wavelet

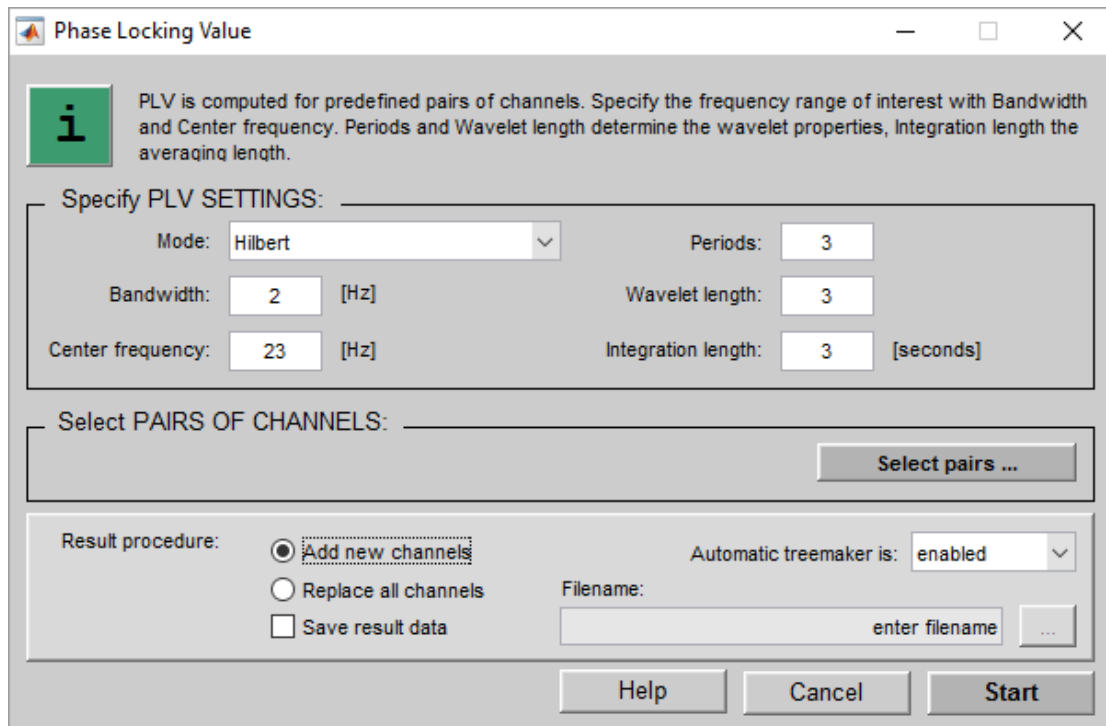
Bandwidth and **Center frequency** determine the lower and upper cut-off frequencies of the butterworth bandpass filter (lower: **Center frequency** – **Bandwidth**/2, upper: **Center frequency** + **Bandwidth**/2)

Periods – changes the standard deviation of the Gaussian bell shape of the Gabor filter (not used for Hilbert). Use **Periods** = 3 for a good time resolution and higher values (6-7) for a better frequency resolution.

Wavelet length – the **Wavelet length** times **Periods** determines the energy under the Gaussian bell curve. If **Wavelet length** is 3, 99 % of the area are covered.

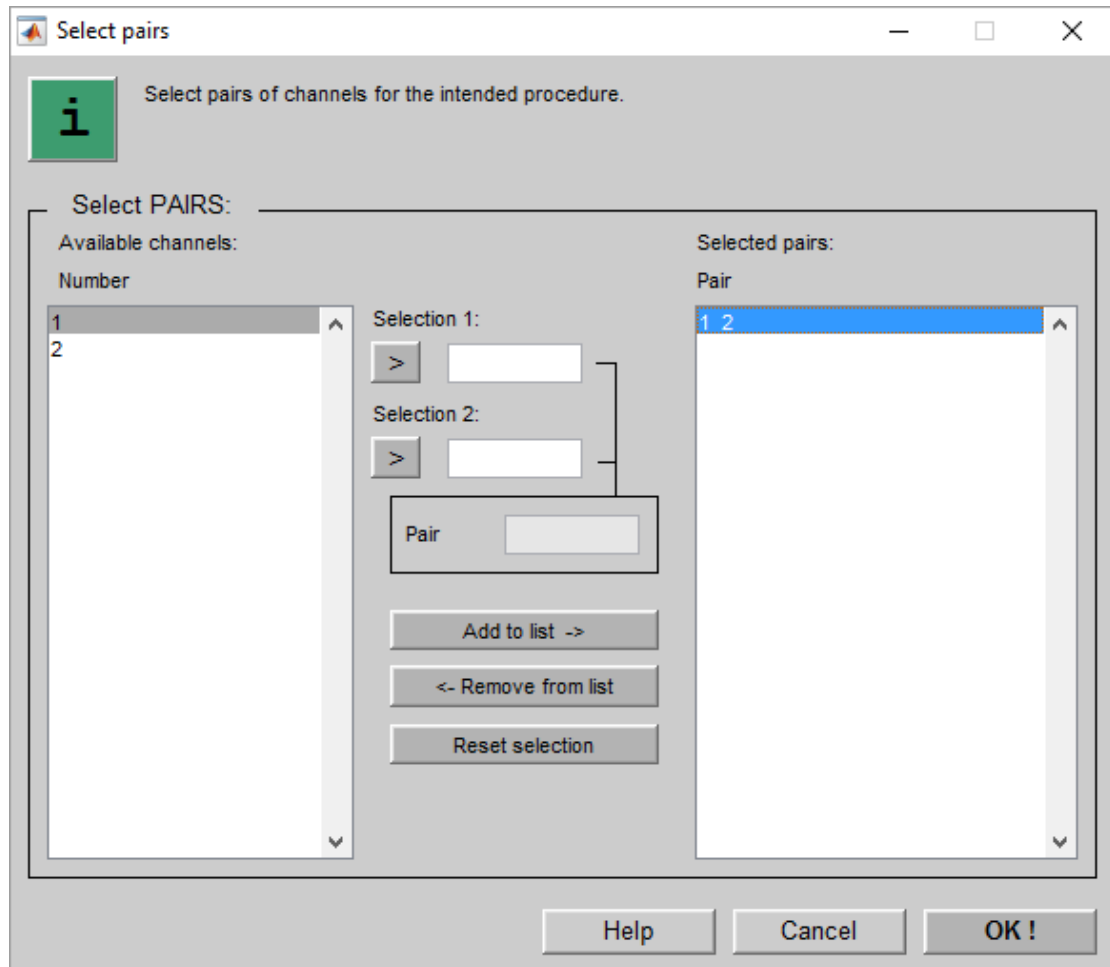
Integration length – length of the integration of the phase differences in seconds.

Note the time shift of the PLV values caused by the **Integration length**. Therefore calculate the PLV before triggering the data-set.



Perform the following steps to calculate the PLV from 2 EEG channels:

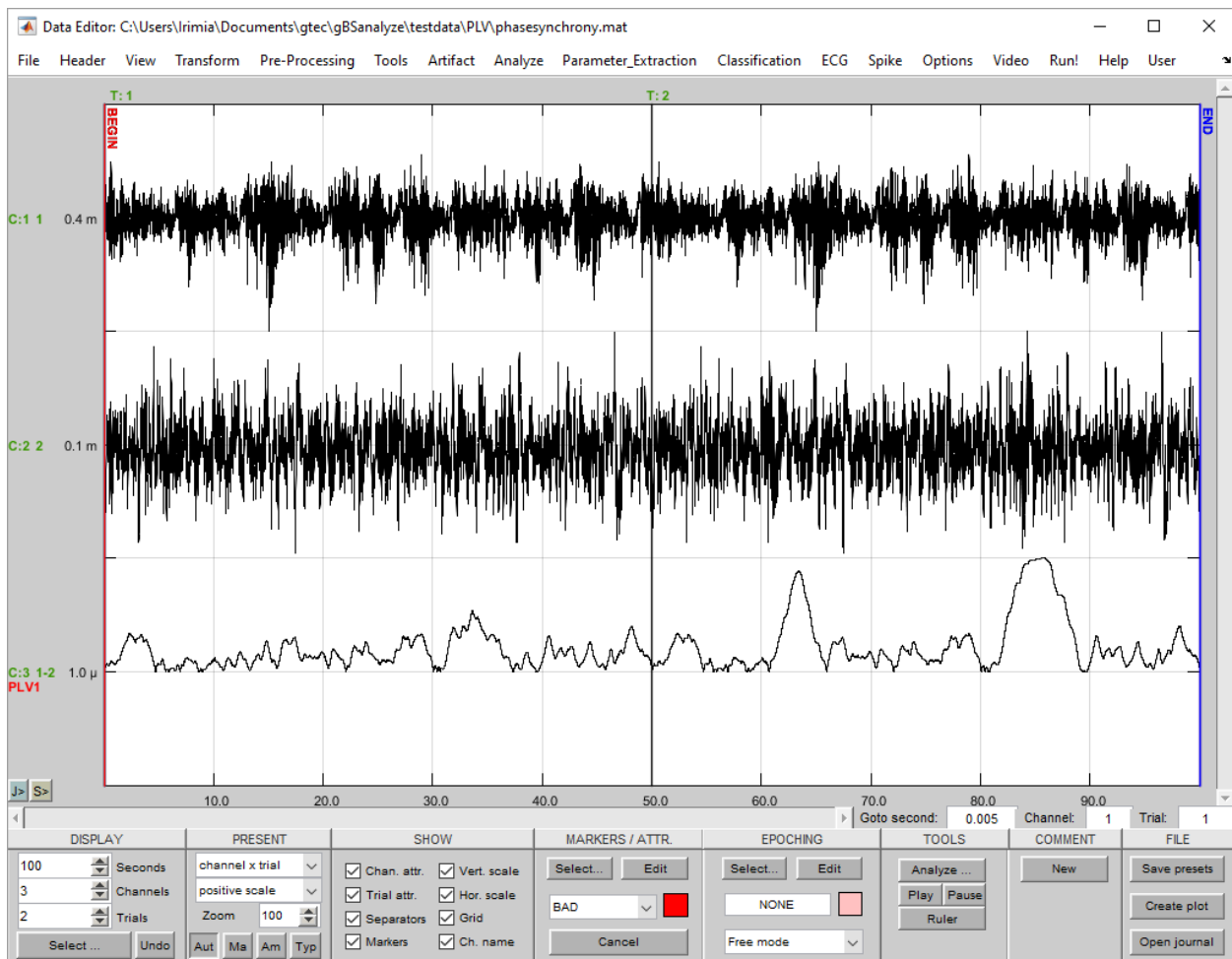
1. Open the file `phasesynchrony.mat` from
`Documents\gtec\gBSanalyze\testdata\PLV`
2. Start **Phase Locking Value** from the **Parameter Extraction** menu and click on **Select pairs ...**



Define channels 1 and 2 as electrode pair for the computation. Close the window with the **OK** button.

3. Select under **Result procedure Add new channels** and start the calculation.

The **Data Editor** shows the newly created PLV channel on channel 3.



The PLV differs between trials 1 and 2. In trial 2 there are 2 segments where the PLV reaches almost 1 which indicates that the phase difference between channels 1 and 2 is very little.

The following code show how to perform the example demonstrated above from the MATLAB command line.

```
%Load Data
P_C=data;
File=['C:\Users\' getenv('USERNAME')
'\Documents\gtec\gBSanalyze\testdata\PLV\phasesynchrony.mat'];
P_C=load(P_C,File);

%Phase Locking Value
T.method='Hilbert';
T.bandwidth=2;
T.centerfrequency=23;
T.periods=3;
T.wavelen=3;
T.intlen=3;
ChannelPairs=[1 2];
Replace=['add channels'];
FileName='';
ProgressBarFlag=0;
P_C=gBSphaselockingvalue(P_C,T,ChannelPairs,Replace,FileName,...
ProgressBarFlag);
```

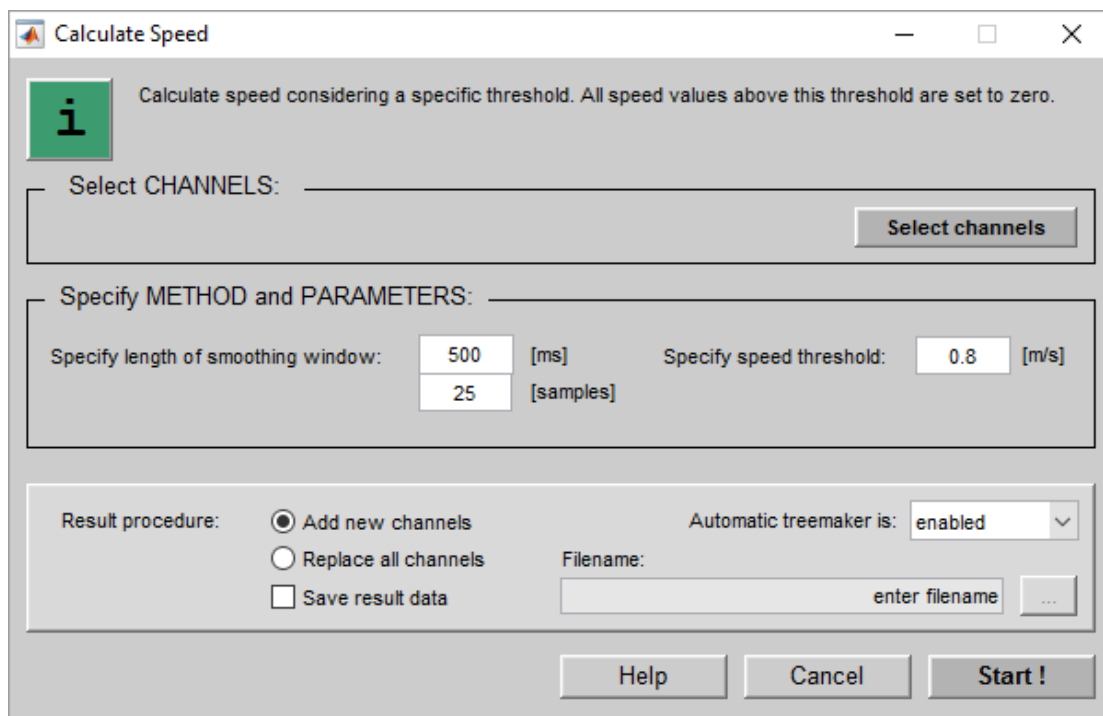

Calculate Speed

This function of the **Parameter Extraction** menu calculates the speed of position data, using the x and y coordinates and a time vector. The time vector is created with help of the sampling frequency. By clicking **Select Channels**, you have to select the x and y channels, where your position information are stored.

The window allows the following settings:

Specify length of smoothing window - specify the length of the smoothing window used for filtering the speed data.

Specify speed threshold - Sometimes, while position recording, errors occur, which are visible in the trajectory. As a result, the distance between two recorded positions increases dramatically. To eliminate these distortions, all speed values above the specified threshold are set to zero.



Perform the following steps to calculate the speed:

1. Import data-set `run1.pos` under
`Documents\gtec\gBSanalyze\testdata\AXONA`
2. Click on **Calculate Speed** under the **Parameter Extraction** menu
3. **Specify length of smoothing window** as 500 ms and set the threshold to 0.8 m/s
4. Click on **Select channels** and add the two channels, containing x and y information, to the list. Confirm the settings with the **OK** button

6. Press **Start** to calculate the parameters

The following code show how to perform the example demonstrated above from the MATLAB command line.

```
%Import Data
P_C = data;
File=['C:\Users\' getenv('USERNAME')
'\Documents\gtec\gBSanalyze\testdata\AXONA\run1.pos'];
PPM = 512;
Arenaxy = [0.5 0.5];
P_C = import(P_C,'AXONA',File, PPM, Arenaxy);
%Calculate Speed
ChannelExclude = [1];
IntervalLength = 25;
Threshold = 0.8;
Replace = 'replace all channels';
FileName = '';
ProgressBarFlag = 0;
P_C = gBScalculatespeed(P_C, ChannelExclude, IntervalLength,...
Threshold, Replace, FileName, ProgressBarFlag);
```

Classification

This section explains the classification of different parameters.

[Generate Classifier](#) – classify the data and generate a classifier

The window supports:

[Multi-Class LDA](#) – linear discriminant analysis

[Minimum Distance Classifier](#) – template matching system

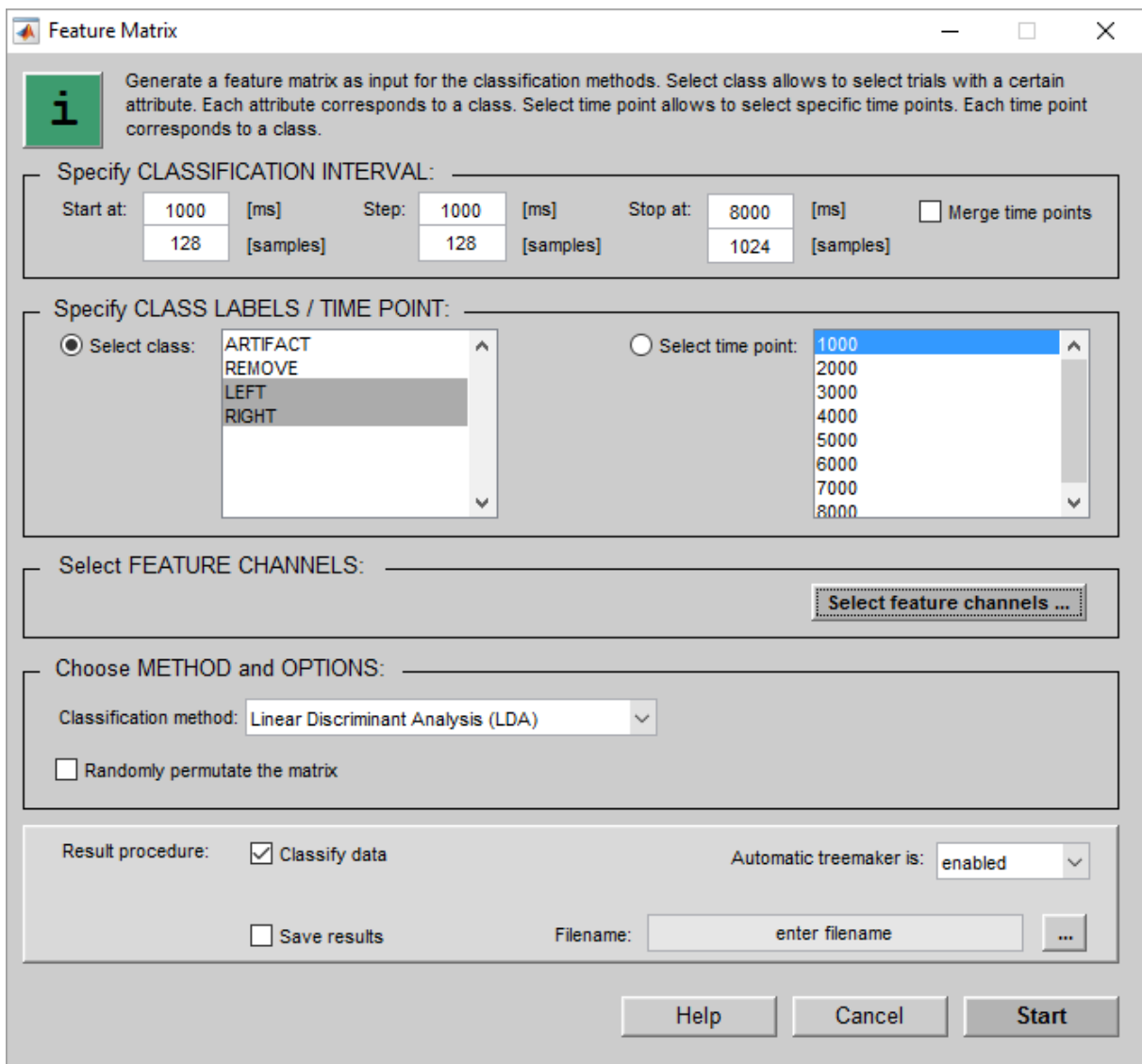
Generate Classifier

The **Generate Classifier** window allows to classify data-sets and to generate a classifier for on-line analysis. The window allows performing a linear discriminant analysis and minimum distance classifier analysis of multiple classes. The data must be triggered and class attributes must be assigned to the trials. The number of trials of each class must be equal.

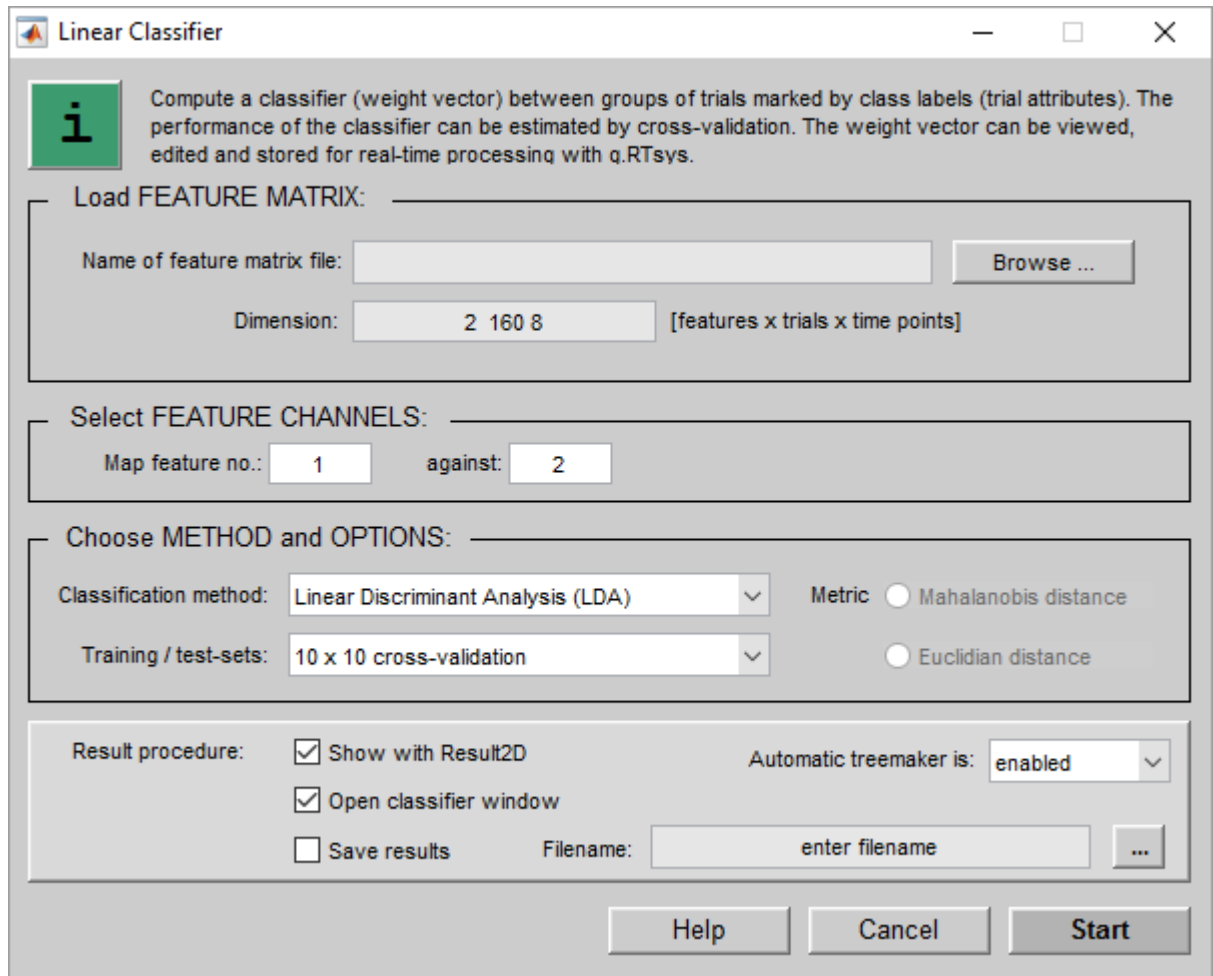
Example:

Perform the following steps to make a classification of an EEG-based brain-computer interface data-set:

1. Load data-set `session1234triggered.mat` from
`Documents\gtec\gBSanalyze\testdata\bci`
2. Perform a bandpower calculation of the 2 EEG channels from the **Parameter Extraction** menu entry. Select a butterworth bandpass filter of order 5, lower cutoff frequency of 14 and upper cutoff frequency of 20 Hz. Set the averaging window to 500 ms.
3. Open the **Feature matrix** window from menu **Classification**
4. Specify an interval that starts at second 1, ends at second 8 with steps of 1 second
5. Select as class 1 trials with the `LEFT` attribute and for class 2 trials with the `RIGHT` attribute
6. Select channels 1 and 2 for classification
7. Select **Linear Discriminant Analysis (LDA)** as classification method and press **Start**



8. The **Linear Classifier** window opens. Select **10 x 10 cross-validation** to randomly mix the training and testing data-set
9. Check the **Open classifier window** to open the MATLAB Editor to view the classification result and the weight vector (not available for 10 x 10 cross validation)



10. Press the **Start** button to start the calculation

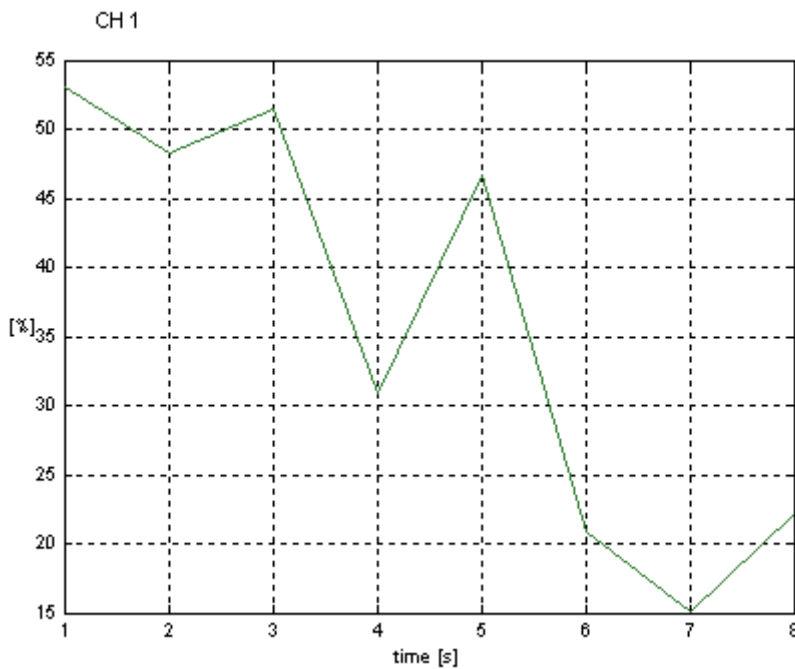
11. g.BSanalyze performs the calculation and starts up the MATAB Editor to view the classification results:

```

C:\Users\Irimia\Documents\gtec\gBSanalyze\testdata\classification.dat
EDITOR VIEW
New Open Save Find Files Compare Go To Insert Comment Indent Breakpoints
FILE NAVIGATE EDIT BREAKPOINTS
1 Classification Method: Linear Discriminant Analysis
2 Training- and testdata option: CV
3 class 1: LEFT
4 class 2: RIGHT
5 Total Nr. of Trials: 160
6 Trials per selected Class 1: 80
7 Trials per selected Class 2: 80
8
9 Second/Sample/Mean Error/Std/[Total Error CV Runs]
10 1.000 128.000 48.6 2.0 46.3 51.2 50.0 48.1 50.6 49.4 46.3 49.4 49.4 45.6
11 2.000 256.000 47.3 1.6 50.0 46.9 46.9 47.5 46.9 45.6 48.1 48.8 47.5 44.4
12 3.000 384.000 52.5 3.2 51.2 50.0 49.4 48.8 56.9 56.9 55.6 53.1 49.4 53.8
13 4.000 512.000 38.6 1.0 39.4 38.8 38.8 38.1 36.9 40.0 38.8 38.8 36.9 39.4
14 5.000 640.000 46.7 1.1 46.3 48.8 48.1 46.9 46.9 45.6 45.6 46.9 45.6 46.3
15 6.000 768.000 23.7 0.2 23.8 23.8 23.8 23.8 23.8 23.8 23.1 23.8 23.8 23.8
16 7.000 896.000 15.0 0.7 13.8 16.3 15.0 15.0 14.4 15.6 15.6 15.0 15.0 14.4
17 8.000 1024.000 18.3 0.3 18.1 18.1 18.1 18.1 18.1 18.8 18.1 18.1 18.8 18.8
18
19
plain text file Ln 11 Col 25

```

The classification is shown over time points and 10 cross validation runs and yields to a minimum at second 7 of about 15 %. Result2D opens with the following output:



12. To calculate a linear classifier select **Train 50% - Test 50%** from the **Training/test - sets** pull-down menu. In this case 50 % of the data are used for training a classifier (and to produce a classifier) and 50 % of the trials are used to test the classifier (and to produce a classification result). Check the **Save results** box to store the classification result and the weight vector. **Start** the calculation.

```

1 Classification Method: Linear Discriminant Analysis
2 Training- and testdata option: 50:50
3 class 1: LEFT
4 class 2: RIGHT
5 Total Nr. of Trials: 80
6 Trials per selected Class 1: 40
7 Trials per selected Class 2: 40
8
9 Second/Sample/Total Error/[Error Class 1/Error Class 2...]
10 1.000 128.000 46.3 28.7 17.5
11 2.000 256.000 47.5 17.5 30.0
12 3.000 384.000 50.0 27.5 22.5
13 4.000 512.000 38.8 16.3 22.5
14 5.000 640.000 47.5 30.0 17.5
15 6.000 768.000 25.0 1.3 23.8
16 7.000 896.000 18.8 0.0 18.8
17 8.000 1024.000 17.5 2.5 15.0
18
19 | separates the classifiers for different classes
20 Second/Sample: 1.000 128.000
21 Classifier 1
22 -0.139 | 0.139 |
23 Classifier 2
24 0.031 |-0.031 |
25 0.034 |-0.034 |
26
27 Second/Sample: 2.000 256.000
28 Classifier 1
29 0.051 |-0.051 |
30 Classifier 2
31 -0.010 | 0.010 |
32 -0.015 | 0.015 |

```

13. Go to the directory where you stored the result and load it into MATLAB with


```
load weightvector.mat
```

type whos to show the variables

out_err	8x1	error rate over time points (rows)
out_wv	8x3	one weight vector for each time point. The first entry of each weight vector is the bias value.

14. out_wv can be used in real-time BCI experiments for classification with g.Rtsys

The following code shows how to perform the example demonstrated above from the MATLAB command line.

```
%Load Data
P_C=data;
File=['C:\Users\' getenv('USERNAME')
'\Documents\gtec\gBSanalyze\testdata\BCI\session1234triggered.mat'];
P_C=load(P_C,File);

% Bandpower
ChannelExclude = [3];
Filter.Name = 'BETA';
Filter.Type = 'BP';
Filter.f_low = [14];
Filter.f_high = [20];
Filter.Realization = 'butter';
Filter.Order = [5];
IntervalLength = 64;
Overlap = 63;
Replace = 'replace all channels';
FileName = '';
ProgressBarFlag = 0;
P_C = gBSbandpower(P_C, ChannelExclude, Filter, IntervalLength,...
Overlap, Replace, FileName, ProgressBarFlag);

%Generate Classifier
Interval=[128 128 1024];
AttrID={'LEFT'; 'RIGHT'};
ChannelExclude=[];
Methods=['LDA'];
TrainTestData=['CV'];
Metric=[''];
FileName=[''];
ProgressBarFlag=[0];
C_O=gBSgenerateclassifier(P_C, Interval, AttrID, Methods, Metric, ...
TrainTestData, ChannelExclude, FileName, ProgressBarFlag);
```

Multi-Class LDA

Linear Discriminant of Fisher

An optimal decision rule for minimizing the probability of misclassification is based on the idea of discriminant functions. The simplest form consists of a linear combination of the inputs. The parameters are obtained with a learning algorithm from a set of training data. Fisher introduced a method that reduces the dimensionality before classification [Bishop 1995].

The dimension reduction is done by projecting the input data \mathbf{x} onto a value y with adjustable weights \mathbf{w}

$$y = \mathbf{w}^T \mathbf{x} \quad (1)$$

Of course this leads to a loss of information but we chose \mathbf{w} in such a way that maximizes the class separation between class 1 and class 2 (e.g. left and right finger movement).

For class „left finger“ the mean vector is

$$\mathbf{m1} = \frac{1}{N1} \sum_{n \in \text{Classleftfinger}} \mathbf{x}_{n\text{leftfinger}} \quad (2)$$

where $N1$ is the length of the input vector.

For the class „right hand“ $\mathbf{m2}$ is

$$\mathbf{m2} = \frac{1}{N2} \sum_{n \in \text{Classrightfinger}} \mathbf{x}_{n\text{rightfinger}} \quad (3)$$

The separation of the two classes is made by separating $\mathbf{m1}$ and $\mathbf{m2}$

$$\mathbf{m2} - \mathbf{m1} = \mathbf{w}^T (\mathbf{m2} - \mathbf{m1}) \quad (4)$$

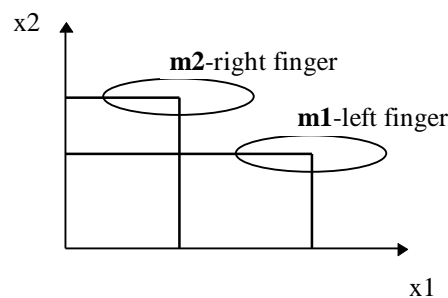
By choosing \mathbf{w} arbitrary large the difference increases, therefore define

$$\sum_i \mathbf{w}_i^2 = 1 \quad (5)$$

and after some calculation we obtain

$$\mathbf{w} \propto (\mathbf{m2} - \mathbf{m1}) \quad (6)$$

But a problem arises with this separation that is shown below.



Separation problem.

If we project $\mathbf{m1}$ and $\mathbf{m2}$ onto the x_1 axes the difference is bigger than in the case of projecting onto the x_2 axes. But there are within-class spreads that cause a better separation when $\mathbf{m1}$ and $\mathbf{m2}$ are projected onto x_2 .

Fisher proposed as solution

$$\mathbf{w} \propto \mathbf{S}_w^{-1}(\mathbf{m2} - \mathbf{m1}) \quad (7)$$

where \mathbf{S}_w is the total within class covariance matrix

$$\mathbf{S}_w = \sum_{n \in \text{Classleftfinger}} (\mathbf{x}_n - \mathbf{m1})(\mathbf{x}_n - \mathbf{m1})^T + \sum_{n \in \text{Classrightfinger}} (\mathbf{x}_n - \mathbf{m2})(\mathbf{x}_n - \mathbf{m2})^T \quad (8)$$

Which is a projection rule for the data down to one dimension.

By choosing a threshold y_0 we can classify a point to class 1 if it is greater zero or to class 2 otherwise.

Obviously the dimension reduction reduces the amount of information, but it can lead to improvements of the classifier performance [Bishop 1995].

References:

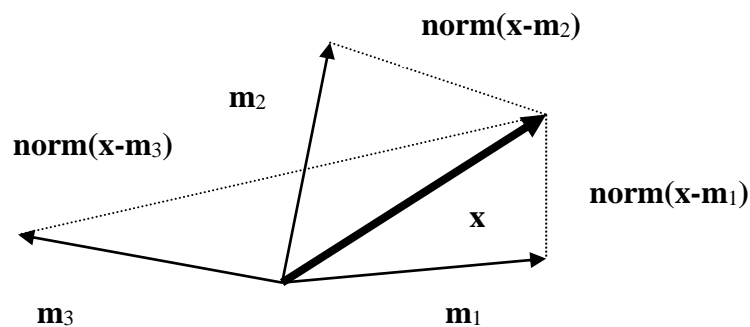
Bishop, C. M., Neural Networks for Pattern Recognition. Clarendon Press, Oxford, 1995.

Minimum Distance Classifier

Let \mathbf{x} be the feature vector for the unknown input. Vectors $\mathbf{m}_1, \mathbf{m}_2, \dots, \mathbf{m}_c$ are the templates (i.e., perfect, noise-free feature vectors) for the c classes. Then the error in matching \mathbf{x} against \mathbf{m}_k is given by

$$\|\mathbf{x} - \mathbf{m}_k\|.$$

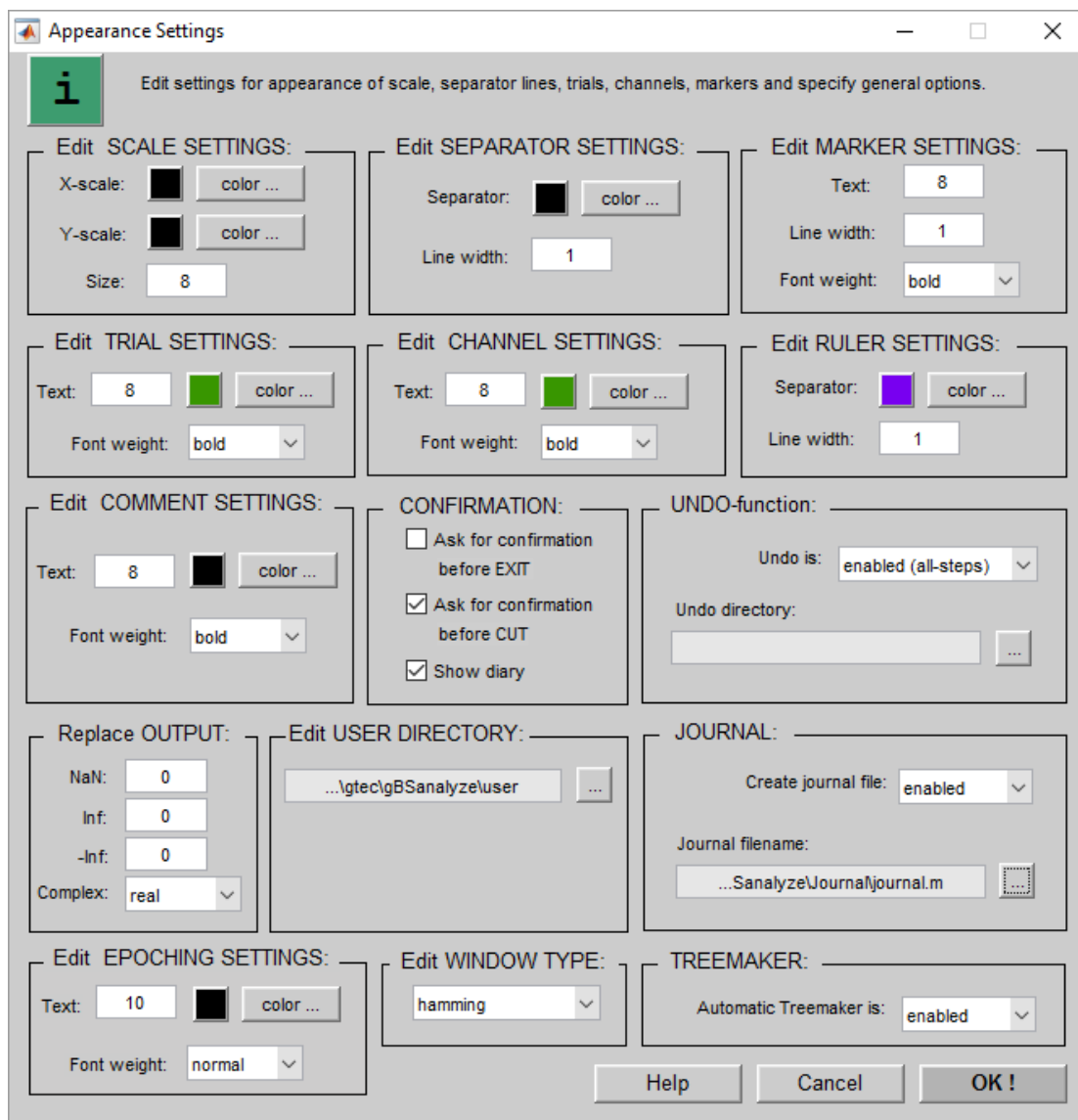
where $\|\mathbf{u}\|$ is called the **norm** of the vector \mathbf{u} . The minimum-error classifier calculates $\|\mathbf{x} - \mathbf{m}_k\|$ for $k = 1$ to c to find the class for which this error is minimum. $\|\mathbf{x} - \mathbf{m}_k\|$ is also the distance from \mathbf{x} to \mathbf{m}_k and therefore the method is called **minimum-distance** classifier.



Appearance Settings

Edit SCALE SETTINGS - settings for x-, and y-scale of Data Editor

Edit SEPARATOR SETTINGS - settings for separator lines between trials and channels



Edit MARKER SETTINGS - settings for marker appearance

Edit TRIAL SETTINGS - settings for trial description

Edit CHANNEL SETTINGS - settings for channel description

Edit RULER SETTINGS – settings for the ruler

Edit COMMENT SETTINGS – settings for comment appearance

CONFIRMATION

Ask for confirmation before EXIT - do you really want to exit dialog

Ask for confirmation before CUT - confirmation for cut samples, channels and trials

Show diary - show commands in MATLAB Editor

UNDO-function – disables or enables the **UNDO** function. The pull-down menu allows to enable a 1-step back or multiple-step back **UNDO** function. Note that g.BSanalyze stores the data-set to harddisk and re-loads this data-set if the **UNDO** button is pressed. To speed up the operation of g.BSanalyze and save harddisk resources disable the **UNDO** function. An Undo-Directory has to be set in order to be able to work with the Undo function.

Replace OUTPUT – if a calculation yields to NaN (Not a Number), Inf (Infinity), -Inf (minus Infinity) or complex results replace these entries by the specified numbers.

Edit USER DIRECTORY – put your own M-files into this directory to start the programs from the **User** menu entry in the Data Editor

JOURNAL – store all outputs of the MATLAB command window to a specific file

Edit EPOCHING SETTINGS – settings for epochs

Edit WINDOW TYPE – windowing function used for **EPOCHING** before merging the marked data-sets. The windowing is performed before calculating the spectrum in the **Analyze** window.

TREEMAKER – creates automatically a subdirectory for storing results. The subdirectory is created under the path where the data-set is stored.

Click **OK!** to confirm the settings and to close the window.

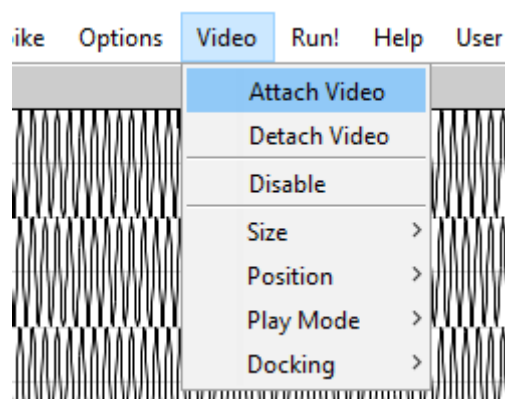
Video

g.BSanalyze allows to attach a video file to an raw data set, analyse the corresponding data frame by frame and view the video sequence which corresponds to a specific event or wave identified from the data.

1. Load the dataset `testdata.mat` that is stored under:

`Documents\gtec\gBSanalyze\testdata\video`

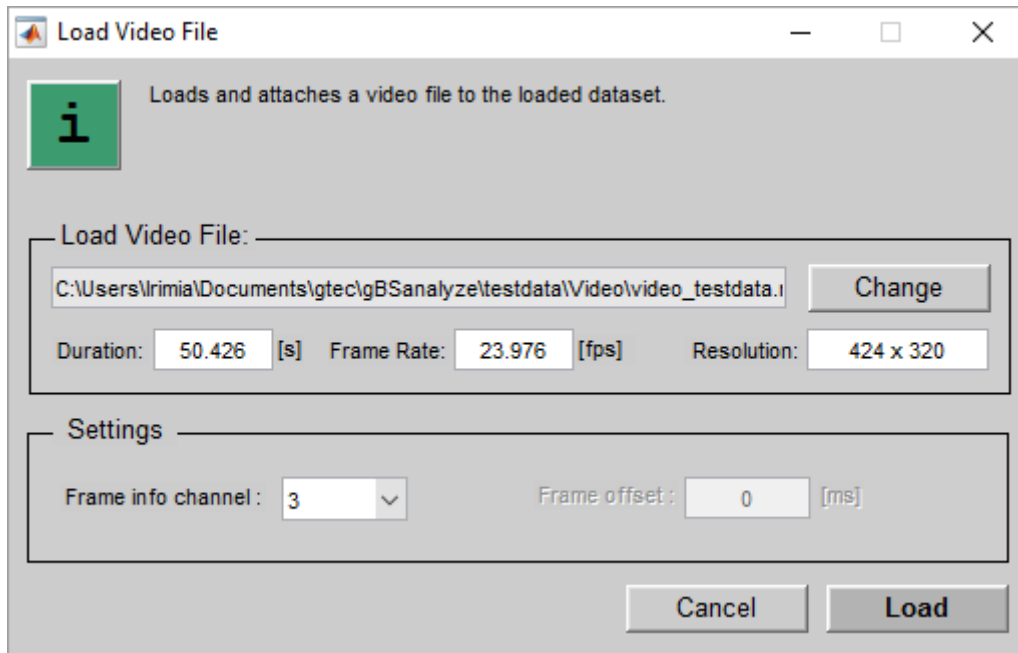
2. select **Attach Video** from the **Video** menu to open the **Load Video File** dialog



3. Select **Browse** to choose the **Video file** to be attached

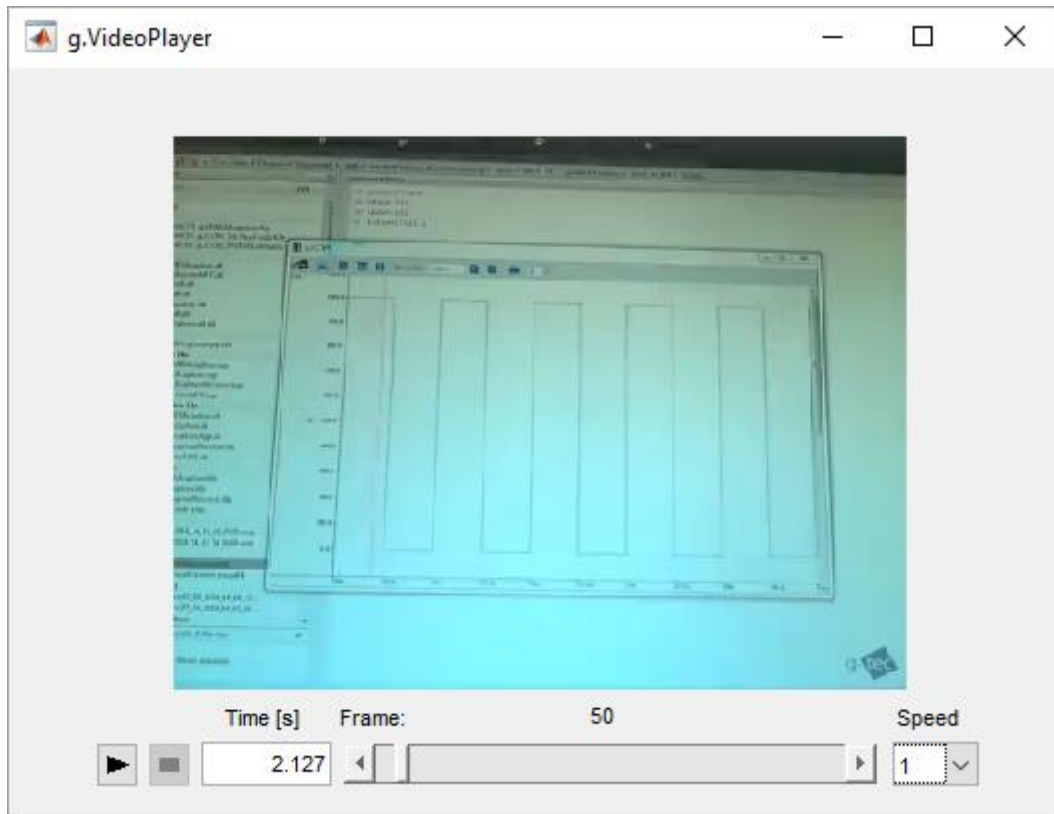
Select video file `testdata_video.mp4` that is stored under `Documents\gtec\gBSanalyze\testdata\video`, and press **OK**.

4. Check the basic information about **Video file**, its **Duration**, **Frame rate** and **Resolution** to conform to the settings specified by the **g.VdieoTool** or **g.Recorder**.
5. Set the **Frame info channel** to 3. For a data set which has been recorded with **g.Recorder** select **Create** instead and specify the **Frame offset** time when the video was started in relation to the recording of the biosignal data.
6. Press **Load** to attach the video and display it.

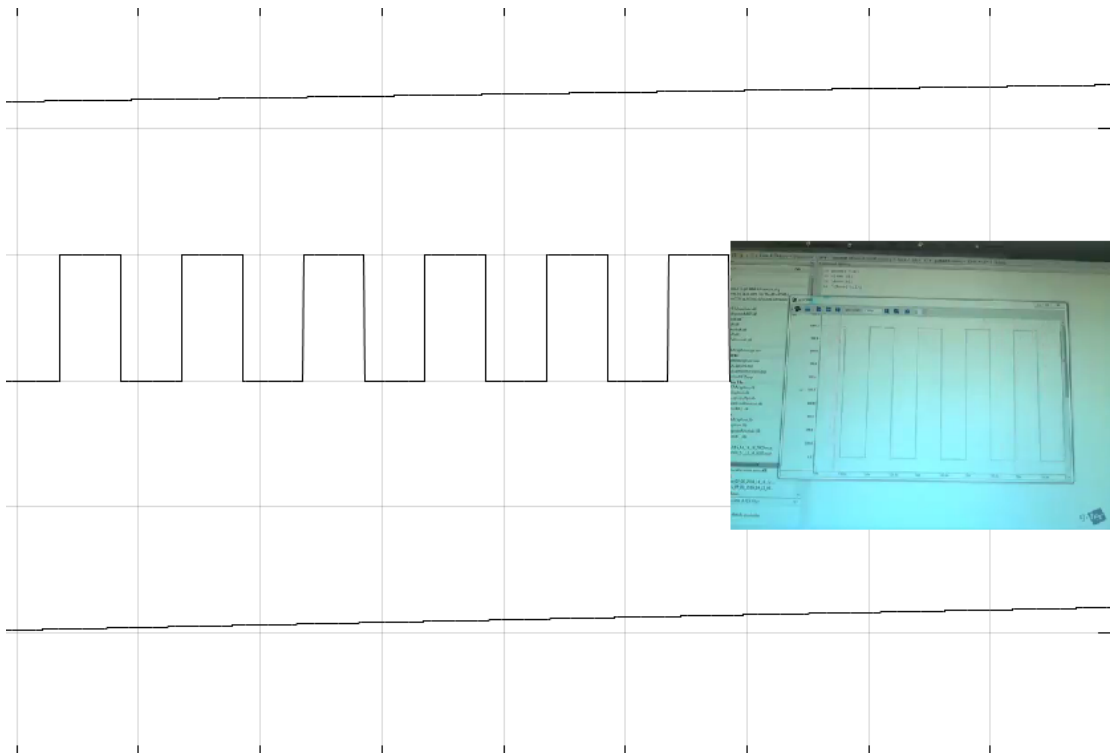


Detach Video – detaches the video from the loaded dataset.

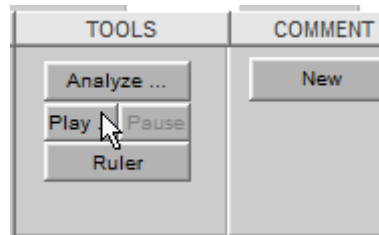
7. To disable and hide the display of the video select **Disable** from the **Video menu**
8. To enable and show the hidden video select **Enable** available from the **Video menu** after it has been hidden before.
9. To adjust the size of the video select either Small, Medium or Large from the **Size** submenu of the **Video menu**.
10. To move the display of the video select either Top, Center or Bottom from the **Position** submenu of the **Video menu**.
11. To adjust the speed at which the video is displayed select either Slow (0.5 x), Normal (1.0x) or Fast (2.0 x) from the **Play Mode** submenu of the **Video menu**.
12. To display the video the external **g.VideoPlayer** window select Undock Video from the **Docking** submenu of the **Video menu**.




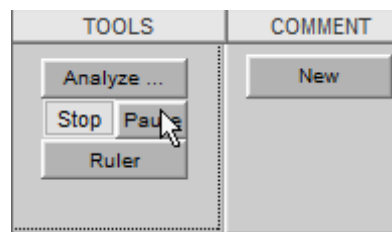
13. When displayed in the external **g.VideoPlayer** window the video can be reembedded within the **Dataeditor** window by selecting Dock Video from the **Docking** submenu of the **Video** menu.




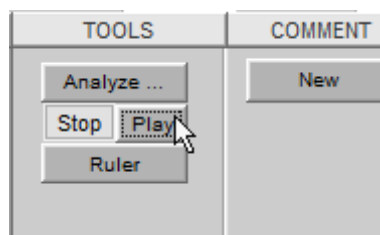
14. To play the video press **Play** on the **Dataeditor** window or  on the external **g.VideoPlayer** window.



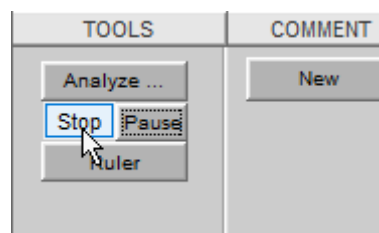
15. To pause the video press **Pause** on the **Dataeditor** window or  on the external **g.VideoPlayer** window.



16. To resume the video press **Play** on the **Dataeditor** window or  on the external **g.VideoPlayer** window.

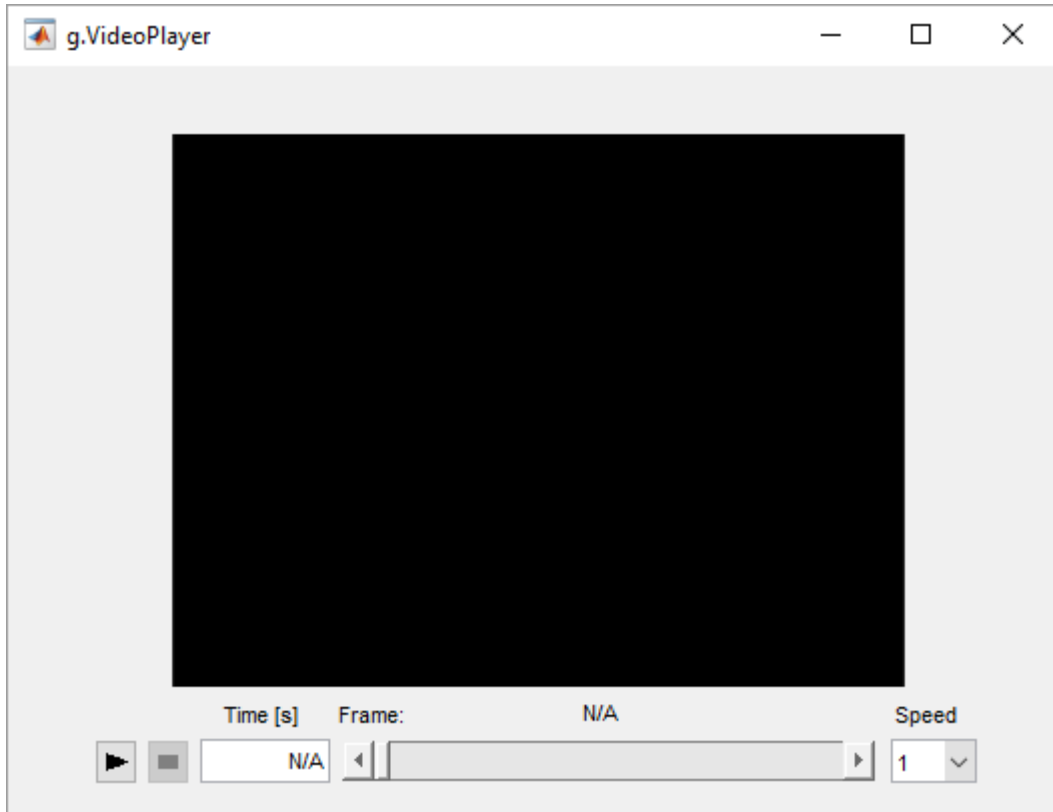


17. To stop the video press **Stop** on the **Dataeditor** window or  on the external **g.VideoPlayer** window.



18. To advance to a specific sample time either adjust the display of the **Dataeditor** to show the corresponding data, enter the **Time [s]** on the external **g.VideoPlayer** window.
19. To jump to a specific **Frame**, drag the handle of the slider on the external **g.VideoPlayer** window, click between the handles of the **Frame** slider to move in steps corresponding to about 1 s or use the arrow handles for the **Frame** slider to move frame by frame.

In all cases the frame displayed on the video and the data displayed by the **Dataeditor** are synchronized with each other on a per frame basis. In case no video has been recorded for a specific time point a black screen will be displayed instead of the video and the sample **Time [s]** and **Frame** number display on the **g.VideoPlayer** window will be set to **N/A**.



gResult2D

gResult2D is a tool for visualizing results like Average, Coherence, Classification,... of the Data Editor.

Perform the following steps:

1. To start gResult2D type under the MATLAB command window

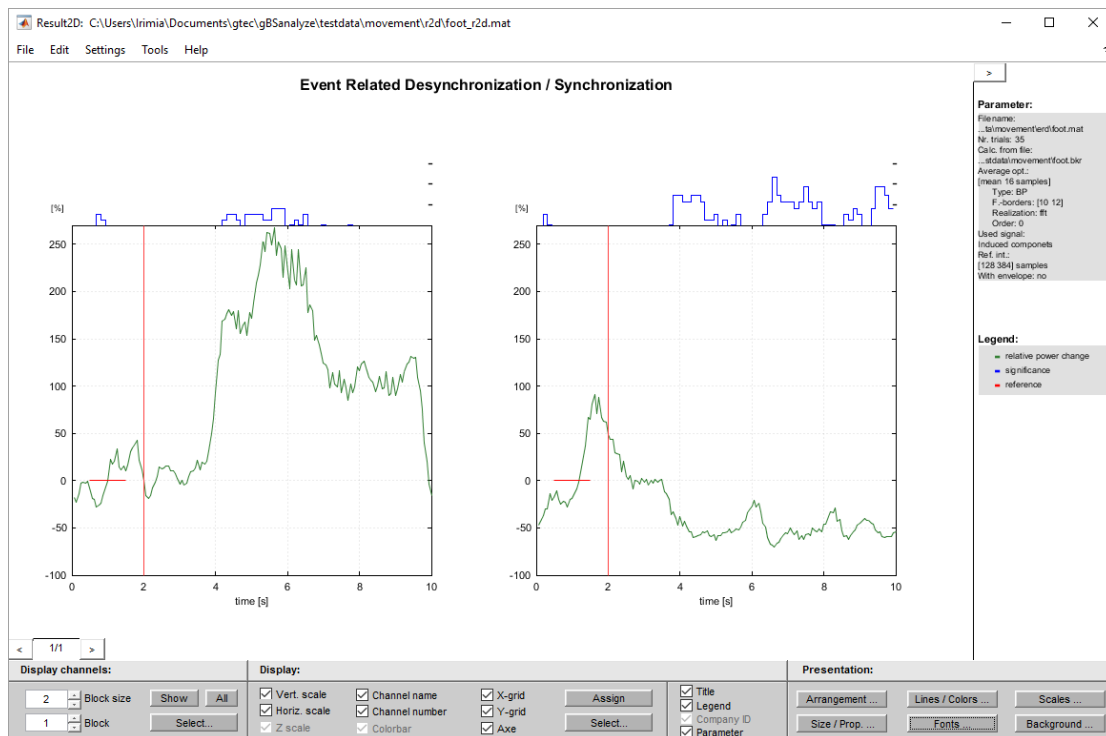
```
gResult2d
```

and press enter to start the application or use the menu of gBSanalyze and select **gResult2D** under **Run!**

2. To load a gResult2D file into the viewer select **Open** of the **File** menu
3. Go to the directory

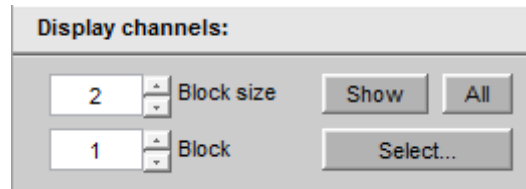
```
Documents\gtec\gBSanalyze\testdata\movement\r2d
```

and select the `foot_r2d.mat` file and press **Open**



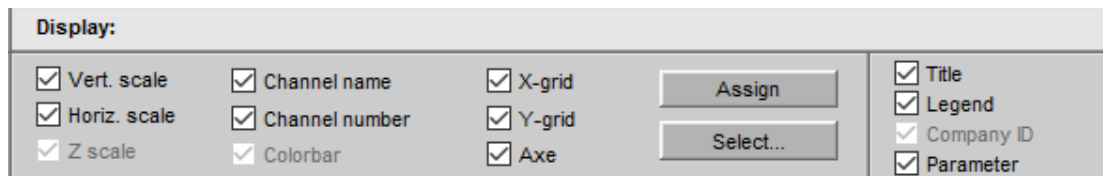
Display Options

To view only specific channels click on **Select** and chose the corresponding channels. The block mode allows switching fast between a selectable amount of channels. Press **All** to view all channels.



To assign specific layout options check the corresponding boxes and channels and press **Assign**.

- **Vertical scale** – show the vertical scale
- **Horizontal scale** - show the horizontal scale
- **Z scale**- show z scale
- **Channel name** - show the name of the channels as plot title
- **Channel number** - show the channel number as plot title
- **Colorbar** – show the colorbar of maps
- **X-grid** - show a vertical grid
- **Y-Grid** - show a horizontal grid

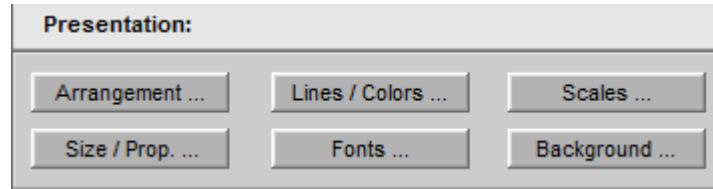


Select channels – allows to show and edit only specific channels

- **Title** - shows a global title
- **Legend** – show a legend
- **Company ID** - show company information
- **Parameter** – show additional parameters

Presentation Options

Presentation allows making the following modifications



- **Arrangement ...** - gives the possibility to arrange the plots by topographic information (x- and y- electrode positions), by channel type specific information (e.g. start with EEG channels followed by EOG channels) and by the channel number.
- **Size / Prop. ...** - gives the possibility to change the size and the x/y proportion of the plots. Select the axis with a left mouse click before starting **Size / Prop.**
- **Lines / Colors ...** - gives the possibility to change curve settings like line thickness, color and style. Select the curve/line with a left mouse click before starting **Lines / Colors.**
- **Fonts ...** - change the font. Select the text with a left mouse click before starting **Fonts.**
- **Scales ...** - change the x-, y- and z-scaling as well as the colorbar of colormaps. Select the axis or colorbar with a left mouse click before starting **Scales.**
- **Background ...** - load a background image

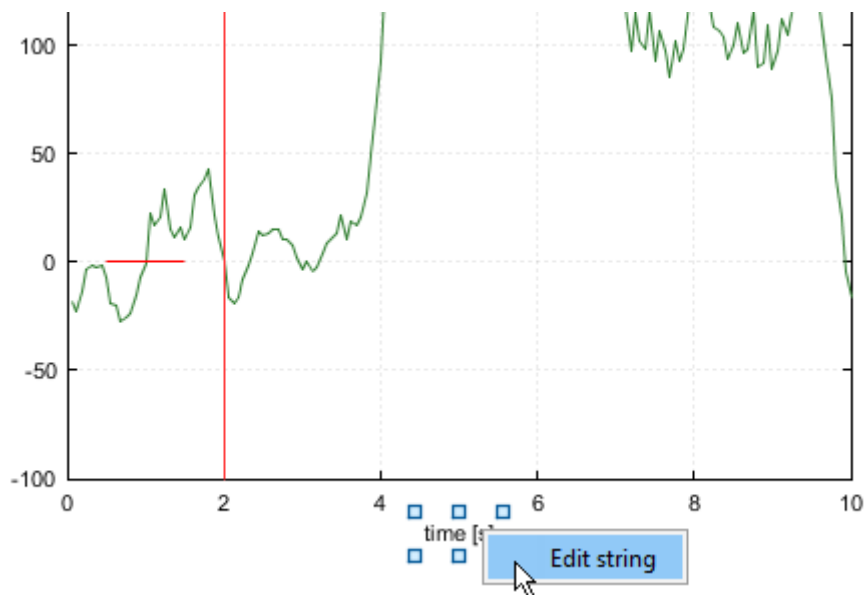
Menu Edit and Settings

The **Edit** menu has the following functions:

Undo last changes – go back one or multiple steps

Copy to clipboard – copy the current gResult2D figure to the clipboard. Note that Windows deletes the contents of the clipboard if e.g. MS Word is started. Therefore, open e.g. MS Word before copying the results to the clipboard.

Edit the Company ID, Legend, Parameter, XLabel, YLabel, ZLabel and Title by using the menu item or the context menu with the right mouse button.



Create plot – create a new MATLAB figure of the current gResult2D figure. This can be used to have full access to the result plot under MATLAB.

The **Settings** menu allows to clone gResult2D properties like line thickness, colors, arrangement,... from one file and to apply it on a new result file. Two options are available:

with current plot – take the settings of the current plot and clone it to the new one

with loaded plot – take the settings of an already stored plot and clone it to the new one

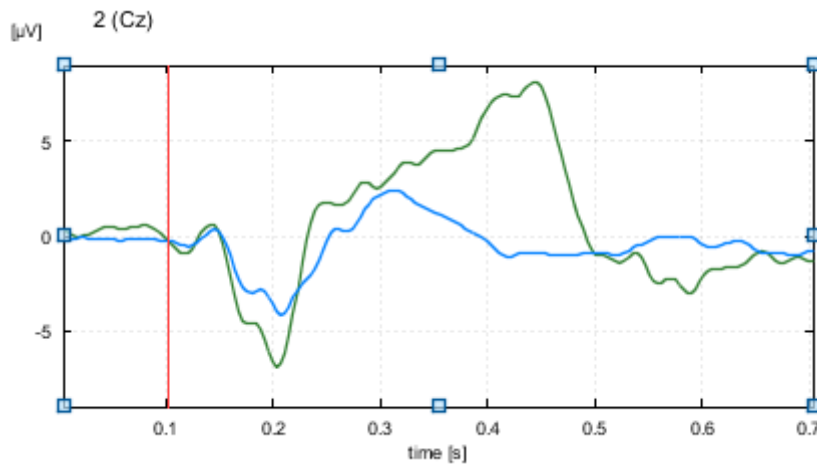
Cloning is used for analyzing many data-sets of e.g. multiple subjects. Therefore the settings are performed for the first subject and are cloned to all other subjects.

Result Measure

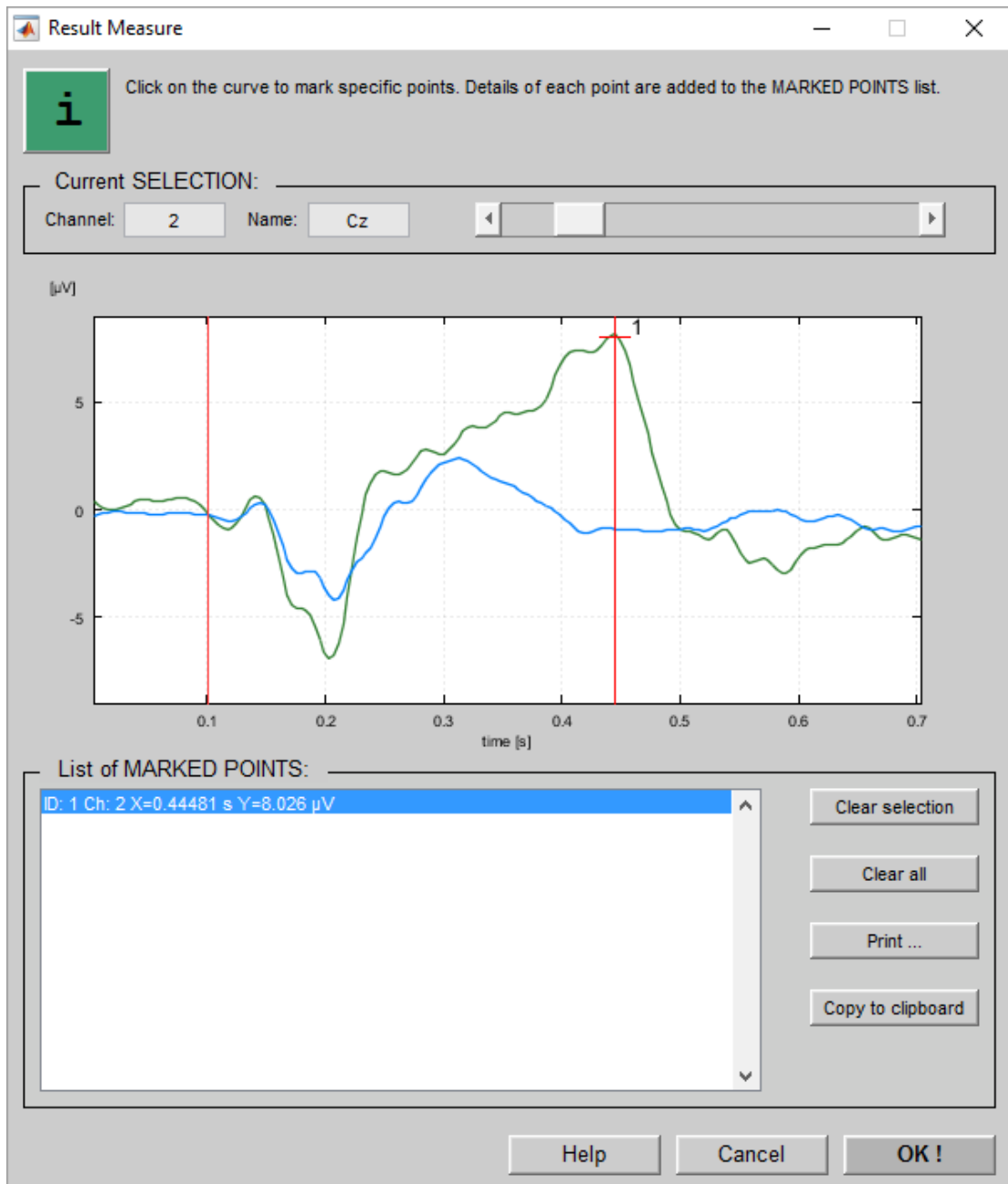
The menu **Tools** has a function called **Measure** which allows to measure specific points of the result plots.

Perform the following steps:

1. Click on the axis that should be investigated



2. Select **Result Measure** from the **Tools** menu to open the following window



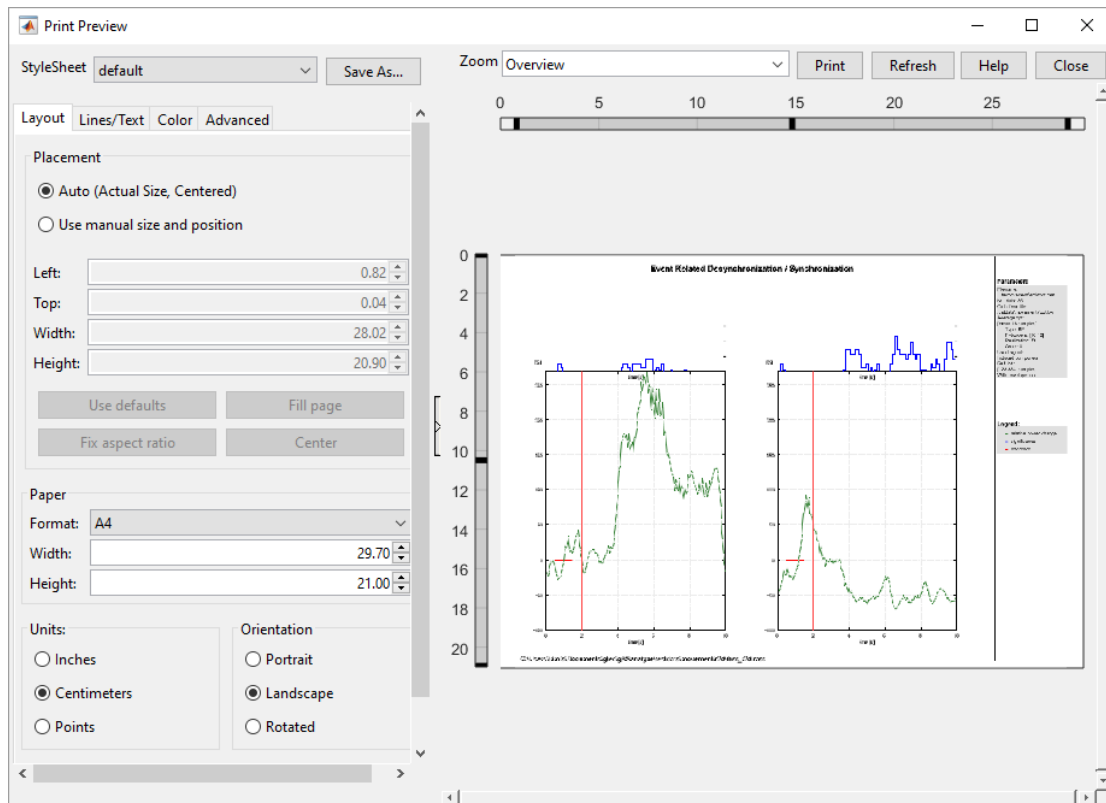
3. Click on the curve to mark specific points. Details of each point are added to the **MARKED POINTS** list.

Use the slider to select different channels.

Print and Preview

Use the **Print Preview** and **Print** menu items under the **File** menu to make hardcopies of your results.

The **Page setup** and **Print Preview** options under the **File** menu allow the user to control the layout and the appearance of the figure before sending it to a printer or print file.

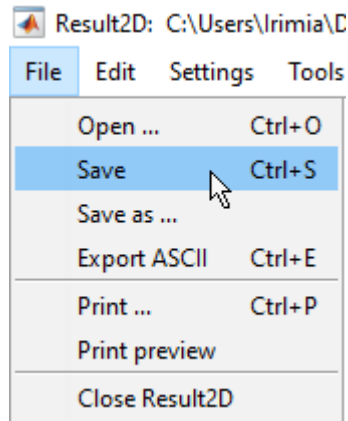


By default the results are printed without the Result2D graphical user interface. Avoid this by enabling the **Print UIControl**s checkbox in the **Advanced** tab.

For details, see MATLAB > Graphics > Printing and Exporting > printpreview help topics.

Save Results

To save changes use the **Save** or **Save as** option of the **File** menu.



If the **Save** option is selected the changes will be saved in the same file as currently loaded. The **Save as** option allows to select a new filename for saving.

Example:

Save the changed Result2D file as a new file named 'erd8_10Hz.mat'.

1. Select **Save as** in the **File** menu to open the dialog
2. Move to directory

```
Documents\gtec\gBSanalyze\testdata\bci\erd
```

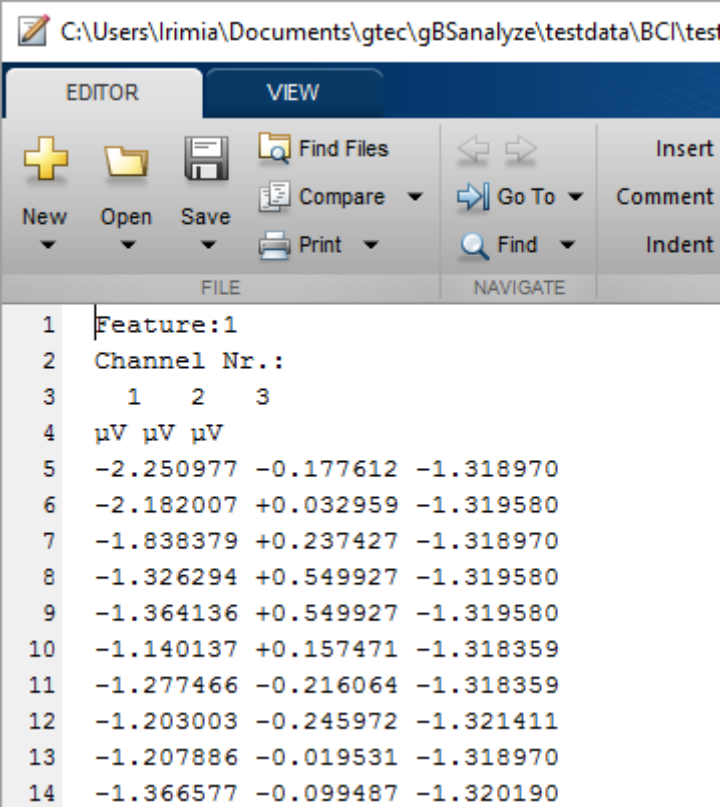
3. Insert the filename `erd8_10Hz.mat` into the **filename** editor field
4. Press the **Save** button

Export ASCII

The **Export ASCII** function allows exporting the data in gResult2D to a file in ASCII format. This ASCII file can be loaded into third-party programs like Microsoft Excel or SPSS.

Perform the following steps:

1. Load `trig1.mat` from the following directory into gBSanalyze
`Documents\gtec\gBSanalyze\testdata\BCI`
2. Select the **Average** function from the **Analyze** menu and calculate a simple average with default settings. gResult2D opens automatically with the averaging results.
3. Click on **Export ASCII** under the **File** menu in gResult2D and specify the filename `test.txt` to export the data. The MATLAB Editor opens with the ASCII data. The first line contains the feature number, followed by the channel numbers and sensitivities. The average values are given for each channel as column. The rows represent the time steps.



```
1 Feature:1
2 Channel Nr.:
3   1   2   3
4  µV µV µV
5 -2.250977 -0.177612 -1.318970
6 -2.182007 +0.032959 -1.319580
7 -1.838379 +0.237427 -1.318970
8 -1.326294 +0.549927 -1.319580
9 -1.364136 +0.549927 -1.319580
10 -1.140137 +0.157471 -1.318359
11 -1.277466 -0.216064 -1.318359
12 -1.203003 -0.245972 -1.321411
13 -1.207886 -0.019531 -1.318970
14 -1.366577 -0.099487 -1.320190
```

Closing gResult2D

To close the **gResult2D** window use the **Close Result2D** option in the **File** menu.

Alternatively use the **ALT+F4** key combination.

Montage Creator

The **Montage Creator** allows defining electrode positions, names and constellations for source derivations.

The section contains the following topics:

[Starting the Montage Creator](#)

[Select Electrode Grid](#)

[Define Electrode Positions](#)

[Read electrode positions using the Polhemus Patriot Digitizer](#)

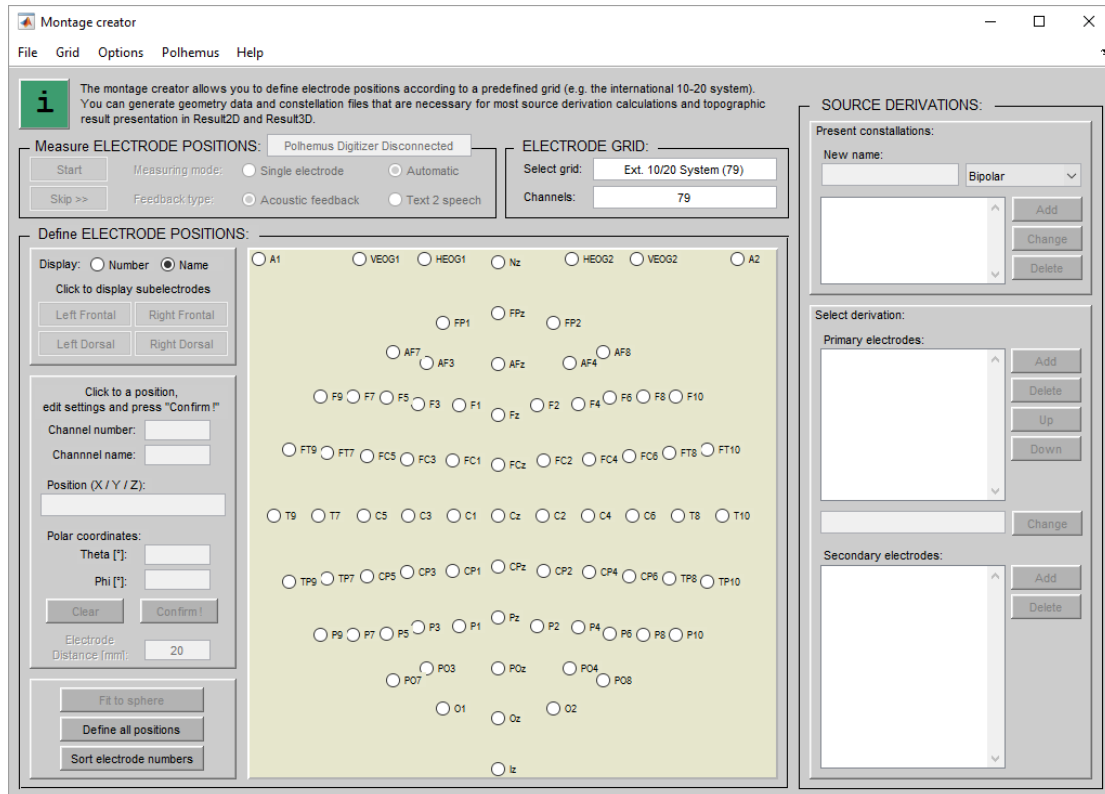
[Define Constellations for Source Derivations](#)

[Import Polhemus](#)

[Save Montage and Exit](#)

Starting the Montage Creator

To start the Montage Creator type gMONcreator in MATLAB or select **gMONcreator** from the **Run!** menu of gBSanalyze. The Montage Creator appears and shows an extended 10-20 electrode system montage with 4 EOG channels.



Select Electrode Grid

A committee of the International Federation of Societies for Electroencephalography and Clinical Neurophysiology recommended a specific system for electrode placement for use in all laboratories [Jasper 1958]. Their recommendation was the system now known as the International 10-20 system. Specific measurements of bony landmarks are used to determine the electrode placement. The electrode placement can be replicated consistently over time and can be replicated between different laboratories.

Jasper, H., "Report of committee on methods of clinical exam in EEG," Electroencephalogr. Clin. Neurophysiol., vol. 10, pp. 370-375, 1958.

American EEG Society, "Guidelines for Standard Electrode Position Nomenclature," J. Clin. Neurophysiol., vol. 8, pp. 200-202, 1991.

Reilly, E.L., "EEG Recording and Operation of the Apparatus," in Electroencephalography: basic principles, clinical applications, and related fields. Editor: Niedermeyer, E., Lopes da Silva, F., 3rd Edition, pp. 104- 124, 1993.

The Montage Creator supports 6 electrode grids:

Simple 10 - 20 system - montage with 25 electrodes according to the international 10 - 20 system. Electrode names and positions are according to the 10 - 20 definition.

Extended 10 - 20 system with 79 electrodes according to the 10 - 20 system. The electrode positions are linear interpolated. This system is also referenced as 10 – 10 system or as a subset thereof.

10 – 10 system diagonal extension with 171 electrodes according to the 10 -10 system with additional positions in the center of the diagonal connections.

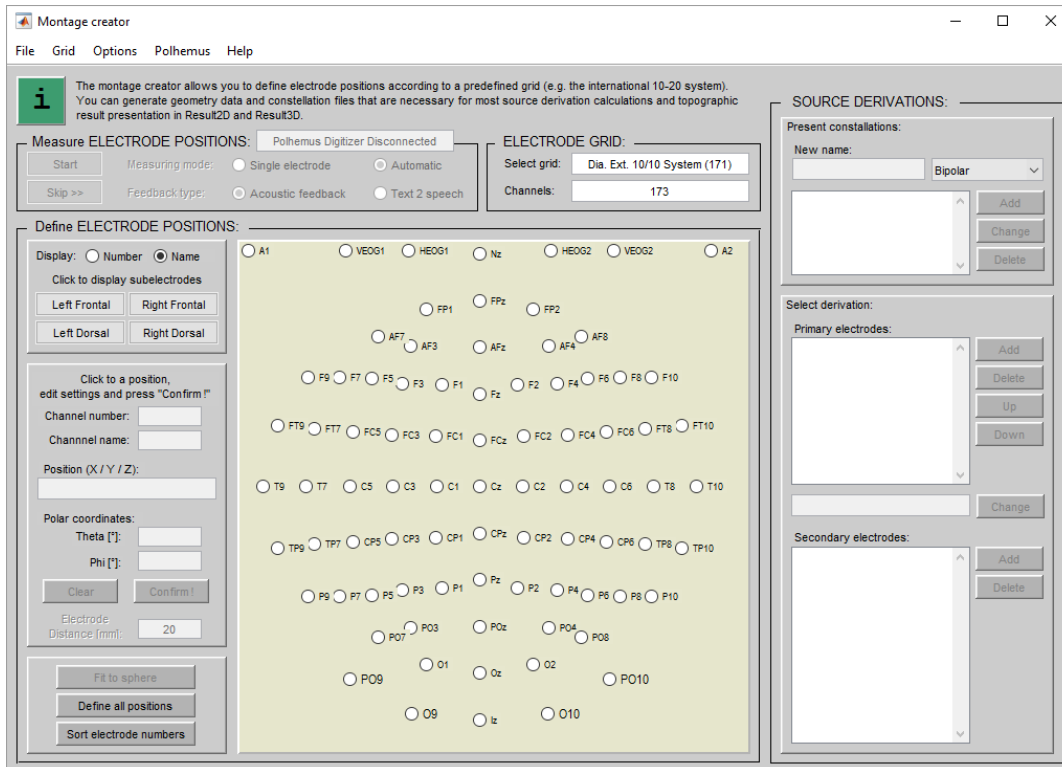
10 -10 system diagonal + rectangular extension with 312 electrodes according to the 10 – 10 system with additional positions in the center of the diagonal and rectangular connections.

Square (11x11) - with 121 electrodes as square matrix with a specific electrode distance.

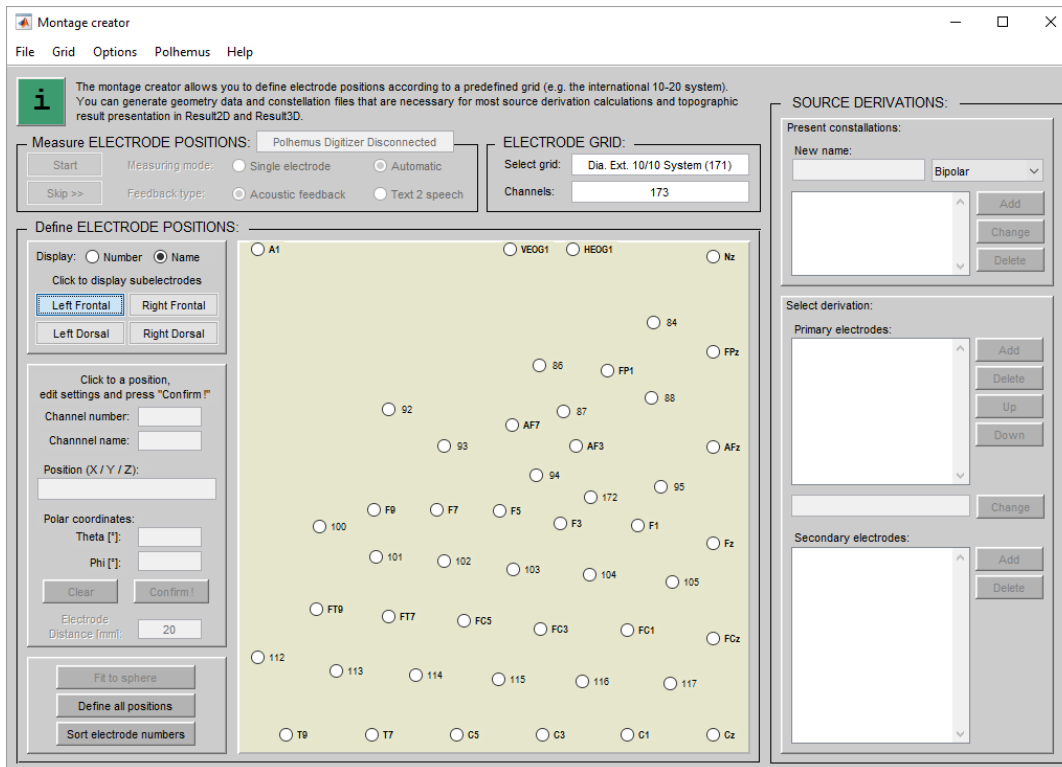
Square (32x32) – with 1024 electrodes as square matrix with a specific electrode distance.

To select a montage go to the **Grid** menu and select the corresponding entry. The Montage Creator changes to the selected montage.

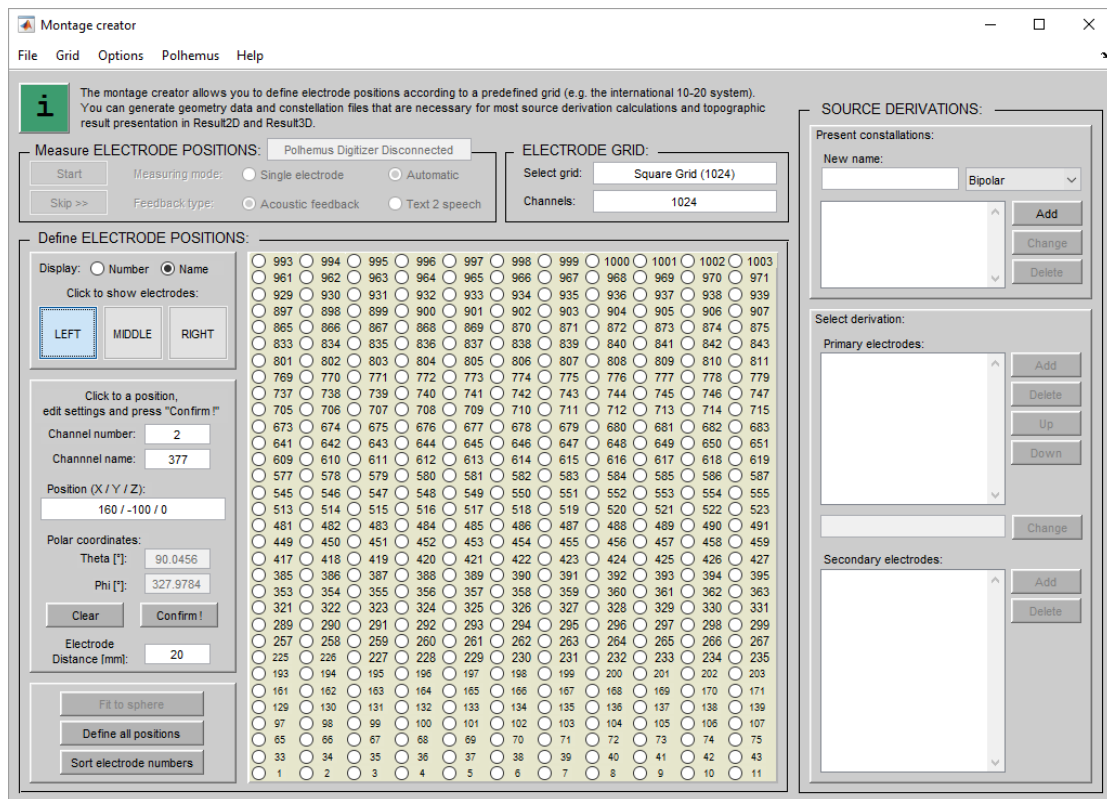
For the **10 – 10 system diagonal extension, 10-10 system diagonal + rectangular extension** initially an overview of the grid is displayed which shows the named positions which are identical to the ones of the **Extended 10 – 20 system**.



The intermediate positions can be displayed by selecting either the **Left Frontal**, **Right Frontal**, **Left Dorsal** or **Right Dorsal** quadrant of the grid. To return to the overview deselect the active quadrant (eg. **Left Frontal**).



The **Square (32x32)** grid is divided into three sections, a **LEFT**, **MIDDLE** and **RIGHT** section. Initially the positions of the **LEFT** section are shown. The remaining positions can be made visible by switching to the **MIDDLE** or **RIGHT** section.



Define Electrode Positions

1. Click on an electrode that should be defined
2. The **Channel number**, **Channel name** and **Position** in x-, y- and z- coordinates as well as in **Polar co-ordinates** appears in the window



3. Click on **Confirm** button to define the electrode. The electrode background gets light-green.
4. Click on **Sort electrode numbers** to sort the electrode numbering from the top left to the bottom right corner of the montage.
5. To view the numbering select **Show Channel number**.

Define Constellations for Source Derivations

The Montage Creator allows to define constellations for **bipolar recordings**

LAR - Local Average Reference

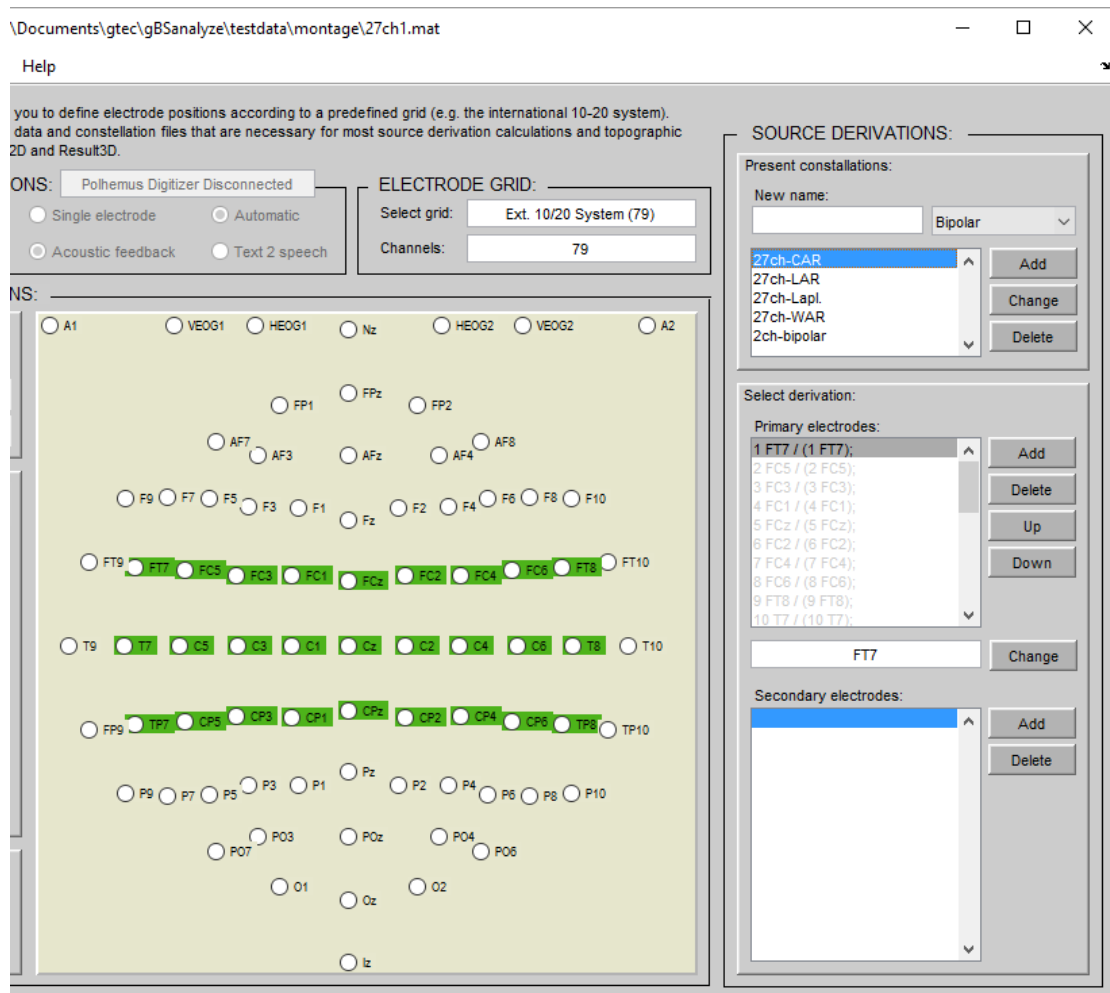
Laplacian - small or large

CAR - Common Average Reference

WAR - Weighted Average Reference

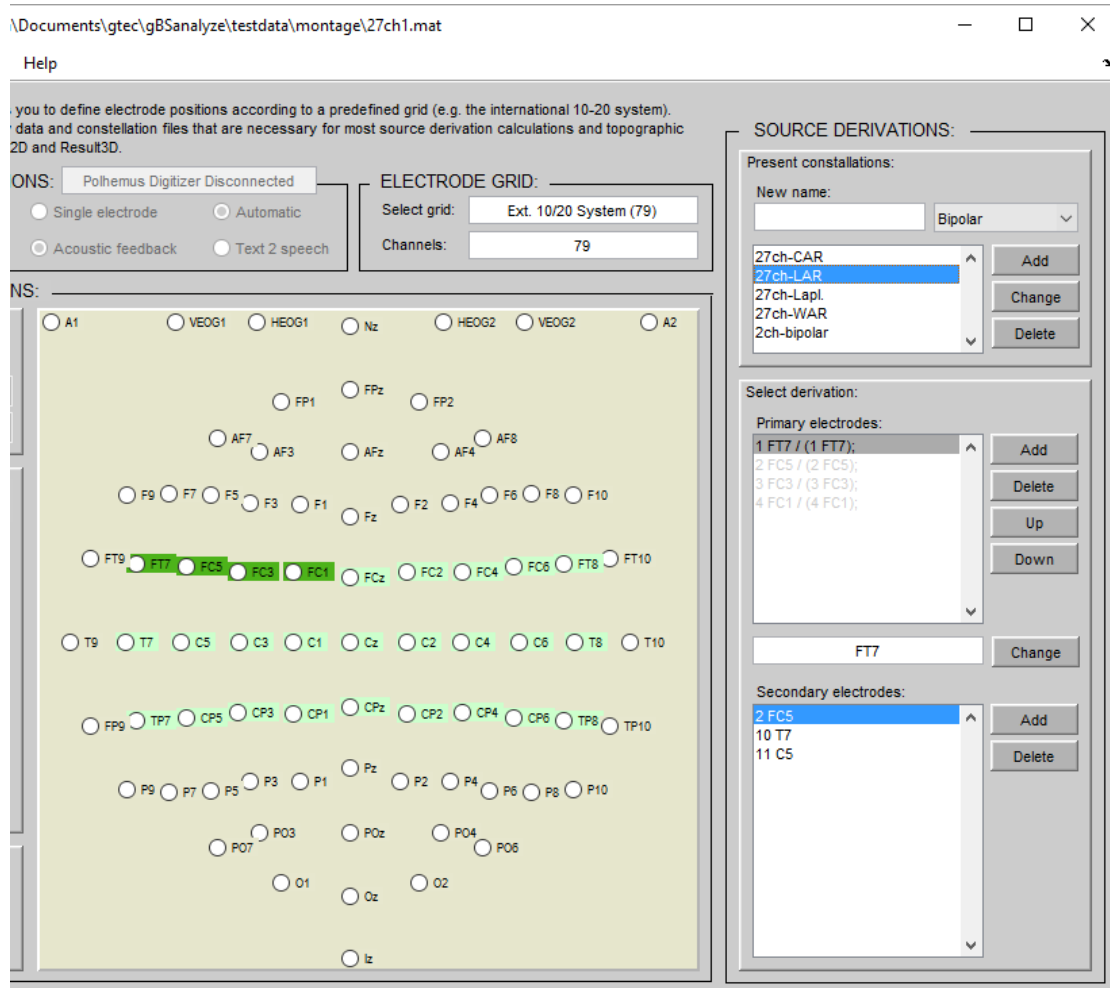
1. Click on **Load montage** from the **File** menu and open the montage file 27ch1.mat from the following directory

Documents\gtec\gBSanalyze\testdata\montage



2. Select under **SOURCE DERIVATIONS** the constellation 27ch-CAR which represents a 27 electrode CAR reference. The list box **Primary electrodes** shows all electrodes which are used for the source derivation. **Primary electrodes** appear in dark green. There are no **Secondary electrodes** defined for the CAR recording because CAR uses the mean of all electrodes for the calculation.

3. Select 27ch-LAR as constellation and click on the first primary electrode FT7. Now the **Secondary electrodes** list box shows all secondary electrodes used for the Local Average Reference calculation. Use the **Add** button to define new Primary and Secondary electrodes. **Up** and **Down** buttons allow to change the order of the primary electrodes which affects the order in g.BSanalyze.



4. Click on the other constellations to view examples for Laplacian, WAR and bipolar recordings

Read electrode positions using the Polhemus Patriot Digitizer

Hardware Requirements

The table below lists the Polhemus Patriot Digitizer hardware and accessories needed for digitizing the coordinates of the electrodes placed on a cap.

Hardware	Properties
Patriot Digitizer	System Electronics Unit (SEU) containing the hardware and software necessary to generate and sense the magnetic fields
Source	TX2 or TX4 magnetic source provided by Polhemus
Stylus	Free-form digitizing tool (3-inch or 8-inch sizes) with single point and continuous line output modes
Sensor	Standard Sensor (RX2)
RS232 cable	Standard RS232 cable with 9-pin female connectors at both ends

Digitizing positions

The Montage Creator allows users to use the Polhemus Patriot Digitizer for reading the real coordinates of the electrodes placed on a cap.

The electrode positions can be recorded in two modes:

- **Single electrode** – define the electrode position for the currently selected electrode.
- **Automatic** – update the positions of all electrodes defined in the loaded montage. After pressing the **Start** button, a message is prompted to the user to measure the nasion, porion and inion positions. Based on the positions of these three points, the reference point coordinates are calculated. After the calculation of the reference point, Montage Creator automatically advances to the next electrode position when the position of the current one has been recorded. The user can manually advance to the next electrode by pressing the **Skip** button.

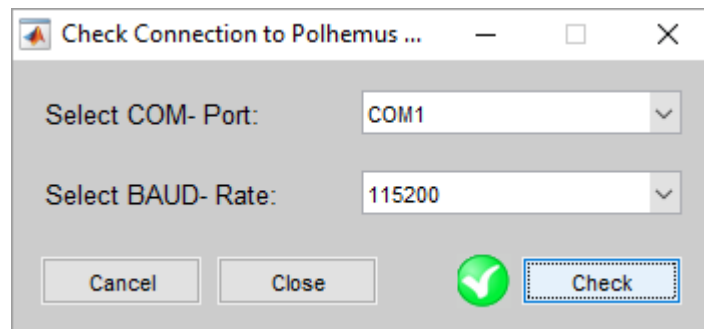
Note: Before starting this experiment, please read the Polhemus Patriot User Manual.

1. Using the provided RS-232 cable, connect the Polhemus Digitizer to the computer serial port;
2. Connect the Stylus Pen to SENSOR 1 input of the digitizing device;
3. Mount the g.GAMMAcap² on the user's head;

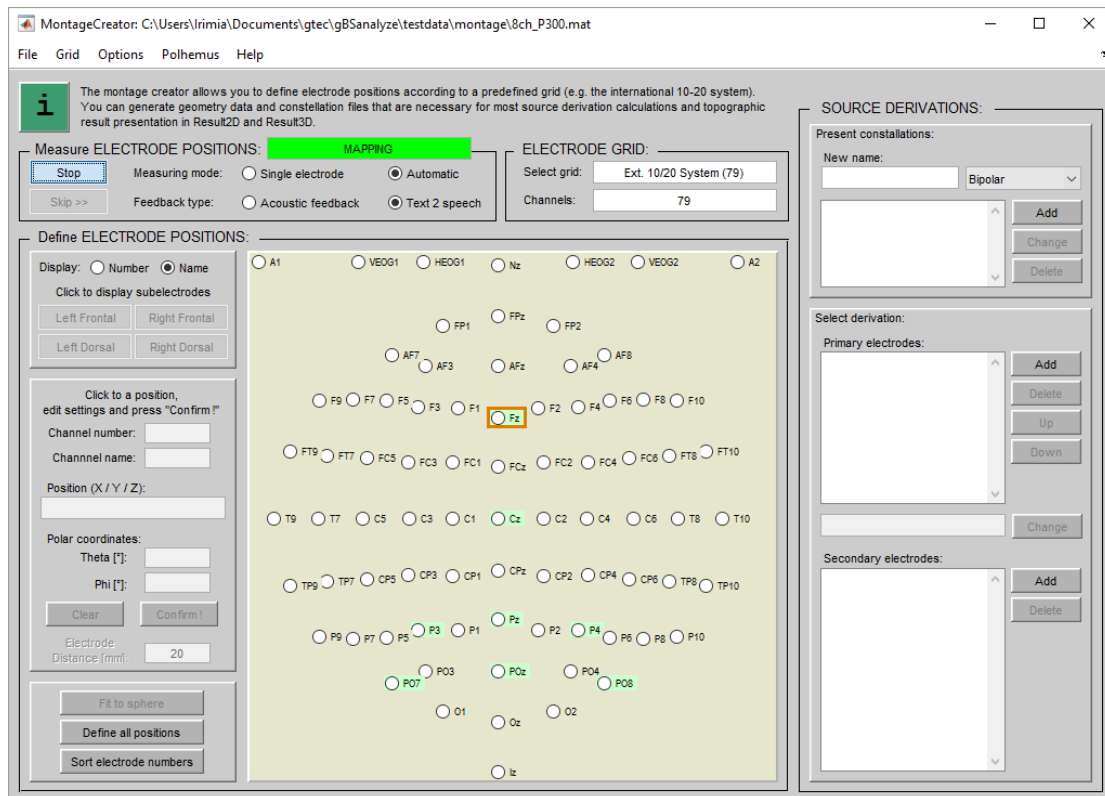
NOTE: When digitizing the electrode positions please make sure that the user

is not moving at all his head during the digitization process, or he keeps it fixed in a special frame.

4. Power-on the Polhemus digitizing device and wait until the **SYS LED** indicator changes its colour to green;
5. Click on **Load montage** from the **File** menu and open the montage file `8ch_P300.mat` from the following directory:
`Documents\gtec\gBSanalyze\testdata\montage`
6. Click on **Check Connection** from the **Polhemus** menu;
7. Select the **COM- Port** to which the digitizer is connected and a **BAUD- Rate** of 115200 bits/second;
8. By clicking the **Check** button, Montage Creator will check the connection to the Polhemus digitizer. If the device is connected, a green checkmark is displayed in the left part of the **Check** button. If the device cannot be found on the selected port, a red cross will be displayed instead of the green checkmark;



9. Click on the **Close** button. The **Measure ELECTRODE POSITIONS** frame will be activated, and the **Polhemus Digitizer Connected** message will be displayed;
10. Select the **Automatic** measuring mode and the **Text 2 speech** feedback type;
11. Click the **Start** button;
12. Based on the prompted message and the acoustic feedback, measure the nasion, inion and porion positions by pressing the Stylus Pen button;
13. Measure one by one the positions of the electrodes indicated by the Montage Creator.



The positions for all measured electrodes will be stored in the montage object. To restore the original positions of all electrodes, use the **Fit to sphere** option with a radius of 1.

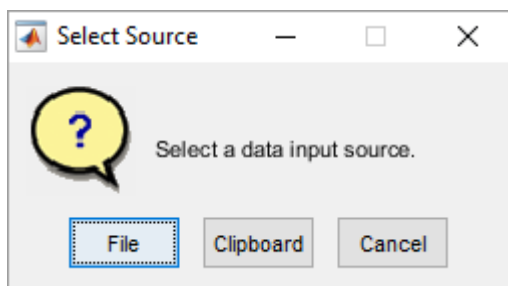
Import Polhemus

Use **Import Polhemus** under the **File** menu to assign electrode co-ordinates to your electrode montage. Note that the number of electrode co-ordinates in your import file must match the number of defined electrodes in the Montage Creator.

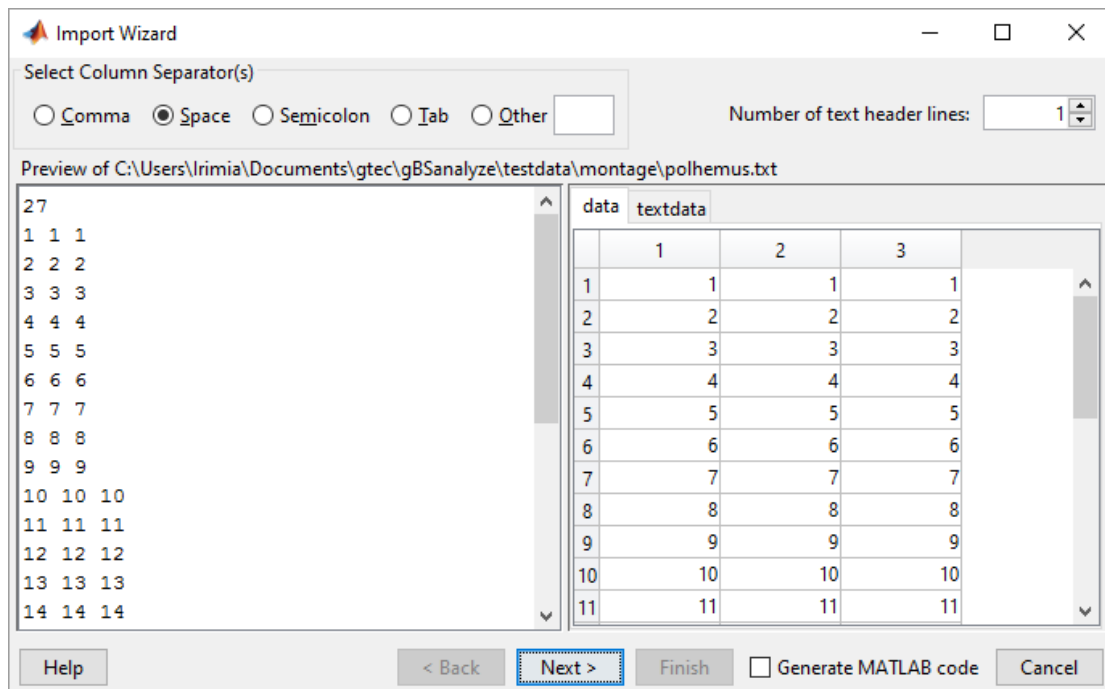
1. From the **File** menu select **Load Montage** and browse to the file 27ch1.mat in the folder

Documents\gtec\gBSanalyze\testdata\montage

2. Select **Import Polhemus** from the **File** menu. The **Select Source** dialog will open, click on **File** and select the polhemus.txt file.



3. The first entry of the file is the number of electrodes followed by x-, y-, and z-co-ordinates for each electrode. **Number of text header lines** has to be set to 1.



4. Click on **Next** and **Finish** to import the co-ordinates and assign it to your electrode definition

Save Montage and Exit

1. Click on **Save Montage as** under the **File** menu and enter a filename for the montage
2. Select **Exit Montage Creator** to close the application

Help

g.BSanalyze provides a printable documentation and a function help.

The printable documentation is stored under

```
C:\Program Files\gtec\gBSanalyze\Help
```

as gBSanalyze.pdf. Use Acrobat Reader to view the documentation.

To view the function help type

```
help gBSfunctionname
```

under the MATLAB command window.

To view all functions which are available in batch mode type

```
gBSfunctions
```

Accessing Data

g.BSanalyze stores data and header information in an object called P_C of class data. To access the data object while g.BSanalyze is running define the data object **P_C** as global in the MATLAB command window:

```
global P_C
```

If the data are stored to harddisk the name of the data object is changed to **P_C_S**.

To access the data object see sections:

[Using the get Command](#)

[Using the set Command](#)

A detailed list of object entries can be found below:

General		
Data	Matrix	Data in format trials x samples x channels
PreTrigger	Scalar	Interval before trigger
PostTrigger	Scalar	Interval after trigger
TrialNumber	Vector	Number of each trial
Version number	Scalar	Current release number
Amplifier and Channel Settings		
Channels	Vector	Channel numbers
ChannelName	Cell of strings	Name of channels
LowPass	Vector	Lowpass settings
HighPass	Vector	Highpass settings
Notch	Vector	Notch settings
Sens	Vector	Sensitivity settings
Unit	Vector	Unit of sensitivity settings
NumberChannels	Scalar	Number of I/O channels
ChannelType	Vector	Channel type (1=N.S. / 2=EEG / 3=EOG / 4=EMG / 5=ECG / 6=ECoG / 7=Resp / 8=Trig)
PhysicalValue	Vector	Physical value of input signal
PhysicalUnit	Vector	Unit for physical value (1... μ V, 2...mV, 3...V)
MatlabValue	Vector	Value of the physical value in MATLAB (calibration)
ScalingSens	Vector	Sensitivity for scaling of data
ScalingUnit	Vector	Unit for ScalingSens
Xposition	Vector	Electrode x position

Yposition	Vector	Electrode y position
Zposition	Vector	Electrode z position
MontageName	String	Name of montage file
AmplifierName	String	Name of amplifier(s)
Subject Information		
SubjectLastName	String	Last name of subject
SubjectFirstName	String	First name of subject
SubjectID	String	Subject identifier
SubjectBirthday	String	Birthday
SubjectDiagnosis	String	Diagnosis
SubjectMedication	String	Medication
SubjectComment	String	Comment
SubjectHand	Scalar	Handiness. (1=not spec., 2=right, 3=left, 4=mixed)
SubjectSex	Scalar	Sex (1='not spec.', 2='male', 3='female')
Company Information		
CompanyName	String	Name of Company
CompanyInstitute	String	Name of Institute
CompanyDepartment	String	Name of Department
CompanyInvestigator	String	Name of Investigator
CompanyOperator	String	Name of Operator
CompanyCopyright	String	Copyright
CompanyYear	String	Year
CompanyLogo	String	Name and path of logo
Session/Paradigm Information		
SessionNumber	String	Session number
SessionTime	String	Session recording start time
SessionDate	String	Session recording date
SessionStudyName	String	Study name
SessionComment	String	Comment
ParadigmName	String	Paradigm name
ParadigmRun	String	Run number
ParadigmCondition	String	Condition
ParadigmISI	String	Inter Stimulus Interval
ParadigmComment	String	Comment
Storage		
FileName	String	Path and file name
DAQ settings		

DAQBoard	Cell of Strings	I/O boards
DAQBoardName	Cell of Strings	I/O board name
DAQChannels	Cell of Scalars	I/O board channels
DAQResolution	Cell of Scalars	DAQ board resolution
DAQSamplingFrequency	Scalar	Sampling Frequency
Attributes, Markers		
Attribute	Matrix	Attributes assigned to trials
AttributeName	Cell	Name of trial attributes
AttributeColor	Cell	Color of trial attributes
Marker	Matrix	Sample markers
MarkerName	Cell	Name of markers
MarkerColor	Cell	Color of markers
ChannelAttribute	Matrix	Attribute of channels
ChannelAttributeName	Cell	Name of channel attributes
ChannelAttributeColor	Cell	Color of channel attributes
Epoching		
EpochingSelect	Cell	Marked epochs in free-mode, multi-channel and multi-trial mode. Column 1: start point sample number Column 2: trial number; 0 means all trials Column 3: channel number; 0 means all channels Column 4: length of epoch in samples Column 5: epoching type: 1..free mode, 3...multi-channel mode, 4...multi-trial mode Column 8: epoch number Column 9: epoch name ID (e.g. SleepStage 1) Column 11: epoch comment all other columns are for internal usage
EpochingName	Cell	Names for different epochs (e.g.: 'ANALYZE', 'SLEEPSTAGE1',...)
EpochingColor	Cell	Colors for different epochs

Using the get Command

The get method provides a way to access the data object entries

Syntax

```
get(P_C_S, 'PropertyName')
```

returns the value of the property *'PropertyName'* of the data object P_C_S

Example

```
get(P_C_S, 'SamplingFrequency')  
get(P_C_S, 'SubjectLastName')
```

Using the set Command

The set method provides a way to set data object properties.

Syntax

```
set(P_C_S, 'PropertyName', 'PropertyValue')
```

assigns the *'PropertyValue'* to the specified *'PropertyName'* of the data object.

Example

```
set(P_C_S, 'SamplingFrequency', 128)  
set(P_C_S, 'SubjectLastName', 'Mr. Miller')
```


User Extensions

g.BSanalyze allows to incorporate your own M-files and graphical user interfaces. The software scans through the directory

```
Documents\gtec\gBSanalyze\user
```

for M-files and makes all of them accessible in the menu **User**.

Open `gTimesTwo.m` in the user directory with the MATLAB Editor to view an example.

```
function gTimesTwo(varargin) ;
global P_C
global V_R

y=P_C.Data;           %import the data
y=y*2;               %make your calculations
P_C.Data=y;          %give the results back

[V_R]=plot(P_C,V_R); %visualize the new data
```

`gTimesTwo.m` reads the biosignal data stored in the data object `P_C` into matrix `y` and multiplies the matrix by 2. The result is passed back to the data object and plotted in the Data Editor.

To view an example for bandpass filtering open `BandpassFilter.m` under the user directory. This example uses a `g.BSanalyze` command to bandpass filter the data.

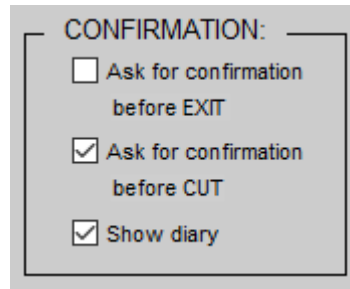
```
function BandPassFilter(varargin) ;
global P_C
global V_R

Filter.Realization='butter';
Filter.Type='BP';
Filter.Order=5;
Filter.f_high=16;
Filter.f_low=12;
TrialExclude=[];
ChannelExclude=[];
P_C=gBSfilter(P_C,Filter,ChannelExclude,TrialExclude);

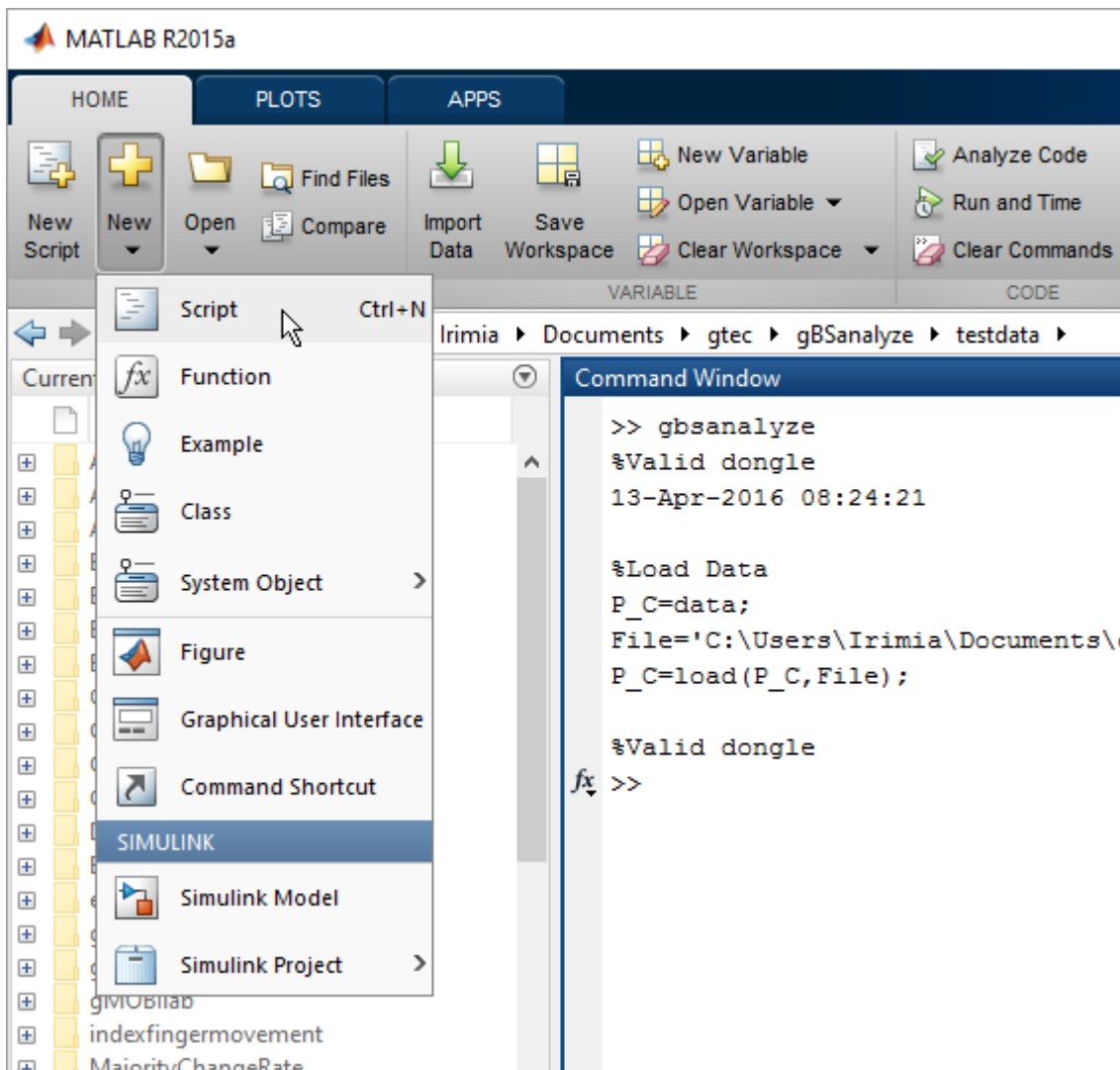
[V_R]=plot(P_C,V_R);
```

Batch Mode

The easiest way to create a batch for data processing is to perform the analysis under the Data Editor with the graphical user interfaces. Make sure that the **Show diary** checkbox is enabled in **Appearance Settings** under the **Options** menu.



This forces g.BSanalyze to report all calculations in the MATLAB command window. After finishing the analysis open a **New Script** and copy and paste all commands into the file.



```

1 %Load Data
2 P_C=data;
3 File=['C:\Users\Irimia\Documents\gtec\gBSanalyze\testdata\Classify\',...
4 'classifieddata\mi_2class_80trials.mat'];
5 P_C=load(P_C,File);
6
7 %ClassificationOutputMapping
8 ClassIndex=[3 4];
9 ChannelExclude=[3 4 5];
10 TrialExclude=[];
11 FileName='';
12 SignificanceLevel=[5];
13 ProgressBarFlag=1;
14 V_O = gBSclassificationoutputmapping(P_C,ClassIndex,ChannelExclude,...
15 TrialExclude,FileName,SignificanceLevel,ProgressBarFlag);
16
17 %Calling g.Result2D
18 result2D = CreateResult2D(V_O);
19 gResult2d(result2D);

```

Save the batch in your own directory as `mybatch.m` and start the batch under the MATLAB command window with

```
mybatch
```

For further data-sets just replace the input data file to perform the same analysis.

Calling `g.Result2d` from the command line

Analyze and classification methods produce specific objects that can be viewed with `g.Result2d` and stored to harddisk. To open such an object from the command line type

```
R_O = CreateResult2D(C_O_S);
```

to create a `result2d` object of the `C_O_S` (e.g. classifier object).

then enter

```
gResult2d(R_O);
```

to view the object or

```
gResult2d(R_O,'print');
```

to print the object.

Starting g.BSanalyze from the command line

To start g.BSanalyze from the MATLAB command line and to open immediately a file use

```
dataedit('filename');
```

Data-sets

This section describes experimental procedures used to acquire the data-sets included in g.BSanalyze:

[Movement Imagination](#)

[Right and Left Hand Movement Imagination](#)

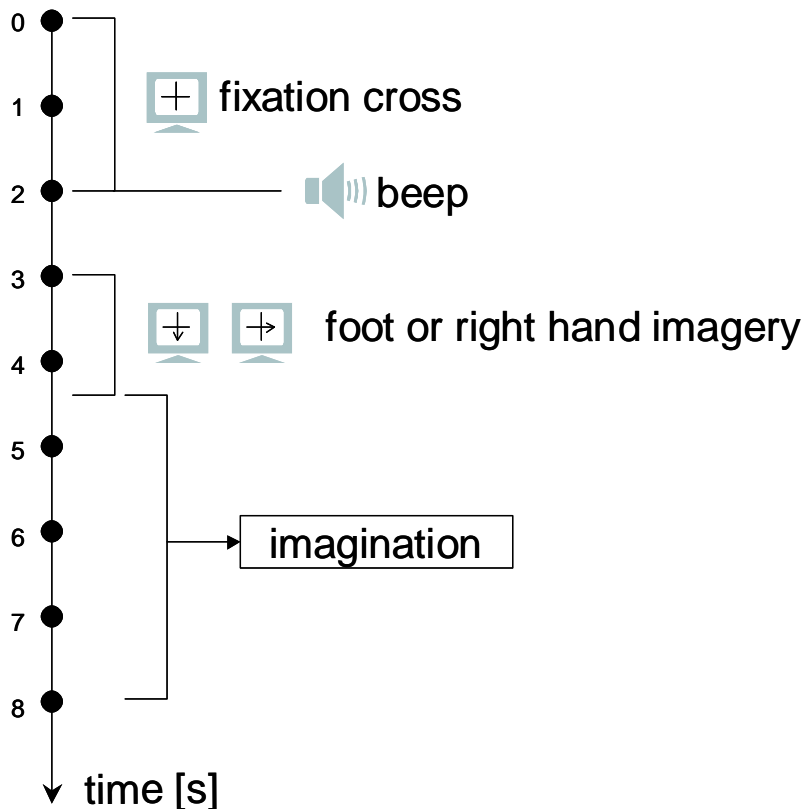
[Index Finger Movement](#)

[Evoked Potential](#)

Movement Imagination

The figure shows the timing of one trial of the experiment. The subject sat in a comfortable armchair 150 cm in front of a computer-monitor and was instructed not to move and to keep both arms and feet relaxed and to maintain throughout the experiment the fixation at the center of the monitor. The experiment started with the display of a fixation cross that was shown in the center of a monitor. After two seconds a warning stimulus was given in form of a "beep". From second 3 until 4.25 an arrow (cue stimulus), pointing down or to the right, was shown on the monitor. The subject was instructed to imagine a foot or right hand movement, depending on the direction of the arrow until second 8.

One trial lasted 8 seconds and the time between two trials was randomized in a range of 0.5 to 2.5 seconds to avoid adaptation. The subject performed 4 runs consisting each of 40 trials.



Path: Documents\gtec\gBSanalyze\testdata\movement

Files: foot.bkr
right.bkr

Channels: C3 - right hand movement representation
Cz - foot movement representation

Derivation: Local Average Reference

Right and Left Hand Movement Imagination

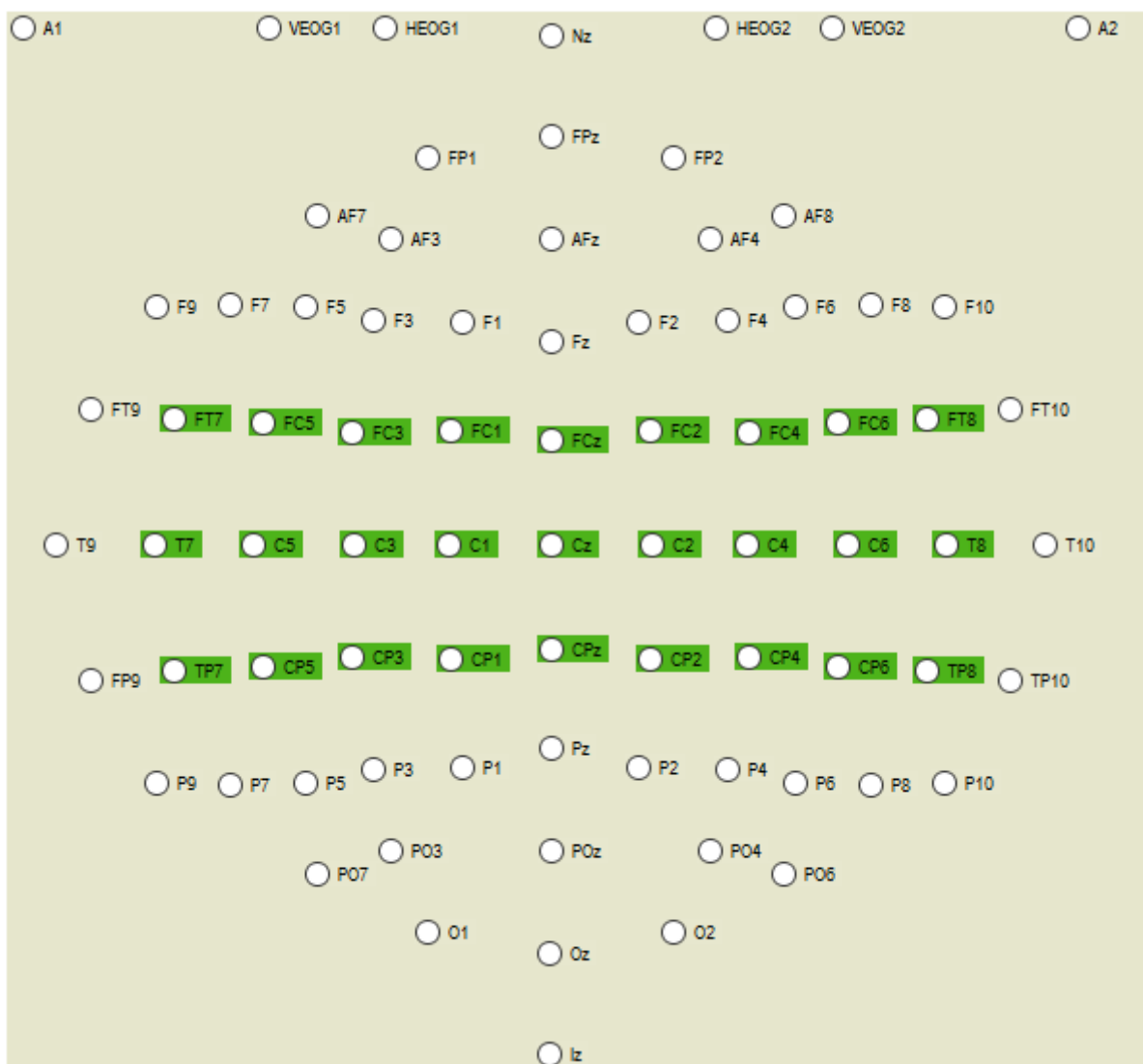
The timing of the experiment is described under section Movement Imagination. But the subject was instructed to imagine a right hand movement if the arrow pointed to the right side and a left hand movement if the arrow pointed to the left side.

Path: Documents\gtec\gBSanalyze\testdata\RightLeftImagination

Files: run1234.mat
run1234_geom.mat

Montage: Documents\gtec\gBSanalyze\testdata\montage\ch271.mat

C3 - right hand movement representation area
C4 - left hand movement representation area



Derivation: referenced to the right ear

Index Finger Movement

The subject sat in a comfortable armchair and was instructed to move the right index finger every 10 seconds. A movement sensor was attached to the right finger and allowed to register movement on-set and off-set. The data presented here is movement off-set triggered with 4 seconds a priori movement and 4 seconds after movement.

Path: Documents\gtec\gBSanalyze\testdata\indexfingermovement

Files: finger.bkr

Montage: Channel 11 is C3_{..} - right hand movement representation
 Channel 14 is Cz - foot movement representation
 Channel 17 is C4 - left hand movement representation

Derivation: small Laplacian

Evoked Potential

The subject sat in a comfortable armchair and the medianus nerve of the right hand was electrically stimulated. 800 trials with 500 ms length are recorded.

Path: Documents\gtec\gBSanalyze\testdata\EP

Files: session1.bkr

Montage: Channel 1 anterior to C3
Channel 2 posterior to C3
Channel 3 anterior to Cz
Channel 4 posterior to Cz

Derivation: referenced to the right ear

Product Page

Please visit our homepage www.gtec.at for

- Update announcements
- Downloads
- Troubleshooting
- Additional demonstrations



contact information

g.tec medical engineering GmbH
Sierningstrasse 14
4521 Schiedlberg
Austria

tel. +43 7251 22240
fax. +43 7251 22240 39
web: www.gtec.at
e-mail: office@gtec.at