1st BCI2000 Workshop

The Desmond Conference Center

Albany, New York, USA
June 13+14, 2005

Organized by

Gerwin Schalk
Wadsworth Center, New York State Department of Health

Faculty:

Dennis McFarland, Dean Krusienski, Bill Sarnacki, Eric Sellers,
Juergen Mellinger, Christoph Guger, Febo Cincotti

1st BCI2000 Workshop
The Desmond Conference Center
Albany, New York, USA
June 13+14, 2005

Final Program: Monday, June 13 – Theory

08:45a-09:15a	**Breakfast and Check-In**
09:15a-10:00a	**BCI2000 – General Concepts and Operation** (*Gerwin Schalk*) • The current situation in BCI research; the need for a general-purpose system • The four components of any BCI system and their implementation in BCI2000 • The benefits of BCI2000 to the user and the software engineer • The current status of BCI2000 implementations and dissemination
10:00a-10:15a	**Break**
10:15a-11:00a	**A Functioning BCI is a System – Not One Component** (*Dennis McFarland*) • Data Acquisition (signal conditioning, filtering, digitization) • Signal Processing (adaptation, etc.) • The User Task (task design, psychological and training aspects)
11:00a-11:15a	**Break**
11:15a-11:45p	**Sensorimotor (i.e., mu/beta) Rhythms in the EEG and their Signal Processing/User Task Support in BCI2000** (*Dean Krusienski*)
11:45p-12:45p	**Lunch**
12:45p-1:45p	**How to Get Quality EEG Recordings** (*Bill Sarnacki*) • Tutorial and hands-on practise
1:45p-1:55p	**Break**
1:55p-2:25p	**The P300 Response in the EEG and its Signal Processing/ User Task Support in BCI2000** (*Eric Sellers*)
2:25p-2:30p	**Break**
2:30p-3:00p	**Sensorimotor Rhythms in ECoG** (*Gerwin Schalk*)
3:00p-3:10p	**Break**
3:10p-4:10p	**Programming and Extending BCI2000** (*Juergen Mellinger*)
4:10p-4:15p	**Break**
4:15p-5:00p	**Hardware Concepts** (*Christoph Guger, g.tec Guger Technologies, Austria*)
6:00p-8:00p	**Optional Social Gathering**

1st BCI2000 Workshop
The Desmond Conference Center

Albany, New York, USA
June 13+14, 2005

Final Program: Tuesday, June 14 – Practise

08:15a-08:45a	**Breakfast**
08:45a-12:00p	**Configuration, Conduction, and Analysis of Sensorimotor Rhythm Experiments** (*Febo Cincotti*)
12:00p-01:00p	**Lunch**
01:00p-01:45p	**Advanced Signal Processing** (*Christoph Guger, g.tec Guger Technologies, Austria*)
01:45p-05:00p	**Configuration, Conduction, and Analysis of P300 Experiments** (*Febo Cincotti*)
05:30p	**Shuttle Service to 3rd International BCI Meeting in Rensselaerville** (*only for registered participants of the 3rd Intl. Meeting*)

BCI2000 – General Concepts and Operation

Gerwin Schalk
Wadsworth Center, New York State Department of Health

1st BCI2000 Workshop
The Desmond Conference Center
Albany, New York, USA
June 13+14, 2005

BCI2000 – General Concepts and Operation

1st BCI2000 Workshop
The Desmond Conference Center
Albany, New York, USA
June 13+14, 2005

Gerwin Schalk
Research Scientist, Chief Software Engineer; Brain-Computer Interface R&D Program
Wadsworth Center, New York State Dept. of Health, Albany, New York, USA

Outline

- **What a BCI is, and What it is Not**
- Current State of BCI Research
- Key Issues for Further Progress
- BCI2000: a General-Purpose Platform

Augmentative Control Options

- Substitute
  (e.g., tongue instead of hand)
- Circumvent
  (e.g., "Freehand" neuroprosthesis)
- Add

Brain-Computer Interface Research

Goal is to provide:

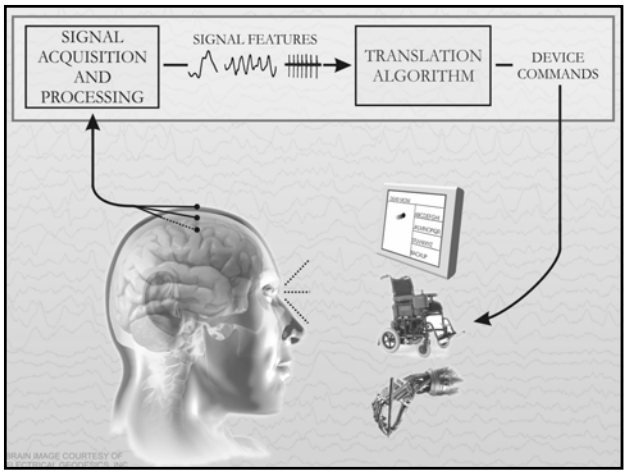
- A new control option (to people with severe motor disabilities)
- Functions that normal muscular control can't address (to able-bodied people)

Brain-Computer Interface Definition

- A control system that does not depend on nerves and muscles
- Messages are carried by electrical or other signals from the brain

Brain-Computer Interfaces (BCIs)

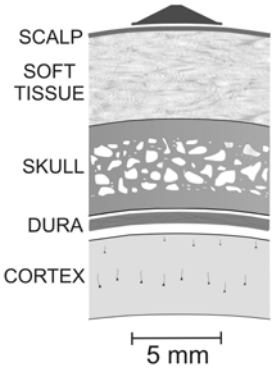
- Add new non-muscular communication channel
- User produces brain signals that encode the message
- System translates these signals into device commands
- Interaction of two adaptive controllers
- BCI use is a new skill that both user and system must master and maintain



Outline

- What a BCI is, and What it is Not
- **Current State of BCI Research**
- Key Issues for Further Progress
- BCI2000: a General-Purpose Platform

Acquisition Method: EEG



Spatial Resolution

centimeters

Safety

very safe

Stability/Durability

practical issues with prolonged application

Non-Invasive BCIs in Humans

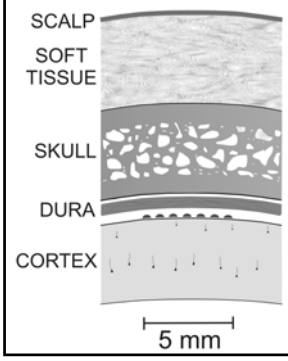
Impressive Demonstrations

- Accurate non-muscular 1D/2D cursor control
- Accurate 2D symbol selection

Significant Impediments

- Prolonged user training (a couple months)
- Not all users learn this skill
- Low information transfer rate
- Susceptibility to non-CNS artifacts

Acquisition Method: ECoG



Spatial Resolution

few millimeters

Safety

manageable risks

Stability/Durability

may provide stable recordings for months

ECoG-Based BCIs in Humans

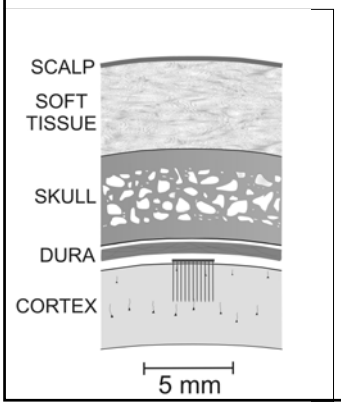
Impressive Demonstrations

- Accurate non-muscular 1D cursor control
- Successful decoding of 2D motor function
- Preliminary results for 2D control
- Manageable clinical risk and may have good electrode stability

Significant Impediments

- Study parameters solely determined by requirements of epilepsy surgery protocol
- Invasive

Acquisition Method: Microelectrodes



Spatial Resolution

tens of microns

Safety

risky procedure?

Stability/Durability

unsolved problems with long-term stability

Invasive BCIs in Monkeys

Impressive Demonstrations

- Accurate non-muscular 3D cursor control

Significant Impediments

- Short- and long-term risks associated with invasive brain surgery
- Electrode stability typically limits recording to few weeks

Outline

- What a BCI is, and What it is Not
- Current State of BCI Research
- **Key Issues for Further Progress**
- BCI2000: a General-Purpose Platform

BCI Research is Interdisciplinary

- Neuroscience
- Physiology/Psychology
- Engineering
- Mathematics
- Computer Science
- Practical Applications

… the Bottom Line

This is a complex problem that needs an integrated and interdisciplinary approach

This is not just a classification problem!!

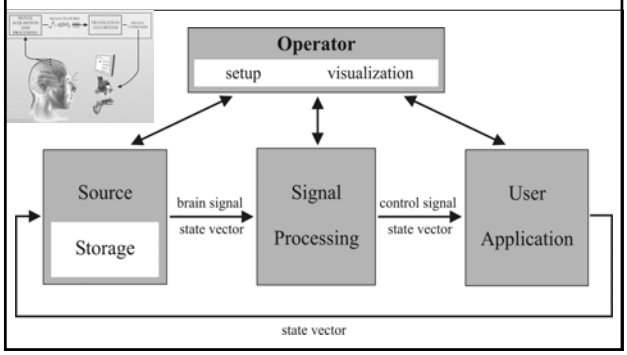
Rationale for Generic R&D Platform

- Progress requires the capacity to compare and contrast different approaches
- Up to now, BCI systems have been very specific
- Method comparison was difficult and expensive

Outline

- What a BCI is, and What it is Not
- Current State of BCI Research
- Key Issues for Further Progress
- **BCI2000: a General-Purpose Platform**

BCI2000 System Model



BCI2000: A General-Purpose System

- Addresses the need for systematic studies
- Can support any BCI method and a variety of other studies
- In use by 50 groups world-wide
- Available with source code and documentation
  http://www.bci2000.org
  Schalk et al., *IEEE Trans Biomed Eng*, 2004

Key Features of BCI2000

- Generic System Model
- Interchangeability and Independence
- Scalability
- Real-Time Capability
- Support of Offline Analyses
- Practicality

Benefits of BCI2000

To the investigator

Availability of a complete system that already allows for a range of studies

To the software engineer

Concentration on the aspects that are unique to the new BCI method that is to be developed

To people with disabilities

System can be configured to the particular brain signal, signal processing method, and application best suited to each person

Current BCI2000 Implementations

Hardware
- Data Translations, Inc.
- National Instruments, Inc.
- Neuroscan, Inc.
- Measurement Computing, Inc.
- Brain Vision (Brain Products)
- USBamp (g.tec)
- AmpServer (Electrical Geodesics, Inc.; in preparation)
- Tucker-Davis Pentusa
- Biosemi (not publicly available)

Brain Signals
- Slow Cortical Potentials (SCPs)
- P300 Potentials
- EEG mu/beta Rhythms (ERD/ERS)
- ECoG Oscillations
- SSVEPs

User Applications
- 1D/2D/3D Cursor Movement
- 2D Cursor Movement with Mouse Click
- n-Choice Speller
- 2D Robotic Arm Control
- Data Glove
- P300 Matrix Speller
- Auditory + Visual Stimulus Presentation incl. Real-Time Feedback

Development System
- Borland C++ Builder 6
- open-source CVS client
- no other 3rd party tool required

Main Tools Supplied with BCI2000

Data Conversion
- BCI2000toASCII (conversion into Matlab and ASCII

Signal Analysis
- BCI2000 Analysis Project (code name "Mario") (frequency-domain and time-domain analyses)

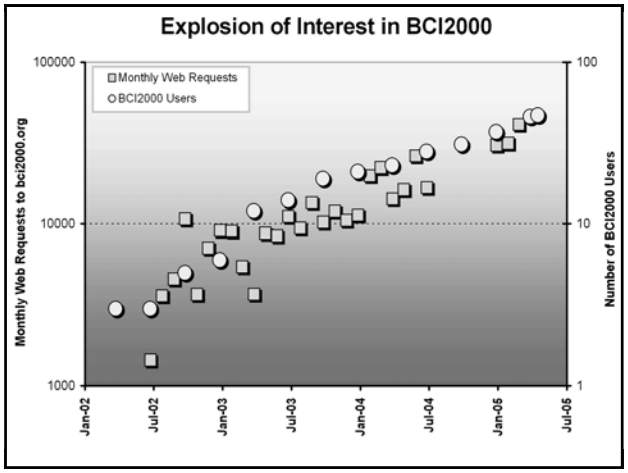
Signal Visualization
- Maxifred (time-domain "polygraph" plotter)

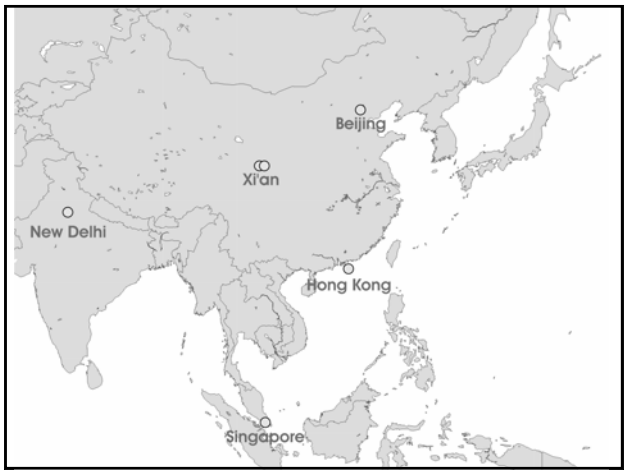
Miscellaneous
- BCI2000FileInfo (shows a data file's system parameters)
- cmdline Tools (use BCI2000 filters from the command line)
- BCI2000 Batcher (start BCI2000 in pre-set configurations)

BCI2000 Scope

BCI2000 is a General-Purpose System

- Ideally suited for research and development
- Clinical applications (but not necessarily clinical research) require ease of use instead of flexibility





Core BCI2000 Contributors

Project Lead
Gerwin Schalk

Chief Scientist
Dennis McFarland

Project Administrator
Dongyan Zhang

Lead Engineer
Jürgen Mellinger

Applications
Melody M. Moore-Jackson
Brendan Allison
Shi Dong Zheng
Janki Vora

Project Sponsors
Jonathan R. Wolpaw
Andrea Kübler
Niels Birbaumer

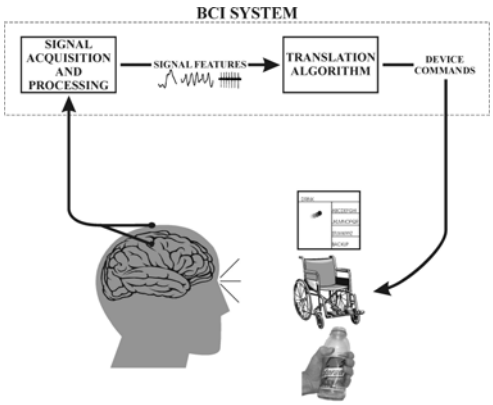
Sites
Wadsworth Ctr, New York St. Dept. of Health, Albany, New York, USA
Inst. for Behav. Neuroscience, University of Tübingen, Germany
Comp. Inf. Sys., Georgia St. Univ., Atlanta, Georgia, USA

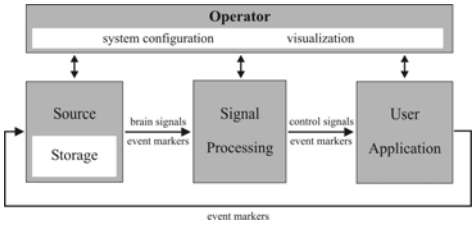
A Functioning BCI is a System – Not One Component

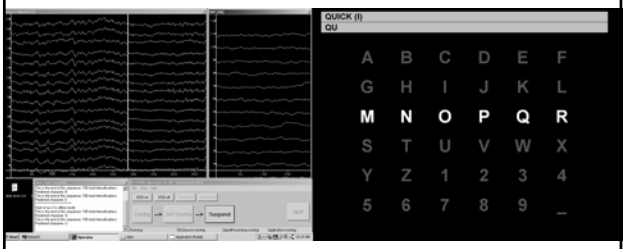
Dennis J. McFarland
Wadsworth Center, New York State Department of Health

1st BCI2000 Workshop
The Desmond Conference Center
Albany, New York, USA
June 13+14, 2005

A Functioning BCI System

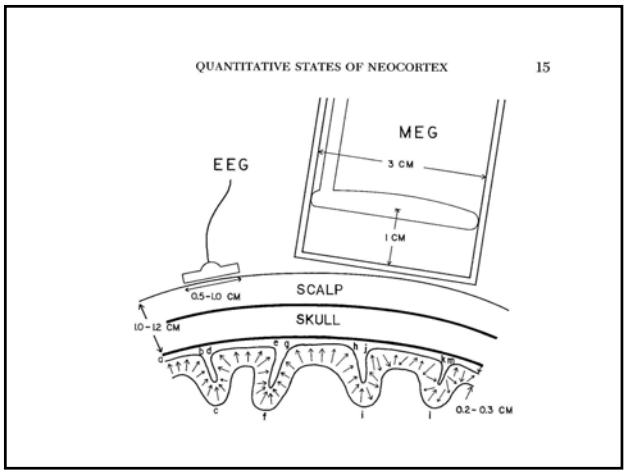


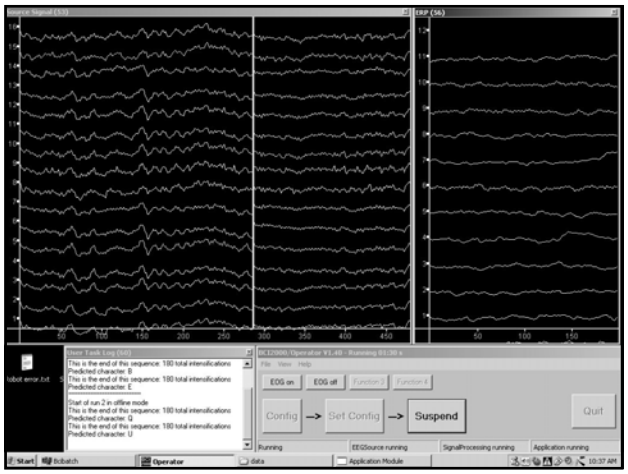


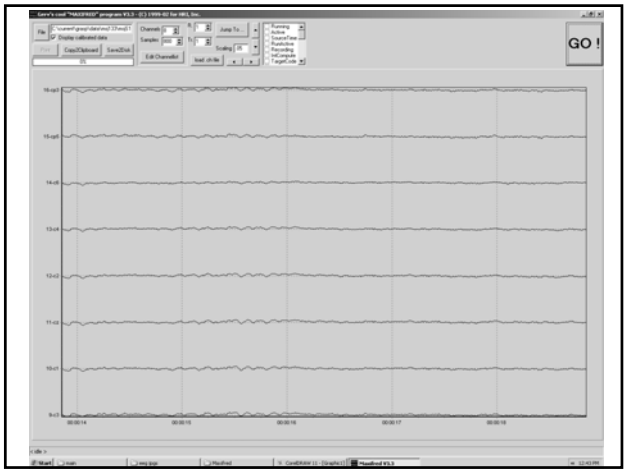


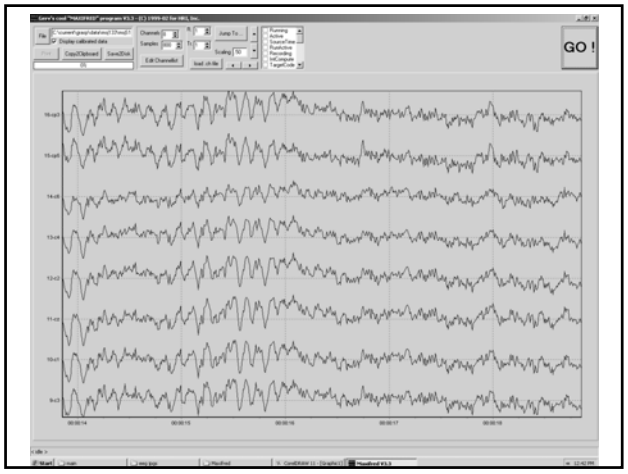
Signal Acquisition

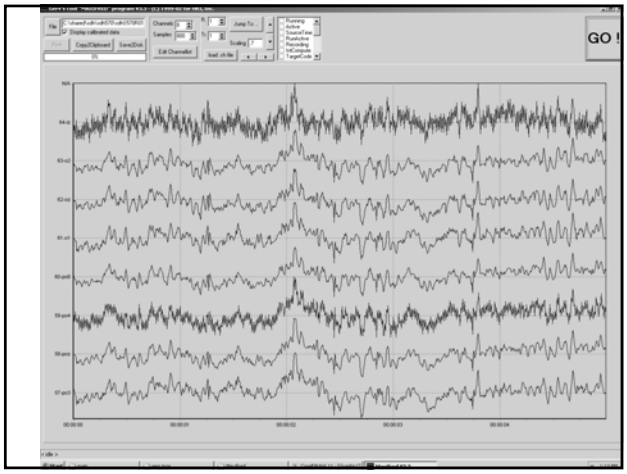
- A Good Recording is Essential
- Proper Parameterization of Recording
- Storage of data and parameters

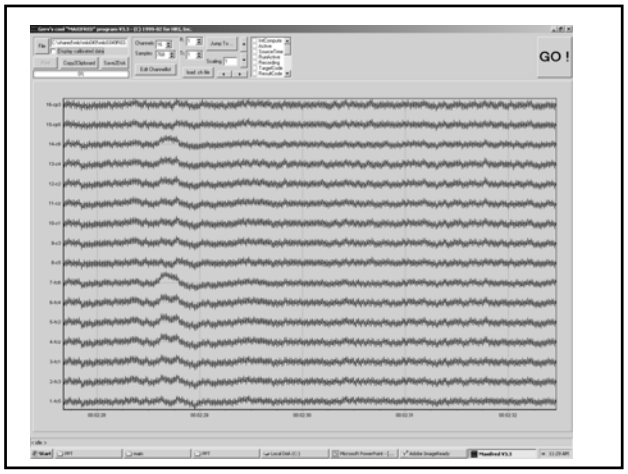

QUANTITATIVE STATES OF NEOCORTEX
15
MEG
EEG
3 CM
1 CM
0.5-1.0 CM
SCALP
SKULL
1.0-1.2 CM
0.2-0.3 CM

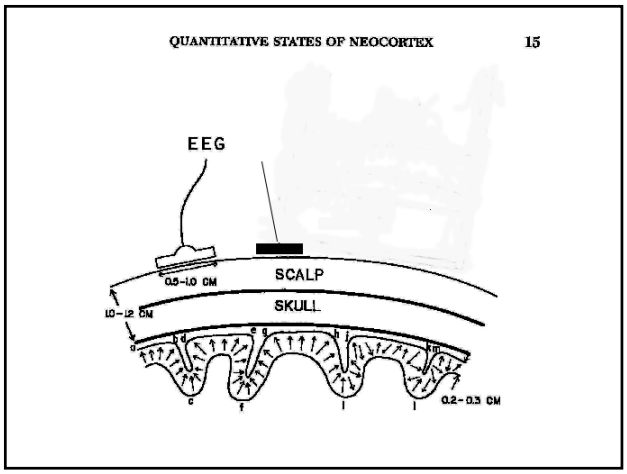

EOG on
EOG off
Config
Set Config
Suspend
Quit
Operator

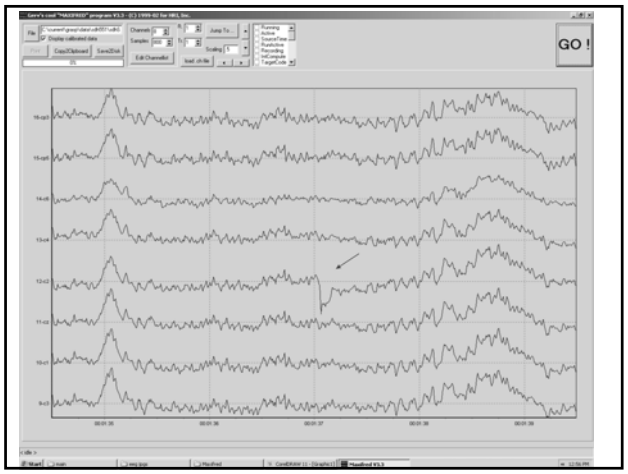

GO !

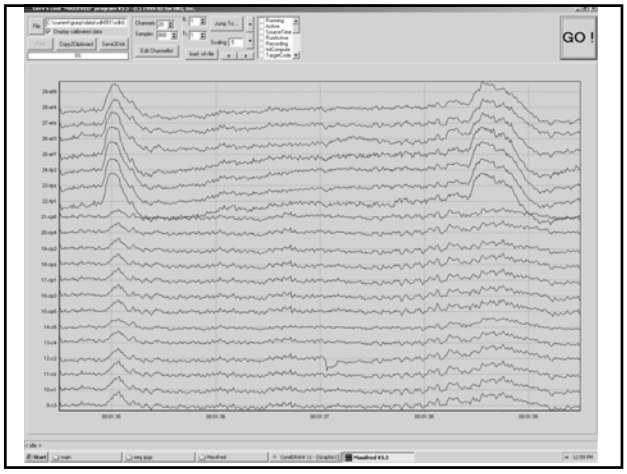

GO !

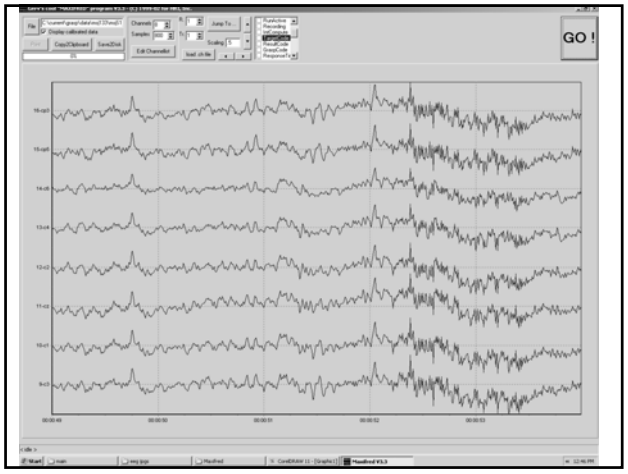

GO !

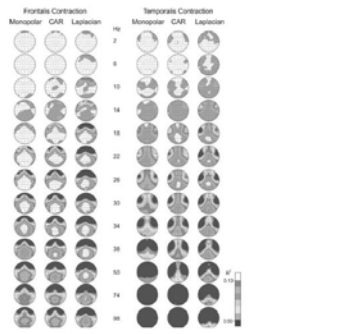

GO !



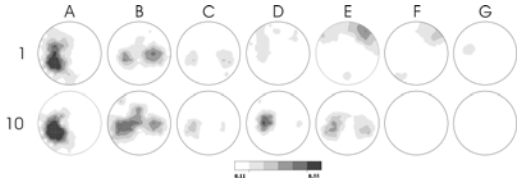




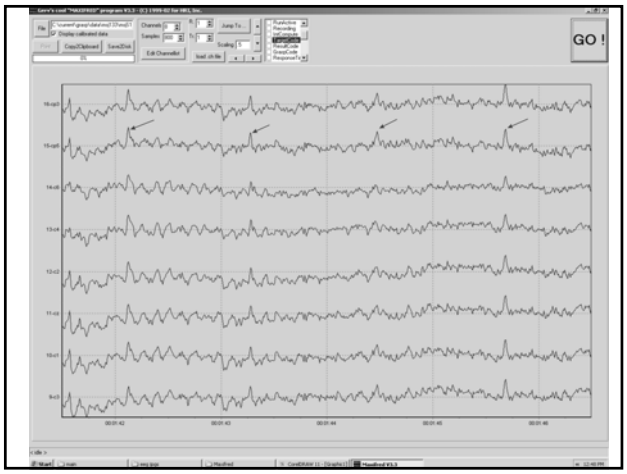


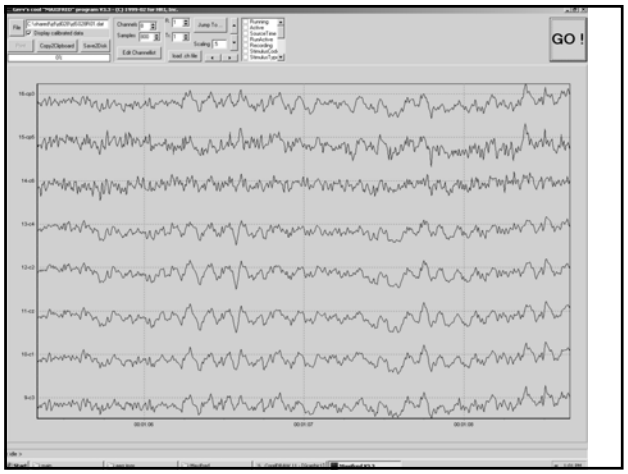


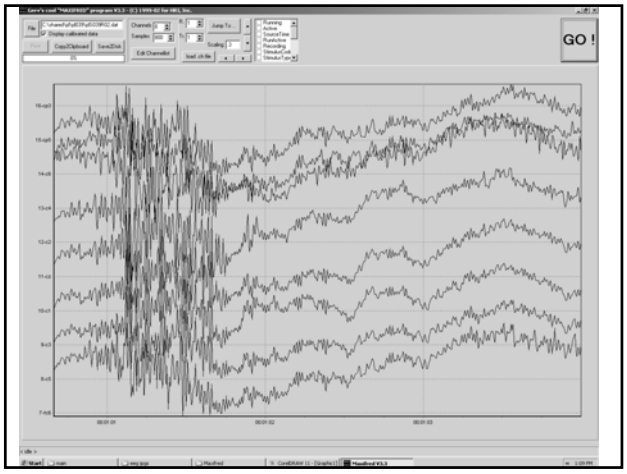
Goncharova, I. I., McFarland, D. J., Vaughan, T. M., & Wolpaw, J. R. (2003) EMG Contamination of EEG: Spectral and Topographical Characteristics. Clinical Neurophysiology, 114, 1580-1593.

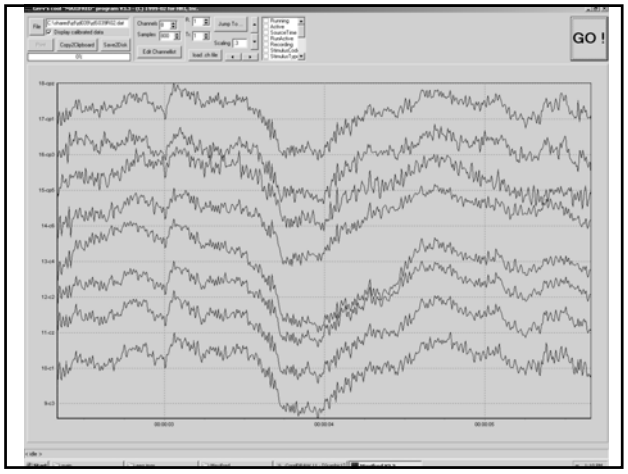


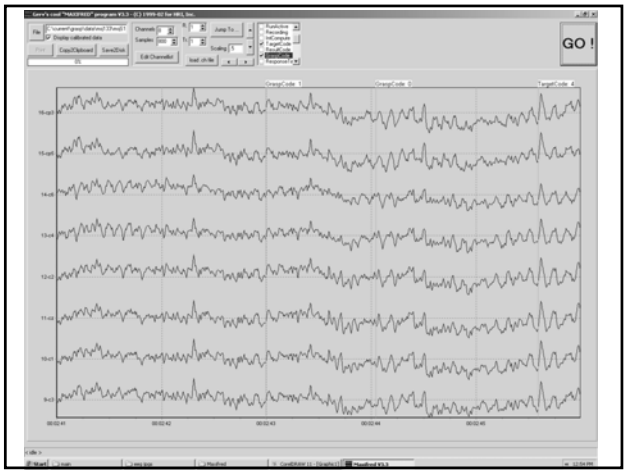
McFarland, D.J., Sarnacki, W.W., Vaughan, T.M., & Wolpaw, J.R. (2005) Brain-computer interface (BCI) operation: Signal and noise during early training sessions. Clinical Neurophysiology, 116, 56-62.



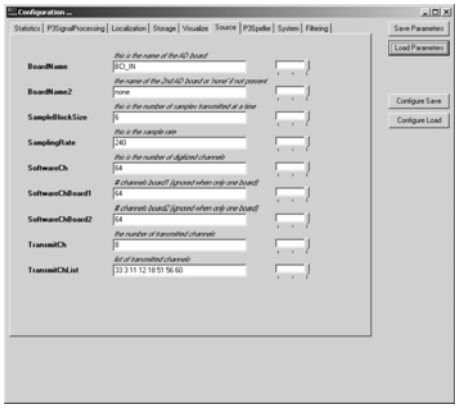




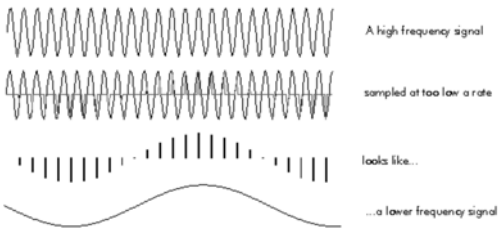


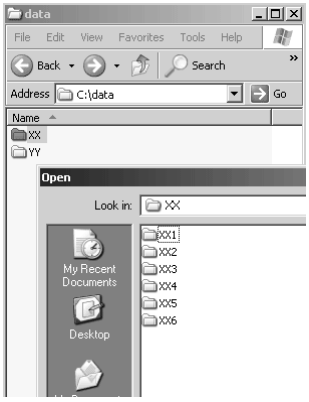


Source Menu for Data Translation 3000 series A/D boards



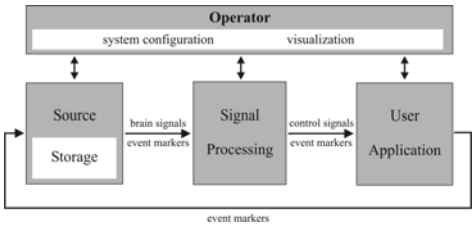
Source Menu for GugerTech System





Signal Processing

- Interchangeable parts
- Proper Parameterization is essential



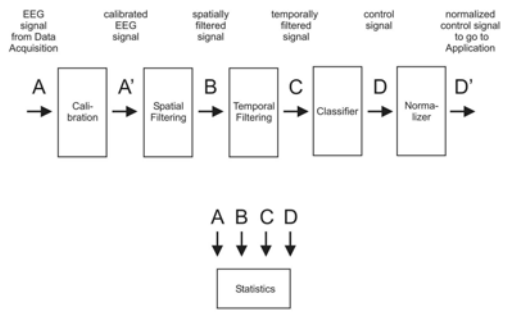
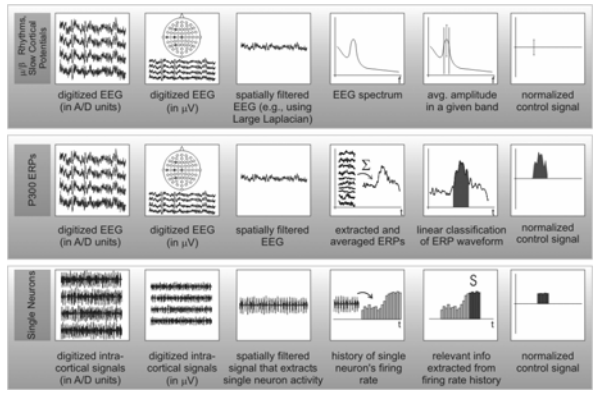
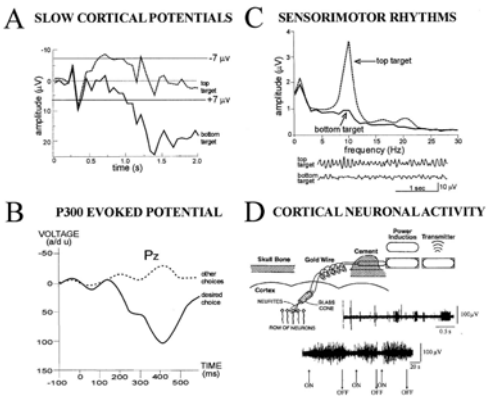
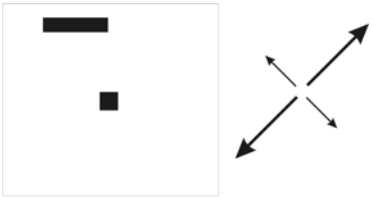


Figure 6.1: Data flow in the signal processing module





$$Y = b(X-a)$$

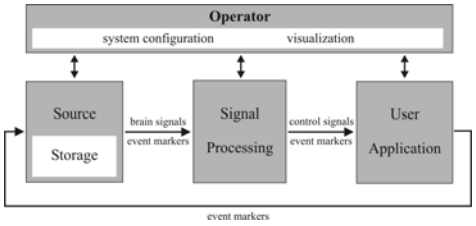


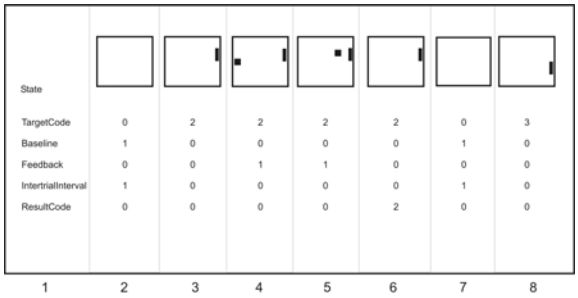
$$Y_1 = b_1(X_1-a_1)$$

$$Y_2 = b_2(X_2-a_2)$$

User Task

- Many Varieties
- Proper Parameterizing is Essential
- Use of standard parameter files and fragments





State	1	2	3	4	5	6	7	8
TargetCode	0	2	2	2	2	0	3	
Baseline	1	0	0	0	0	1	0	
Feedback	0	0	1	1	0	0	0	
IntertrialInterval	1	0	0	0	0	1	0	
ResultCode	0	0	0	0	2	0	0	

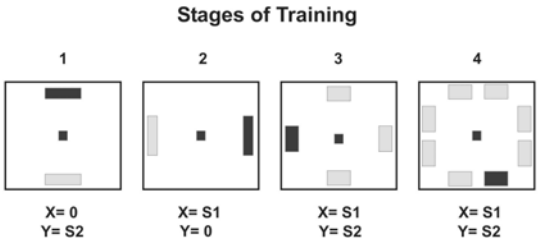
Four target rjb task. 1 is the intertrial interval were the screen is blank and baseline data can be collected (optional). In 2 the target is presented but the cursor is absent (no feedback). In 3 the cursor is present and it's vertical movement is controled by the user's EEG. Horizontal cursor movement is under computer control and is at a constant rate. In 4 the cursor approaches a the target and in 5 the target has been hit and trial-outcome feedback is presented. In 6 the screen is blank (intertrial interval again) and in 7 another target is presented.

	A	B	C	D	E
StimulusCode	4	4	10	10	0
StimulusType	1	1	0	0	0
Flashing	1	0	1	0	0
SelectedTarget	0	0	0	0	16

Comparison of States

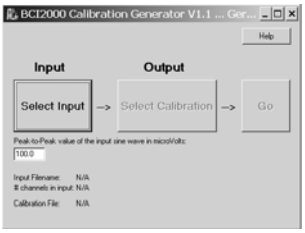
- RJB
- TargetCode
- Baseline
- Feedback
- IntertrialInterval
- ResultCode

- P3
- StimulusCode
- StimulusType
- Flashing
- SelectedTarget



Parameter Files and Fragments

- Standard Files are useful
- File Fragments can add flexibility
  - Calibration Fragments
  - Task Fragments



Calibgen Output

- Filtering floatlist SourceChOffset= 64 -69 99 -35 -240 223 -112 -69 54 101 -153 72 167 53 21 -38 -184 88 3 187 -34 100 -186 -46 76 -33 1 15 11 28 124 65 84 23 -34 178 69 85 173 43 -138 67 -13 -84 51 115 76 320 214 20 -114 -32 179 -137 87 -70 107 80 -56 -154 79 -324 78 -236 -189 0 -500 500 // offset for channels in A/D units
- Filtering floatlist SourceChGain= 64 0.00771 0.00770 0.00772 0.00772 0.00778 0.00769 0.00772 0.00769 0.00769 0.00768 0.00774 0.00769 0.00771 0.00769 0.00772 0.00771 0.00768 0.00769 0.00770 0.00770 0.00768 0.00769 0.00770 0.00772 0.00772 0.00775 0.00771 0.00772 0.00771 0.00770 0.00771 0.00773 0.00772 0.00771 0.00771 0.00770 0.00760 0.00758 0.00770 0.00777 0.00763 0.00773 0.00766 0.00765 0.00768 0.00773 0.00772 0.00773 0.00771 0.00774 0.00769 0.00767 0.00773 0.00770 0.00773 0.00770 0.00770 0.00773 0.00773 0.00769 0.00769 0.00769 0.00770 0.00770 0.033 -500 500 // gain for each channel (A/D units -> muV)
- Source int TransmitCh= 64 4 1 128 // the number of transmitted channels
- Source intlist TransmitChList= 64 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 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 1 1 128 // list of transmitted channels (# of channels MUST equal TransmitCh)

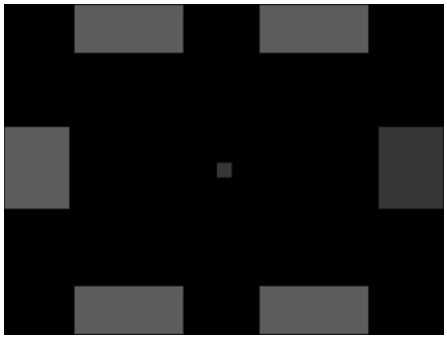
Rjb2.prm

```
UsrTask int FeedbackDuration= 20000 0 0 0 // Max Trial Duration
Targets int IncludeAllTargets= 1 0 0 1 // Test all target positions?
Targets int NumberTargets= 2 0 0 1 // Number of Targets
Targets float StartCursorX= 0.0 0 0 100.0 // Horizontal Start of Cursor
Targets float StartCursorY= 50.0 0 0 100.0 // Vertical Cursor Starting Position
Targets matrix TargetPos= { x%20start y%20start x%20length y%20height x%20code y%20code adapt%20code } { target%201
target%202 ) 90 90 0 51 10 10 50 50 0 0 -1 1 3 3 0 0 0 // Target Position Matrix - Values are 0-100
Statistics matrix BaselineCfg= 2 4 TargetCode 1 Feedback 1 TargetCode 2 Feedback 1 0 0 0 // states to watch for baseline
Statistics matrix BaselineHits= 2 2 1 0.50 2 0.50 0 0 0 // proportion correct for each target
Statistics float XGain= -40 5.0 0.0 100.0 // Normal Filter Left/Right Slope
Statistics float XMean= 10 -5.0 -100.0 100.0 // Normal Filter Left/Right Intercept
Statistics float XPixelsPerSec= 630 70 0 400 // Horizontal Pixel Rate
Statistics int XTrendControl= 0 0 0 2 // X Intercept Adapt 0 none 1 mean 2 mean prop 3 slope
Statistics float YGain= 1000 5.0 -100.0 100.0 // Normal Filter Up / Down Slope
Statistics float YPixelsPerSec= 472 70 0 400 // Desired pixels per second
Statistics int YTrendControl= 1 0 0 2 // Y Intercept Adapt 0 none 1 mean 2 mean prop 3 slope
```

Rjb2.prm

Ud.prm

Six 2D Targets.prm



Sensorimotor Rhythms in the EEG and their Signal Processing/User Task Support in BCI2000

Dean Krusienski
Wadsworth Center, New York State Department of Health

1st BCI2000 Workshop
The Desmond Conference Center
Albany, New York, USA
June 13+14, 2005

Signal Processing / User Task Support of Sensorimotor Rhythms in BCI2000

Dean Krusienski
Wadsworth Center

Outline

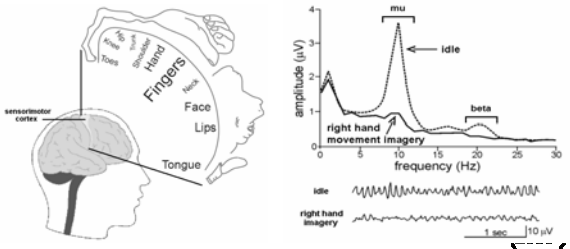
- Sensorimotor Rhythms
- BCI2000 Signal Processing
- BCI2000 Sensorimotor Rhythm User Applications

Spectral Content of Scalp recorded EEG

- **0-4Hz (Delta):** young children, sleep, underlying lesions
- **4-8Hz (Theta):** adolescence, drowsiness, light sleep
- **8-14Hz (Alpha):** relaxed and alert, visual
- **8-12Hz (Mu):** sensorimotor stimulation, movement, imagery
- **12-30Hz (Beta):** sensorimotor, active concentration
- **30-80Hz (Gamma):** higher mental activity

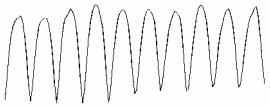
Sensorimotor Rhythms (Mu & Beta)

- Localized, narrowband amplitude modulations corresponding to movement, stimulation, or mental imagery



The Mu Rhythm

- 8-12Hz Event Related Desynchronization
- Evident in most healthy adults
- Prominent over the primary sensorimotor cortex
- **Blocked by:**
  - **contralateral movement**
  - **tactile stimulation**
  - **movement imagery**
- Tends to be asynchronous across hemispheres
- Tends to often exhibit an arch shaped pattern:

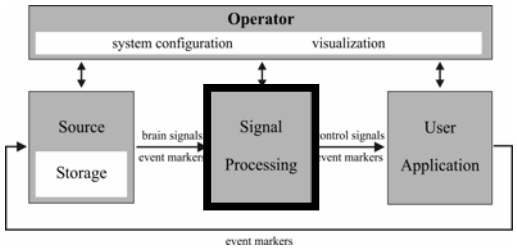


- Can be actively modulated and trained for control
- Exact relation to 12-30Hz beta rhythm not clear

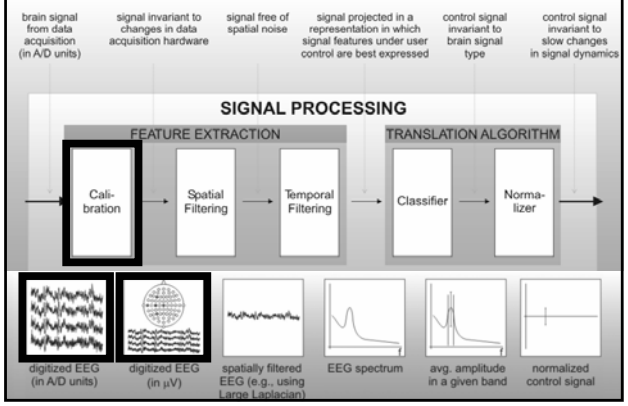
Outline

- Sensorimotor Rhythms
- **BCI2000 Signal Processing**
- BCI2000 Sensorimotor Rhythm User Applications

BCI2000 Modules



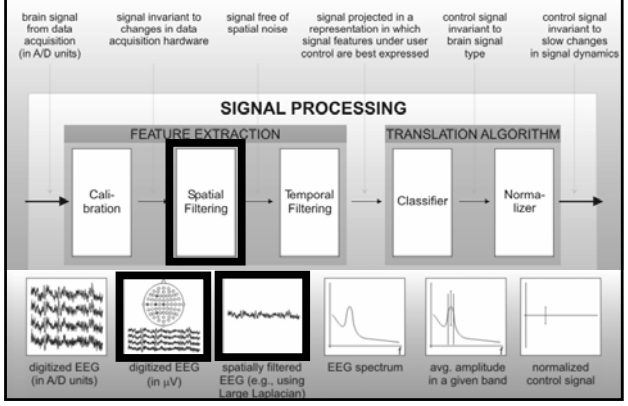
BCI2000 Signal Processing Module



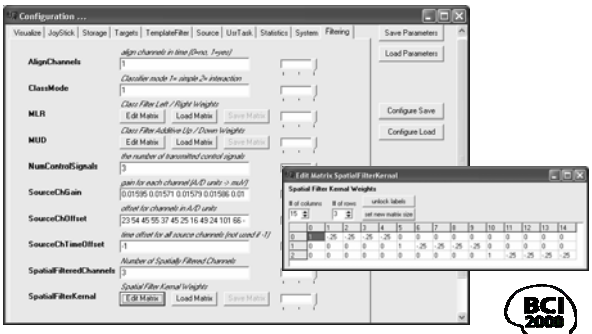
Calibration

- Converts from analog-digital converter units to microvolts (μV)
- Adjusts for any amplifier gain or offset discrepancies between input channels
- Calibration gains and offsets specified by investigator

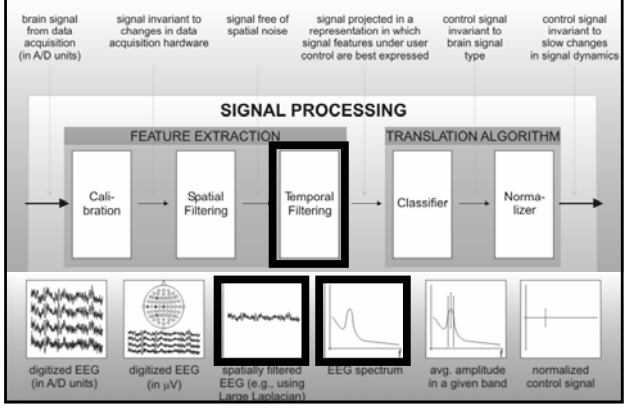
BCI2000 Signal Processing



Spatial Filtering



BCI2000 Signal Processing



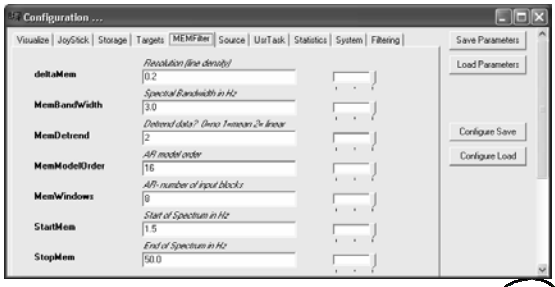
Temporal Filtering

- Basics
  - Incoming multi-channel digitized signal is processed in real-time using segments of data consisting of current and past samples
  - Segment lengths and overlap tend to be application dependent and can easily be specified in the BCI2000 user interface
  - The specified features are generated continuously for each incoming data segment at a rate of: sampling rate/overlap
  - Spectral resolution, locations, and other spectral parameters tend to be subject dependent and can easily be specified in the BCI2000 user interface

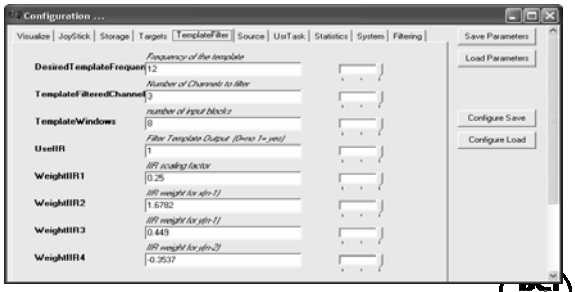
Temporal Filtering

- Feature Extraction options for sensorimotor rhythms:
  - **Autoregressive (AR) Spectral Modeling:** general identification and tracking of one or more frequency components across entire frequency spectrum
  - **Template Filtering:** tracking waveforms having a particular temporal morphology such as the mu rhythm

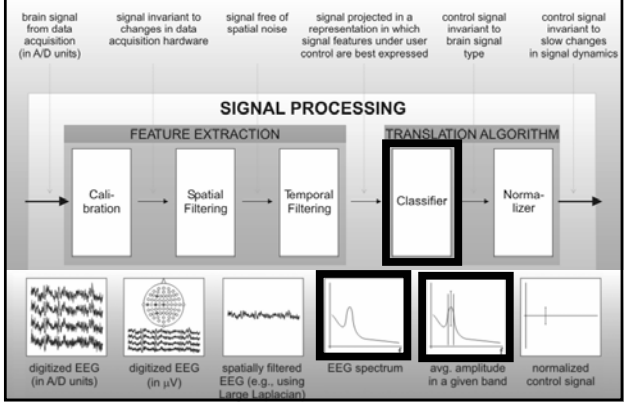
BCI2000 AR Filter Parameters



BCI2000 Template Filter Parameters



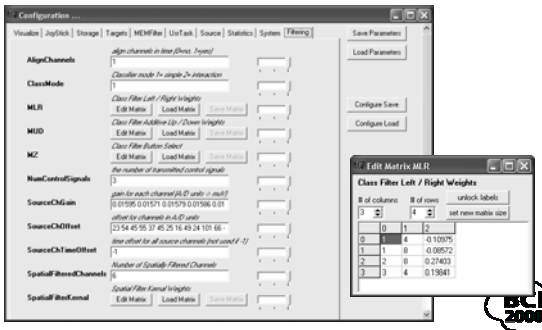
BCI2000 Signal Processing



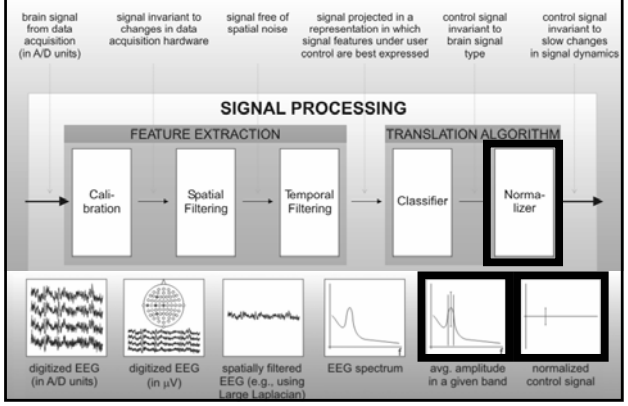
Classifier

- Linear combination of temporal filter features selected for control
  - Examples:
    - Average signal amplitude at FCz, Cz, and CPz from 6-20Hz
    - Difference of template filter amplitudes between C3 and C4
    - Sum of 12Hz and 24Hz bands at C3
- Feature weight adaptation available

BCI2000 Classifier Features



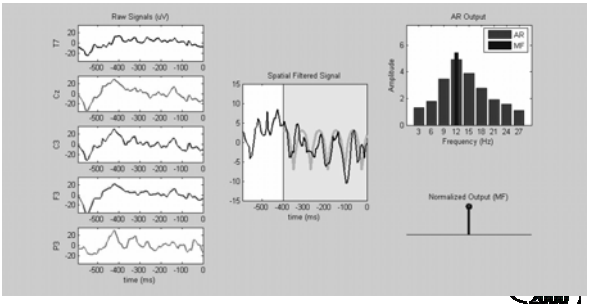
BCI2000 Signal Processing



Normalizer

- Normalizes feature(s) based on feature statistics to account for changing signal dynamics
  - Feature mean estimated and removed
  - Feature standard deviation estimated and used to scale the dynamic range

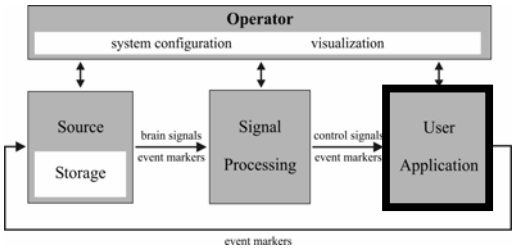
Signal Processing Illustration



Outline

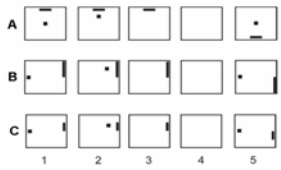
- Sensorimotor Rhythms
- BCI2000 Signal Processing
- **BCI2000 Sensorimotor Rhythm User Applications**

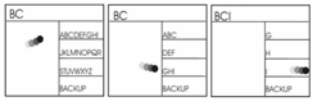
BCI2000 Modules



Sensorimotor Rhythm Cursor Control Tasks in BCI2000

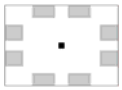
- One-dimensional cursor control
  - Linear combination of specified control features
  - Multiple vertical/horizontal targets
  - Multiple right justified targets & speller

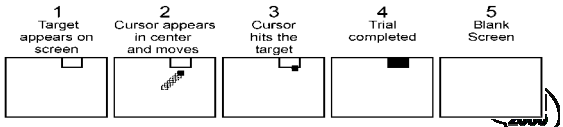




Sensorimotor Rhythm Cursor Control Tasks in BCI2000

- Two-dimensional cursor control
  - Linear combination of specified control features for each dimension
  - Single/simultaneous target modes
  - Button click in progress





BCI2000 2D Cursor Control

BCI2000 2D Robotic Arm

Sensorimotor Rhythm Cursor Control Tasks in BCI2000

- Features
  - Adjustable cursor/target sizes and locations
  - Adjustable cursor rate
  - Button click mode
  - Adjustable user feedback options
  - Adjustable trial lengths and timeouts
  - All parameters, state codes, and results stored in BCI2000 data file

The P300 Response in the EEG and its Signal Processing/User Task Support in BCI2000

Eric Sellers
Wadsworth Center, New York State Department of Health

1st BCI2000 Workshop
The Desmond Conference Center
Albany, New York, USA
June 13+14, 2005

BCI2000 and the P300 Event-Related Potential

Eric W. Sellers
Wadsworth Center, NYSDOH
June 13, 2005

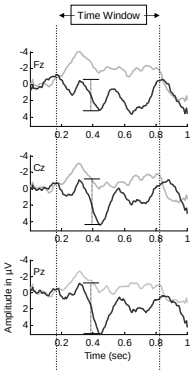
Outline

- The P300
- The Oddball Paradigm
- Modified Oddball Paradigm – P300 Speller
- BCI2000 P300 Signal Processing
- P300 Speller Module
- P300 Sequential Module (P3AV)

The P300

- A positive deflection in the scalp recorded EEG approximately 300 ms after stimulus presentation
- The P300 component is defined in terms of a typical amplitude, latency, and scalp distribution
  - Amplitude – 3 to 20 µV
  - Latency – 200 to 700 ms
  - Scalp distribution – Fz < Cz < Pz
- Realized by averaging
- Process NOT stimulus dependent

The P300



Outline

- The P300
- **The Oddball Paradigm**
- Modified Oddball Paradigm – P300 Speller
- BCI2000 P300 Signal Processing
- P300 Speller Module
- P300 Sequential Module (P3AV)

The Oddball Paradigm

Oddball paradigm elicits a P300

- Typical requirements
  - Random sequence of events
  - Classification rule to separate events into categories
  - Task that requires using the rule
  - One category is presented infrequently

Standard Oddball Experiment

Realized with the P3AV Sequential Presentation Module

Outline

- The P300
- The Oddball Paradigm
- **Modified Oddball Paradigm – P300 Speller**
- BCI2000 P300 Signal Processing
- P300 Speller Module
- P300 Sequential Module (P3AV)

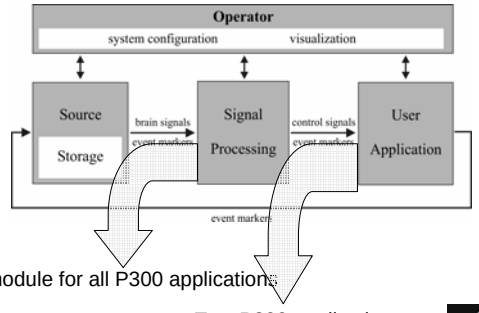
Modified Oddball Paradigm – P300 Speller

- Variable matrix of characters
- Each row and each column intensify briefly
- Order of intensifications random
- Subject focuses attention on one cell

Outline

- The P300
- The Oddball Paradigm
- Modified Oddball Paradigm – P300 Speller
- **BCI2000 P300 Signal Processing**
- P300 Speller Module
- P300 Sequential Module (P3AV)

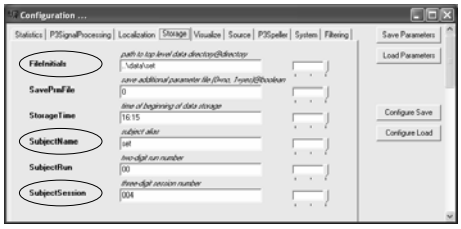
BCI2000 Signal Processing



One module for all P300 applications

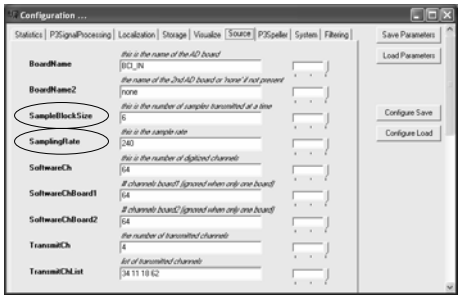
Two P300 applications

Generic BCI2000 Parameters



BCI2000 Workshop 2005

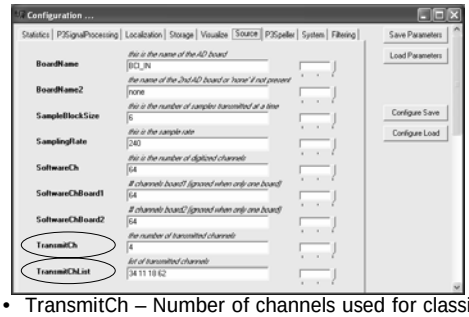
Generic BCI2000 Parameters



- SamplingRate – Number of digitized samples/sec
- SampleBlockSize – Number of samples sent from source to signal processing at a time
- At 240 Hz: 1 sample = 4.1667 ms x 6 = 25 ms

BCI2000 Workshop 2005

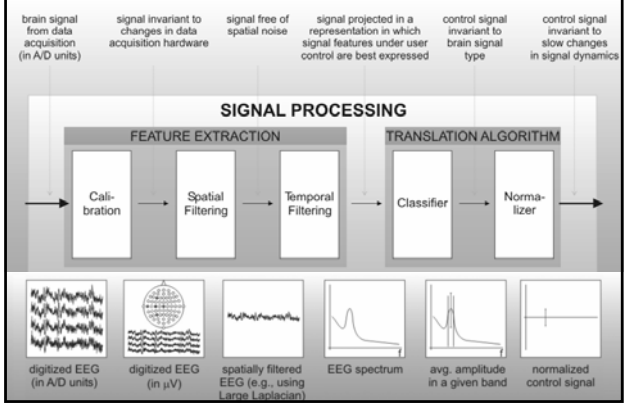
Generic BCI2000 Parameters



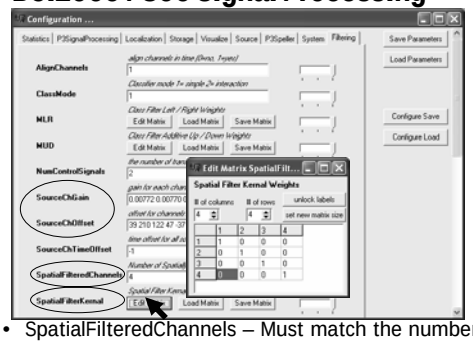
- TransmitCh – Number of channels used for classification and/or filtering
- TransmitChList – Exact channel numbers used for classification and/or filtering

BCI2000 Workshop 2005

BCI2000 P300 Signal Processing



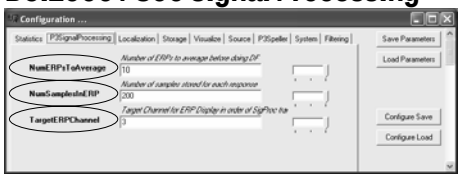
BCI2000 P300 Signal Processing



- SpatialFilteredChannels – Must match the number of rows in the SpatialFilterKernal
- SpatialFilterKernal – Spatial filter to be used before classification

BCI2000 Workshop 2005

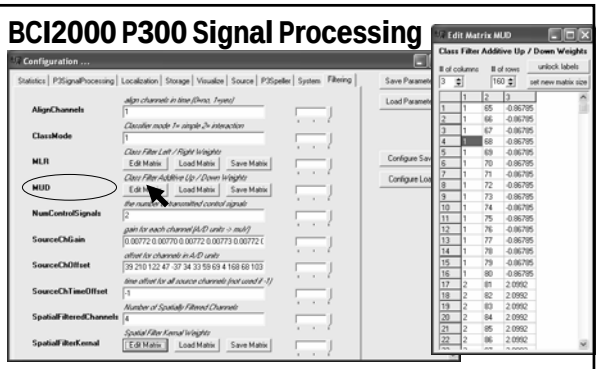
BCI2000 P300 Signal Processing



- NumERPsToAverage – Number of stimuli/sequences to collect before averaging
- NumSamplesInERP – Samples for each time epoch
- TargetERPChannel – The channel visualized in VisualizeP3TemporalFilter

BCI2000 Workshop 2005

BCI2000 P300 Signal Processing



- MUD –Channels x time points Matrix of weights for online classification

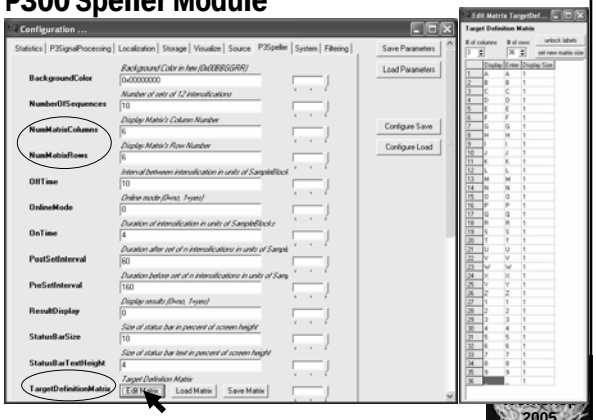
BCI2000 Workshop 2005

Outline

- The P300
- The Oddball Paradigm
- Modified Oddball Paradigm – P300 Speller
- BCI2000 P300 Signal Processing
- **P300 Speller Module**
- P300 Sequential Module (P3AV)

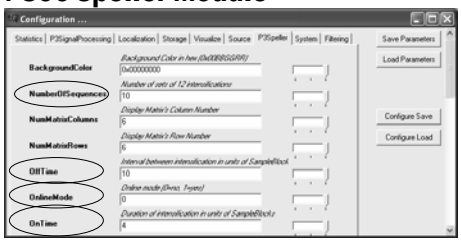
BCI2000 Workshop 2005

P300 Speller Module



2005

P300 Speller Module



- NumberOfSequences – Number of times each stimulus flashes (equal to or > NumERPsToAverage)
- OffTime – Amount of time between stimulus presentations (sample blocks)
- OnTime – Duration of stimulus presentation (sample blocks)
- OnlineMode – 0 Copy Spelling; 1 Free Spelling

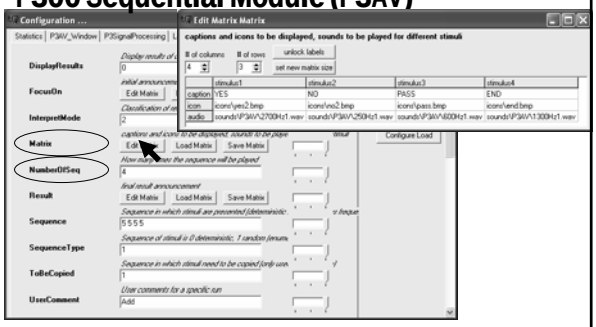
BCI2000 Workshop 2005

Outline

- The P300
- The Oddball Paradigm
- Modified Oddball Paradigm – P300 Speller
- BCI2000 P300 Signal Processing
- P300 Speller Module
- **P300 Sequential Module (P3AV)**

BCI2000 Workshop 2005

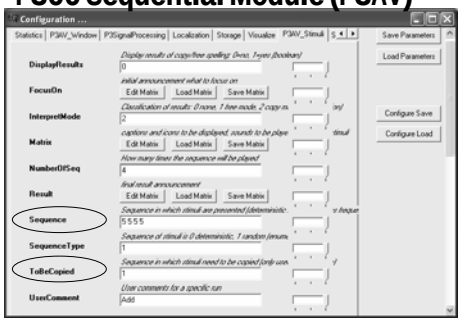
P300 Sequential Module (P3AV)



- Matrix – Define the list of possible stimuli
- NumberOfSeq – Num of times the stimulus sequence is repeated (must => NumERPsToAverage)

BCI2000 Workshop 2005

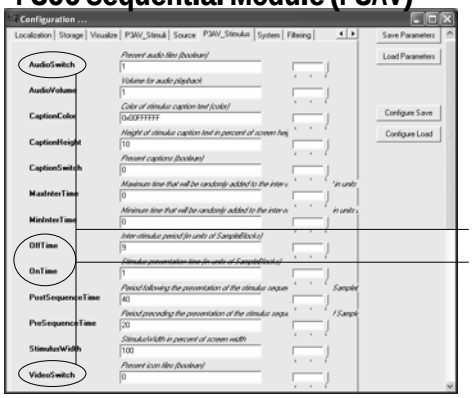
P300 Sequential Module (P3AV)



- Sequence – Determines number of times to present each stimulus
- ToBeCopied – Defines which stimulus is coded as target

BCI2000 Workshop 2005

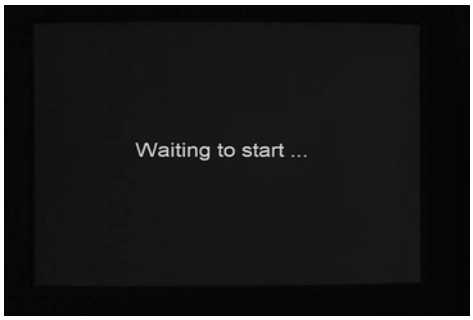
P300 Sequential Module (P3AV)



AV On/Off

Function of sample rate and sample block size

BCI2000 Workshop 2005



BCI2000 Workshop 2005

Sensorimotor Rhythms in ECoG

Gerwin Schalk
Wadsworth Center, New York State Department of Health

1st BCI2000 Workshop
The Desmond Conference Center
Albany, New York, USA
June 13+14, 2005

Sensorimotor Rhythms in ECoG

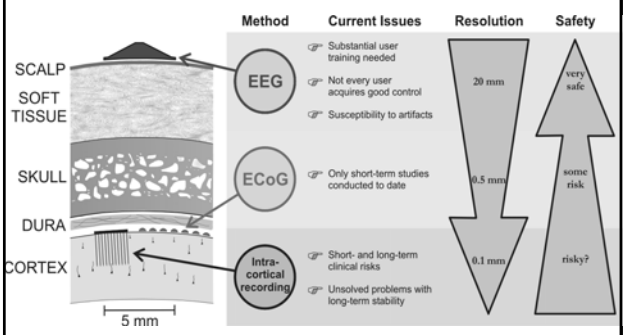
1st BCI2000 Workshop
The Desmond Conference Center
Albany, New York, USA
June 13+14, 2005

Gerwin Schalk
Research Scientist, Chief Software Engineer; Brain-Computer Interface R&D Program
Wadsworth Center, New York State Dept. of Health, Albany, New York, USA

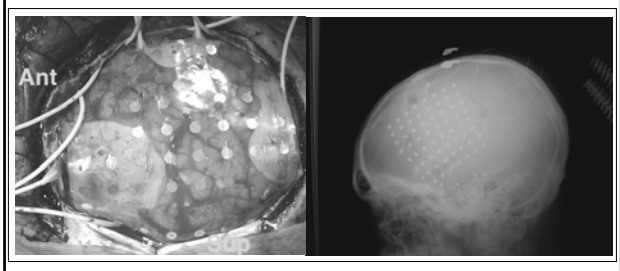
Outline

- **Electrocorticography (ECoG)**
- Brain-Computer Interfaces using ECoG
- SIGFRIED: Rapid Detection and Use of ECoG
- Cognitive Paradigms with ECoG and BCI2000

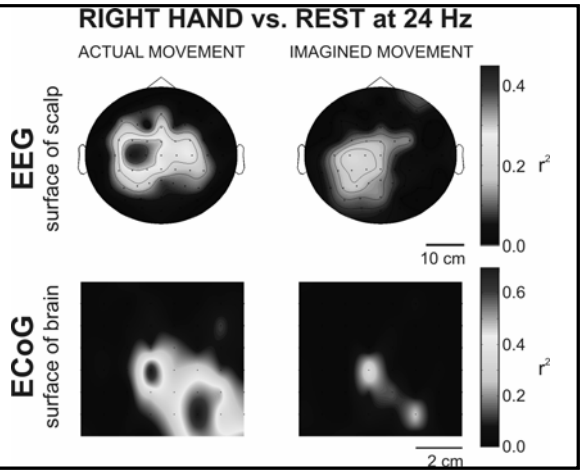
BCI Recording Alternatives



Electrocorticography (ECoG)



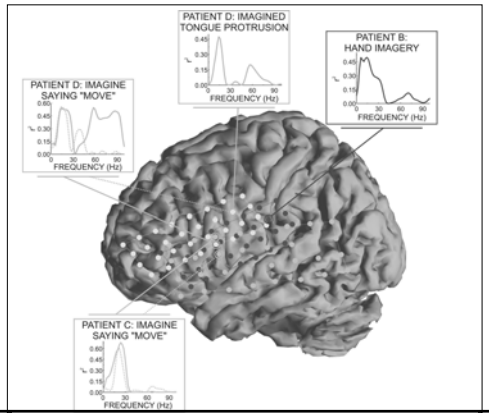
Signals recorded from patients who are temporarily implanted with electrode grids on the cortical surface prior to epilepsy surgery.



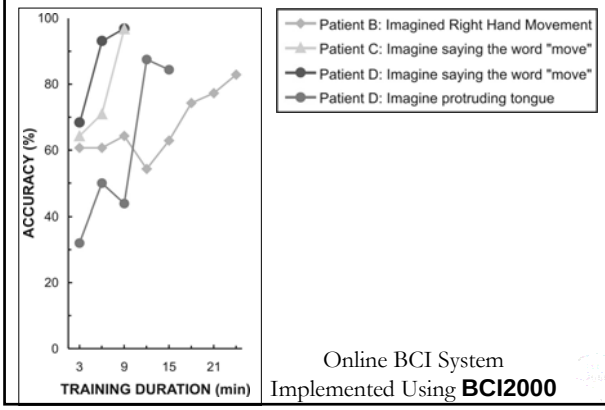
Outline

- Electrocorticography (ECoG)
- **Brain-Computer Interfaces using ECoG**
- SIGFRIED: Rapid Detection and Use of ECoG
- Cognitive Paradigms with ECoG and BCI2000

1D Movement Control: ECoG

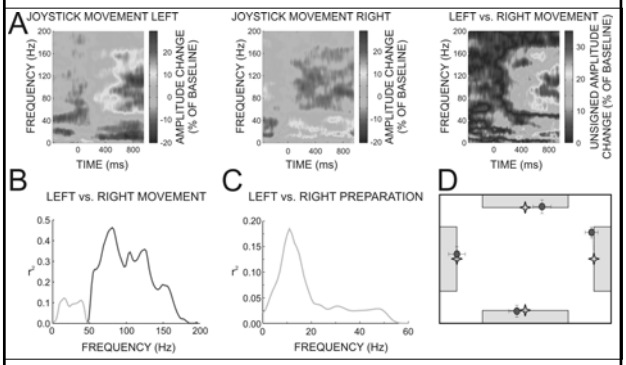


1D Movement Control: ECoG



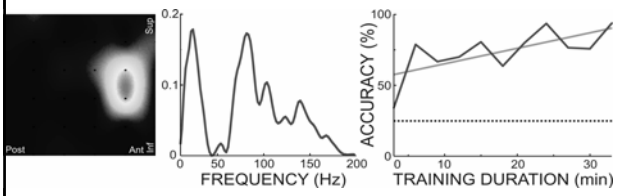
Online BCI System Implemented Using **BCI2000**

2D Movement Prediction: ECoG

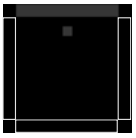


Leuthardt, Schalk, Wolpaw, Ojemann, Moran (JNE, 2004)

2D Control Using Actual Movements



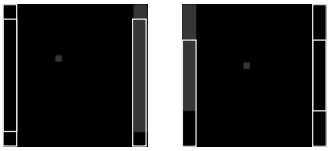
Avg. Accuracy	74.0 %
Total # Trials	362
Training Duration	33 min
Horiz. Control (r^2)*	0.19
Vert. Control (r^2)*	0.23



* r^2 assessed at optimal location/frequency

The user used actual shoulder and middle finger movements to move a cursor towards one of four targets.

2D Control Using Imagined Movements



Avg. Accuracy	64.5 %	81.7 %
Total # Trials	146	84
Training Duration	24 min	9 min
Horiz. Control (r^2)*	0.21	0.65
Vert. Control (r^2)*	0.06	0.11

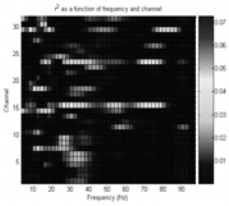
* r^2 assessed at optimal location/frequency

The user used shoulder and finger imagery to move a cursor towards one of four targets.

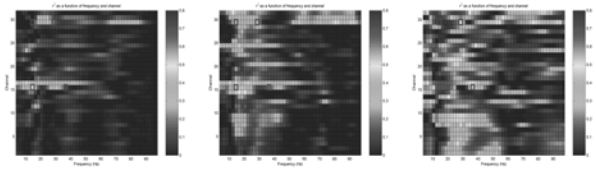
Outline

- Electrocorticography (ECoG)
- Brain-Computer Interfaces using ECoG
- **SIGFRIED: Rapid Detection and Use of ECoG**
- Cognitive Paradigms with ECoG and BCI2000

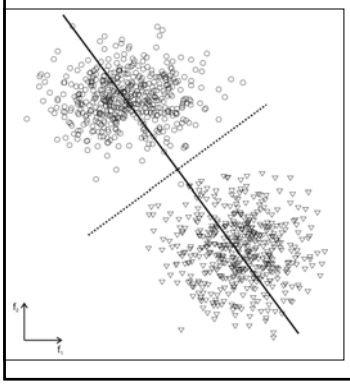
Imaging saying "move" vs. rest



Imagining saying "move" to control a cursor in 1D



Current Signal Translation Methods

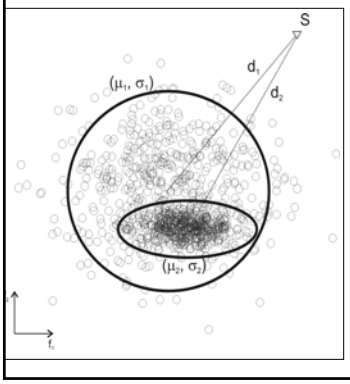


All current BCI approaches:

- Determination of classification / regression function
- Use a distance metric to produce control signals
- Assumes stationary signal AND background
- Requires *a-priori* knowledge of signal

SIGFRIED

SIGnal Modelling FoR Identification and Event Detection

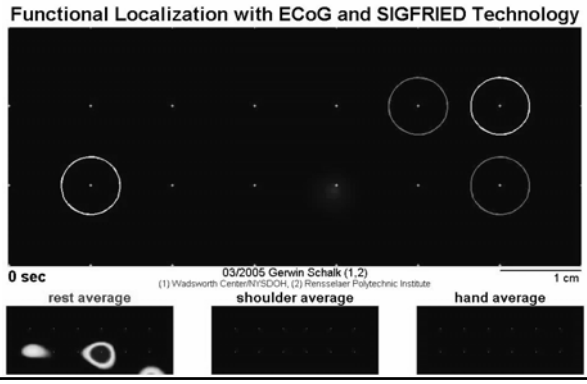


- Modelling *background* rather than describing signal
- Use distance metric (e.g., Mahalanobis distance, p-values) for control
- Only assumes stationary background
- Does NOT require *a-priori* knowledge of signal

Background modelling is currently not used in biosignal processing!!

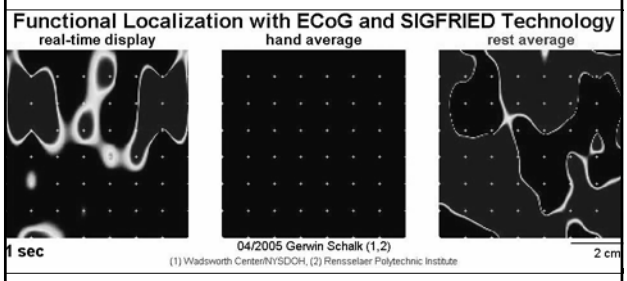
SIGFRIED

Signal Modelling FoR Identification and Event Detection

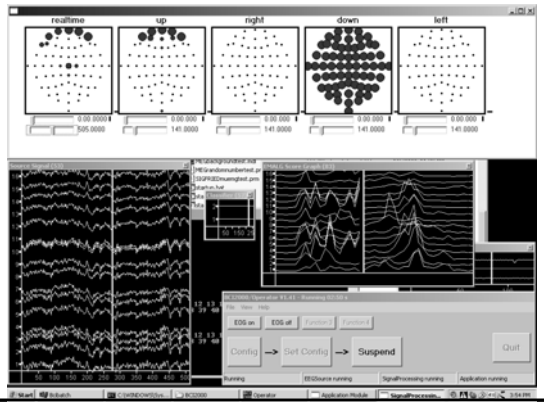


SIGFRIED

Signal Modelling FoR Identification and Event Detection



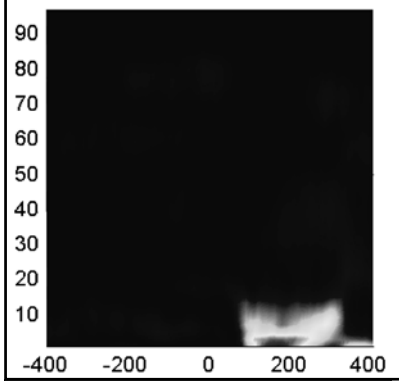
SIGFRIED in BCI2000



Outline

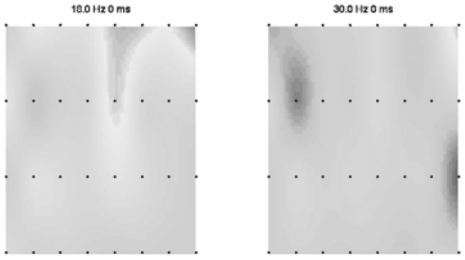
- Electrocorticography (ECoG)
- Brain-Computer Interfaces using ECoG
- SIGFRIED: Rapid Detection and Use of ECoG
- **Cognitive Paradigms with ECoG and BCI2000**

Cognitive Paradigms using BCI2000 and ECoG: Faces and Houses



- Intensity-matched images of faces and houses randomly presented using P3AV
- Faces produce a response that is markedly different to houses at a location over fusiform gyrus

Cognitive Paradigms using BCI2000 and ECoG: Verb Generation



- Subject is presented with nouns
- The subject's task is to say a verb that corresponds with the noun
- There are detectable brain signals in response to verb generation

Wadsworth Center
Gerwin Schalk
Peter Brunner
Jonathan R. Wolpaw

University of Washington
Kai J. Miller
Rajesh P.N. Rao
Jeff G. Ojemann

Washington University in St. Louis
Nick Anderson
Daniel W. Moran
Eric C. Leuthardt

Albany Medical Center
Sam Koszer
Anthony Cacace

University of Wisconsin-Madison
Elisabeth Felton
Adam Wilson
Charles Garell
Justin Williams

Programming and Extending BCI2000

Juergen Mellinger
University of Tuebingen

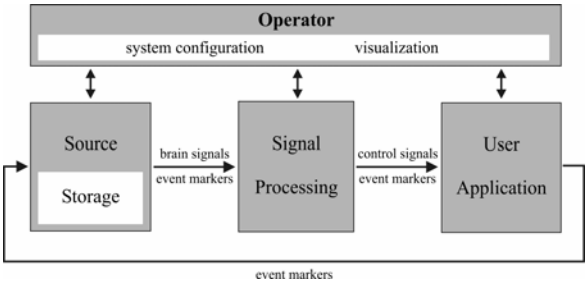
1st BCI2000 Workshop
The Desmond Conference Center
Albany, New York, USA
June 13+14, 2005

Programming and Extending BCI2000

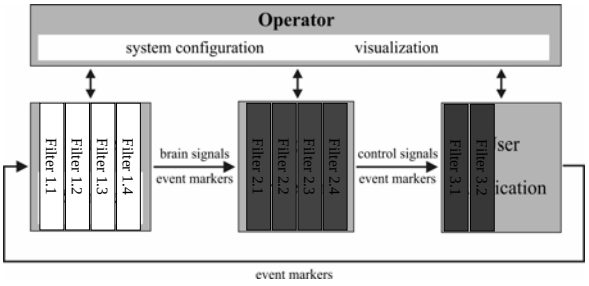
Talk at the BCI2000 Workshop
Albany, NY, USA 2005

juergen.mellinger@uni-tuebingen.de

BCI2000 System Model



BCI2000 System Model



Overview

- Introduction to the **`GenericFilter`** class interface
- Tutorial example: Implementing a low pass filter
- Recapitulation of framework facilities

References:
BCI2000 software design document, c 6.6
`http://www.bciresearch.org/implementation.pdf`

`GenericFilter` Class Interface

- **`GenericFilter`** defines an interface
  - implemented by your code
  - called from the BCI2000 framework
- Interface elements are event handlers
  - Initialization
  - Processing
  - Helpers
- **C++** lingo: virtual inheritance

`GenericFilter` Class Interface

```
class GenericFilter
{
 public:
  virtual void Preflight( const SignalProperties& Input,
                          SignalProperties& Output ) const = 0;
  // Initialize() performs initialization required when
  // parameter settings have changed.
  virtual void Initialize() = 0;
  // StartRun() performs initialization required when
  // starting a new run.
  virtual void StartRun()  { Initialize(); }
  // StopRun() performs actions required when a run ends.
  virtual void StopRun()   { Resting(); }

  virtual void Process( const GenericSignal* Input,
                        GenericSignal* Output ) = 0;
  virtual void Resting() {}
  virtual void Halt() {}
};
```

`GenericFilter::Process()`

- A filter‘s main function
- Called once for each block of data
- Single chain of filters:
  - input received from preceding filter
  - output forwarded to subsequent filter
- Filters not modifying their signal must do `*output=*input;`

`GenericFilter::Initialize()`

- Called when parameter settings have changed
- Adapt your filter’s member variables to reflect new parameter settings
- No need to check for configuration errors

`GenericFilter::Preflight()`

- Called when parameter settings need verification
- Prevent configuration errors
  - report an error if settings will lead to a crash
  - report a warning if parameters appear inconsistent

`GenericFilter::StartRun()`

- Called when a new run starts – user clicks “Start” or “Resume”
- Typical **`StartRun()`** examples
  - writing a run number into a log file
  - opening a new output file
- Any initialization should *either* fit into **`Initialize()`** *or* into **`StartRun()`**

`GenericFilter::StopRun()`

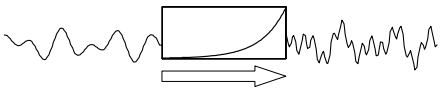
- Called when a run ends
- Only place where parameter values may be modified
- Modified parameter values will be propagated to operator and other modules automatically

Tutorial Example: Low Pass Filter

RC-like low pass filter with time constant T, input and output samples obeying

$$S_{out,0} = \left(1 - e^{-1/T}\right) S_{in,0}$$

$$S_{out,t} = e^{-1/T} S_{out,t-1} + \left(1 - e^{-1/T}\right) S_{in,t}$$



Tutorial Example: Low Pass Filter

- Derive a new class **`LPFilter`** from **`GenericFilter`**
- Implement its **`Process()`**, **`Initialize()`**, **`Preflight()`**, and constructor functions
- Add it to the filter chain

Declaring a **`LPFilter`** Class

Contents of **`LPFilter.h`**:

```
#ifndef LPFilterH
#define LPFilterH

#include "UGenericFilter.h"

class LPFilter : public GenericFilter
{
 public:
   LPFilter();
   ~LPFilter();

   void Preflight( const SignalProperties&,
                   SignalProperties& ) const;
   void Initialize();
   void Process( const GenericSignal*, GenericSignal* );
};
#endif // LPFilterH
```

`LPFilter::Process()`

```
void LPFilter::Process( const GenericSignal* input,
                        GenericSignal* output )
{
  // The first line of the filter prescription:
  mPreviousOutput.resize( input->Channels(), 0 );
  // This implements the second line for all channels:
  for( size_t channel = 0; channel < input->Channels(); ++channel )
  {
    for( size_t sample = 0; sample < input->Elements(); ++sample )
    {
      mPreviousOutput[ channel ] *= mDecayFactor;
      mPreviousOutput[ channel ] +=
              ( *input )( channel, sample ) * ( 1 - mDecayFactor );
      ( *output )( channel, sample ) = mPreviousOutput[ channel ];
    }
  }
}
```

Data Members

```
#include <vector>
...
class LPFilter : public GenericFilter
{
 public:
   ...
 private:
    float               mDecayFactor;       = e^(-1/T)
    std::vector<float>  mPreviousOutput;    = S_(out,t-1)
};
```

`LPFilter::Initialize()`

```
void LPFilter::Initialize()
{
  float timeConstant = Parameter( "LPTimeConstant" );
  mDecayFactor = ::exp( -1.0 / timeConstant );
  mPreviousOutput.clear();
}
```

`LPFilter::Initialize()`

User friendliness of the time constant parameter

```
#include "MeasurementUnits.h"
...
void LPFilter::Initialize()
{
  // Get the time constant in units of a sample block's
  // duration:
  float timeConstant = MeasurementUnits::ReadAsTime(
                         Parameter( "LPTimeConstant" ) );
  // Convert it into units of a sample's duration:
  timeConstant *= Parameter( "SampleBlockSize" );
  ...
}
```

LPFilter::Preflight()

Identifying possible error conditions:

- `timeConstant == 0` – division by zero
- `timeConstant < 0` – numeric overflow
- are `input` and `output` valid pointers?
- does the output signal hold at least as much data as the input signal?

LPFilter::Preflight()

```
void
LPFilter::Preflight( const SignalProperties& inputProperties,
                     SignalProperties& outputProperties ) const
{
  float LPTimeConstant = MeasurementUnits::ReadAsTime(
                              Parameter( "LPTimeConstant" ) );
  LPTimeConstant *= Parameter( "SampleBlockSize" );
  // Check for configuration error:
  if( LPTimeConstant <= 0 )
    bcierr << "The LPTimeConstant parameter must be greater 0"
           << endl;
  // Alternatively:
  PreflightCondition( LPTimeConstant > 0 );
  // Request output signal properties:
  outputProperties = inputProperties;
}
```

LPFilter::LPFilter()

```
LPFilter::LPFilter()
: mDecayFactor( 0 ),
  mPreviousOutput( 0 )
{
  BEGIN_PARAMETER_DEFINITIONS
    "Filtering float LPTimeConstant= 16s"
      " 16s 0 0 // low pass time constant "
      " in blocks or seconds",
  END_PARAMETER_DEFINITIONS
}
```

Making `LPFilter.cpp` compile

```
#include "PCHIncludes.h" // Make the compiler's Pre-Compiled
#pragma hdrstop          // Headers feature happy

#include "LPFilter.h"

#include "MeasurementUnits.h"
#include "UBCIError.h"
#include <cmath>

using namespace std;

...
```

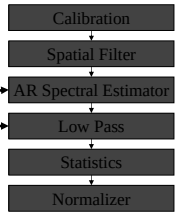
Instantiating `LPFilter.cpp` in a Signal Processing module

UFilterHandling.cpp:

```
...
#include "LPFilter.h";

Filter( CalibrationFilter, 2.A );
Filter( SpatialFilter, 2.B );
Filter( LPFilter, 2.D );
Filter( ARTemporalFilter, 2.C );
Filter( StatFilter, 2.E1 );
Filter( NormalFilter, 2.E2 );
```

Filter chain:



Instantiating `LPFilter.cpp` in a different type of module

- Other possible environments for filter instantiation
  - non-signal processing module
  - command line tool
- Then: **`RegisterFilter()`** statements in the filters' own cpp files
- All filters linked into the executable will be instantiated
- Filter chain built according to lexical string comparison

Tutorial Example: Low Pass Filter

- Code is available to play with in
  `BCI2000/SignalProcessing/LPFilter`
- Additionally: Data visualization
- Build it as a command line tool:

```
cd BCI2000\Tools\cmdline
make LPFilter.exe
LPFilter --help
```

Framework Facilities

- Parameter access
- State access
- Reporting errors and warnings

Parameter Access

- Getting parameter values:
  `float myFloat = Parameter( "SamplingRate" );`
- Setting parameter values:
  `Parameter( "TimeConstant" ) = myFloat;`
- Matrix parameters:

```
myFloat = Parameter( "MUD", 1, 2 );
myFloat = Parameter( "Audio", "Task", 2 );
myFloat = OptionalParameter( 0, "MUD", 20, 30 );
```

State Access

- Getting state values:
  `short targetCode = State( "TargetCode" );`
- Setting state values:
  `State( "ResultCode" ) = targetHit;`
- Optional states:
  `short artifact = OptionalState( 0, "Artifact" );`
- Beware of typos in names – you won't get an error message for
  - Optional states and parameters (obviously)
  - Non-existent row/column indices for matrix or list parameters (less obvious)

Reporting Errors and Warnings

- C++ command-line programs use

```
cout << "The result is " << result << "." << endl;
cerr << "This is an error message." << endl;
```

- BCI2000 filters use

```
bciout << "The result is " << result "." << endl;
bcierr << "TimeConstant must be greater 0." << endl;
```

- Side effects
  `endl` is an `'\n'` forcing immediate display
  `bcierr` from `Preflight()`: Prevents `Initialize()`
  `bcierr` elsewhere: Terminates module

Miscellaneous

- Measurement units
- Data visualization
- GenericFilters and VCL windows
- Playing sounds
- Log files

Measurement Units

- Natural units in a BCI2000 system
  - time: duration of a block
  - frequency: sampling rate
- User inconvenience
  - error-prone/time-consuming manual conversion required for setup
  - hardware setup changes will affect many parameters
- Automatic parameter conversion

```
oneSecondInBlocks = MeasurementUnits::ReadAsTime("1000ms");
twoBlocks = MeasurementUnits::ReadAsTime("2");
```

Data Visualization

- Purpose: displaying data or text in an operator window
- Class **`GenericVisualization`**
  - unique ID in **`defines.h`**, constructor takes ID
  - display a signal: **`GenericVisualization::Send(GenericSignal)`**
  - display text: **`GenericVisualization::Send("your message")`**
  - various configuration options: **`GenericVisualization::Send(cfgid, value)`**

`GenericFilters` with Windows

- Borland Visual Class Library (VCL)
  - + nice to use, mostly intuitive
  - – restricted use of C++ when VCL classes come into play
- Problem: VCL windows (forms) cannot be **`GenericFilters`**.
- Suggestion: Use a VCL form that 'belongs' to a **`GenericFilter`**, and is invisible from outside.

`GenericFilters` with Windows

```
class Task : public GenericFilter
{
 public:
  Task();
  ~Task();
  ...
 private:
  ...
  class TTaskForm* mpForm;
};

class TTaskForm : public TForm
{
  ...
  // Auto-generated declaration
  // of TTaskForm
};
```

```
Task::Task()
: mpForm(new TTaskForm((TComponent*)NULL)),
  ...
{
  ...
}

Task::~Task()
{
  delete mpForm;
}

void Task::Initialize()
{
  mpForm->Width =
   Parameter("WindowWidth");
  ...
}
```

Playing Sounds

- **`WavePlayer`** class handles reading and playing windows wave files using DirectX 8+.
- Load a file in **`Initialize()`** using **`WavePlayer::AttachFile()`**, play it with **`WavePlayer::Play()`**.
- Use STL containers to store multiple **`WavePlayer`** instances.

Log Files

- Purpose: logging output to a file side-by-side with brain signal data files
- Individually implementing a log file is cumbersome and error prone
  - path and file name construction from parameters
  - opening/closing, keeping track of file handles
  - avoiding crashes from invalid file handles

Log Files

- Class **`LogFile`**
- Add a member **`mLogFile`** to your filter
- Ctor-initialize it with an individual file type extension
- Call **`mLogFile`**'s **`Preflight()`** and **`Initialize()`** from your filter's **`Preflight()`** and **`Initialize()`**
- Write to the log file:

```
mLogFile << "Result Code is "
         << resultCode << '\n';
```

Conclusion

- Extending BCI2000 is done by deriving your own filter class from **`GenericFilter`**.
- Begin with coding the core functionality into your filter's **`Process()`** member function, then derive the others.
- The framework tries to make error handling, state and parameter access easy, safe, and user friendly – give it a try!
- Share your ideas, suggestions, annoyances **`juergen.mellinger@uni-tuebingen.de`**

Hardware Concepts

Christoph Guger
g.tec Guger Technologies

1st BCI2000 Workshop
The Desmond Conference Center
Albany, New York, USA
June 13+14, 2005

Christoph Guger
g.tec medical engineering GmbH
Guger Technologies OEG
Herbersteinstrasse 60
8020 Graz

Hardware Concepts

- USB based amplifier
- amplifier with serial interface
- stand-alone amplifier with standard DAQ board
- electrode caps and electrodes
- medical systems

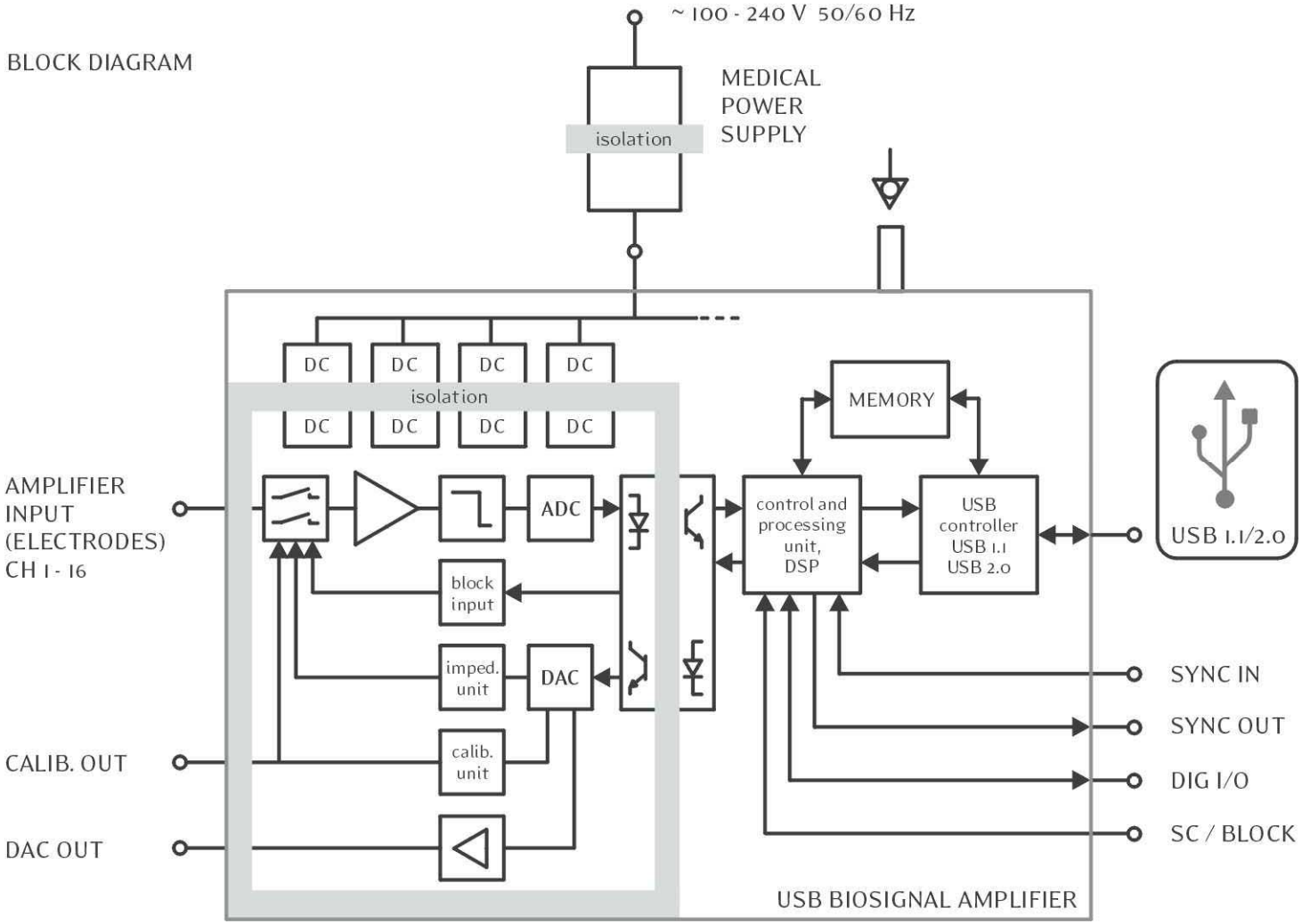
g.USBamp - Highlights

- EEG, ECoG, ECG, EMG and EOG recording via USB
- 16 analog inputs with 24 Bit and 38.400-Hz sampling frequency per channel
- digital filtering of the biosignal data
- over-sampling to achieve high SNR
- can be connected to a PC or notebook
- simultaneous sample and hold of 16 channels
- direct connection of electrodes with standard safety connector
- system connectors for user specific patient cables
- DC input channels, ±250 mV sensitivity

Notified body
0636

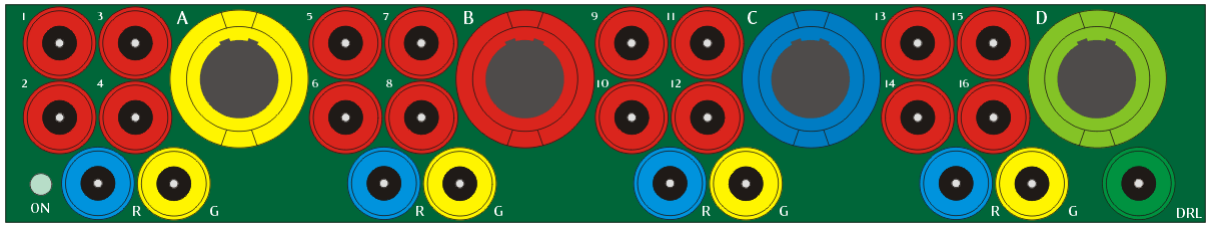
www.gtec.at

Hardware Architecture



www.gtec.at

Front Side



g.USBamp has 4 mono-polar amplification groups with separated ground and reference

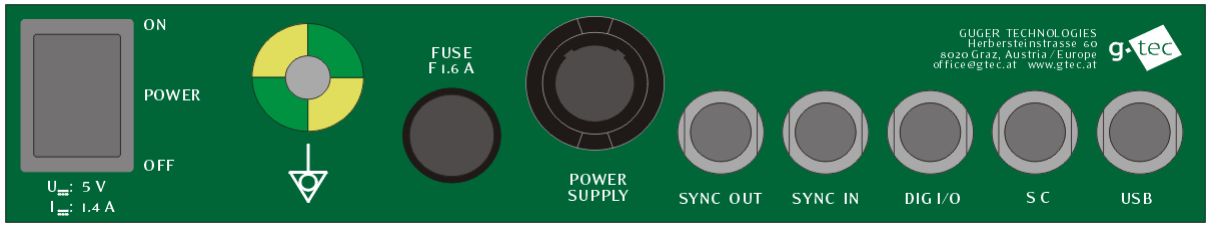
Group A: Safety sockets 1 - 4: 4 analog input channels for
EEG, ECoG, EMG, EOG or ECG
Push-pull connector A is connected to safety sockets 1 – 4
Group B: Safety sockets 5-8,…
Group C: Safety sockets 9-12,…
Group D: Safety sockets 13-16,…

The 4 ground sockets and reference sockets can be connected

The green LED on the left side (ON) indicates power on

The calibration signal is available at push-pull connector D

Rear Side



POWER	switch for switching ON/OFF the device
⏚	potential equalization conductor
FUSE	1.6 A fuse, type 20 mm, fast-acting
POWER SUPPLY	socket for the connection of the external power supply unit
SYNC IN/OUT	socket for the synchronization with another g.USBamp
DIG I/O	socket for digital inputs and outputs
SC	socket to disconnect the input electrodes from the amplification units. The amplifier unit inputs are connected to ground potential
USB	USB socket for the connection with the PC or notebook

Certification

IP 41: protected against particulate matter of degree 4 (wire with diameter of 1 mm). Protection against ingress of water of degree 1 (protection against vertical falling water drops)

S1: permanent operation

CE mark, notified body 0636

applied part of type CF – invasive usage: ECoG, directly on the heart

attention to instructions

safety class II

Overampling

$$OS = 2^{2M}$$

OS … oversampling rate
M … gain of bits
OS = 4 -> M=1 -> gain of 1 Bit

• each ADC operates with 2,4576 MHz and performs a 64 times oversampling -> 38.400 Hz per channel

• floating point DSP performs additional oversampling and filtering

• 128 Hz sampling rate has an oversampling rate of 19.200

• noise is reduced by SQRT(19200) = 139

• very high signal to noise ratio

Modi

Measure biosignal data

amplification of the input channels
analog to digital conversion with 24 Bit
overampling
bipolar derivation
bandpass filtering
notch filtering (50/60 Hz)
transmission of data over USB

Test signal

DSP generates a sine-, rectangular-, sawtooth- or noise-signal connected to each input signal
used for system testing, algorithm testing, calibration

Calibrate

adjust the gain and offest of each channel

Impedance measurement

apply a sine wave signal to each electrode and measure its impedance
the measurement frequency can be changed

Modi

Short-cut

TTL high impulse on the SC input socket

-> disconnect all electrode input sockets from the input amplifiers and connect the inputs to ground potential

protection of the amplifier against noise (gamma knife)

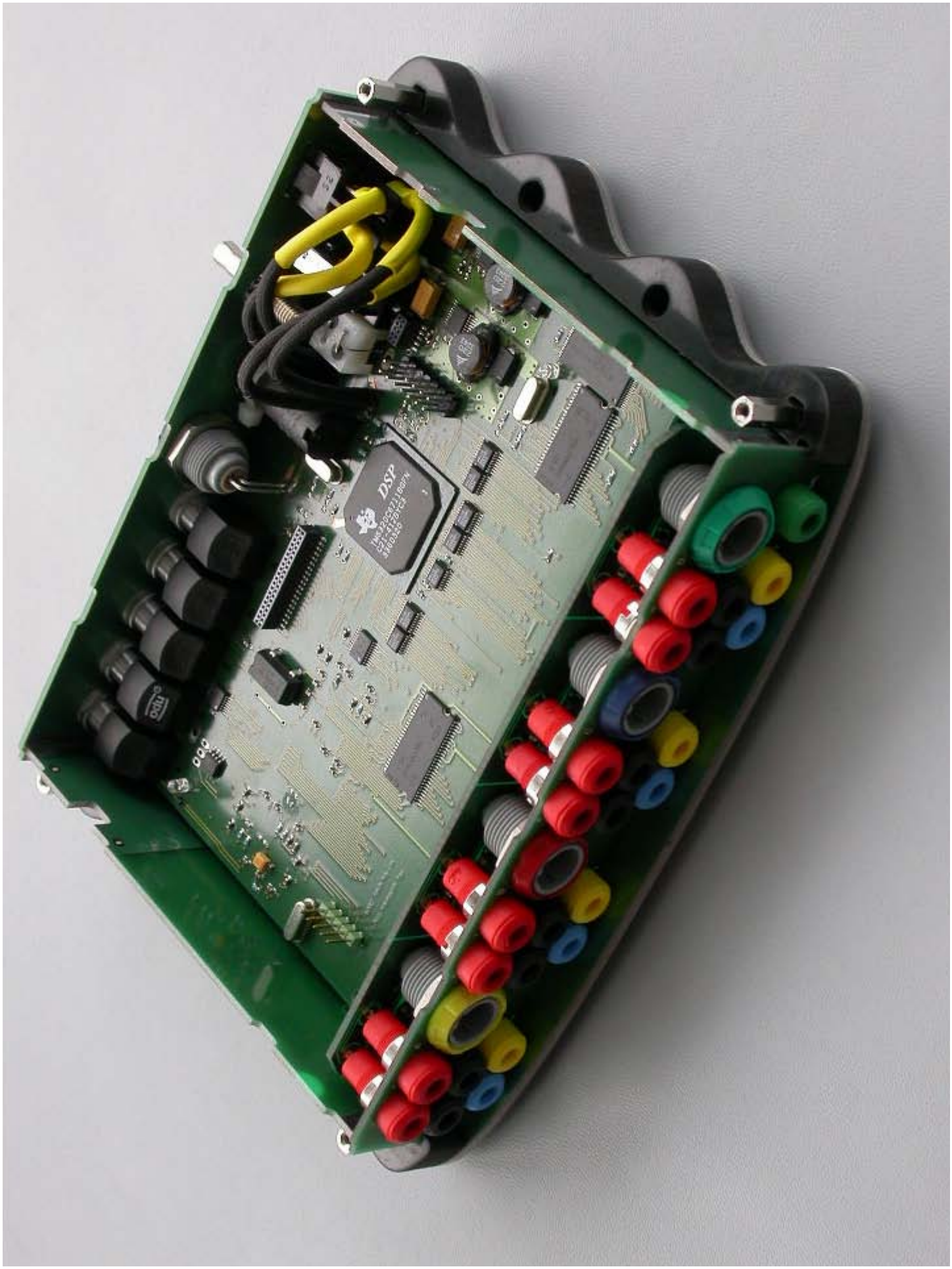
Synchronization of multiple g.USBamps

amplifiers can be stacked to 16-, 32-, 48-, 64-channel systems

connect socket SYNC OUT (MASTER device) to SYNC IN (SLAVE devices)

MASTER drives the slave devices

all channels have the same sampling frequency


DSP

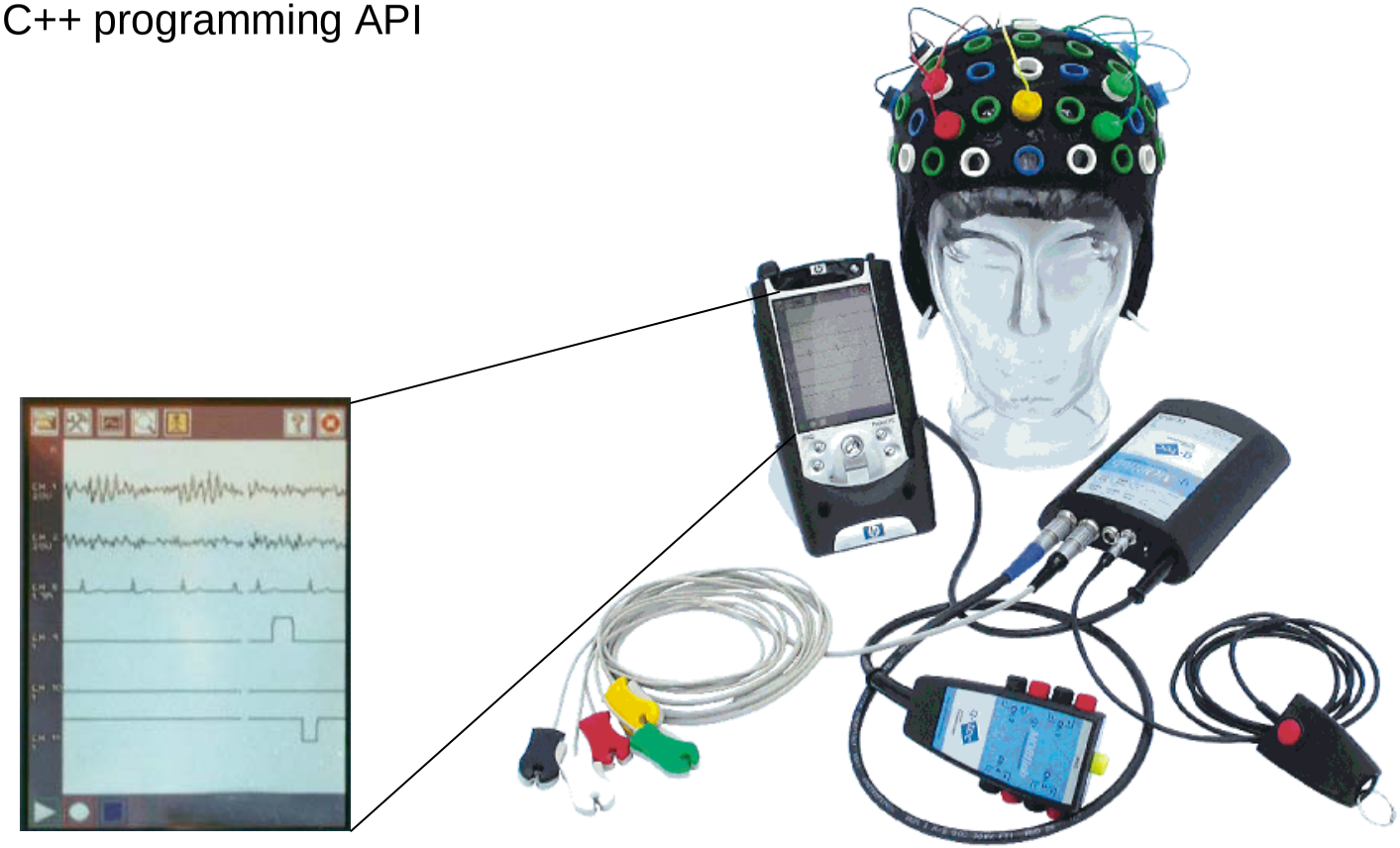
g.MOBIlab – Mobile Laboratory

Mobile Pocket PC biosignal acquisition system

uses standard serial interface

TCP/IP remote control unit

C++ programming API



Hardware Architecture

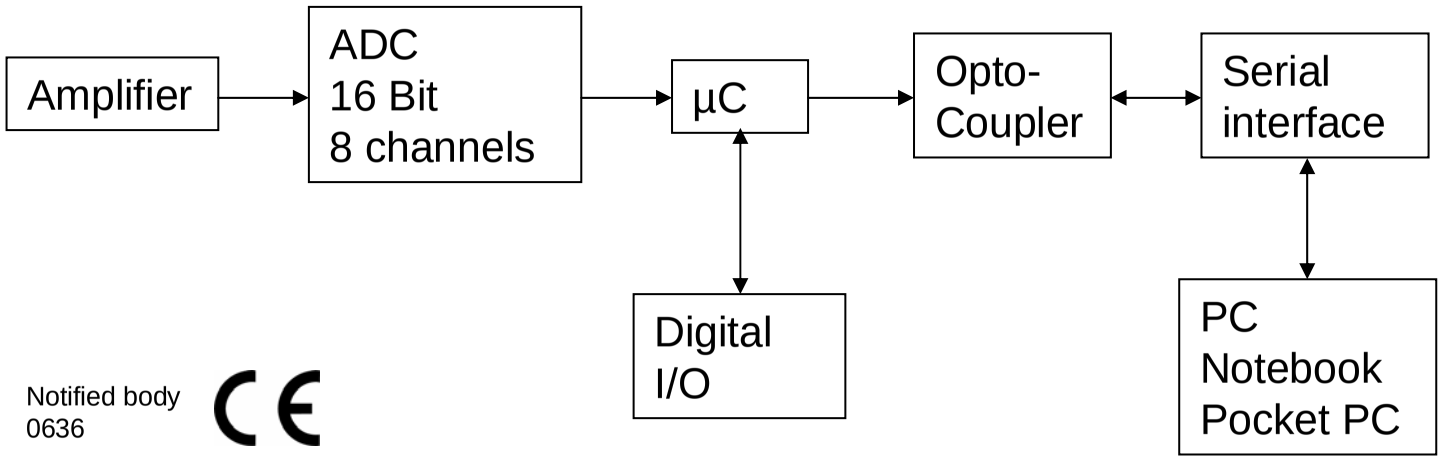
256 Hz sampling frequency

multiplexed 16 Bit ADC

supplied by 2-4 AA batteries

applied part BF

operation time: 1 week



Notified body
0636

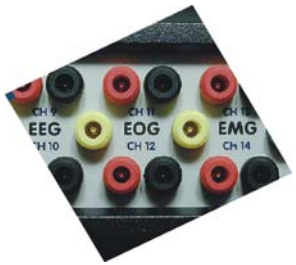
www.gtec.at

g.BSamp – Biosignal Amplifier

GUGER TECHNOLOGIES

- multi-modal amplifier system
  - Electroencephalogram – EEG
  - Electrocardiogram – ECG
  - Electromyogram – EMG
  - Electroocculogram – EOG
  - Respiration
  - Other sensors

- 1 to 16 bipolar/unipolar channels per amplifier
- 16 channel systems are stack-able
- portable due to battery supply
- stand-alone system – combine it to other user-specific systems (e.g. National Instrument boards)



design the amplifier exactly to your needs

Notified body 0636

CE

BCI 2000

www.gtec.at

Hardware Architecture

amplifier has analog output and is stand-alone device

DAQ boards available from different manufacturers

- different resolution (typically 12 or 16 bit)
- sampling frequency (typically 100 kHz to 300 kHz)
  - 100 kHz / 64 channels = 1,5 kHz per channel
- C++ API available

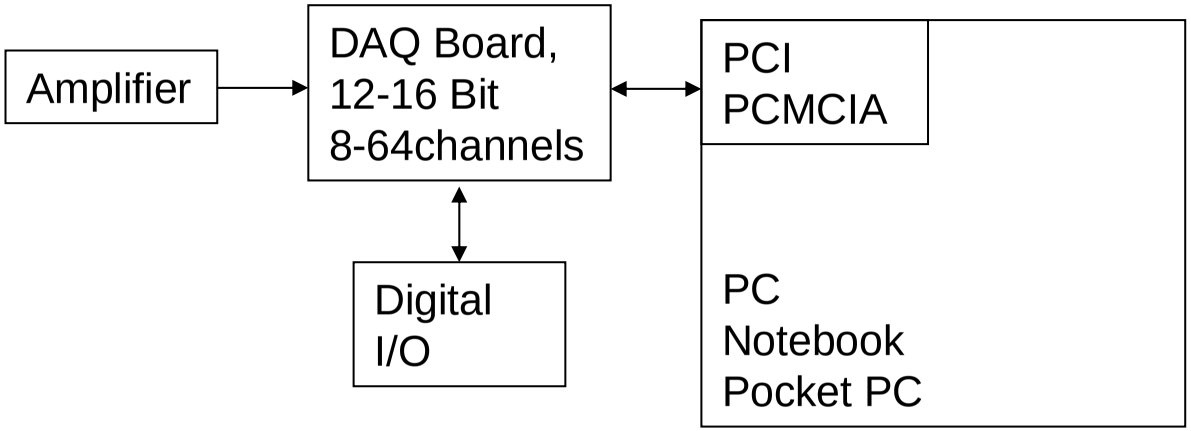
signal is NOT digitized in the amplifer

multiple DAQ boards can be used

- device must be synchronized

important that all channels are scanned as fast as possible

digital I/Os often NOT scanned with analog inputs



Electrode Caps and Electrodes

Example 1: g.EEGcap with screwable electrodes

electrode positions according to 10/20 system

each electrode has safety connector with 1.5mm diameter

electrodes can be screwed into cap for optimal adjustment

each electrode has its own cable and can be replaced

Example 2: Electrocap with built in electrodes

electrode positions according to 10/20 system

one/two Sub-D connector cables for all electrodes

connector cable to amplifier necessary

fast assembling

if one electrode breaks -> cap must be removed

Medical System

If an amplifier is connected to other devices like a PC the following leakage currents have to be checked:

ground leakage current (transformer to ground)
enclosure leakage current (housing to ground)
patient leakage current (applied part - patient - ground)

The leakage currents must be checked if several amplifiers are interconnected

What is the BF and CF?

applied part of type BF – medical device with isolated applied part
(e.g. EEG, ECG)
ground leakage current ≤ 100 µV
patient leakage current ≤ 100 µV

applied part of type CF – medical device for intracardial applications
(e.g ECoG)
ground leakage current ≤ 100 µV
patient leakage current ≤ 10 µV

Medical System: Who is responsible?

Example 1: Stand-alone amplifier (e.g. g.Bsamp) is bought by University Darwin. The University has its own DAQ board and PC.

Step 1: Check if the amplifier has an applied part of type BF/CF and is a certified medical product

Step 2: Connect the amplifier to the DAQ board and to the PC

Step 3: Check the leakage currents

If the leakage currents are small enough the device can be used.

Otherwise use an isolating transformer and check the leakage currents again.

Example 2: Stand-alone amplifier is bought by University Darwin together with DAQ board and PC.

The company has to provide the documents

g.tec medical engineering GmbH
Guger Technologies OEG
phone: +43-316-675106
email: office@gtec.at
web: www.gtec.at
Herbersteinstrasse 60
8020 Graz
Austria

g.tec - GUGER TECHNOLOGIES ◆ Herbersteinstr. 60, 8020 Graz, Austria ◆ tel. (+43) 316 675106 ◆ office@gtec.at ◆ www.gtec.at

Configuration, Conduction, and Analysis of Sensorimotor Rhythm Experiments

Febo Cincotti
Fondazione Santa Lucia

1st BCI2000 Workshop
The Desmond Conference Center
Albany, New York, USA
June 13+14, 2005

Tutorial

BCI 2000

Mu Rhythm Acquisition and Analysis Using *Mario*

MU RHYTHM ACQUISITION **3**
OBJECTIVE 3
PREREQUISITES 3
PLANNING YOUR EXPERIMENT 3
BCI2000 SET-P 4
MONTAGE INSTRUCTIONS 4
CONDUCTING THE EXPERIMENT 5
Instructions to the subject *5*
Monitoring the recording session *6*
THE SCREENING 6
MANAGEMENT OF THE SUBJECTS OVER THE SESSIONS 7
BCI2000 SETUP FOR MU RHYTHM **8**
BCI2000 SETUP FOR MU RHYTHM **8**
SCREENING AND FOLLOWING SESSIONS 8
SYSTEM CONFIGURATION 9
SUMMARY OF SYSTEM OPERATION 13
Acquisition *13*
Spatial Filtering *14*
Spectral estimation *14*
Combination of features into a logical signal *14*
Signal translation *15*
Application and classification *15*
GENERAL CONFIGURATION 16
Storage settings *16*
Display settings *16*
SETTING UP THE MU PIPE **17**
MU RHYTHM ANALYSIS **18**
MU RHYTHM ANALYSIS **18**
STARTING AN ANALYSIS SESSION 18
OPENING FILES WITH MARIO 18
RUN COMPUTATIONS 20
VIEW RESULTS 21
R-square matrix *21*
Power Spectra *22*
Topographic plots *23*
INTERPRETATION OF THE RESULTS 24
BEWARE OF ARTIFACTS! 25
IMPROVING THE ANALYSIS 28
ANALYZING THE SCREENING 28
CONCLUSIONS 28
D2BOX APPLICATION STATES **29**

Mu Rhythm Acquisition

Objective

This tutorial intends to give you a step-by-step guide on how to configure BCI2000 to control a cursor on the screen using Mu Rhythm modulation. Since this is an introductory tutorial, only one-dimensional control will be covered.

Prerequisites

This tutorial assumes:

- that you have installed the latest version of BCI2000
- that you were able to connect the specific BCI2000 Source Module (e.g. DT2000.exe, Neuroscan.exe, RDAClient.exe, etc.) to your acquisition system
- that you were able to start any of the demo configurations (e.g. running `startupD2Box.bat`)

Planning your experiment

You are going to test Brain Computer Interface control mediated by a physiological phenomenon named mu rhythm desynchronization / synchronization. Please refer to other parts of this documentation to have further information about the physiology.

Briefly, the mu rhythm is a rhythmic component at approximately 10 cycles per second that is generated over sensorymotor areas and that tends to be suppressed when a motor task is performed or imagined.
Spectral analysis can reveal quite neatly the presence of a peak in the alpha (i.e., 8-12 Hz) and possibly in the beta (18-24 Hz) band that is reduced even by a factor of 10 during motor performance. Training may allow a subject to gain voluntary control over this spectral component, and to modulate it at will. A time-resolved spectral analysis can provide the instantaneous amplitude of the mu rhythm as a control signal for the movement of a cursor.

The BCI2000 package provides an experimental environment in which a horizontally sweeping cursor is presented and its vertical deflection is controlled by mu amplitude.
The speed of the cursor may be adjusted to allow the subject to take several seconds to hit the target, and may be decreased as training proceeds.

This system is in fact particularly suited for the early training of subjects who have to learn to modulate their mu rhythm.

Usually, a trial lasts about six second (considering cursor movement time, inter-trial interval and other periods); a subject is then asked to try and hit about thirty targets in a row (i.e., one experimental run).

In fact, as a rule of thumb the experiment must be designed such that continuous recording does not last more than 3-5 minutes: because subjects are typically told to minimize movements (to minimize artifacts), sessions are physically and psychologically stressing, and thus it is not indicated to record for longer time periods. Recording "runs" will be repeated until the amount of data collected suffices. Consider to keep the overall recording time below an hour (about half an hour recording, plus preparation and cleaning), so that experimental subjects still enjoy participating in the study.

Knowing this, we must now set our system so that it receives the relevant EEG data and performs the proper analyses.

BCI2000 set-p

We assume by now that for some reason you already know the most relevant locations and frequency bands. This information is typically determined by analyzing the data acquired during a previous session or during a screening session.
In the following paragraphs, we assume that electrode Cz shows the maximally discriminant difference between target and non-target responses in a frequency band between 10 and 14 Hz.

Experimental sessions with a Mu rhythm controlled cursor are supported by BCI2000 through the use of, for example, the ARSignalProcessing and the D2Box modules, together with the Source module appropriate for the available EEG equipment.
Parameters for each module must be set up in accordance to the current EEG sampling frequency and to the specifications set out above regarding the responsive EEG features. Step-by-step guide to setting up BCI2000 for a mu rhythm experiment are given in another section.

Montage instructions

Electrode placement is covered more extensively in an Appendix to this tutorial. Here we just underline how important it is that:

1. the cap has been carefully cleaned. Dried residuals of gel will likely result in higher electrode impedances.
2. the cap is placed with precision. If in a previous session you found that a certain electrode is responsive, you want to put that electrode over the very same position as in the session before.

3. the conductive gel is injected in the right quantity; too little will not allow a good contact; too much will spread out to large areas, thus resulting in a spatially averaged signal.

1. Reference and ground electrodes are correctly utilized, so to minimize the mains disturbance.

Conducting the experiment

Instructions to the subject

After the subject is ready for EEG acquisition, he/she is given instructions about the experimental task.

A screen with a T in the center is initially presented. As soon as the subject is ready and the EEG traces are stabilized, the investigator will start the acquisition, and the T will disappear. For each trial, four phases will occur:

1. *Target presentation.* A target appears on the right edge of the screen for about one second.
2. *Cursor movement.* The subject receives feedback about his/her mu amplitude by the trajectory of the cursor on the screen. The time needed for the cursor to move from the left to the right edge of the screen should be approximately three seconds.
3. *Result.* If the cursor hits the target, the latter changes its color; otherwise, the screen will blank. In either case, this period lasts one second.
4. *Intertrial Interval.* Screen will turn/remain black for one second, after which the cycle starts over.

When a target is presented, the subject must concentrate on whatever he feels can modify the cursor trajectory towards the desired direction. In other words, it is not the subject's task to perform a particular mental imagery, but it is the subject's task to control the cursor.
In the early part of the training, the experimenter will suggest that up/down movement may be achieved by imagining movement of upper/lower limbs. In fact, this is likely to produce a differential modulation of mu rhythm over the hands (C3, C4) or feet (Cz) motor areas. After a few sessions, motor imagery typically becomes less important.

The second set of instructions to the subject regards the minimization of artifacts. Since in a typical recording session EEG data are acquired for off-line analyses, it is important that all efforts are made to obtain the highest quality EEG. The main causes of artifacts are:

- Contraction of the muscles of the face/head, swallowing;
- Eye blinks and eye movements;
- Motion

Provided that subjects are asked to minimize artifacts (e.g. asked to try to swallow only during the pause between letters), he/she should be further assisted in these efforts by providing a comfortable chair and a dimly lit room. We will return to the role of reinforcement, but it is already clear that the experimenter must carefully monitor the EEG and alert the subject in the case he/she has forgotten some of the instructions.

When the experimenter is sure that his/her instructions have been well understood, the recording session may start.

Monitoring the recording session

After the recording session has started, one might feel the temptation to leave the subject alone during the run since most of the experimental activities are automated in BCI2000. On the contrary, the experimenter has several tasks:

- Filling in the run-sheet to report information that is not automatically recorded by BCI2000 and that will later help when data are analyzed.
- Monitoring the EEG waves, checking that the quality of recording does not decrease (electrode contact failure, muscular, EOG or motion artifacts, etc.)
- Reinforce the subject: warn him/her if he/she is producing artifacts, keep him/her alert if getting drowsy, give him/her feedback about his/her performance so that interest is kept high.

The run sheet is the most important mean of communication between the technician who performs the recordings and the person who analyzes the data (or possibly the historical memory of whoever does both). It is important that it is compiled carefully and that it is rich of appearingly obvious details: only time will say what is standard and what changes from session to session, and if you will need to analyze data acquired years before, you will probably have a problem if you did not note down all information.

The screening

The first time the subject is recorded, it is unlikely that the experimenter can guess a responsive EEG feature by chance. Thus, during the initial session, the subject will not be able to control anything. The purpose of this first session is rather to determine which signals change when the subject engages in particular tasks.

To avoid frustration, and to provide the highest amount of information to the experimenter, an experiential paradigm named screening was developed.

It consists of alternate presentation of cues on opposite sides of the screen (either up/down (vertical) or left/right (horizontal)). In the first run, the subject is asked to move his/her hands when the top target appears and to

move his/her feet when the bottom target appears. The second run is similar to the first, except that the movement is just imagined. This sequence is repeated three times.
From the seventh run on, the targets appear on the left or right side of the screen, and the subject is asked to move (odd runs) or imagine to move (even runs) his/her hands.
The screening ends at the twelfth run.

Management of the subjects over the sessions

Ideally, the subject will show some kind of responsive feature in the screening data, and that response should be enhanced in amplitude (larger R-square) with training.

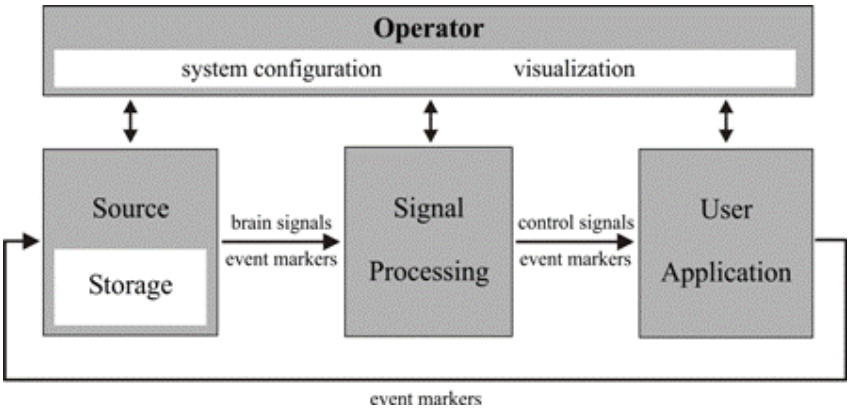
In practice, screening will probably not show a clear and unique responsive feature, and a "promising" feature may reveal not to be as good in the following sessions. Moreover, the development of new control strategies may slightly modify the responsive channel or the frequency.

For this reason, at the end of a training session it is strongly advised that you analyze the recorded data. By doing this, you can assess the predicted trend of the training, or realize that something different is going on and undertake the proper action. In neither case, system parameters should never be changed during one experimental (i.e., one hour) session so as to not confuse the subject. Changes should always occur between sessions after a particular trend (e.g., change in frequency or location) has been clearly evident in several sessions.

BCI2000 setup for Mu Rhythm

Pipe configuration

In this section we will use the following modules:



Source	Signal Processor	User Application
RDAClient	ARSignalProcessing	D2Box

The RDAClient source was chosen only as an example, and you should replace it with the specific module for your data acquisition hardware.

If you are not familiar with pipe setup, please have a look at [Setting up the mu Pipe]

Screening and following sessions

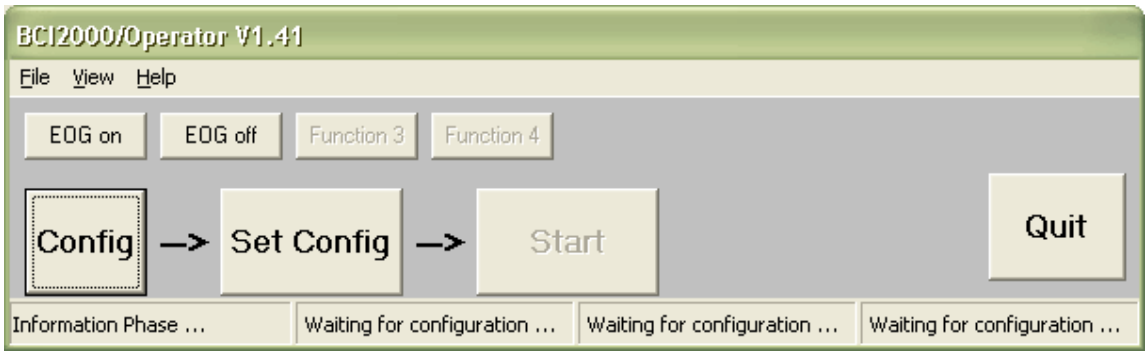
Before the system can reliably control the cursor with mu rhythm modulation, you need to determine the individual EEG feature (e.g. channel and frequency bin) that most reliably separates responses where the mu rhythm is either present (i.e., synchronized) or absent (i.e., desynchronized).

As an example, we will now assume that you would like the system to use a frequency band between 10 and 14 Hz at channel Cz to enable the user to control the cursor.

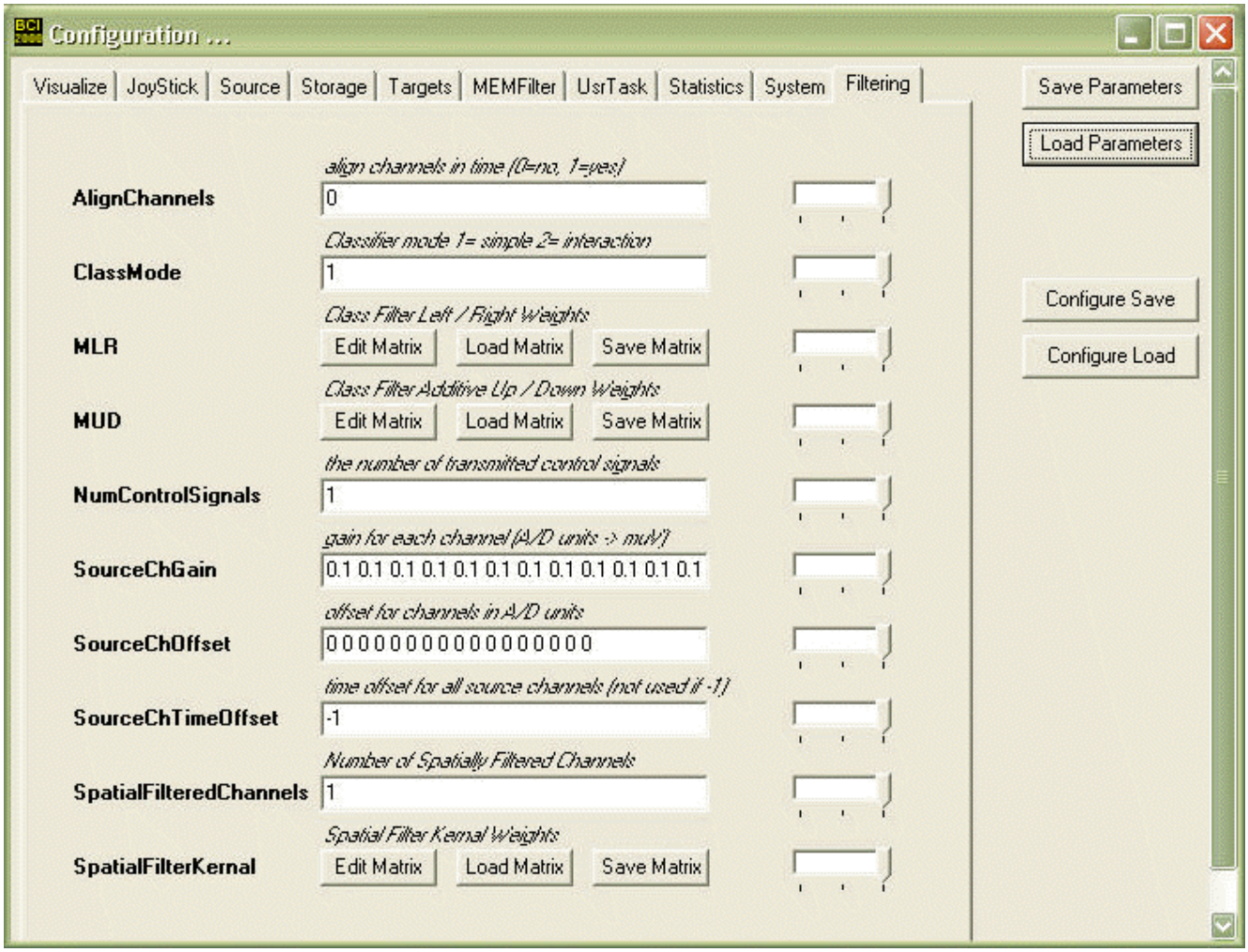
After the subject has been recorded once, off-line analysis will allow us to select the correct values.

System configuration

On the Operator window, press the Config button in order to access the Configuration dialog.



In the Configuration dialog press Load Parameters and choose the file `C:\BCI2000\Tutorial\prm\D2Btutorial.prm`. this will automatically set the proper value for each parameter. Close the Configuration dialog, return to the Operator window and press Set Config.



At this point the system has been configured as follows:

Group	Parameter	Value	Description
Source	SamplingRate	256	The sampling rate per channel. a number above 200 suffices for most EEG applications.
	SampleBlockSize	8	In this example, the source module will transmit data through the processing pipe in blocks of 16 samples.

			This results in a system update of 256/8=32 times per second (or 31.25 ms per update).
	TransmitCh	16	This is the subset of the total number of channels you want to transmit to Signal Processing. Unless your system is slowed down by processing all channels, you should transmit all the source channels.
	TransmitChList	1 2 3 ... 16	Enumeration of all *TransmitCh* channels.
	SourceChOffset	0.1 0.1 0.1 ... 0.1	The first step in Signal Processing is calibration. Signals are converted from A/D units into µV. The used transformation is: result in µV=(A/D units-offset)*gain. Specify these variables as appropriate for your system.
	SourceChGain	0 0 0 ... 0	
Filtering	SpatialFilteredChannels	2	This number (e.g., 2) specifies how many output channels result from the spatial filtering operation

			(see SpatialFilterKernal) and will be available to the linear classifier.
	SpatialFilterKernal	[SpatialFilteredChannels x TransmitCh] matrix	BCI2000 can provide any linear spatial filtering operation (e.g. Common Average Reference, Surface Laplacian, etc). Specify weights that determine which linear combination of input channels make up the output channels.
	MUD	[#of features x 3] matrix	See details in the text below this table
	MLR	0 0 0 0 0	
MEMFilter	MemWindows	8	The length of the EEG epoch utilized for each spectral estimation (in samples) is MemWindows x SampleBlockSize = (in this example) 64 samples = 250 ms. You should use periods between 200 and 400 ms.
	MemDetrend	1	Remove signal mean prior to frequency analysis.
	MemBandWidth	2	Bandwidth of frequency bins.
	StartMem	0	Band of

	StopMem	60	frequencies covered by spectral analysis. This band is broken up in *MemBandWidth* bins.
	MemModelOrder	16	Order of the autoregressive model. Should never be more than half the number of samples of MemWindows x SampleBlockSize. Typical values are between 12 and 25.
	deltaMem	0.2	Spectral sampling (for internal use, must be < MemBandWidth). Just leave at 0.2.
Statistics	XMean	-1	Baseline of the horizontal control signal
	XGain	400	Gain of the horizontal control signal
	YMean	0	Baseline of the vertical control signal
	YGain	100	Gain of the vertical control signal

Summary of system operation

Acquisition

When the Analog-to-Digital Converter on your acquisition system sends a sample to the Source module, it is buffered until a block of SampleBlockSize (32) samples is ready, which happens every SampleBlockSize/SamplingRate*1000 ms. This rate is the pacemaker for the rest of the processing pipe. Only a

subset (TransmitChList) of the acquired (and saved) channels is transmitted to the processor, where values in ADC units are first converted into microvolt.

Spatial Filtering

The transformation matrix can implement different spatial transformation. In the simplest case, i.e. if it is a unity matrix, the output of the SpatialFilter will be the same raw EEG data that were input.

If the transformation matrix is a rectangural "quasi-unit" matrix, e.g.:

0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0

the output will only have two channels, that are identical to input channels 13 and 14 (see the ones in column 13 and 14), respectively. If the matrix is as follows:

-1/16	-1/16	-1/16	-1/16	-1/16	-1/16	-1/16	-1/16	-1/16	-1/16	-1/16	-1/16	**1**-1/16	-1/16	-1/16	-1/16
-1/16	-1/16	-1/16	-1/16	-1/16	-1/16	-1/16	-1/16	-1/16	-1/16	-1/16	-1/16	-1/16	**1**-1/16	-1/16	-1/16

it will implement the Common Average Reference, i.e., from each channel the average of all other channels is subtracted prior to further processing.

Spatial filters that are more complicated (e.g. ones derived from Independent Component Analysis) can be easily implemented by computing the corresponding linear transformation matrix. The output of the spatial filter is a matrix of *SpatialFilteredChannels* by *SampleBlock* size samples. These samples are submitted to frequency analysis.

Spectral estimation

At every new block of spatially filtered signal, the latest MemWindows blocks are collected and used to identify an autoregressive model. This model is use to calculate the frequency spectrum of the time signals, and sampled from StartMem to StopMem in steps of deltaMem Hz. These values are averaged in bins of MemBandWidth Hz. Thus, the output of spectral estimation is a matrix of *SpatialFilteredChannels* by (*StopMem*-*StartMem*)/*MemBandWidth* frequency bins. This matrix is submitted to the next stage, the linear classifier, that then extracts from this matrix the frequency amplitudes at the desired location(s) and frequency(ies).

Combination of features into a logical signal

The aim of the MUD matrix is to tell the program which of these features to use and how to combine them into a single-dimensional signal.

You need to convert the frequency(es) that you would like to use (e.g., 11 Hz and 13 Hz) into bins, using StartMem and MemBandWidth SamplingRate. (e.g. (11-0)/2+1/2=6 and (13-0)/2+1/2=7). Since we want to use spatially filtered channel 1 (corresponding to transmitted channel 5, Cz) at the same time, MUD must be set as follows

	chan	bin	weight
Feature 1	1	6	1
Feature 2	1	7	1

In this example, the program will compute the output of the Filtering stage as the sum of Feature1 (bin 6 at channel 1) times a weight of 1 + Feature2 (bin 7 at channel 1) times a weight of 1. The output of this stage is a 3-dimensional control signal (corresponding to matriced MUD, MLR, and MZ). The subsequent stage then transforms this signal into a signal with (ideally) zero mean and a particular standard deviation.

Signal translation

At the statistics stage, the system subtracts YMean from the control signals, and multiplies the result by YGain. A mechanism of Automatic Gain Control allows to estimate the best value for these parameters, so that there is no bias in the number of Up and Down targets (which should be 50%); plus, the gain is set so that the cursor sensibly deviates from the horizontal motion, but only occasionally goes out of the screen.

Gain and intercept are updated after every trial, using the data of the last 3 "Up" trials and the last 3 "Down" trials. (These, and other adaptation parameters, are merely defaults and can be changed. Make sure you obtain a correct set of parameter for these variables, because otherwise the system will adapt improperly.) The initial value set at the beginning of the session only influences the first trials. A reasonable starting point is 1 for YMean and 500 for YGain.

The same kind of transformation holds for the horizontal movement of the cursor. From the definition of the MLR matrix, the horizontal control signal is always zero (because the third column contains only zeros and thus there is no influence of brain signals on left/right movement); thus, subtracting -1 and multiplying by 400 induces a uniform movement to the right that will be completed in 32768/400 blocks, i.e. 2.56 s.

Application and classification

The system will then transmit the normalized control signal to the User Application. The latter will update the position of the cursor, increasing its coordinates proportionally to the horizontal and vertical control signals.

General configuration

Some of the parameters have not been discussed so far. Some of these are technically advanced, and should be left to their default value unless the experimenter knows how to use them. Other parameters are, on the other side, of very general use and will be covered here.

Storage settings

These parameters define where on the disk the acquired data will be saved.

FileInitials	`..\Tutorial\data\P300\`	Base folder for storing the current session. It can be an absolute path, or can be relative to the position of the program files.
SubjectName	TestSubj	Subject initials
SubjectSession	001	Session number
SubjectRun	04	Run number

Assuming that operator.exe is installed in `c:\bci2000\Operator\`, the files will be stored in the folder: `c:\bci2000\Operator\TestSubj001\TestSubjS001R04.dat`. The run number will increase automatically, so the experimenter does not have to care about changing it after each run.

Display settings

WinHeight	512	Width and Height of the User Application screen (e.g. the one where the matrix flashes)
WinWidth	512	
WinYPos	0	Horizontal and Vertical position of the top-leftmost corner of the User Application screen. Y coordinates increase from top to bottom; X coordinates increase from left to right
WinXPos	512	

On a 1024x768 monitor, the above setting would put the user screen in the top right quarter. Another option is to use the “Extend Desktop” facility of Windows to run BCI2000 on a dual monitor system. If the main monitor has a resolution of 1280x1024 and the 2nd monitor has a resolution of 800x600 (positioned to the right of the main one, the top edges lined up) the appropriate settings are:

WinHeight	600
WinWidth	800
WinYPos	0
WinXPos	1280

Setting Up the Mu Pipe

You can start the Operator, and then run each of the previous module, but you can obtain the same results if you create a bat file like the following

```
start operator\operat.exe
wait 0
start EEGsource\RDAClient\RDAClient.exe AUTOSTART 127.0.0.1
start SignalProcessing\AR\ARSignalProcessing.exe AUTOSTART
127.0.0.1
start Application\D2Box\D2Box.exe AUTOSTART 127.0.0.1
```

Save this file to your root BCI2000 folder (i.e. `C:\BCI2000\`) and name it `P3tutorial.bat`. You can double-click it whenever you want to run the whole system. You can also create a link to this file and copy it to your desktop for your convenience

Back to to setup

Mu Rhythm Analysis

Starting an analysis session

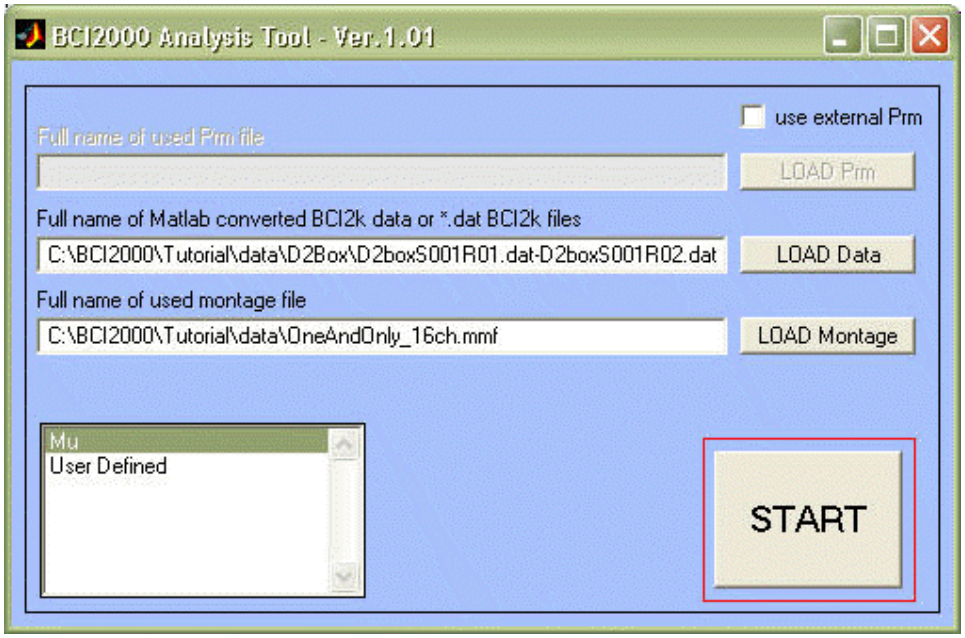
Every recording session has its own history. It is very difficult to say in advance what might happen during the recording. So we will assume that everything was perfect (the subject collaborated, no channel went lost, etc); later on we will consider the main causes of contamination, how to recognize it and what to do.

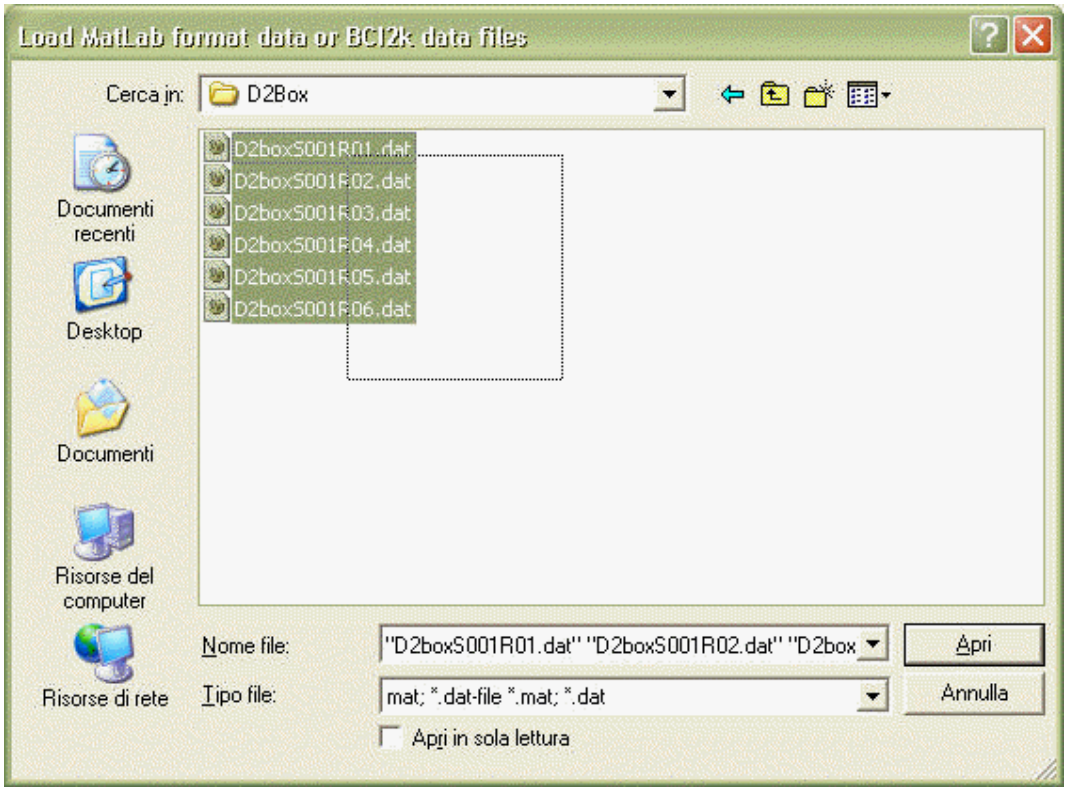
If the recording session run smoothly, in the folder `c:\bci2000\data\Mu\TestSubj` you should have now eight files named `TestSubjS001R01.dat` to `TestSubjS001R08.dat`, corresponding to the eight runs, respectively.

Before you start the analysis, you should locate on your hard disk the correct montage file. This is a file that describes the list of channels that were acquired. In this tutorial we will use `OneAndOnly_16ch.eloc.txt`, which is located in the folder `C:\BCI2000\Tutorial\Montages\`. If you need to edit the `*.mmf` file, check its data format.

Opening files with Mario

After having run `mario.exe`, press LOAD Data. A dialog window will ask you to select the dat files. You can either select a single file, or select multiple files. This time, we will select all files in the dataset.





As an optional (but recommended) operation, push the LOAD MONTAGE button and choose the mmf file. This will allow you to see the labels of each channel the following steps, and to allow to apply spatial filters that depend on the electrode position.

At this point, you can check that Mu is selected in the bottom left listbox (indicating that a Mu dataset has been recognized) and pushing the START button will open the Mu parameters dialog.

Run computations

The Filt Params window will also appear automatically. It reports the settings of the parametric (autoregressive) spectral estimation stage. Most of the values are set after the corresponding values used on-line (see BCI2000 setup for Mu). With settings as the ones reported in figure, the analysis software will:

- take a 1 s long epoch of data
- remove the mean value
- Identify an auroregressive model of order 16
- compute the analytic expression of the spectrum
- sample the spectrum at points starting from 0 Hz to 60 Hz every 0.2 Hz.
- average these values into 2 Hz wide bins

You can modify these settings according to your aims. Changing the model order, for instance, brings sometimes to interesting results. Remember though that in the online version, spectral estimation is performed in epochs as short as 200 ms (i.e. 40 samples, at 200 Hz sampling rate); thus, you might use epochs of similar length and you should also avoid using model order higher than half the samples available in the epoch.
When you are done, remember to close the Filt Params window, otherwise the program will not allow you to launch the analysis (i.e., the Go button will not be enabled).

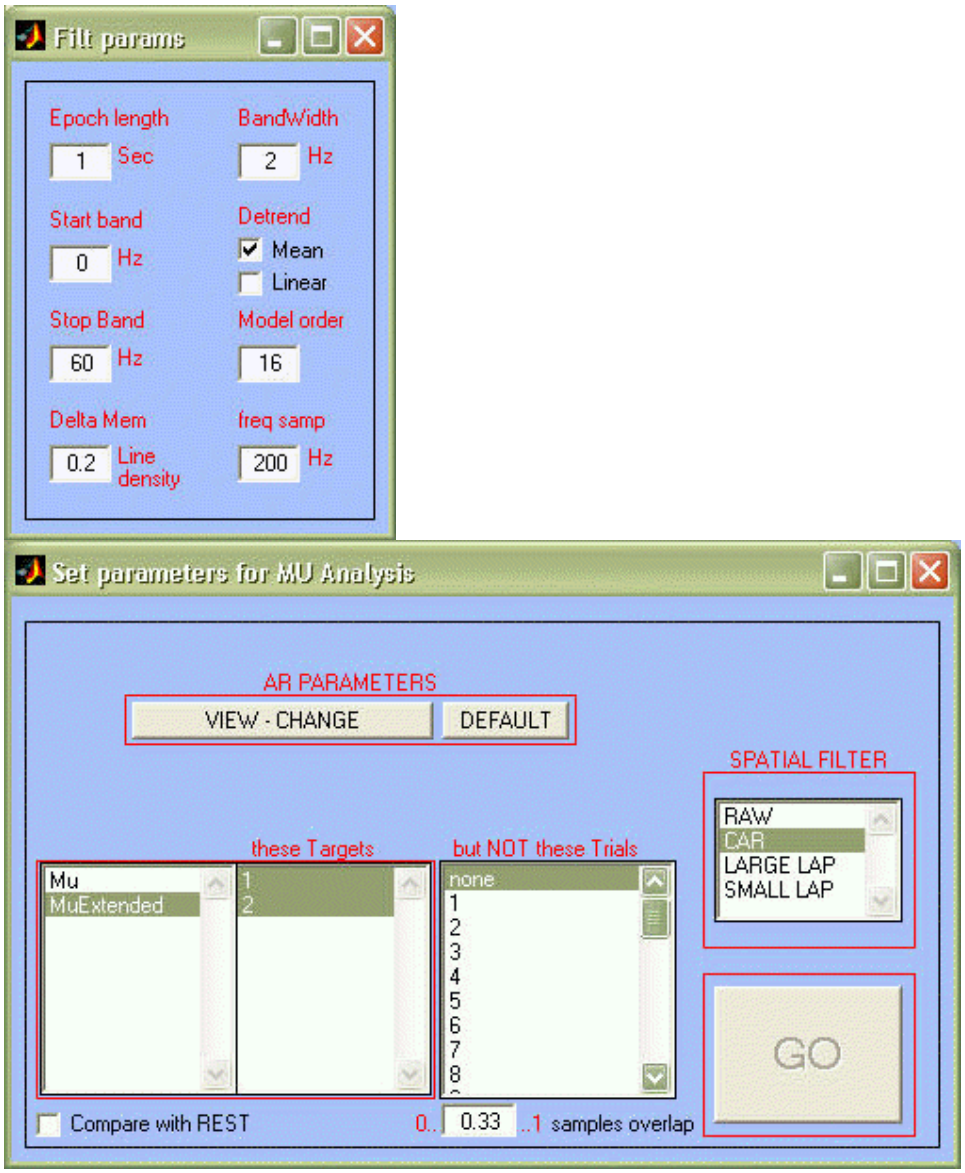
In the Mu Analysis Parameters window, we will now set the spatial filter that you want to apply. Common Average Reference (CAR) will be fine for most situations.
Buttons in the AR Parameters frame will bring up again the Filt Params window or revert the values to the original.

Now you have to specify:

- Which data (in each trial) you want to take into account. Mu will only use the part of the trial when the cursor is visible, while MuExtended will take into account all the period when the target is visible. To learn more, see section D2Box Application States
- The conditions you want to contrast. In this specific experiment, only two targets were present, so you can only compare one to the other. As a extension, you may compare each of the two with the rest condition (defined as the EEG segments acquired when no target was visible).
- If you have made an artifact rejection (or noted on the run sheet during the acquisition which trials contain artifacts), you can instruct the software not to use them.
- Finally, you must decide how the software must extract EEG epochs from the continuous data. Usually, the epochs are partially overlapped. This

attenuates the data loss in case the length of a trial is not a multiple of the length of an epoch.

At this point you can start the analysis by pressing the GO button.

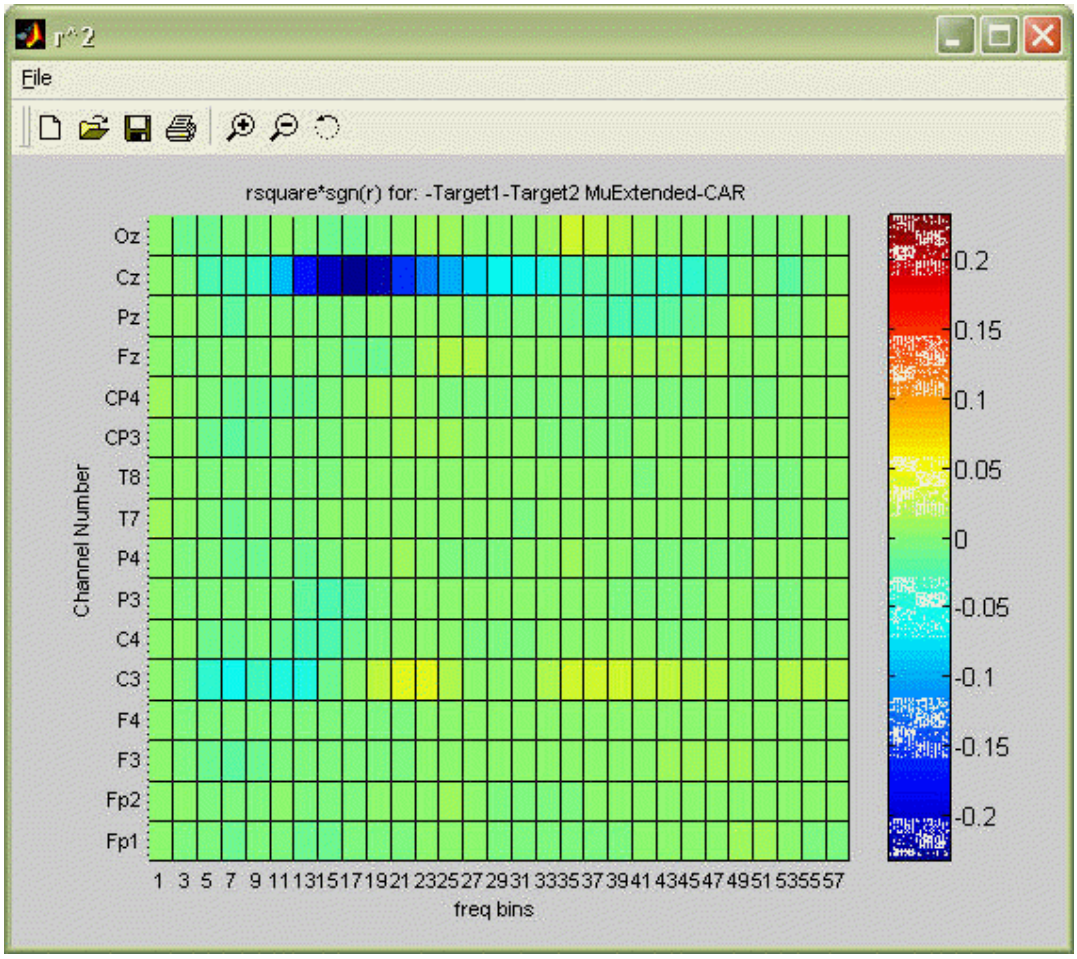


View results

The first figure that appears is the R-square matrix (channels x frequency bins)

R-square matrix

The r-square matrix highlights the most relevant spectral features for the separation of the two classes of EEG - Cursor Up (target 1) and Cursor Down (target 2). In other words, cells that are either blue or red change between the two experimental conditions and thus are potentially useful for feedback control.



Each row of the matrix is related to a single channel, while a column represents a frequency bin (labeled with its central frequency). The color codes the statistical significance of the difference between the two kind of evoked potentials.

Thus a red color would mean that, for that channel and for that frequency, the "Up" EEG is significantly synchronized with respect to "Down" EEG.

Clicking on a cell of the matrix will open two more windows:

- Power spectral distribution of the selected channel
- Topographic maps at the selected frequency

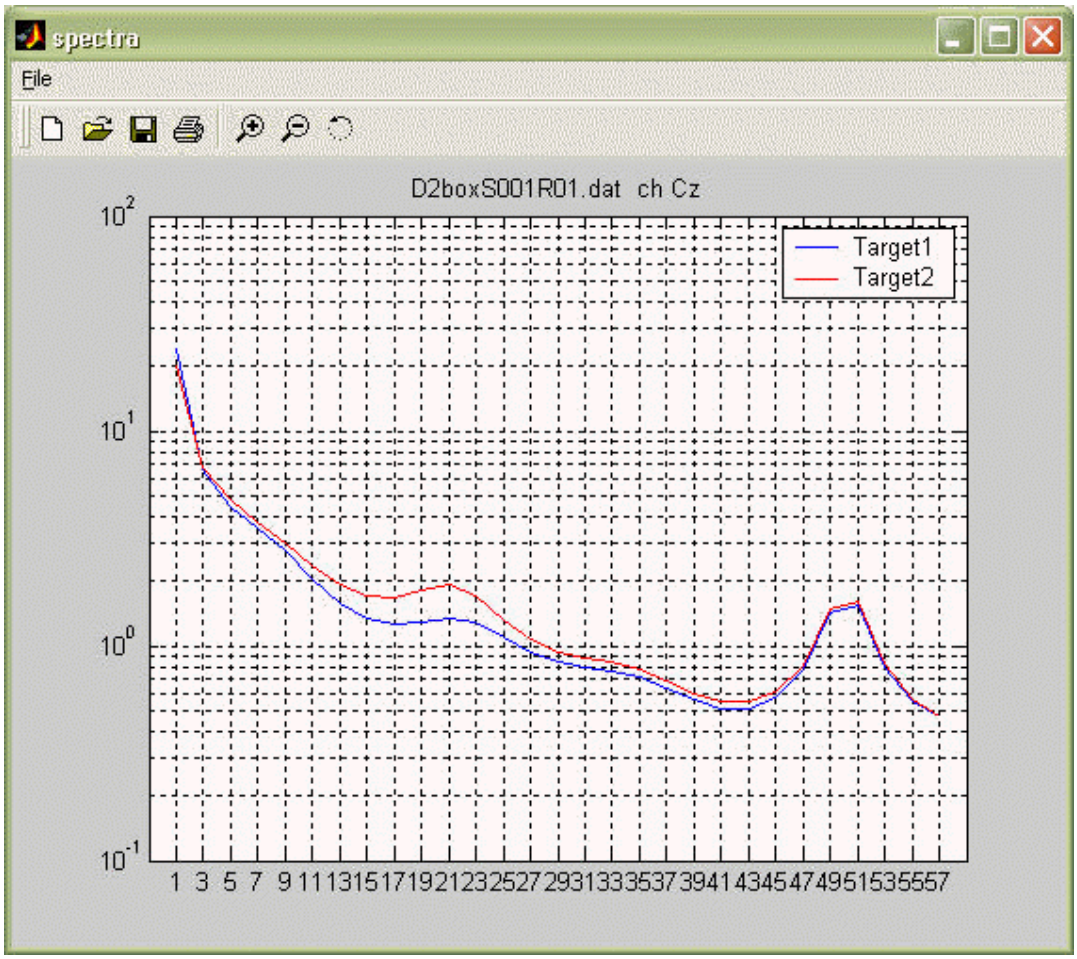
You can choose whether to overwrite or to put the most recently evoked figures beside the previous one. Two sets of waveform/topography figures are available, and are linked to the click of the left or right mouse button.

Power Spectra

Here you can find the spectral density of power for both classes of EEG. Blue line refers to "Up" condition, while Red line to "Down".
We can see that at the vertex, the "Down" condition is generally more

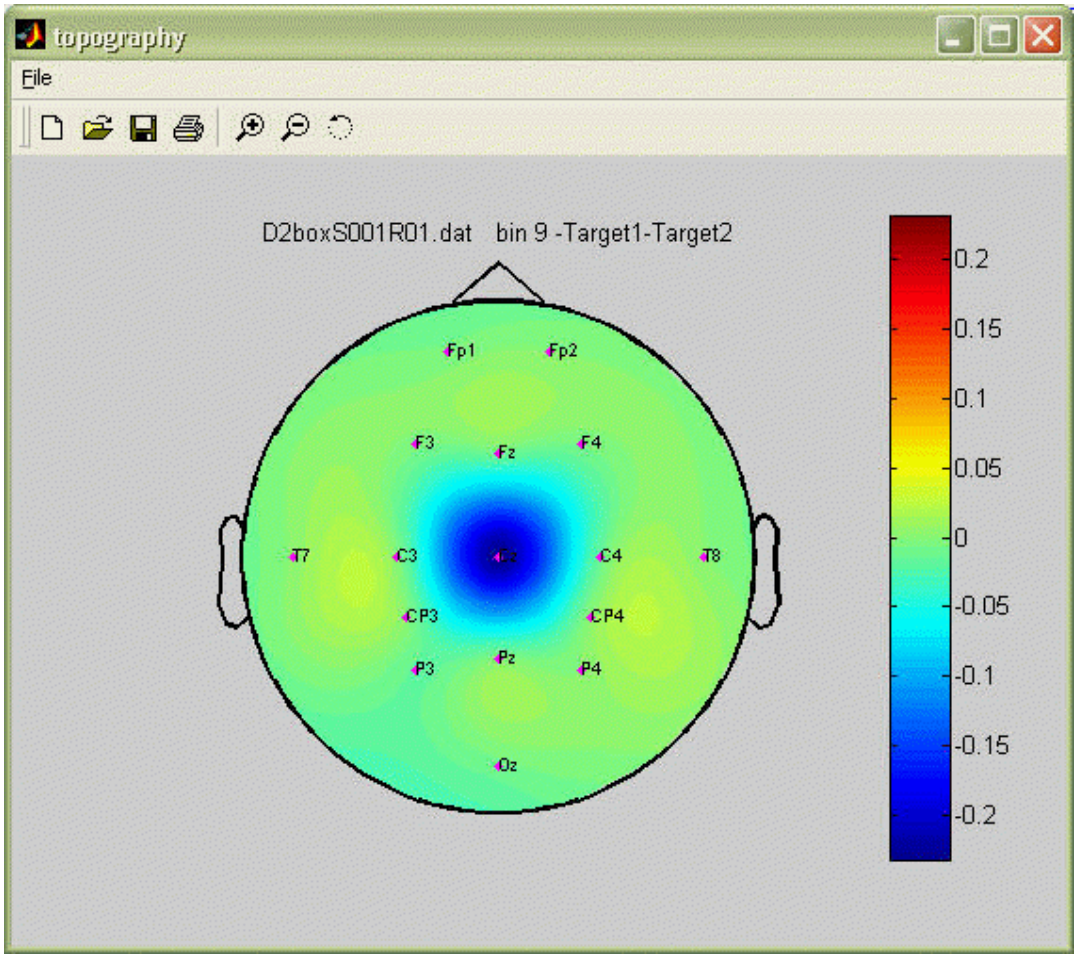
synchronized than the "Up", with maximal difference in the beta (i.e., 18-24 Hz) band, consistently with what is shown by the R-square matrix. The correspondence between spectral differences and R-square is not perfect since r-square is sensitive to the dispersion (variance) of single trials; thus, small differences between spectra could bring to a high r-square if they were very reproducible. This is why we use r-square to determine the best locations and frequencies for control. Simple difference measure would be biased to (typically very low) frequencies that have high amplitudes.

The peak around 50 Hz is due to mains disturbance; this also allows toestimate the leakage introduced by the spectral estimation.



Topographic plots

This figure represents the scalp distribution of the r-square.



The color coding is the same as the r-square matrix figure, and values on each channel are interpolated to create a continuous bidimensional map. The more electrodes electrodes, the more accurate is the map. The number of channels shown on the scalp may not coincide with the whole number of electrodes in case of a Surface Laplacian spatial filtering, since in that case the value of the laplacian is not computed on the border channels.

Interpretation of the results

The first step in the preliminary analysis of mu rhythm BCI data is to find the absolute maximum of r-square. This simple statement must be mediated with a proper knowledge of physiological phenomena. In fact, we do not expect any sensorimotor activity at 3 Hz, so if the absolute maximum is at that frequency, you must suspect that it is actually an artifact. The same holds if the spatial localization of the peak is far away from the centro-parietal electrodes.

From a practical point of view, it is not important now to understand the physiological basis for the desynchronization peak at Cz and 17 Hz, as far as we are confident that this component is stable enough to be used during the next session to control the cursor.

The absolute value of R-square is mathematically bound to lie in the interval between 0 and 1. Values above 0.4 should produce highly accurate control in a 2-target scenario. Values around 0.1 are promising. Values below 0.03 are possibly due to random fluctuations. All the previous figures are referred to analyses made on 240 trials.

Beware of artifacts!

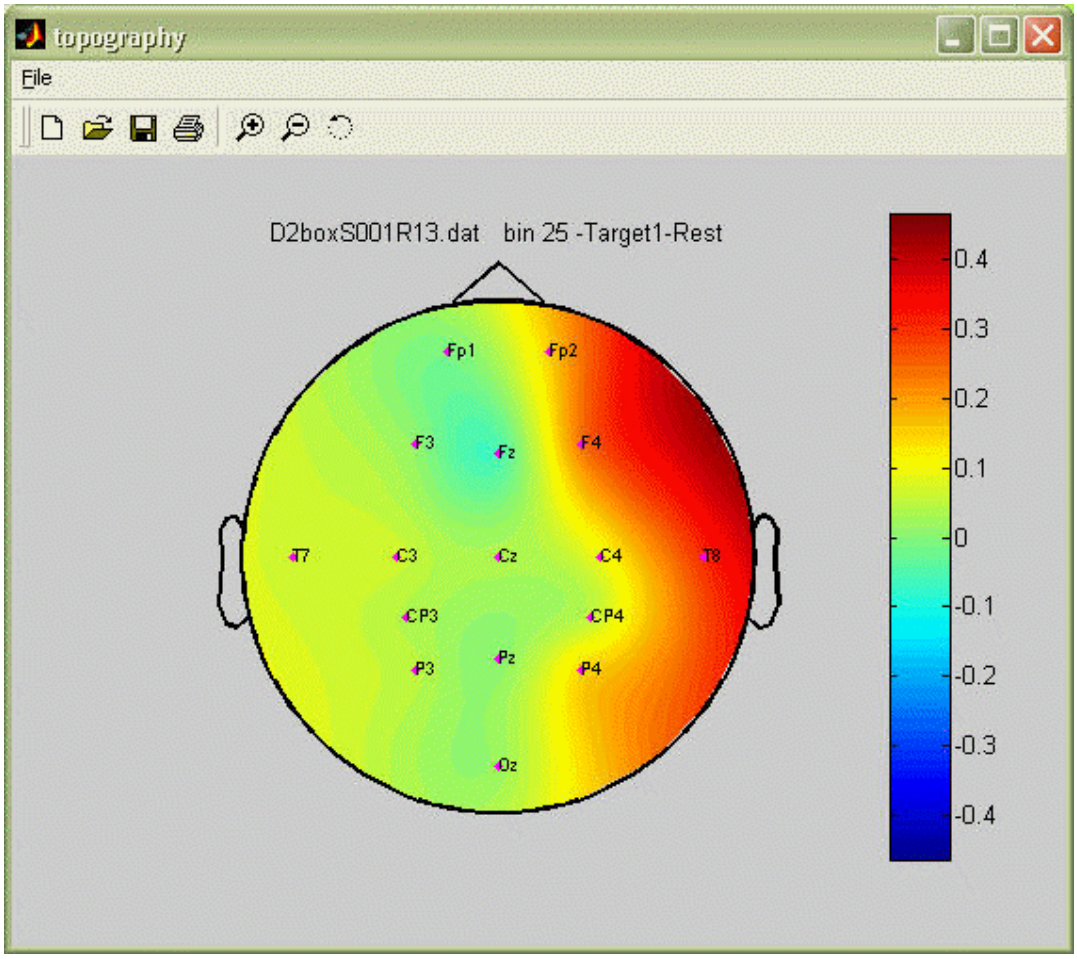
During the acquisition, you should be particularly careful to avoid artifacts. While EOG and blink artifacts are confined to very low frequencies, EMG artifacts are mostly on the beta band and could largely overlap with the frequency band that characterizes the mu rhythm.

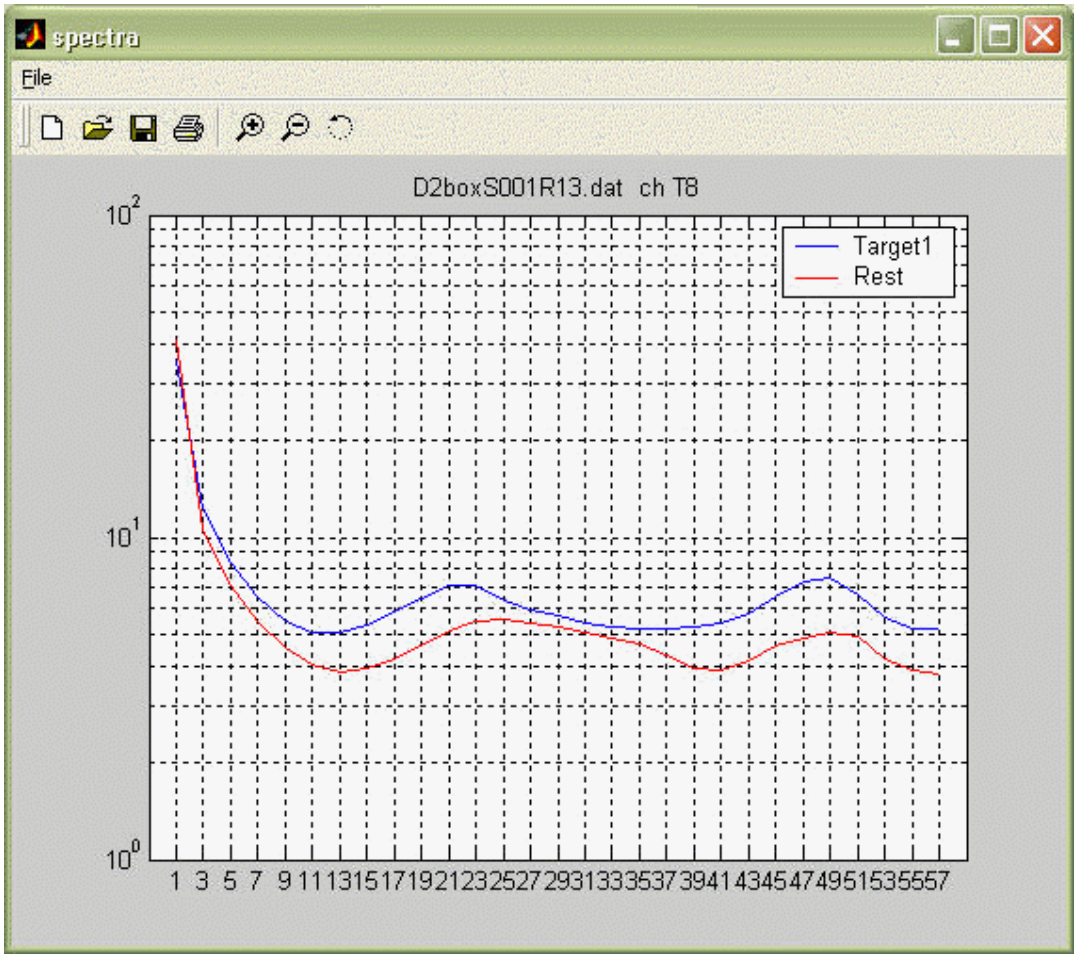
If contaminated data was acquired, then you have two problems:

- Realize that EMG is superimposed to data
- Distinguish between spectral modulation introduced by EEG and by EMG.

The first task is not as trivial as it might appear. If you did not acquire the data yourself (or even if you did it) you might complete the analysis procedure without giving a single glance to the raw data. So you must be particularly careful when you analyze the R-square maps, and always wonder whether the peak you are seeing might be due to an artifact. If this analysis is not done carefully, you might have a subject control the system with muscle artifacts instead of with brain activity. Because the value of BCI systems lies in the independence of muscle control, such experiments would be of very little, if any, value.

Discriminant features between EEG and EMG are both in the frequency distribution and in the spatial distribution.


topography
File
D2boxS001R13.dat bin 25 -Target1-Rest
Fp1
Fp2
F3
Fz
F4
T7
C3
Cz
C4
T8
CP3
CP4
P3
Pz
P4
Oz
0.4
0.3
0.2
0.1
0
-0.1
-0.2
-0.3
-0.4



EMG that strongly affects the recordings is mostly generated by muscles at the forehead or close to the ears/jaws. Thus, their spatial distribution is such that the most responsive channel is one of those at the frontal or parietal edges of the montage. Of course, due to volume conduction, the effects can be seen even on electrodes on the opposite part of the montage, but they show an attenuating pattern.
On the other hand, a peak that shows its maximum on a central channel can hardly be generated by a muscle or by other non-cephalic source. It could, however, be EEG activity that corresponds to actual, rather than imagined, movements. Again, careful analyses of the data, as well as careful monitoring of the subject to ensure that he/she does not use movements during the experiments to control the system, are essential.

The spectrum of EEG is mainly concentrated in the alpha band, with a possible flatter and lower peak in beta. With the exception of the beta peak/plateau, the EEG spectrum decreases almost linearly (if measured in dB) after 12 Hz. On the other hand, EMG spectrum becomes larger at about 20 Hz and is still very high at the highest frequencies we usually analyze (60 Hz). A spectrum with a pattern more similar to the latter should cause suspicion that non-EEG activity is present in the data.

Improving the analysis

The first cause of an unsuccessful analysis is poor recording quality. For this reason it is highly recommended that, at least for the first experiments, the subject is highly motivated and cooperative. If this is the case, data of low quality are usually present in one or a few runs. And if the experimenter is careful enough, he/she should have noted down the occurrence of strong artifacts on the run sheet.
In this hypothesis, the following step is to repeat an unsatisfactory data analysis after having excluded the (putative) bad runs.
To do this, just go back to the file selection dialog, choose LOAD Data again and load only the runs that you believe are clean. The number of runs should not be too low (as a rule of thumb, the data set should contain at least 100 trials, or 50 for each condition). Otherwise the r-square analysis might not be sufficiently statistically significant.

Analyzing the Screening

The screening experiment is not different from a regular training session, from the point of view of data format. You will have four sets (horizontal movement, horizontal imagination, vertical movement, and vertical imagination) of three runs, containing EEG acquired in two conditions (up and down, or left and right).

Start analyzing the vertical movement execution, and mark a few possible resonsive features. The analysis on vertical movement imagination should confirm (though possibly with a lower R-square) those that are actually related to sensorimotor cognitive states.

In the hypothesis that no reliable responsive EEG feature is found, the analysis can be repeated on the horizontal dataset.

If both the vertical and horizontal dataset show responsive features, and these are different, the subject should be trained for some session on the vertical training until he/she decreases the false positive below 10%; at that point the training of the horizontal modulation can begin, with the aim that at a certain point the subject can control both the vertical and the horizontal dimension simultaneously in a two-dimensional task.

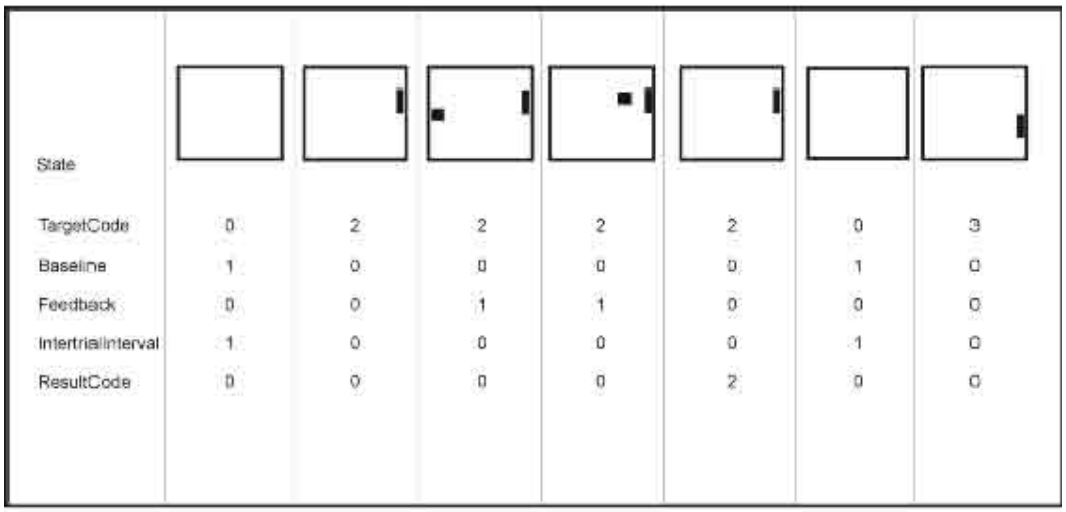
Conclusions

If you are confident that you have found a significant difference between conditions that is due to EEG rather than an artifact and that reflects a cognitive process that is likely to be reproduced (or even enhanced with training) in the next session, then you have reached your goal.
The next time the same subject practices with the D2Box Application, you will have to update the MUD matrix so that it reflects the feature that you just discovered.

D2Box Application States

The figure below is taken from the BCI2000 official documentation.

It describes how the system labels the EEG data using five different states, and the values for these states at every stage of the trial.



State	1	2	3	4	5	6	7	8
TargetCode	0	2	2	2	2	0	3	
Baseline	1	0	0	0	0	1	0	
Feedback	0	0	1	1	0	0	0	
Intertrialinterval	1	0	0	0	0	1	0	
ResultCode	0	0	0	0	2	0	0	

Four target rjb task. 1 is the intertrial interval were the screen is blank and baseline data can be collected (optional). In 2 the target is presented but the cursor is absent (no feedback). In 3 the cursor is present and it's vertical movement is controled by the user's EEG. Horizontal cursor movement is under computer control and is at a constant rate. In 4 the cursor approaches a the target and in 5 the target has been hit and trial-outcome feedback is presented. In 6 the screen is blank (intertrial interval again) and in 7 another target is presented.

Advanced Signal Processing

Christoph Guger
g.tec Guger Technologies

1st BCI2000 Workshop
The Desmond Conference Center
Albany, New York, USA
June 13+14, 2005

Christoph Guger

g.tec – medical engineering GmbH

8020 Graz, Herbersteinstrasse 60, Austria

Tel.: (43)-316-675106-0

Fax: (43)-316-675106-39

office@gtec.at, http://www.gtec.at

1st BCI2000 Workshop
The Desmond Conference Center
Albany, New York, USA
June 13+14, 2005

Advanced Signal Processing

USER MANUAL

organized by Gerwin Schalk
Wadsworth Center, NYSDOH

Advanced Signal Processing will discuss with hands-on exercises the following topics:

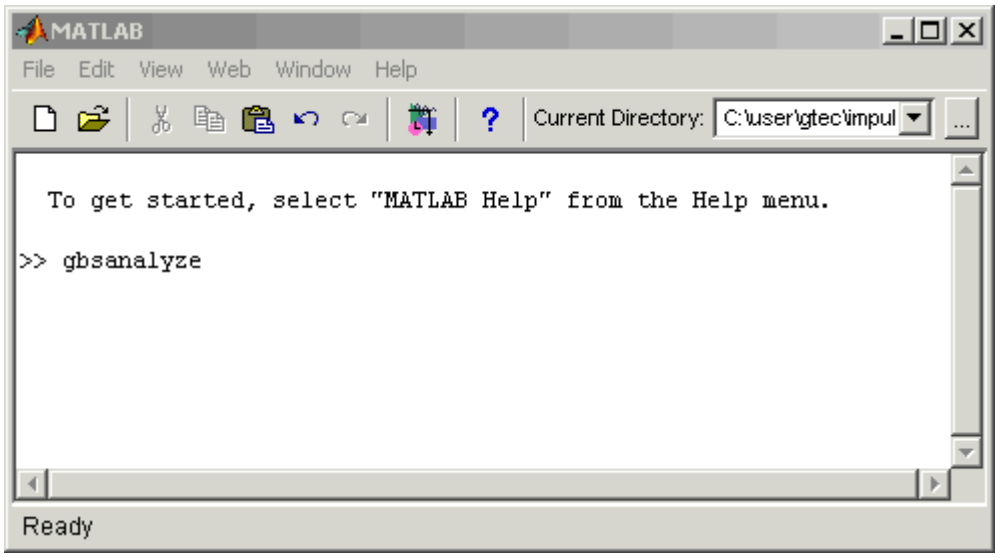
Running g.BSanalyze 3
Loading and Viewing Data 4
Common Spatial Patterns 5
Independent Component Analysis 16
Spatial Filter 20
DSLVQ Feature Weighting 27

Running g.BSanalyze

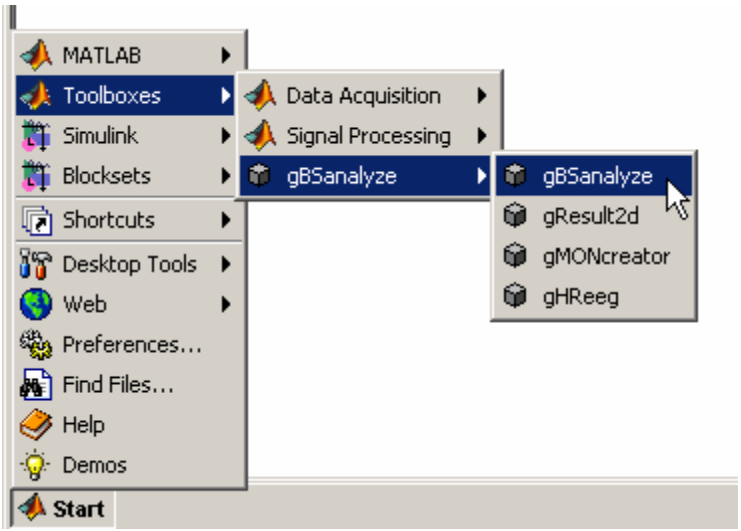
After starting MATLAB type:

```
gbsanalyze
```

in the MATLAB command line



or run the application from the MATLAB **Start** button:



g.BSanalyze starts with a blank data window.

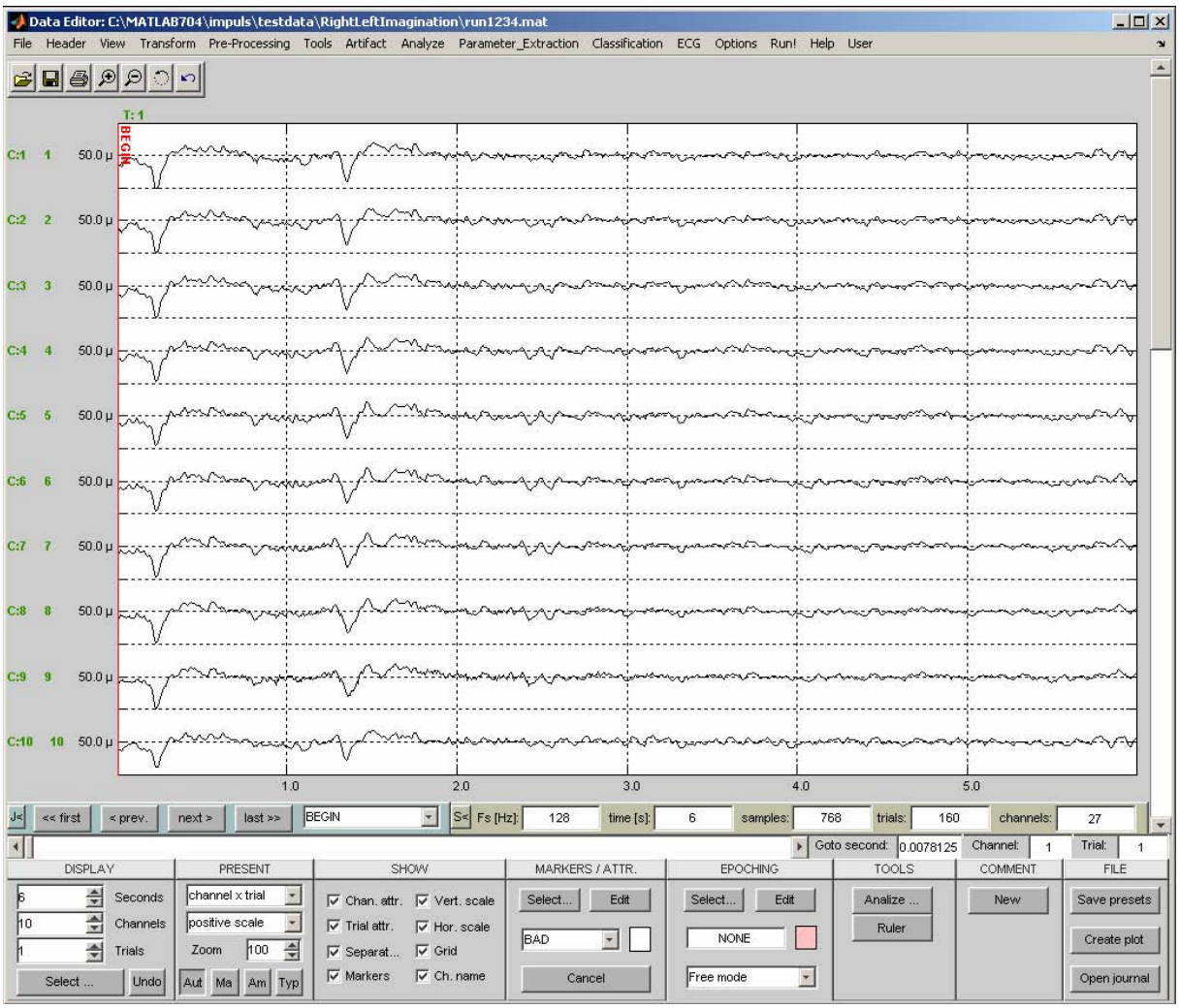
Loading and Viewing Data

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

   ```
   Your MATLAB path:\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.

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



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 **X** of class 1 and class 2, the CSP method gives an NxN projection matrix **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 **W** the decomposition of a trial **X** is described by

$$\mathbf{Z}=\mathbf{W}\mathbf{X} \qquad (1)$$

This transformation projects the variance of **X** onto the rows of **Z** and results in N new time series. The columns of **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 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.

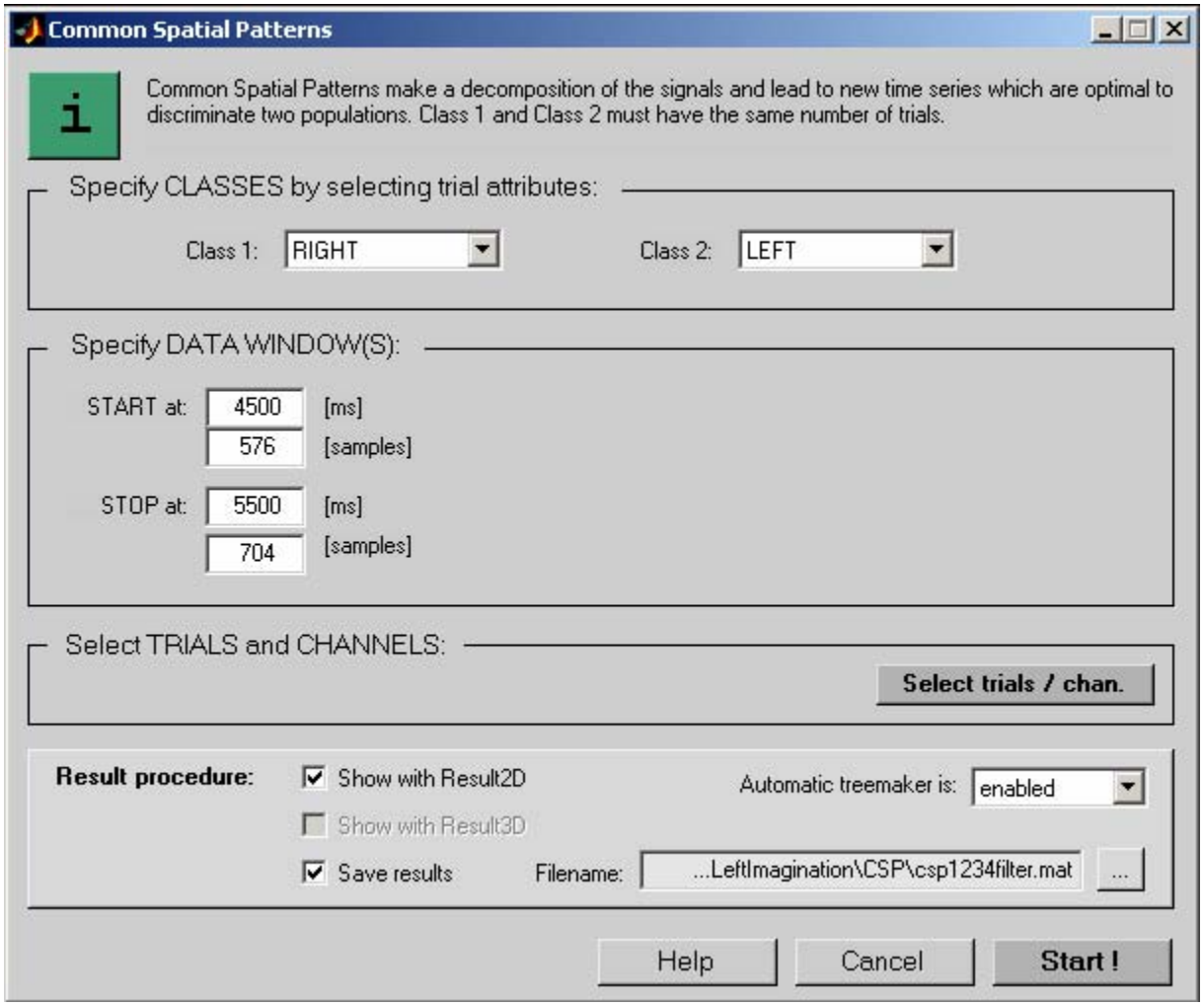
Z.J. Koles, "The quantitative extraction and topographic mapping of the abnormal components in the clinical EEG," Electroenceph. Clin. Neurophysiol., vol. 79, pp. 440-447, 1991.

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

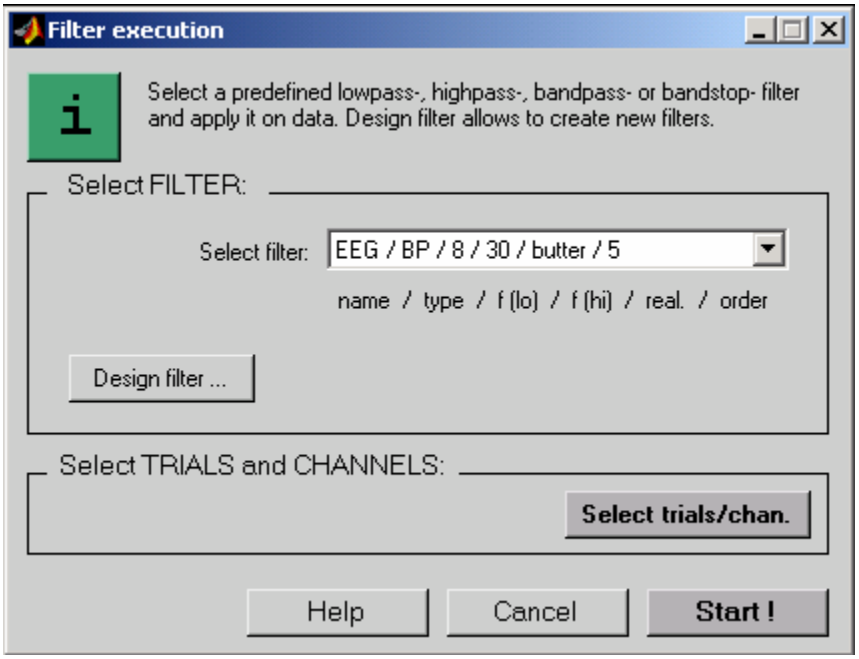


1. Select **Load Class Information** from the **File** menu to assign the class attributes stored in `class160.mat`. Assign the name `RIGHT` and `LEFT` to the trials.

2. Open the **Geometry** window under **Header** and **Browse** for the file `27ch1.mat` under `Your MATLAB path:\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. 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.



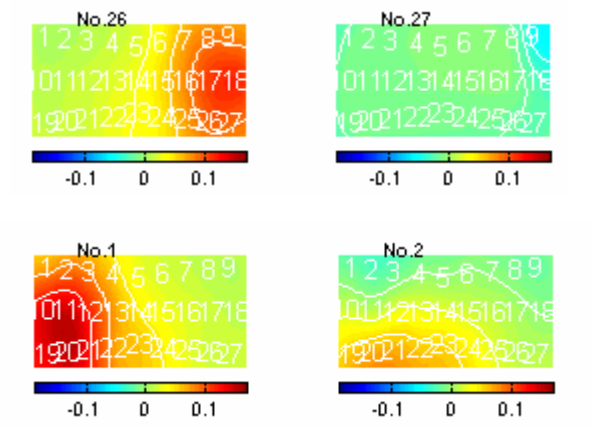
4. Click **Start** to filter your EEG data

5. 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.

6. Check **Open Result2D** to view the common spatial patterns

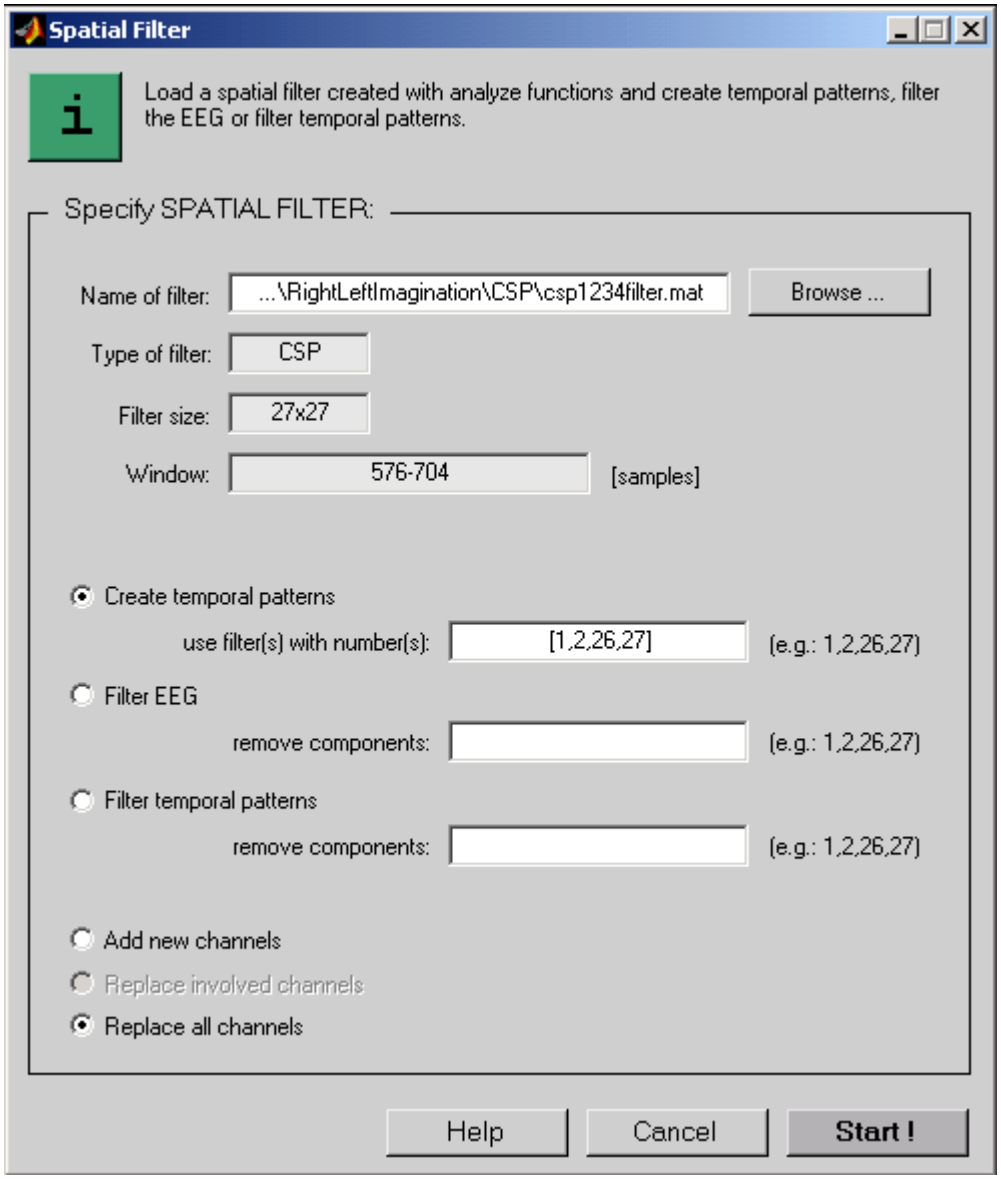
7. Click **Save results** and enter a path and filename to save the maps. The **Automatic treemaker** creates the `CSP` subdirectory.

8. 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.

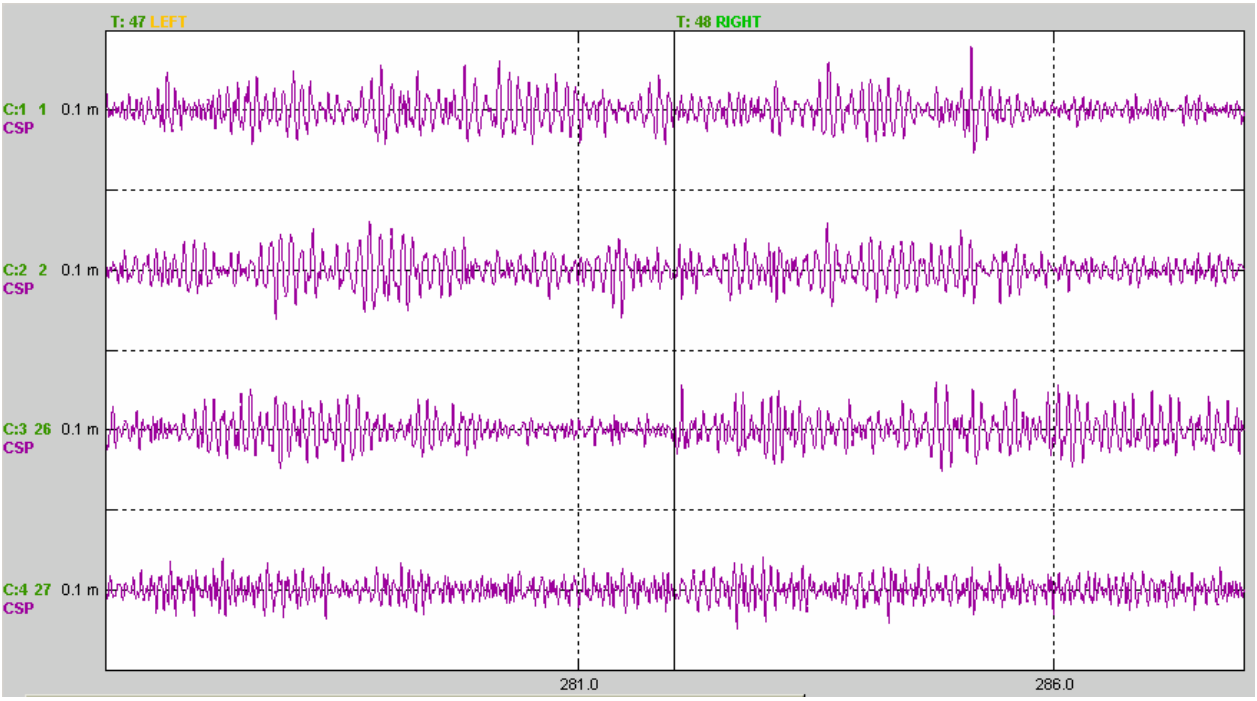


9. Open **Spatial Filter** under the menu **Pre-Processing** and load the created CSP filter by using the **Browse** button

10. Check **Create temporal patterns** and enter the 4 most important filters: 1, 2, 26 and 27

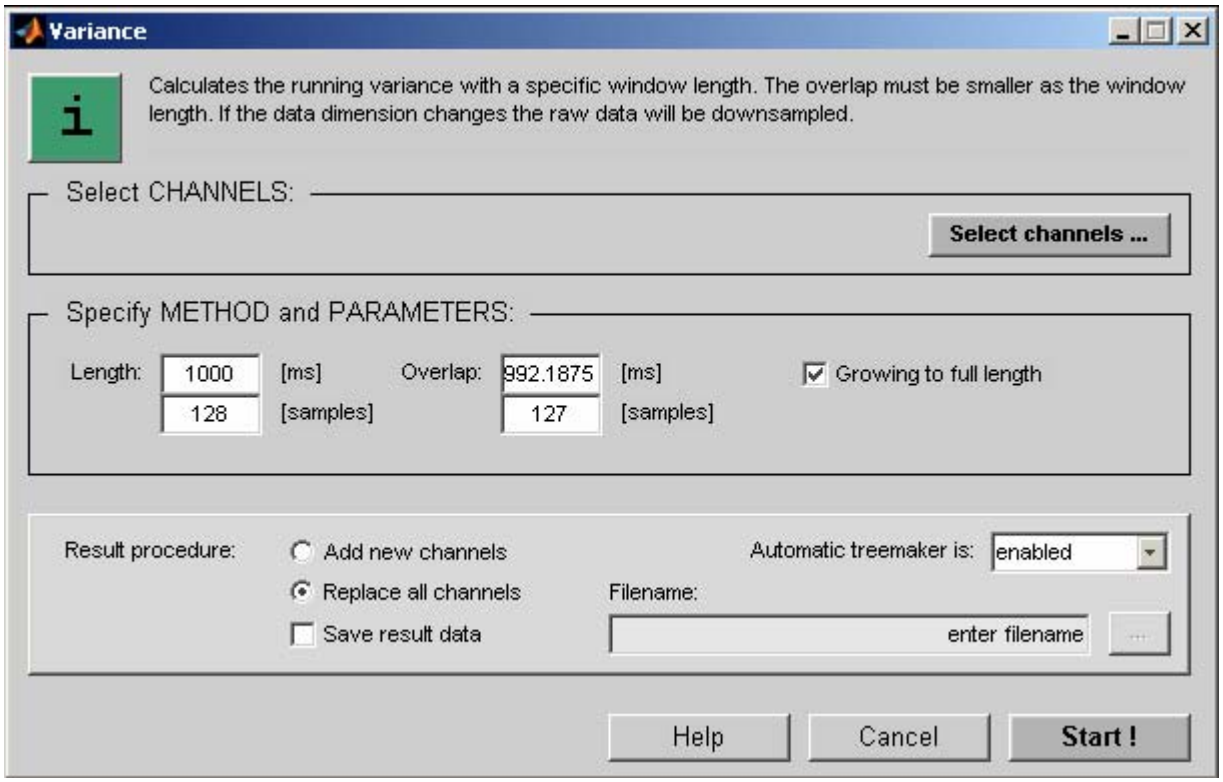


11. Press **Start** to convert the EEG data in temporal patterns. The figure below shows the resulting time series after filtering with the two most important (1, 27) and two second most important (2, 26) common spatial patterns according to (1).

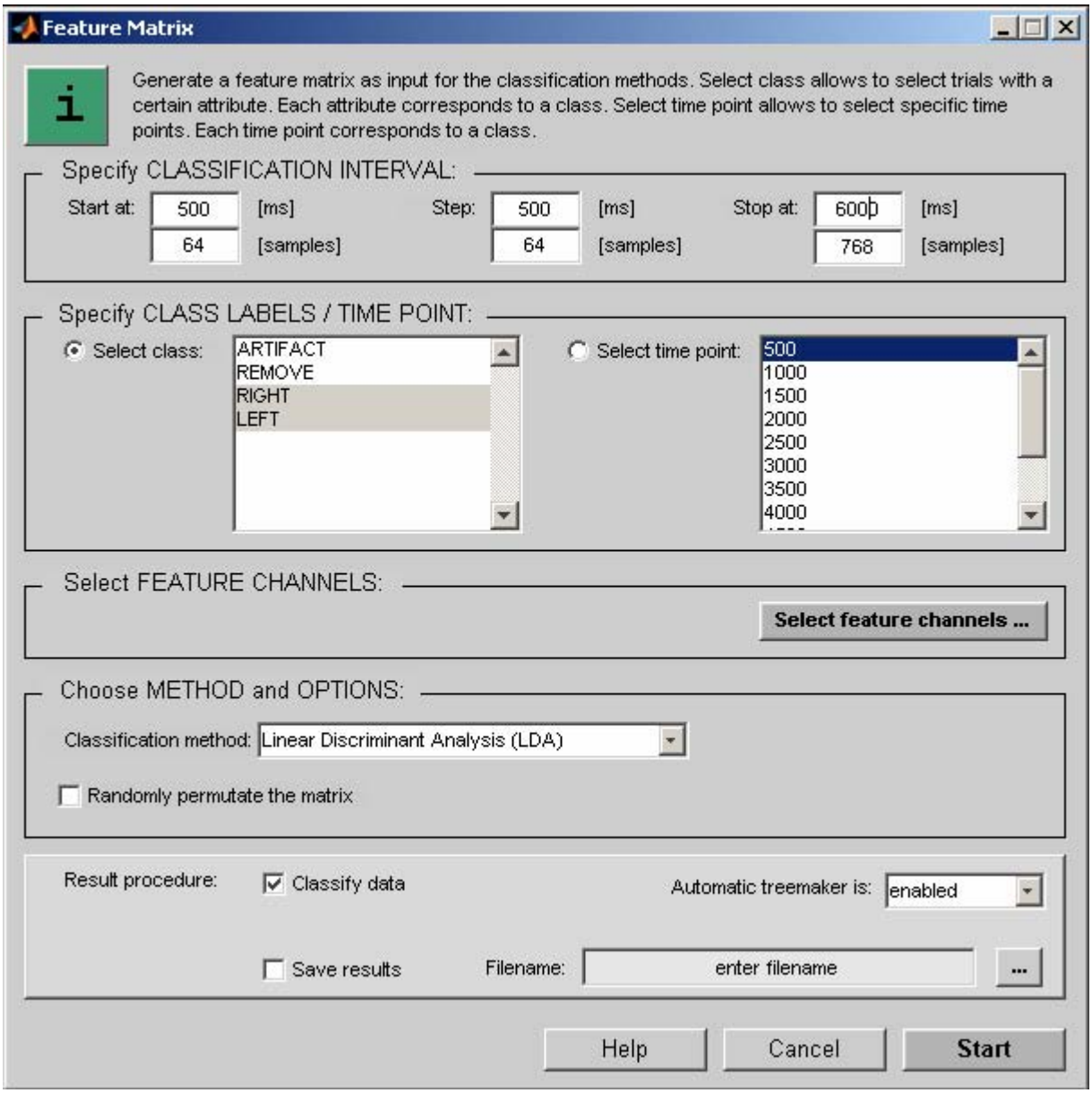


The filter was constructed in such a way that the variance in filter 1 and 2 will be maximized during left-hand hand movement imagination and minimized in filter 26 and 27. The left column shows the new time series of a left trial, the right column of a right trial. By construction in the most discriminating 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.

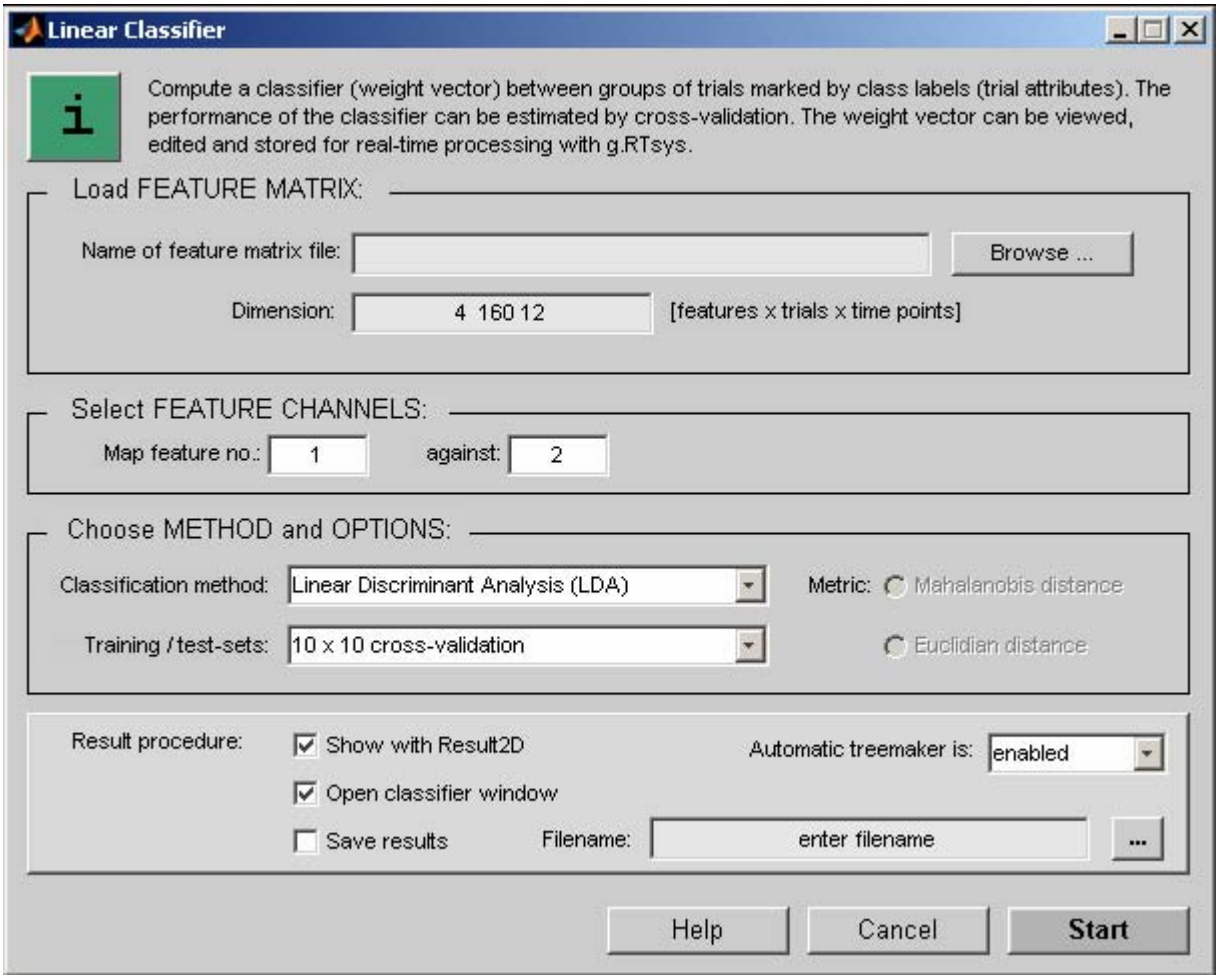
12. Then the variance of the time series is calculated. Therefore, open **Variance** under **Parameter Extraction** and select a time window of 1 second with an overlap of 127 samples. **Start** the calculation.



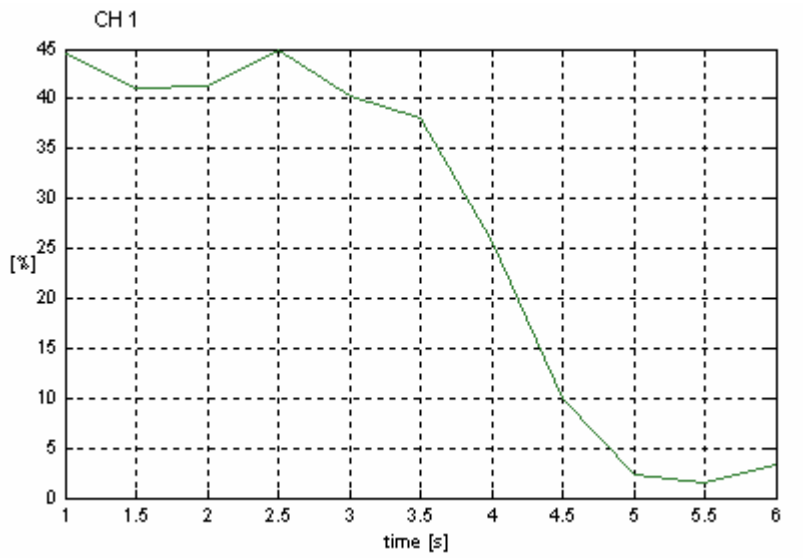
13. The classification accuracy is calculated with a 10 times 10 fold cross-validation of a linear discriminant for `500` ms steps. Therefore open window **Feature Matrix** from the **Classification** menu. Perform the settings as shown in the window below and press **Start**.



14. The **Linear Classifier** window opens. To train a linear classifier on the data select `Train 100 % - Test 100 %` from the **Training / test-sets** pull down menu. Press **Start**.



gResult2d opens automatically with the classification error. 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 MATLAB Editor show the classification result and weight vectors for each time point. Select the best classification accuracy time point at second 5.5 to investigate the weight vector. Classifier 1 is the bias value, Classifier 2 the weight vector for the variance features.

```
Second/Sample: 5.500 704.000
Classifier 1
 1.724 |
Classifier 2
-0.021 |
-0.011 |
 0.021 |
-0.001 |
```

This weight vector can be used in BCI2000 for the on-line classification of the features (see next page).

Perform the following steps to extract the CSP filter:

1. To load the CSP, type into the command window

   `filter=load(spf);`

   and browse for the `CSP.mat` file in the following directory:

   ```
   Your MATLAB path:\gtec\gBSanalyze\testdata\...
   RightLeftImagination\CSP
   ```

2. Extract filters 1, 2, 26 and 27:

```
F=get(filter,'D.W');
filter1 = W(1,:)
filter2 = W(2,:)
filter26 = W(26,:)
filter27 = W(27,:)
```

Each command returns a vector with 27 elements which are used for the on-line weighting of the electrodes.

Using these results for online processing in BCI2000:

This signal processing procedure yields a spatial filter that extracts time series that optimally separate the two classes (see above). It also produces classification weights that are used on the variance of these time series (filtered between 8 and 30 Hz). The output of this classifier will be small for one class (e.g., left hand imagery), and large for the other (e.g., right hand imagery).

Using the spatial filtering matrix:

Transmit all channels from EEGSource to BCI2000 SignalProcessing: Set *TransmitCh* to 27 and *TransmitChList* to the numbers 1-27, delimited by spaces. Then, change the size of the *SpatialFilterKernal* matrix to 4 rows (four output time series) and 27 columns (27 input time series). Then, enter the F matrix into *SpatialFilterKernal*. Finally, set the value of *SpatialFilteredChannels* to 4 as well. This way, frequency analysis will be calculated on the 4 spatially filtered channels. (See the BCI2000 User Manual for more information.)

Configure frequency analysis:

Configure frequency analysis to calculate the band power between 8 and 30 Hz: Set *StartMem* to 8, *StopMem* to anything larger than 30, and *MemBandWidth* to 22. The output of this filter then is a 4x1 matrix of frequency estimates (i.e., one for each spatially filtered channel).

Classification:

As discussed on the previous page, the weights associated with the four channels are: -0.021, -0.011, 0.021, and 0.001. These weights need to be programmed into the BCI2000 *MUD* matrix as follows:

1	1	-0.021
2	1	-0.011
3	1	0.021
4	1	0.001

The first column determines the channel, the second the frequency bin (in this configuration, we only have one from 8-30 Hz), and the third column determines the weight. Using this configuration, the user could control a computer cursor in one dimension with right and left hand imagery.

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

$$\mathbf{X}=\mathbf{AS} \qquad (1)$$

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

$$\mathbf{Y}=\mathbf{BX} \qquad (2)$$

Several implementations for ICA exists. 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:

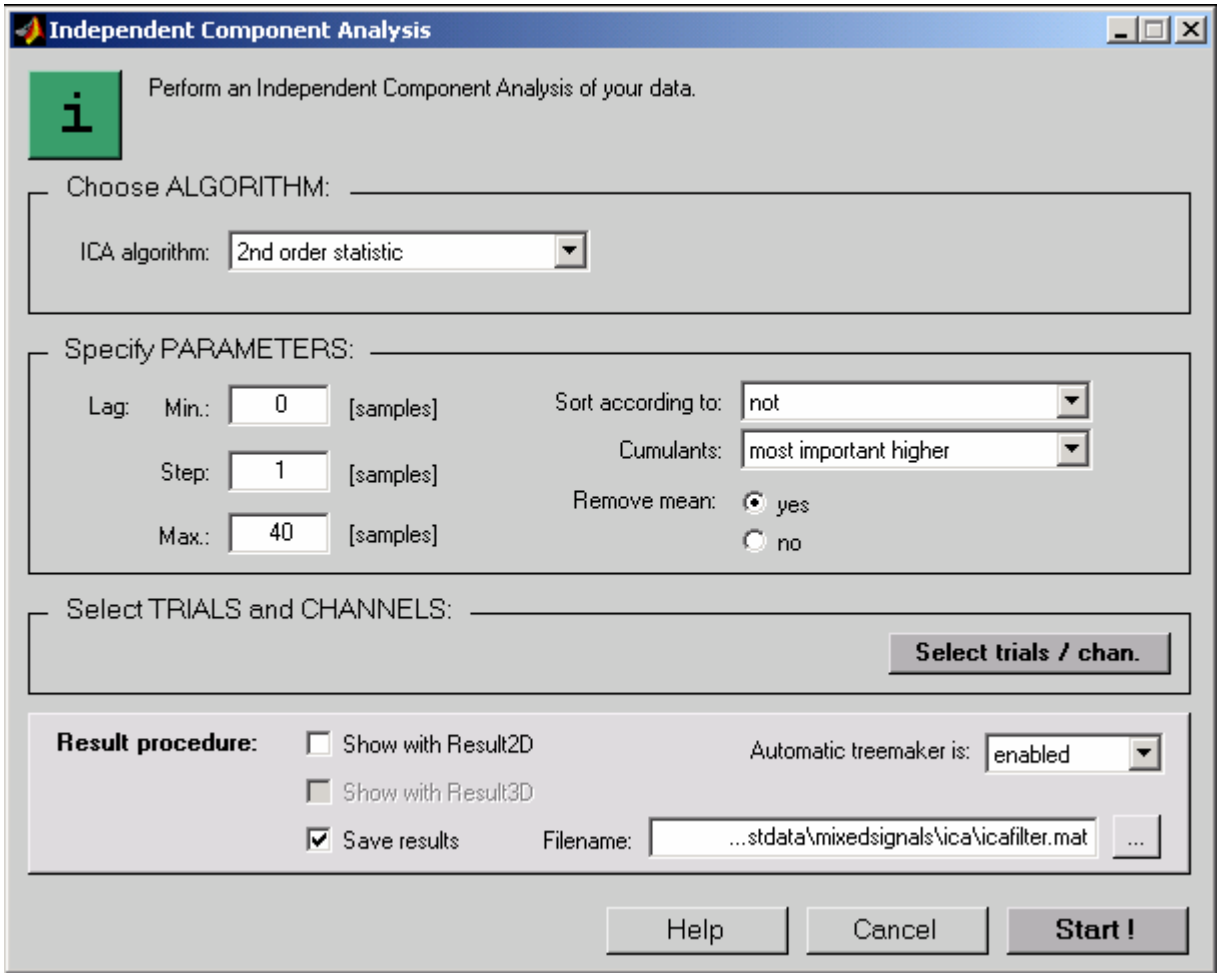
- **ICA algorithm** - use 2nd order statistic or higher order statistic
- **Lag –** time lags used for time-delayed correlations
  min - minimum time lag
  step - lag step size
  max - maximum time lag
- **Sort according to**

  not - do not sort the components
  column norm of mixing matrix

- **cumulants**

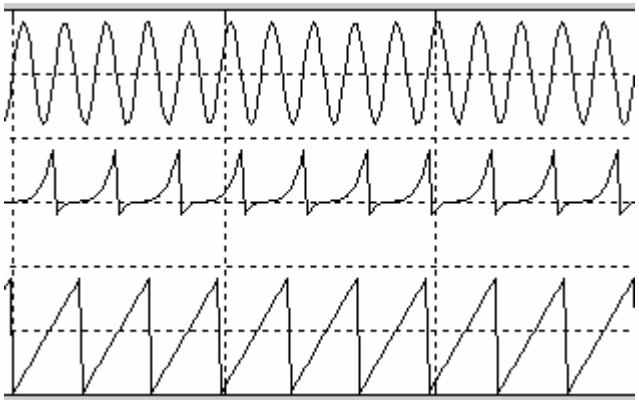
most important higher – use only the most important cumulants
n(n+1)/2 – use n(n+1)/2 cumulants with n is the number of channels

- **remove mean -** remove mean of each channel before calculation



1. Load the data-set `signal1.mat` from the directory

`Your MATLAB path:\gtec\gBSanalyze\testdata\mixedsignals`



Channels 1, 2 and 3 are artificial signals created with

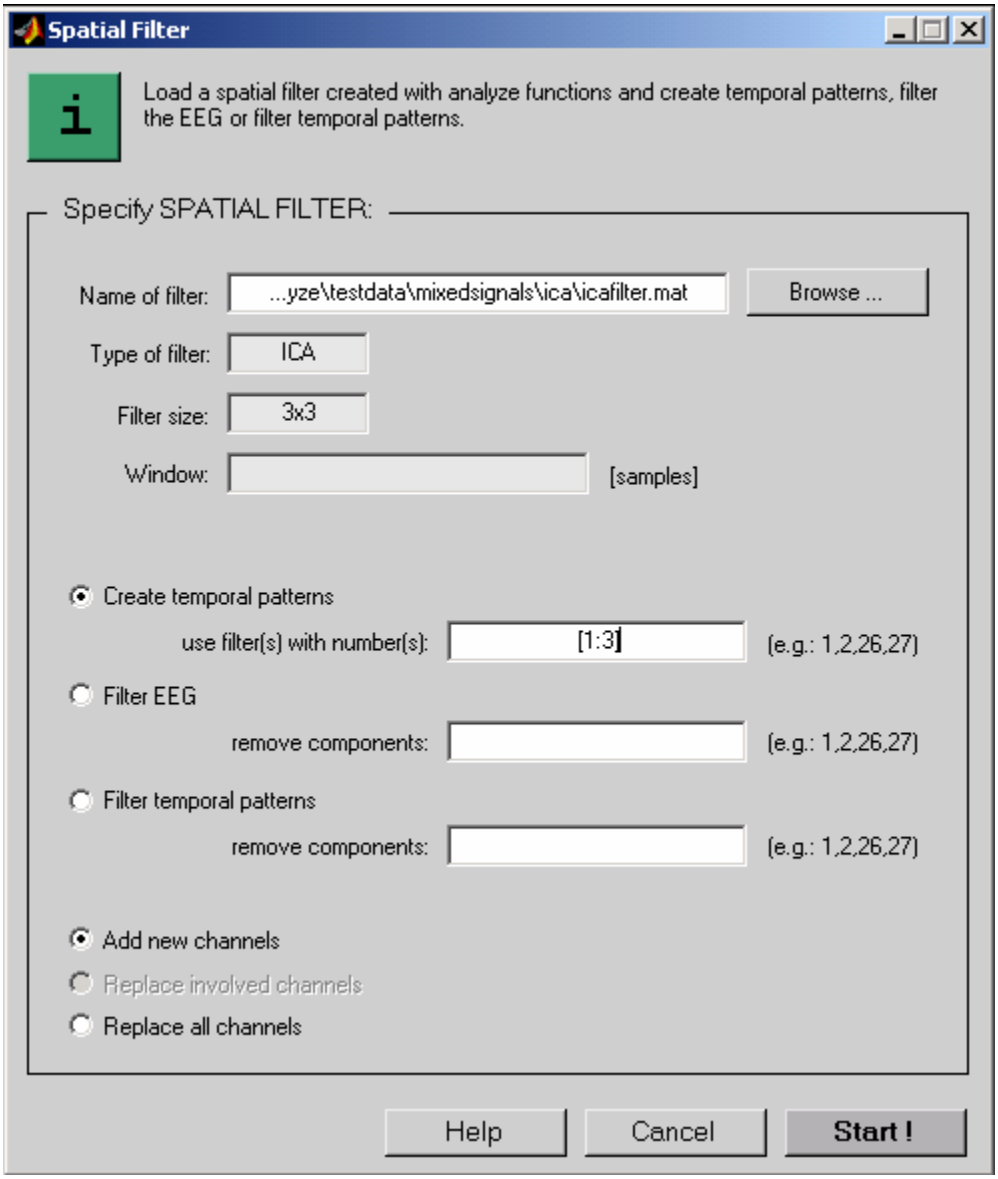
```
T=450;
sig(1,:) = sin(t/2);                    % sinusoid
sig(2,:) = ((rem(t,19)-7)/7).^3;        % a curve
sig(3,:) = ((rem(t,21)-11)/7);          % saw-tooth
```

2. To investigate a mixture of the 3 signals open `signal2.mat`

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.

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.

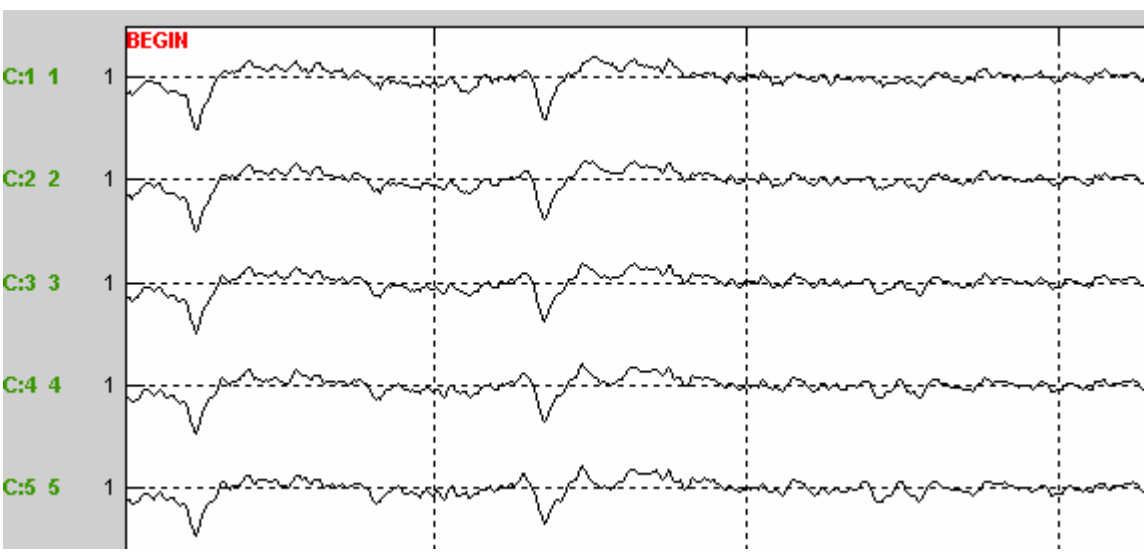
Method

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
   `Your MATLAB directory:\gtec\gBSanalyze\testdata\RightLeftImagination`

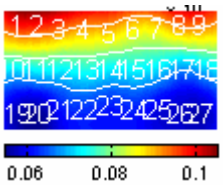


The figure shows a 3.2 second portion o the recorded EEG time series from 5 electrodes, all referred to the right ear. Two EOG artifacts within the first 2 seconds on all channels can be found.

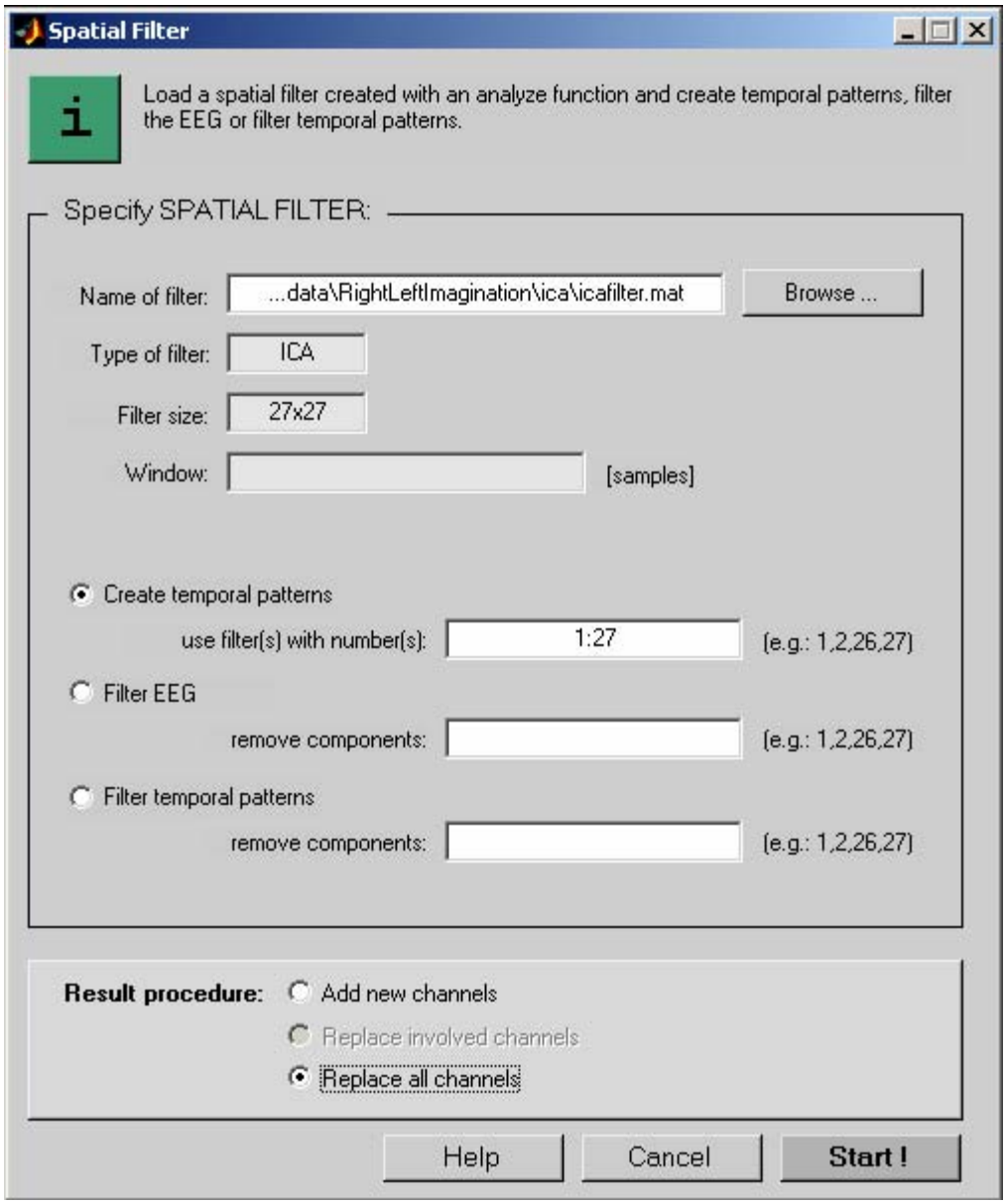
2. Click on **ICA** under the **Analyze** menu to perform an independent component analysis of the data. Store the calculated ICA filter under `icafilter.mat` in the directory

   ```
   Your MATLAB directory:\gtec\gBSanalyze\testdata\…
   RightLeftImagination\ica
   ```

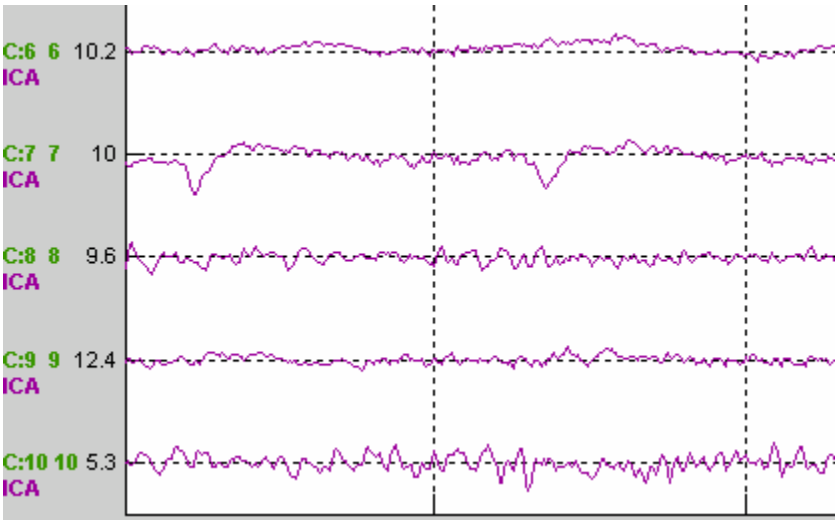
3. Start the calculation and Result2D opens automatically with 27 ICA filter projections. The columns of the inverse matrix W^{-1} give the projection strengths of the components on the scalp electrodes. Filter number 7 with strong frontal activity is presented:



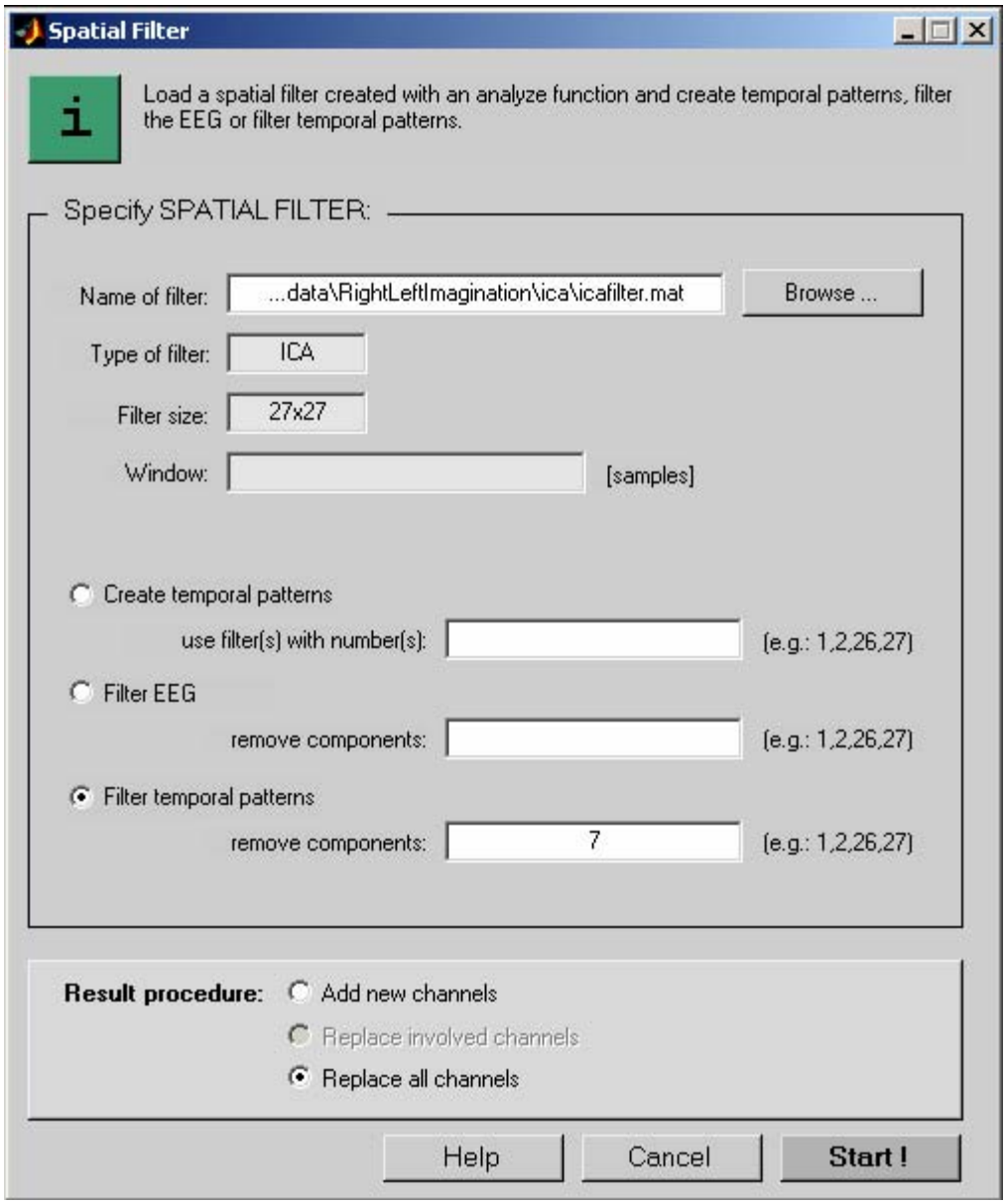
4. Click on **Spatial Filter** under **Pre-Processing** and load with the **Browse** button the created ICA filter



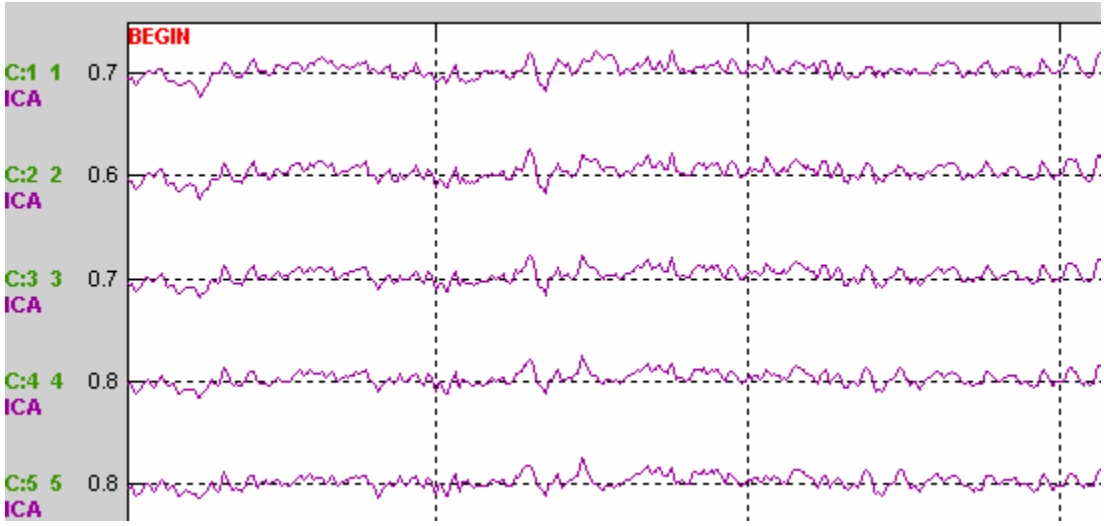
5. Select **Create temporal patterns** with filter number 1 to 27 to transform the EEG data into the ICA space. Click on **Start** to perform the operation. Each channel transformed to the ICA space is marked with the channel attribute ICA. The eye movement artifacts at second 0.2 and second 1.3 are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



6. 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.



7. 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.

1. To load the ICA filter, type into the command window

   ```
   filter=load(spf);
   ```

   and browse for the `icafilter.mat` file in the following directory:

   ```
   Your MATLAB path:\gtec\gBSanalyze\testdata\...
   RightLeftImagination\ica
   ```

2. Extract the filter

   ```
   F=get(filter,'D.W');
   ```

   The filter has the dimension 27 x 27.

3. Set the column of component 7 to zero

   ```
   F(:,7)=0;
   ```

   and multiply the data with the new filter

   ```
   y = F * y;
   ```

   to remove the EOG artifact.

This matrix F can be used on-line in BCI2000 (i.e., by programming the *SpatialFilterKernal* as described before for the CSP example) to eliminate EOG artifacts.

DSLVQ Feature Weighting

DSLVQ can be used to analyze the importance of specific features to a discrimination task.

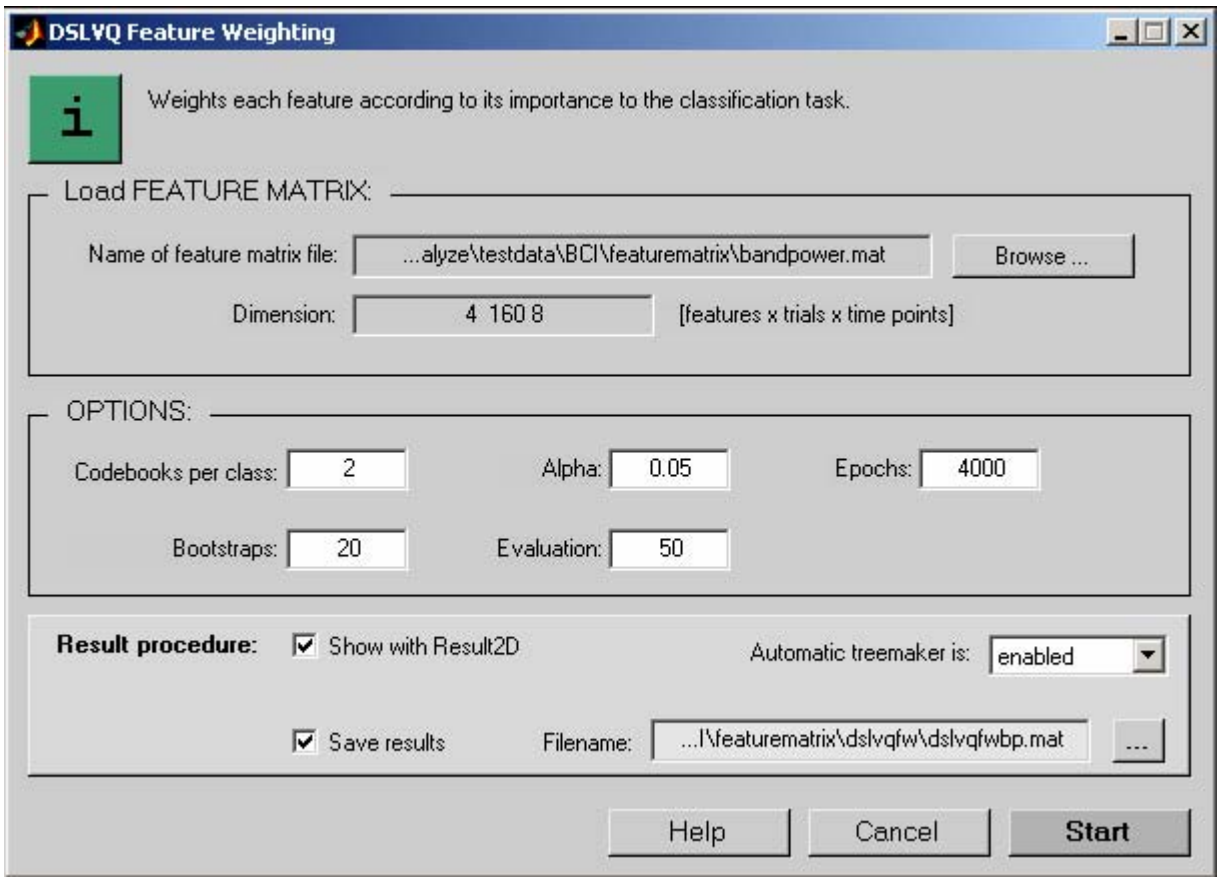
The window has the following OPTIONS settings:

Codebooks per class … define the number of codebooks for each class
Alpha … define the learning speed of the algorithm
Epochs … define the number of iterations
Bootstraps … number of bootstrap repetition
Evaluation … percentage of data used for testing'

Perform the following steps:

1. Open **DSLVQ Feature Weighting** from the **Classification** menu and **Browse** for the feature matrix `bandpower.mat` under

   `Your MATLAB path:\gtec\gBSanalyze\testdata\BCI\featurematrix`

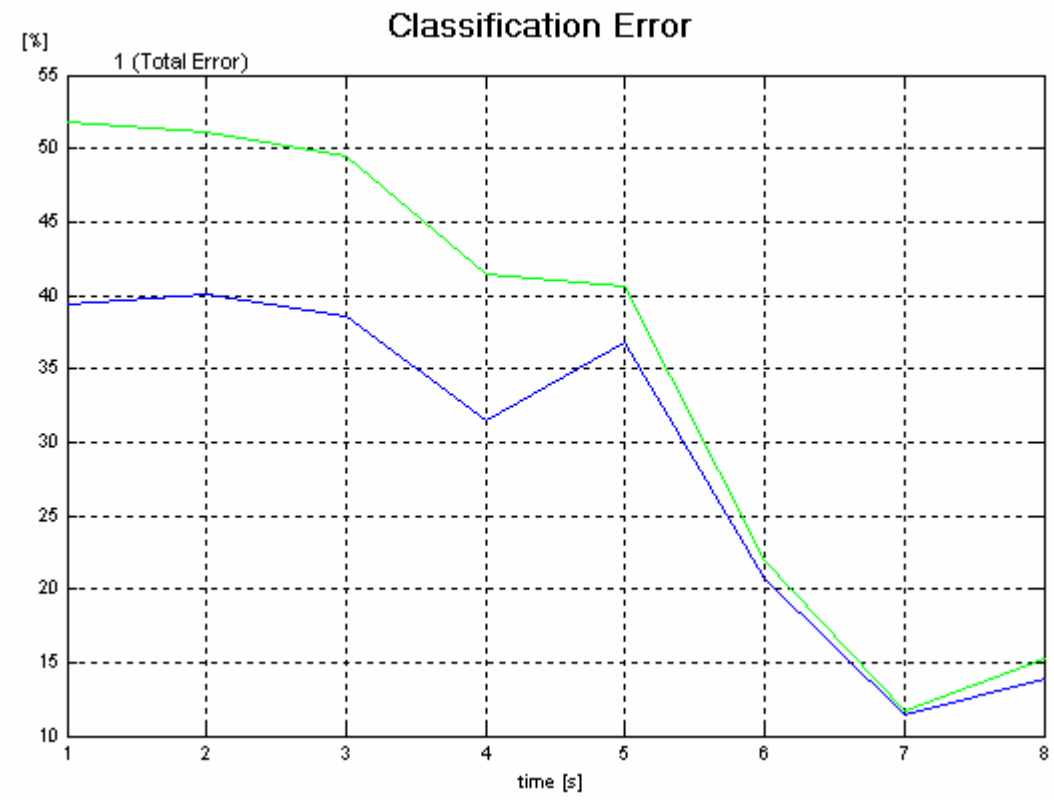


2. Set the number of **Epochs** of `4000`

3. Check **Save results** and enter the filename `dslvqfwbp.mat`

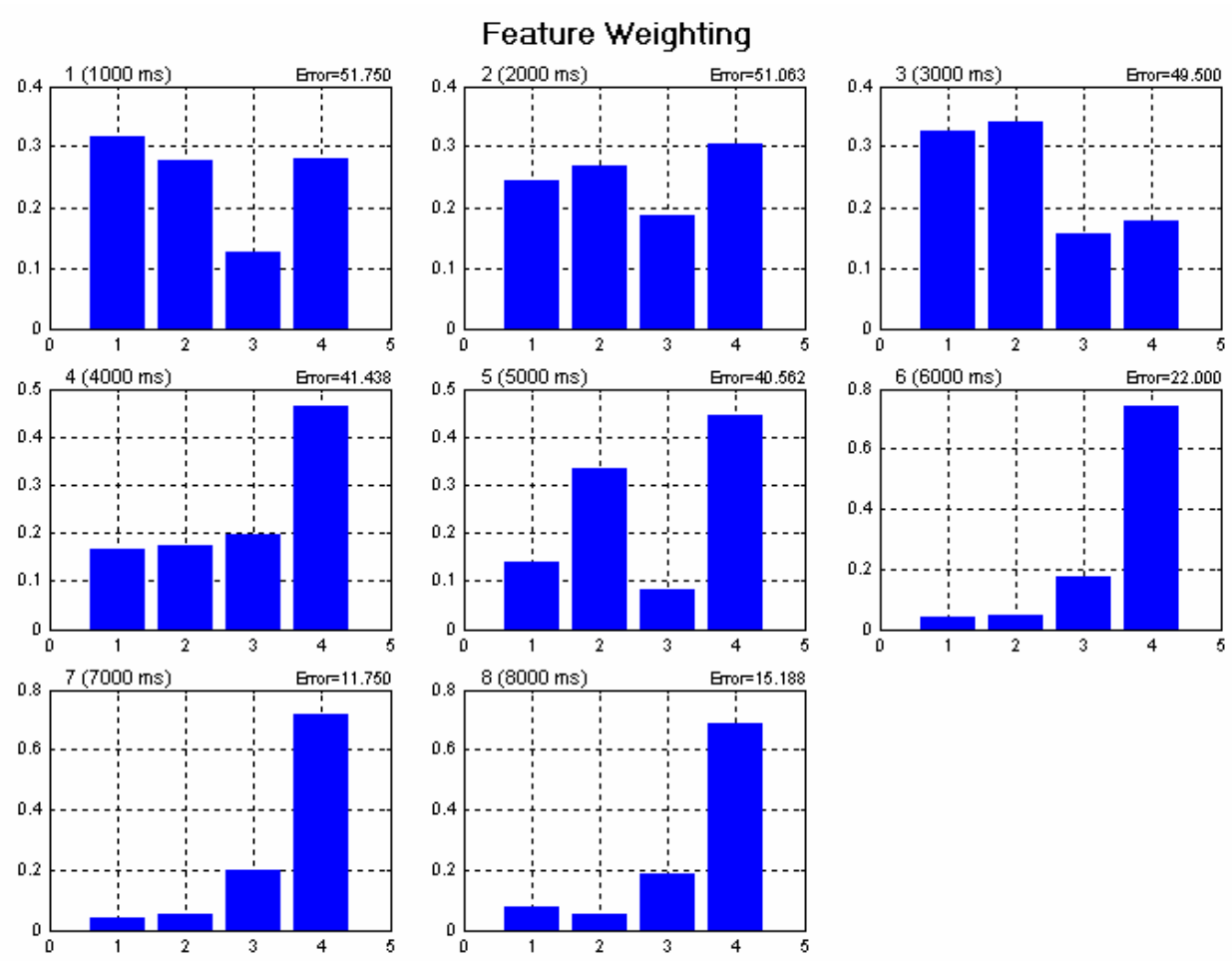
4. Press **Start** to perform the feature weighting

Page 1 of gResult2d show the classification error of the training data (blue line) and of the testing data (green line). The minimum is reached at second 7. During the first 5 seconds the error rate of the testing data is higher as the training error.



The second page of gResult2d shows the feature weights for each time point. From second 1 to 3 the error is around 50 % and drops down to a minimum of 11.75 % at second 7.

The blue bars represent the importance of each feature to the discrimination task. At second 1 features 1 is the most important one, followed by 4, 2 and 3. But the classification error is 51,75 % and therefore the result is random. At second 7 the error is 11,75 % and therefore the feature weighting can be considered as reliable. Therefore, feature 4 (bandpower in the beta range of channel 2) is the most important one. The bar of feature 3 is much smaller but the feature can still be considered as important for the classification task. Features 1 and 2 are not important and should not be considered for the discrimination.



Configuration, Conduction, and Analysis of P300 Experiments

Febo Cincotti
Fondazione Santa Lucia

1st BCI2000 Workshop
The Desmond Conference Center
Albany, New York, USA
June 13+14, 2005

Tutorial

BCI 2000

P300 Signal Acquisition and Analysis using *Mario*

P300 ACQUISITION **3**

FOREWORD 3
OBJECTIVE 3
PREREQUISITES 3
PLANNING YOUR EXPERIMENT 3
BCI2000 SET UP 4
MONTAGE INSTRUCTIONS 5
CONDUCTING THE EXPERIMENT 5
Instructions to the subject *5*
Monitoring the recording session *6*

BCI2000 SETUP FOR P300 ..... **7**

PIPE CONFIGURATION 7
SCREENING AND FOLLOWING SESSIONS 7
SYSTEM CONFIGURATION 8
SUMMARY OF SYSTEM OPERATION 13
Acquisition *13*
Spatial Filtering *13*
Combination of features into a logical signal *13*
Signal translation *14*
Application and classification *14*
GENERAL CONFIGURATION 14
Storage settings *15*
Display settings *15*

P300 ANALYSIS **17**

STARTING AN ANALYSIS SESSION 17
OPENING FILES WITH MARIO 17
RUN COMPUTATIONS 19
VIEW RESULTS 19
R-square matrix *20*
Amplitude waveforms *21*
Topographic plots *22*
INTERPRETATION OF THE RESULTS 23
OFFLINE "PREDICTION" 25
SAVING PARAMETERS 26

P3SPELLER APPLICATION STATES **27**

P300 Acquisition

Foreword

Most of the activities described in the following are very similar to those described with regard to the Mu Rhythm analysis. Since this is a tutorial and not a manual, and since some people may be interested in P300 and not in Mu Rhythm experimentation, we chose not to avoid repetitions, but rather to have a P300 section that can be read independently from the corresponding Mu Rhythm section.

Objective

This tutorial intends to give you a step-by-step guide on how to configure BCI2000 for the classification of P300 ERPs and their use with the P300 speller.

Prerequisites

This tutorial assumes:

- that you have installed the latest version of BCI2000
- that you were able to start any of the demo configurations (e.g. running `c:\bci2000\startupP3speller.bat`)
- that you were able to connect the specific BCI2000 Source Module (e.g. DT2000.exe, Neuroscan.exe, RDAClient.exe, etc.) to your acquisition system

Planning your experiment

You are going to test Brain Computer Interface control mediated by a physiological phenomenon named "P300 Evoked Potential." Please refer to other parts of this documentation to have further information about P300.

Briefly, the P300 potential is a positive deflection of brain signal potentials that last up to few hundreds of milliseconds with an amplitude of a few microvolts, which can be measured on the vertex of the scalp when a "relevant" stimulus is presented. To produce this phenomenon, the subject is asked to concentrate on one of more icons. The target items will seldomly flash; occurrence of a P300 potential in correspondence with flashes of that specific icon allow the system to identify which stimulus was attended to by the subject.

The BCI2000 package can administer such stimuli in a number of ways. Indeed, several applications were designed to produce visual and/or auditory cues that a subject can concentrate on to spell a word or to choose a command. Among these, the P3Speller represents a very standardized and effective utility for word spelling.

The experimental task will consist of suggesting a word to the subject, and having him try to spell it with BCI2000.
In order to spell each letter, the system will flash rows and columns of the letter matrix on the screen for a number of times; this takes (in the default configuration) about 20 seconds per character. Suggested words should be no longer than 10-12 characters.

In fact, as a rule of thumb the experiment must be designed such that continuous recording does not last more than 3-5 minutes: because subjects are typically told to minimize movements (to minimize artifacts), sessions are physically and psychologically stressing, and thus it is not indicated to record for longer time periods. Recording "runs" will be repeated until the amount of data collected suffices. Consider to keep the overall recording time below an hour (about half an hour recording, plus preparation and cleaning), so that experimental subjects still enjoy participating in the study.

Knowing this, we must now set our system so that it receives the relevant EEG data and performs the proper analyses.

BCI2000 set-up

We assume by now that for some reason you already know the location and latency of the P300 potential. You usually determine this information by analyzing the data acquired during a previous session or during an initial screening session. In the following sections, we assume that locations Fz and Cz show the maximally discriminant difference between target and non-target responses at a latency of 350 ms after stimulus delivery.

P300 experimental sessions are supported by BCI2000 using the P3SignalProcessing and the P3Speller modules, along with the Source module appropriate for the available EEG equipment.
Parameters for each module must be set up in accordance with the current EEG sampling frequency and to the specifications set out above regarding the responsive EEG features. A step-by-step guide to setting up BCI2000 for a P300 experiment is given in another section.

Montage instructions

Electrode placement is covered more extensively in an Appendix to this tutorial. Here we just underline how important it is that:

1. the cap has been carefully cleaned. Dried residuals of gel will likely result in higher electrode impedances.
2. the cap is placed with precision. If in a previous session you found that a certain electrode is responsive, you want to put that electrode over the very same position as in the session before.
3. the conductive gel is injected in the right quantity; too little will not allow a good contact; too much will spread out to large areas, thus resulting in a spatially averaged signal.
4. Reference and ground electrodes are correctly utilized, so to minimize the mains disturbance.

Conducting the experiment

Instructions to the subject

After the subject is ready for EEG acquisition, he/she is given instructions about the experimental task.

A matrix of letters and numbers will be presented to him, and a letter will appear on the top left corner that suggests what the subject will have to spell next. After a few seconds, the rows and the columns of the matrix will start flashing. At that time, the subject will have to concentrate on the suggested letter, and count how many times it flashed (meaning that either the row or the column that contain that letter has flashed). After about 30 s the checkerboard will stop flashing, and the system will attempt to predict the letter the subject was concentrating on from his/her ERPs. The predicted letter is noted between the cue letter and the matrix.
This cycle is repeated for each letter contained in the TextToSpell variable.

The second set of instructions to the subject regards the minimization of artifacts. Since in a typical recording session EEG data are acquired for off-line analyses, it is important that all efforts are made to obtain the highest quality EEG.
The main causes of artifacts are:

- Contraction of the muscles of the face/head, swallowing;
- Eye blinks and eye movements;
- Motion

Provided that subjects are asked to minimize artifacts (e.g. asked to try to swallow only during the pause between letters), he/she should be further assisted

in these efforts by providing a comfortable chair and a dimly lit room. We will return to the role of reinforcement, but it is already clear that the experimenter must carefully monitor the EEG and alert the subject in the case he/she has forgotten some of the instructions.

When the experimenter is sure that his/her instructions have been well understood, the recording session may start.

Monitoring the recording session

After the recording session has started, one might feel the temptation to leave the subject alone during the run since most of the experimental activities are automated in BCI2000. On the contrary, the experimenter has several tasks:

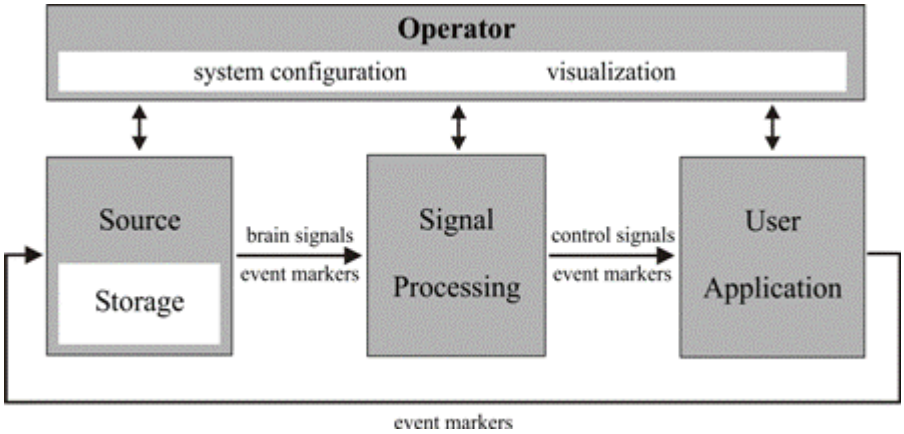
- Filling in the run-sheet to report information that is not automatically recorded by BCI2000 and that will later help when data are analyzed.
- Monitoring the EEG waves, checking that the quality of recording does not decrease (electrode contact failure, muscular, EOG or motion artifacts, etc.)
- Reinforce the subject: warn him/her if he/she is producing artifacts, keep him/her alert if getting drowsy, give him/her feedback about his/her performance so that interest is kept high.

The run sheet is the most important mean of communication between the technician who performs the recordings and the person who analyzes the data (or possibly the historical memory of whoever does both). It is important that it is compiled carefully and that it is rich of appearingly obvious details: only time will say what is standard and what changes from session to session, and if you will need to analyze data acquired years before, you will probably have a problem if you did not note down all information.

BCI2000 setup for P300

Pipe configuration

In this section we will use the following modules:



Source	Signal Processor	User Application
RDAClient	P3SignalProcessing	P300 speller

The RDAClient source was chosen only as an example, and you should replace it with the specific module for your data acquisition hardware.

If you are not familiar with the pipe setup, please have a look at Setting up the P300 Pipe.

Screening and following sessions

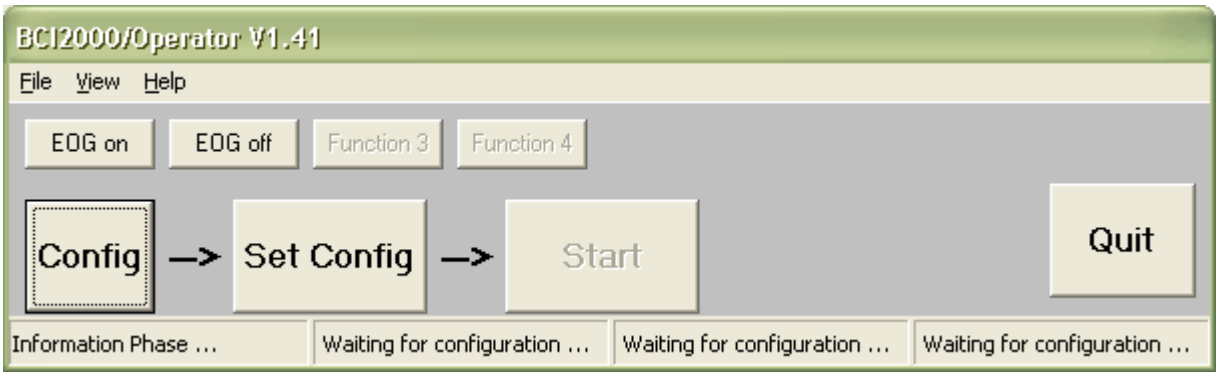
Before the system can correctly classify the P300 Event Related Potentials (ERPs), you need to determine the individual EEG feature (e.g. channel and latency) that most reliably separates responses where the P300 component is present from those where it is not.

As an example, we will now assume that you would like the system to use the samples at channels 13 and 14, both at 350ms, for classification.

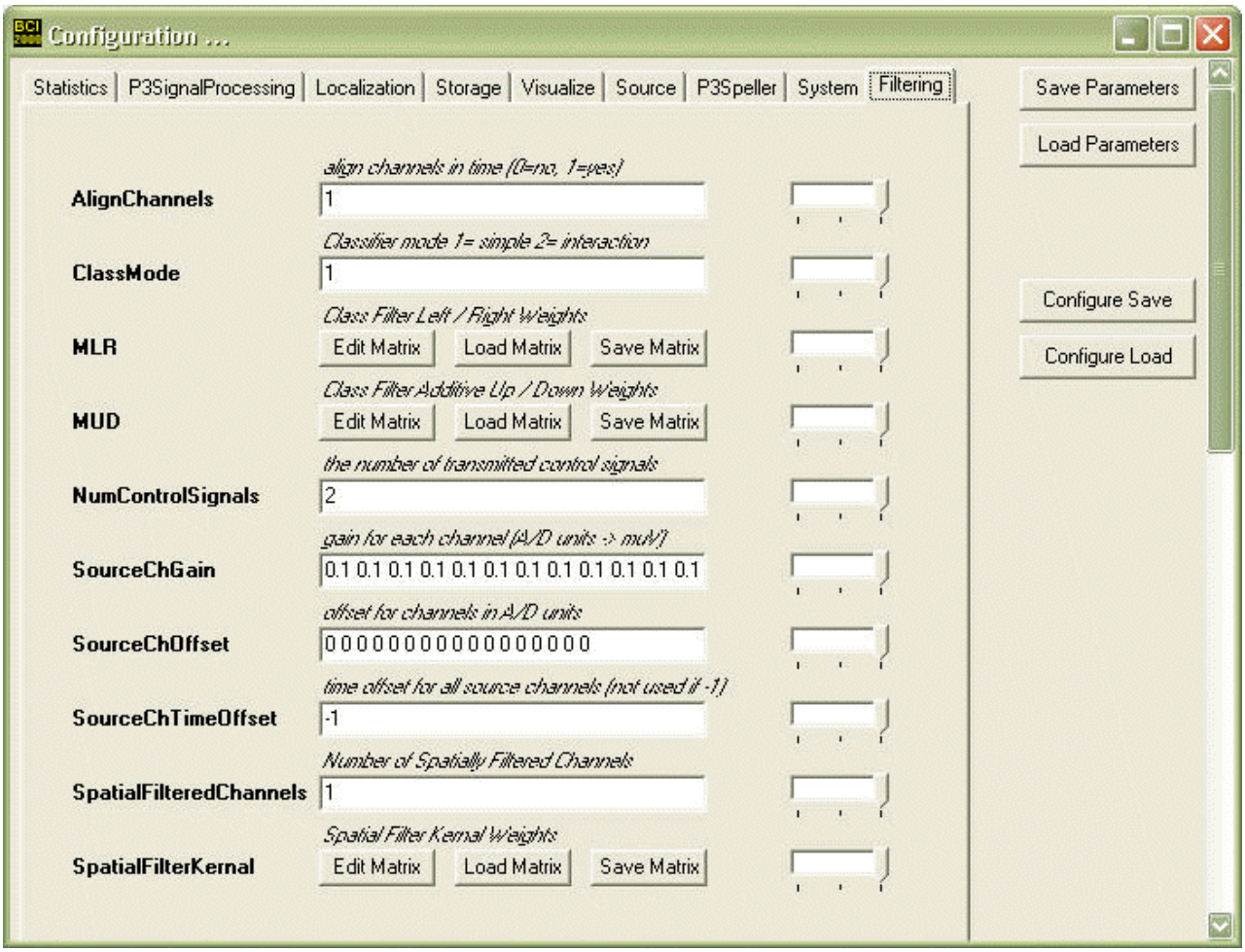
After the subject has been recorded once, off-line analysis will allow us to select the correct values.

System configuration

On the Operator window, press the Config button in order to access the Configuration dialog.



In the Configuration dialog press Load Parameters and choose the file `C:\BCI2000\Tutorial\prm\P3tutorial.prm`. This will automatically set the proper value for each parameter. Close the Configuration dialog, return to the Operator window and press Set Config.



At this point the system has been configured as follows:

Group	Parameter	Value	Description
Source	SamplingRate	256	The sampling rate per channel. A number above 200 suffices for most EEG applications.
	SampleBlockSize	8	In this example, the source module will transmit data through the processing pipe in blocks of 16 samples. This results in a system update of 256/8=32 times per second

			(or 31.25 ms per update)
	TransmitCh	16	This is the subset of the total number of channels you want to transmit to Signal Processing. Unless your system is slowed down by processing all channels, you should transmit all the source channels.
	TransmitChList	1 2 3 ... 16	Enumeration of all *TransmitCh* channels.
	SourceChOffset	0.1 0.1 0.1 ... 0.1	The first step in Signal Processing is calibration. Signals are converted from A/D units into µV. The used transformation is: result in µV=(A/D units-offset)*gain. Specify these variables as appropriate for your system.
	SourceChGain	0 0 0 ... 0	
Filtering	SpatialFilteredChannels	2	This number (e.g., 2) specifies how many output channels result from the spatial filtering operation (see SpatialFilterKernal) and will be available to the linear classifier.

	SpatialFilterKernal	[SpatialFilteredChannels x TransmitCh] matrix	BCI2000 can provide any linear spatial filtering operation (e.g. Common Average Reference, Surface Laplacian, etc). Specify weights that determine which linear combination of input channels make up the output channels.
	MUD	[#of features x 3] matrix	See details in the text below this table
	MLR	0 0 0 0 0	
P3SignalProcessing	NumERPsToAverage	15	This number specifies how many ERPs, for each stimulus, P3SignalProcessing averages before it classifies the waveform. Unless you are an expert, this number should match NumberOfSequences
	NumSamplesInERP	154	The number of samples after stimulus elicitation that can be used for classification. 154 samples corresponds to ~600ms at a 256Hz sampling rate.)
	TargetERPChannel	6	This is the channel (inside Signal Processing) whose average waveforms will be displayed in

			the ERP display. In this example, it will display the sixth spatially filtered channel.
P3Speller	NumberOfSequences	15	see NumERPsToAverage
	OffTime	1	These parameters specify the duration (in units of SampleBlocks) that the stimuli (e.g., rows/columns in the P300 Speller) will be intensified. Using 1 and 3, respectively, corresponds in our example to a stimulus intensification of ~94ms (3*31.25ms) and a non-intensified period of ~31ms (1*31.25ms).
	OnTime	3	
	OnlineMode	1	0=copy mode (user copies TextToSpell); 1=free spelling mode
	TextToSpell	(any text)	The text that the user has to spell
Statistics	XMean	-1	Baseline of the horizontal control signal
	XGain	400	gain of the horizontal control signal
	YMean	0	Baseline of the vertical control signal
	YGain	100	Gain of the vertical control signal

Summary of system operation

Acquisition

When the Analog-to-Digital Converter on your acquisition system sends a sample to the Source module, it is buffered until a block of SampleBlockSize (32) samples is ready, which happens every SampleBlockSize/SamplingRate*1000 ms. This rate is the pacemaker for the rest of the processing pipe. Only a subset (TransmitChList) of the acquired (and saved) channels is transmitted to the processor, where values in ADC units are first converted into microvolt.

Spatial Filtering

The transformation matrix can implement different spatial transformation. In the simplest case, i.e. if it is a unity matrix, the output of the SpatialFilter will be the same raw EEG data that were input.

If the transformation matrix is a rectangural "quasi-unit" matrix, e.g.:

0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0

the output will only have two channels, that are identical to input channels 13 and 14 (see the ones in column 13 and 14), respectively.

If the matrix is as follows:

-1/16	-1/16	-1/16	-1/16	-1/16	-1/16	-1/16	-1/16	-1/16	-1/16	-1/16	-1/16	**1**-1/16	-1/16	-1/16	-1/16
-1/16	-1/16	-1/16	-1/16	-1/16	-1/16	-1/16	-1/16	-1/16	-1/16	-1/16	-1/16	-1/16	**1**-1/16	-1/16	-1/16

it will implement the Common Average Reference, i.e., from each channel the average of all other channels is subtracted prior to further processing.

Spatial filters that are more complicated (e.g. ones derived from Independent Component Analysis) can be easily implemented by computing the corresponding linear transformation matrix.

Combination of features into a logical signal

The aim of the MUD matrix is to tell the program which features (i.e. voltages at particular channels and time points) to use and how to combine them into a

single-dimensional signal, i.e. a signal that should be large for stimuli containing a P300 response, and small for ones that do not.

In order to program the MUD matrix, you need to convert the latency(ies) that you would like to use (e.g., 350 ms) into samples using the current SamplingRate. (e.g. 256*350/1000~=90). Since we want to use channels 13 and 14 at the same time, MUD must be set as follows

	chan	samp	weight
Feature 1	13	90	1
Feature 2	14	90	1

Make sure that you set ClassMode to 1 (otherwise you need to specify more parameters).

In this example, the program will compute the output of the Filtering stage as the sum of Feature1 (sample 90 at channel 13) times a weight of 1 + Feature2 (sample 90 at channel 14) times a weight of 1.

Signal translation

At the statistics stage, the system subtracts YMean from this number, and it will the multiply the result by YGain. You could use 0 for YMean and 100 for YGain. Because this is a linear transformation and because in the end, the system only determines which stimulus (i.e. row or column in the matrix) elicited the highest P300 response, it is essentially arbitrary (for the purpose of P300 processing but not for other brain signals) what you put into these parameters.

Make sure that InterceptControl is turned off so that the system doesn't adapt these values.

Application and classification

The system will then transmit the classification results (i.e. output of the MUD matrix) for each stimulus to User Application (e.g. P3Speller or P3AV). After having averaged the NumSamplesInERP following each of the NumERPsToAverage stimuli, it will then pick the highest classification result for all rows, and the highest result for all columns to determine the predicted character.

General configuration

Some of the parameters have not been discussed so far. Some of these are technically advanced, and should be left to their default value unless the experimenter knows how to use them. Other parameters are, on the other side, of very general use and will be covered here.

Storage settings

These parameters define where on the disk the acquired data will be saved.

FileInitials	`..\Tutorial\data\P300\`	Base folder for storing the current session. It can be an absolute path, or can be relative to the position of the program files.
SubjectName	TestSubj	Subject initials
SubjectSession	001	Session number
SubjectRun	04	Run number

Assuming that operator.exe is installed in `c:\bci2000\Operator\`, the files will be stored in the folder:

`c:\bci2000\Operator\TestSubj001\TestSubjS001R04.dat`

The run number will increase automatically, so the experimenter does not have to care about changing it after each run.

Display settings

WinHeight	512	Width and Height of the User Application screen (e.g. the one where the matrix flashes)
WinWidth	512	
WinYPos	0	Horizontal and Vertical position of the top-leftmost corner of the User Application screen. Y coordinates increase from top to bottom; X coordinates increase from left to right
WinXPos	512	

On a 1024x768 monitor, the above setting would put the user screen in the top right quarter.

Another option is to use the “Extend Desktop” facility of Windows to run BCI2000 on a dual monitor system. If the main monitor has a resolution of 1280x1024 and the secondary monitor has a resolution of 800x600 (positioned to the right of the main one, the top edges lined up) the appropriate settings would be:

WinHeight	600
WinWidth	800
WinYPos	0
WinXPos	1280

Setting up the P300 Pipe

You can start the Operator, and then run each of the previously mentioned modules, but you can obtain the same results if you create a bat file like the following

```
start operator\operat.exe
wait 0
start EEGsource\RDAClient\RDAClient.exe AUTOSTART 127.0.0.1
start SignalProcessing\P3\P3SignalProcessing.exe AUTOSTART 127.0.0.1
cd Application\P3Speller
start P3Speller.exe AUTOSTART 127.0.0.1
cd ..\..
```

Save this file to your root BCI2000 folder (i.e. `C:\BCI2000\`) and name it `P3tutorial.bat`. You can double-click it whenever you want to run the whole system. You can also create a link to this file and copy it to your desktop for your convenience.

Back to to setup

P300 Analysis

Starting an analysis session

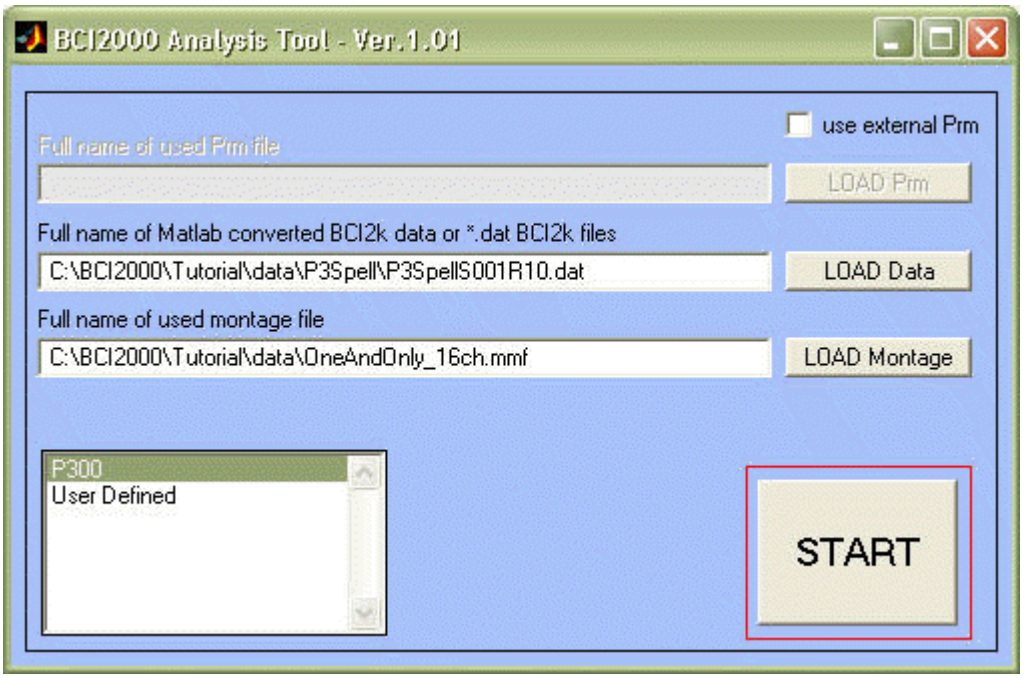
Every recording session has its own history. It is very difficult to say in advance what might happen during the recording. So we will assume that everything was perfect (the subject collaborated, no channel went lost, etc); later on we will consider the main causes of contamination, how to recognize it and what to do.

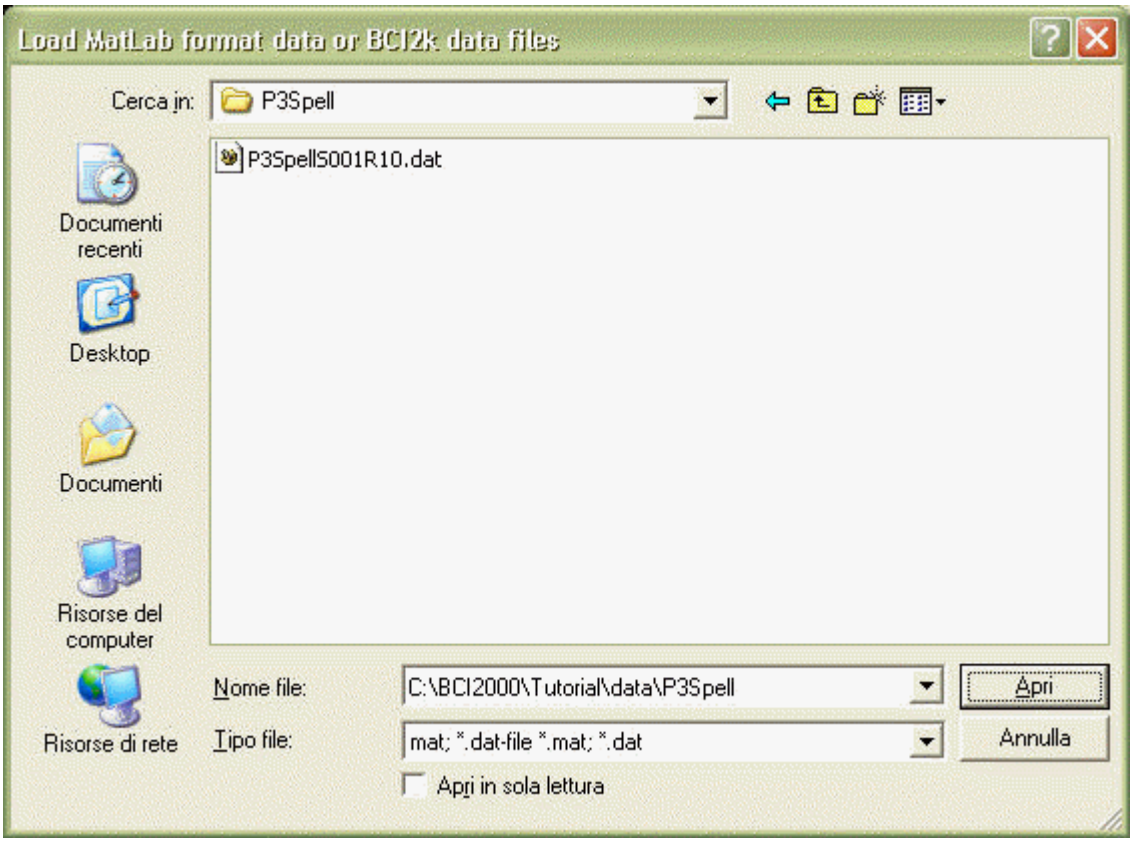
If the recording session ran smoothly, in the folder `c:\bci2000\data\P300\TestSubj` you should have now eight files named `TestSubjS001R01.dat` to `TestSubjS001R08.dat`, corresponding to the eight runs, respectively.

Before you start the analysis you should locate on your hard disk the correct montage file. This is a file that describes the list of channels that were acquired. In this tutorial we will use `OneAndOnly_16ch.eloc.txt`, which is located in the folder `C:\BCI2000\Tutorial\Montages\`. If you need to edit the `*.mmf` file, check its data format.

Opening files with Mario

After having run `mario.exe`, press LOAD Data. A dialog window will ask you to select the dat files. You can either select a single file, or select multiple files. This time, we will select all files in the dataset.



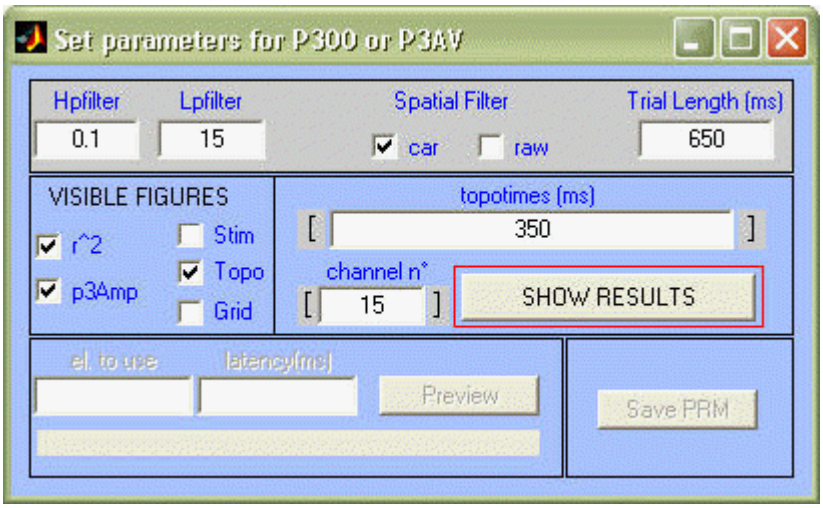


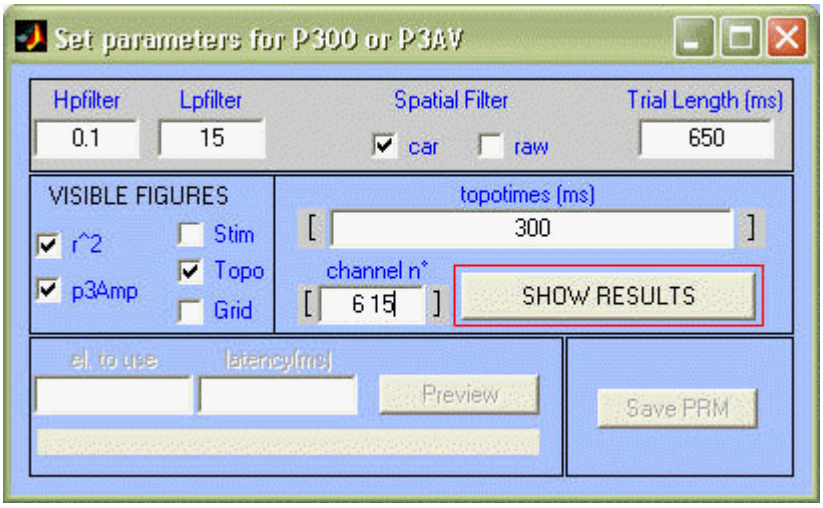
As an optional (but recommended) operation, push the LOAD MONTAGE button and choose the mmf file. This will allow you to see the labels of each channel the following steps, and to allow to apply spatial filters that depend on electrode positions.

At this point, you can check that P300 is selected in the bottom left listbox (indicating that a P300 dataset has been recognized) and push the START button that will open the P300 parameters dialog.

Run computations

In this window, we will now set the latency and channel that have been used for the on-line experiment (see BCI2000 setup for P300). Under topotimes (ms) type 350; under channel n type 15 (corresponding to Cz). Then, press SHOW RESULTS to let the program process the data and display the results.





View results

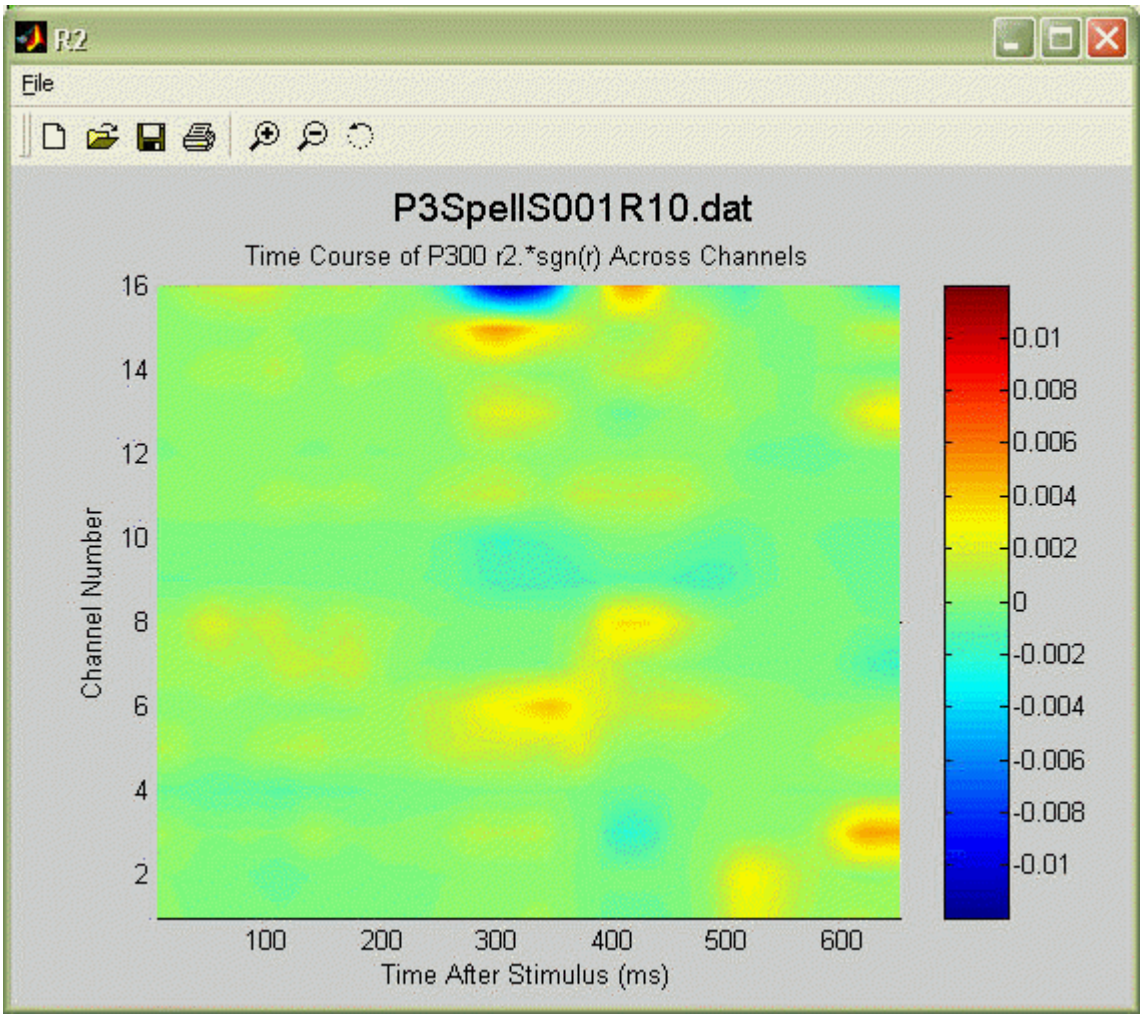
Since three checkboxes were checked in the VISIBLE FIGURES panel, three types of results will be displayed.

1. R-square matrix (channels by latencies)
2. Amplitude waveforms at selected channel(s)

3. Topographic plots at selected latency(es)

R-square matrix

The r-square matrix highlights the most relevant ERP features for the separation of potentials due to frequent or rare stimuli.



In this graph, the X axis is time after stimulus presentation and the Y axis represents the different channels. The color codes the statistical significance of the difference between the two kinds of evoked potentials (i.e. ones that correspond to attended stimuli, and others that correspond to non-attended stimuli).

Thus, a red color would mean that the ERP consequent to a rare stimulus is significantly more positive than the ERP consequent to a frequent stimulus.

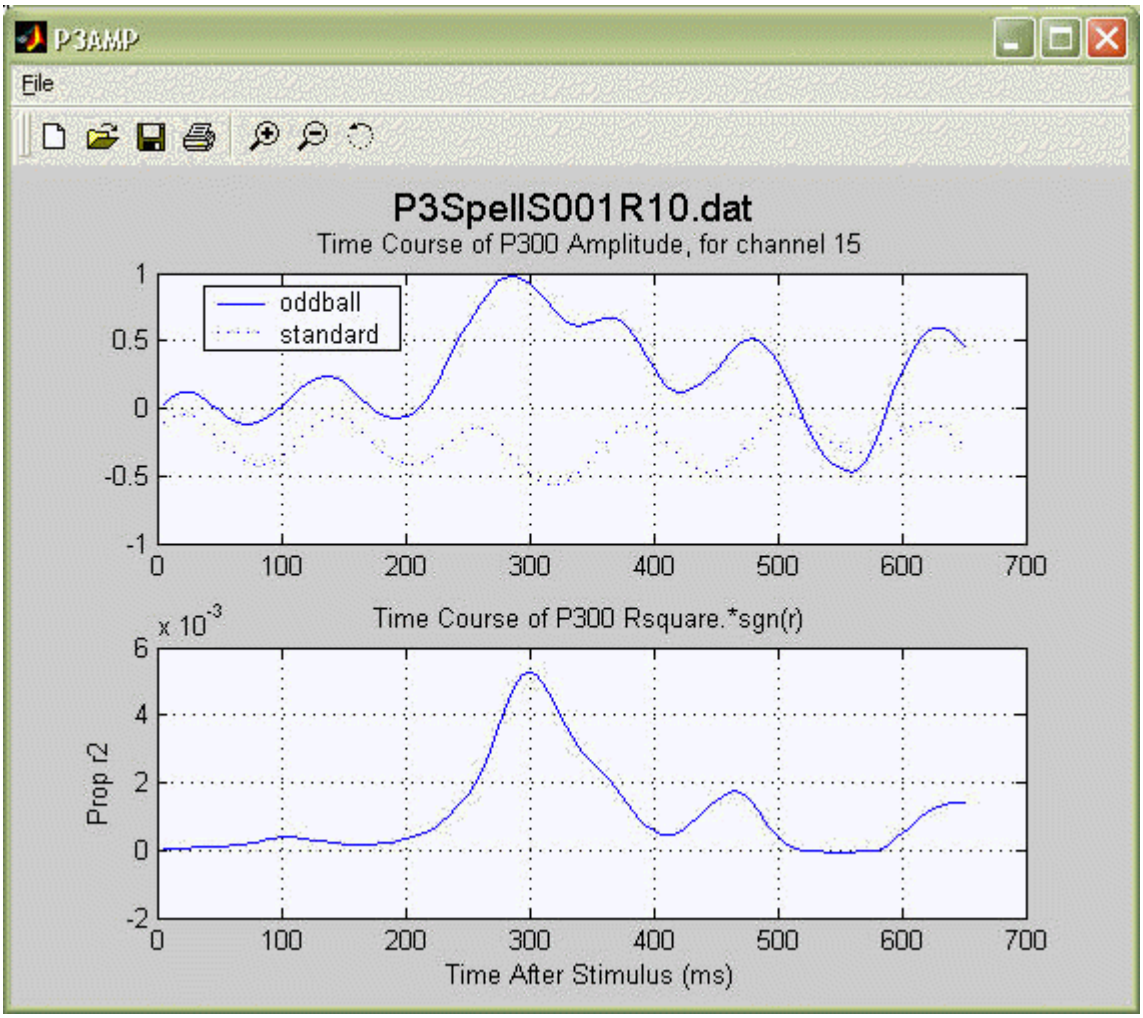
This figure is independent of the channel and latency setting that was set in the parameter dialog (since it summarizes all results).

Clicking on the matrix will modify the channel/latency settings for the other figures; in fact the cell of the matrix that was clicked on is characterized by one channel and one latency whose waveform and topography (respectively) will be shown in the other two figures.

You can choose whether to overwrite or to put the most recently evoked figures beside the previous one. Two sets of waveform/topography figures are available, and are linked to the click of the left or right mouse button.

Amplitude waveforms

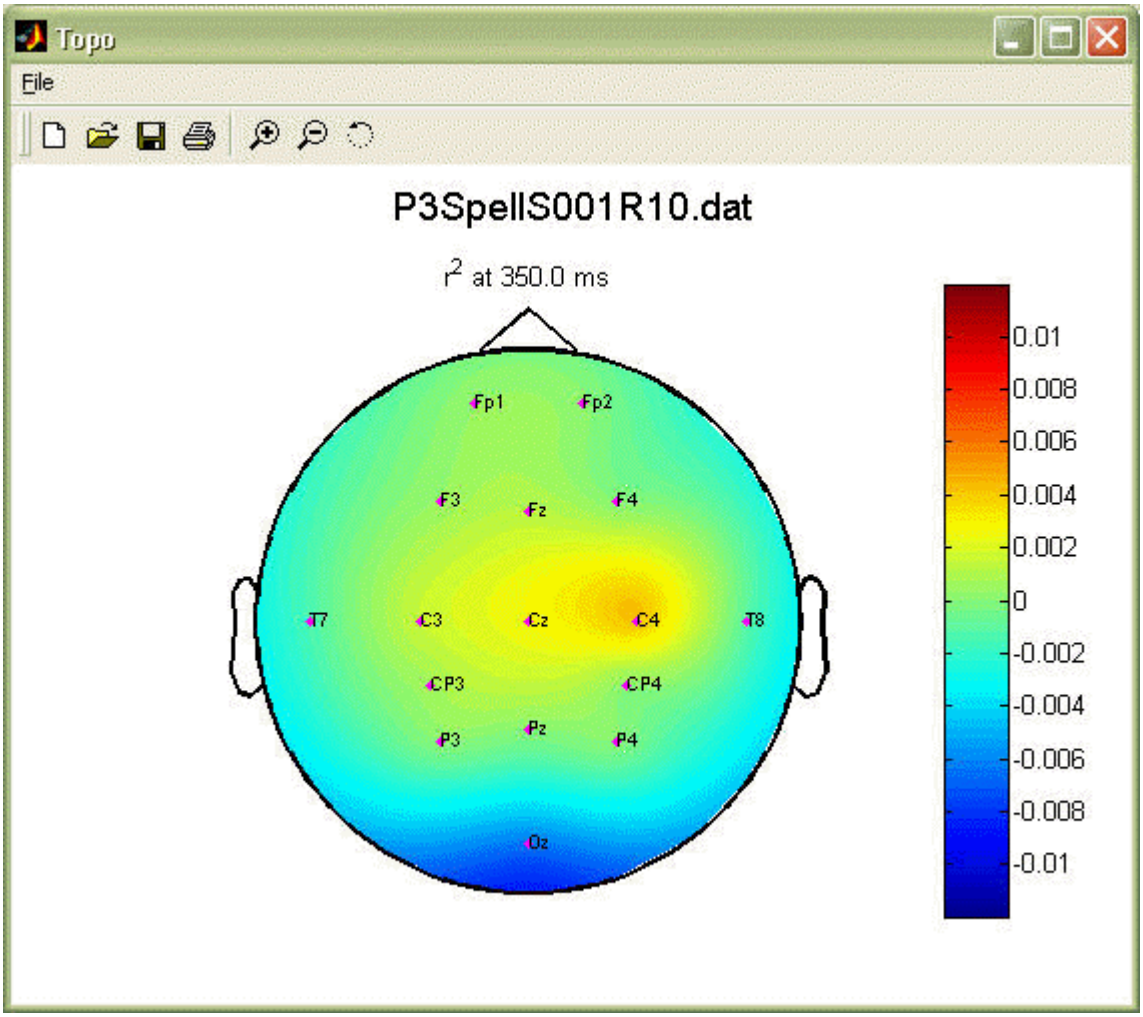
The whole time course of generated ERPs are displayed in the top panel, regardless of the specific stimulus (line of letters) that generated it. Rare ERPs are plotted in solid lines, while frequent ERPs are dotted. If two channels were selected in the P300 parameter dialog, they are plotted in red and in blue, respectively.



The bottom panel shows the r-square time course. It is roughly proportional to the difference between rare and frequent ERPs, but since r-square is sensitive to the dispersion (variance) of single trials, small differences in amplitude produce a high r-square value if they are very reproducible.

Topographic plots

This figure represents the scalp distribution of the r-square.

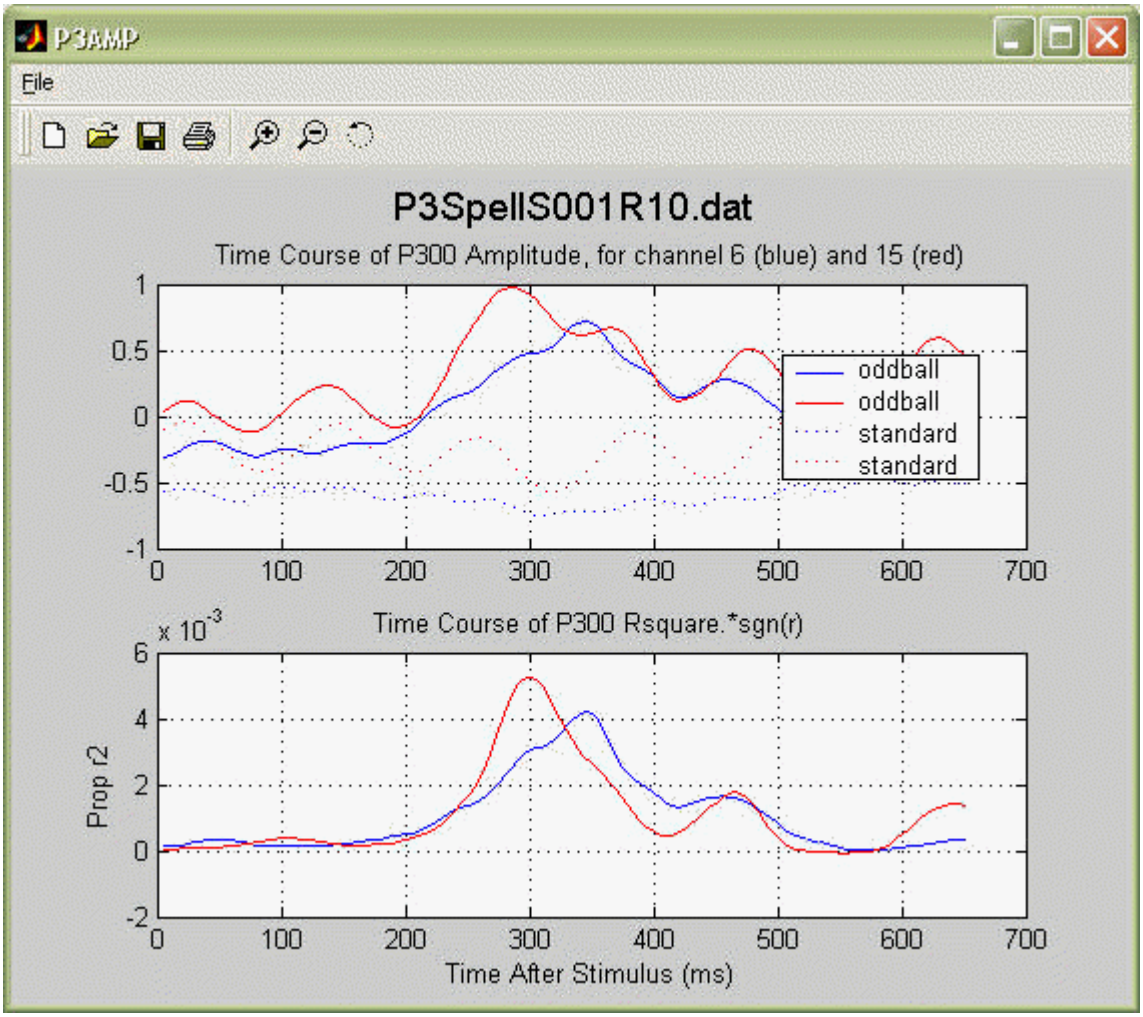


The color coding is the same as the r-square matrix figure, and values on each channel are interpolated to create a continuous bidimensional map. The more electrodes, the more accurate is the map. The number of channels shown on the scalp may not coincide with the whole number of electrodes in case of a Surface Laplacian spatial filtering, since in that case the value of the Laplacian is not computed on the border channels.

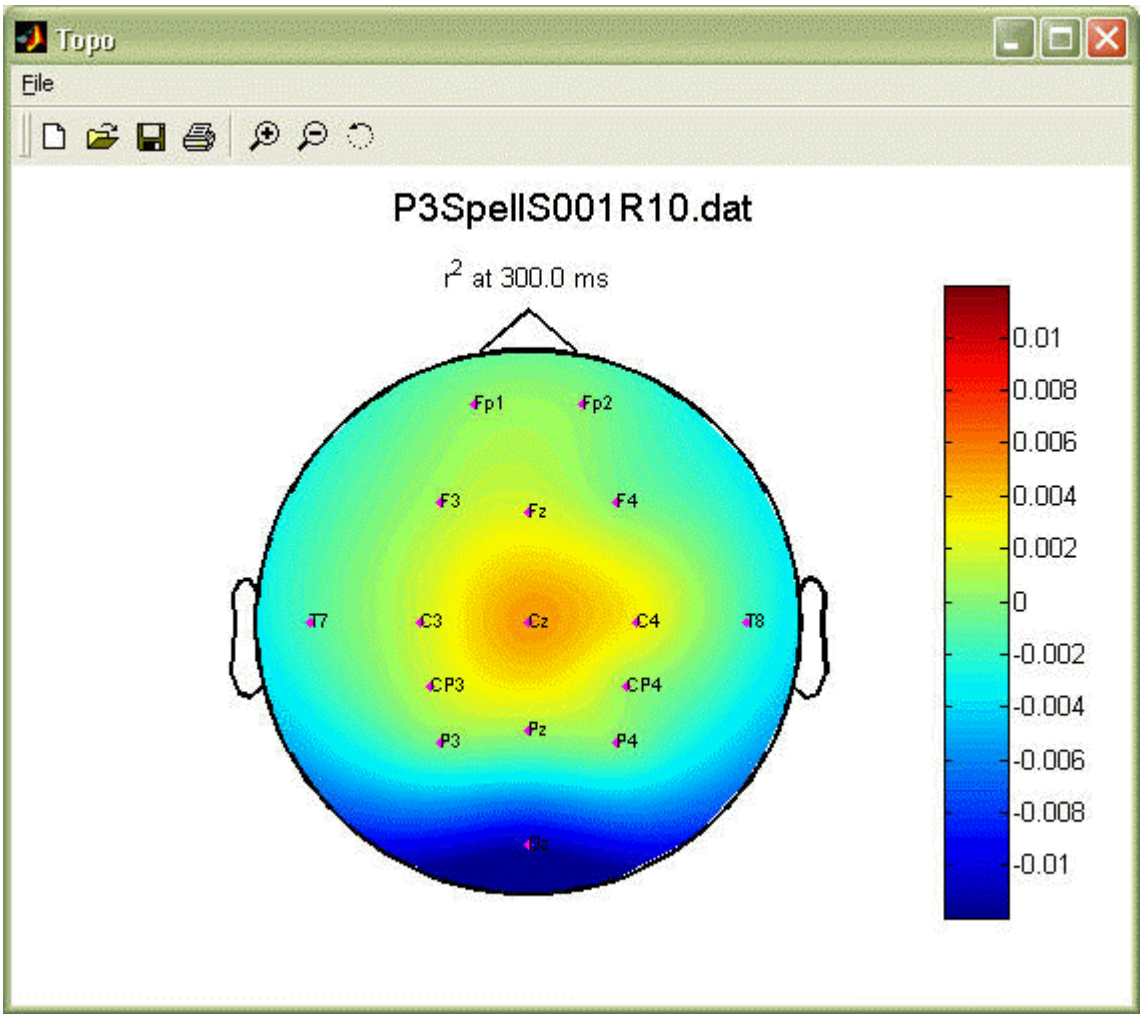
Interpretation of the results

The first step in the preliminary analysis of P300 BCI data is to find the absolute maximum of r-square. This simple statement must be mediated with a proper knowledge of physiological phenomena. In fact, we do not expect any P300 at 50 ms after the stimulus, so if the absolute maximum is at that latency, you must suspect that it is actually an artifact. The same holds if the spatial localization of the peak is far away from the vertex.

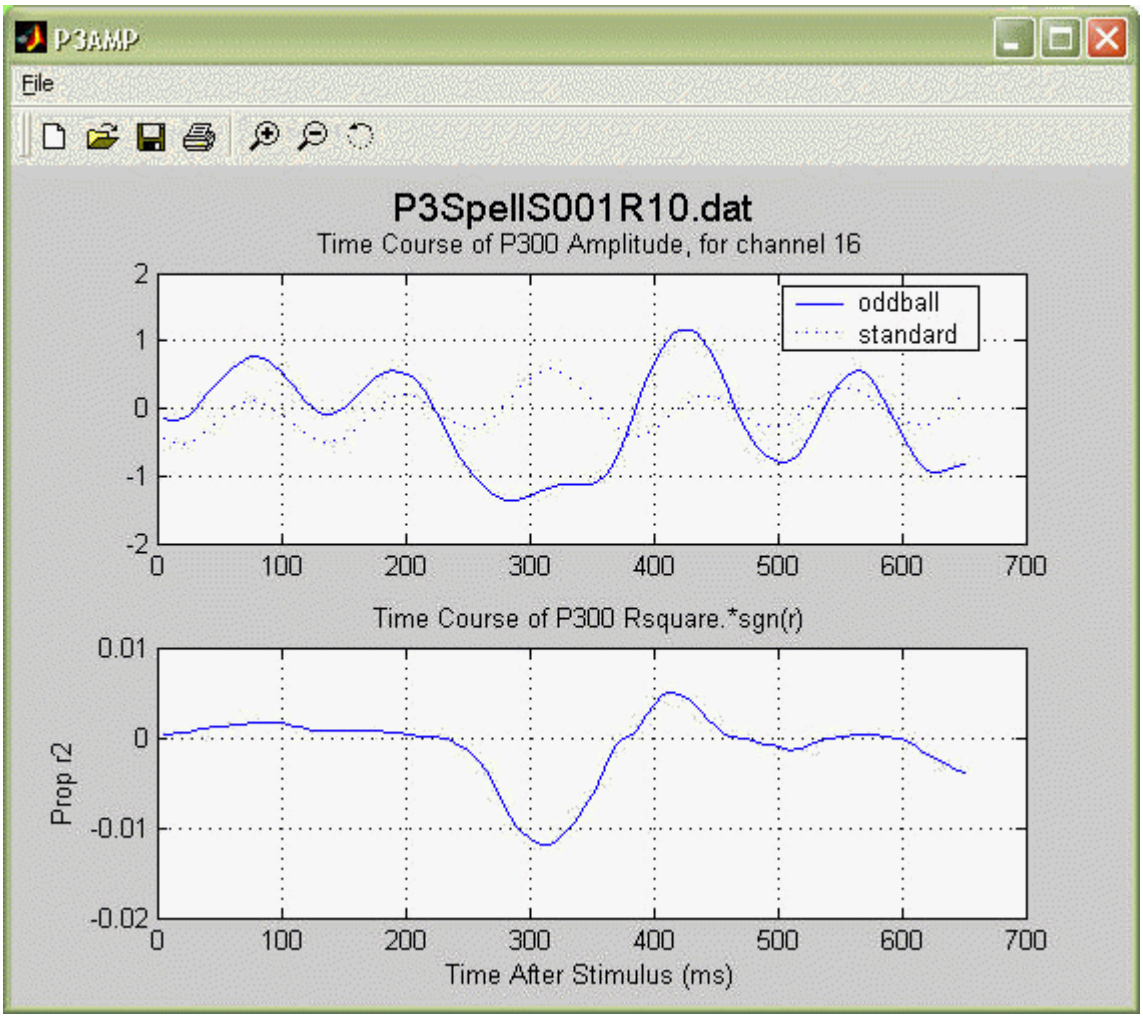
In the following figure, the time courses of channel 15 (Cz) and channel 6 (C4) are presented



From a practical point of view, it is not important now to understand the reasons for different component that explain the peak on Cz at exactly 300ms followed by a peak on C4 at 350ms. What is important is to understand that there are two components that can possibly be used. Because we are trying to predict which is the best, the next recording will help us understand which is the stable one and which one was only a transient phenomenon.



There is another peak that we have disregarded - the negative peak on Channel 16 (Oz). See below its waveforms

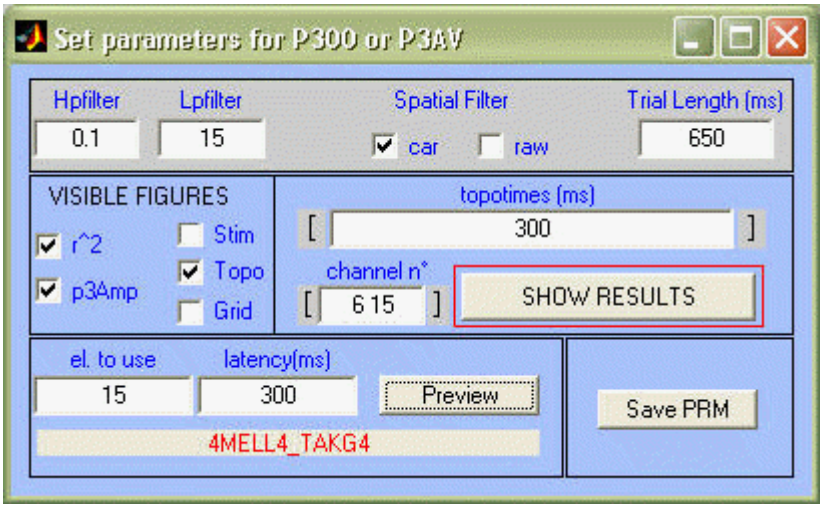


Having disregarded it was intentional, since we are looking for positive peaks in central areas.

Question for the audience: Should we take into account this component?

Offline "prediction"

Mario also allows to check what is the word that the BCI2000 system would have predicted if the current feature had been used online.
To do this, click on a point on the r-square matrix (R2 window) or type channel and latency in the edit boxes in the bottom left corner of the SetParameters window; then press Preview.
Just below, the predicted sequence of characters will appear.



In the case in example, since the sequence should have been `SPELLWITHBCI`, only 4 characters out of 12 (33%) have been correctly predicted. This score is well above chance, though quite inefficient.

S	P	E	L	L	W	I	T	H	B	C	I
4	M	E	L	L	4	_	T	A	K	G	4

In summary, we must select the latency and the channel with the maximum ERP component whose spatio-temporal shape is compatible with the P300 phenomenon. Testing the speller can suggest wich component has the highest probability to predict the user's selected letter.

Saving Parameters

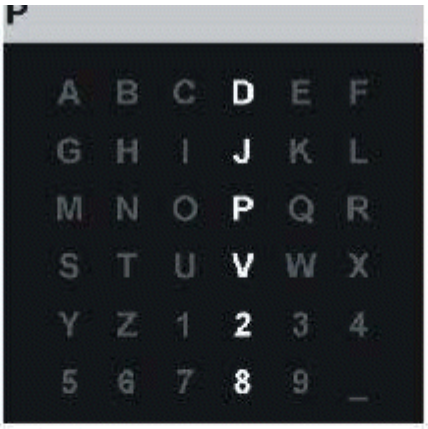
In principle, you should note down the channel and latency that gave the best performance. If in the next session you are going to use exactly the same setup, you may consider to save a partial parameter by pressing the Save PRM button when the correct latency and channel are reported on the left.

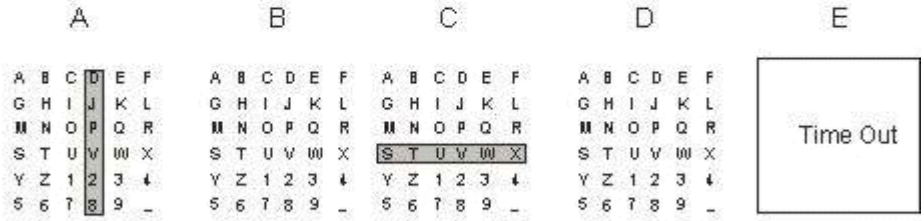
In this way, before starting the next session you may reload the same `prm` as the previous time, and then overwrite the ERP feature information by loading this partial parameter.

P3Speller Application States

The figure below is taken from the BCI2000 official documentation.

It describes how the system labels the EEG data using four different states, and the values for these states for every repetition of stimulus delivery.





	A	B	C	D	E
StimulusCode	4	4	10	10	0
StimulusType	1	1	0	0	0
Flashing	1	0	1	0	0
SelectedTarget	0	0	0	0	16

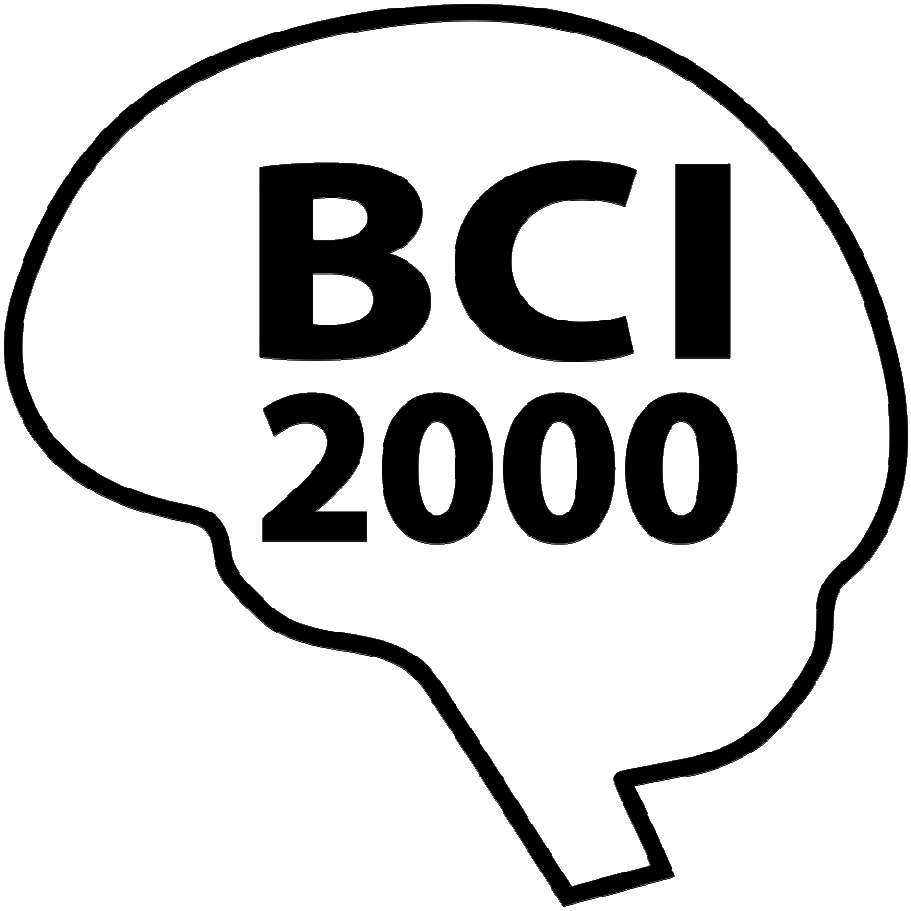
Appendix

BCI Data Acquisition and Analysis

Febo Cincotti
Fondazione Santa Lucia

1st BCI2000 Workshop
The Desmond Conference Center
Albany, New York, USA
June 13+14, 2005

Tutorial



Acquisition and Analysis:

Appendices

THE 10-20 INTERNATIONAL SYSTEM **3**

DATA FORMATS **5**

MARIO MONTAGE FILE 5
BCI2000 PARAMETER FILE 6
Syntax *7*

SOURCE INT SIGNALTYPE= 0 0 0 3 // NUMERIC TYPE OF OUTPUT SIGNAL: 0: INT16, 1: FLOAT24, 2: FLOAT32, 3: INT32 (ENUMERATION)MONTAGE INSTRUCTIONS **8**

MONTAGE INSTRUCTIONS **8**

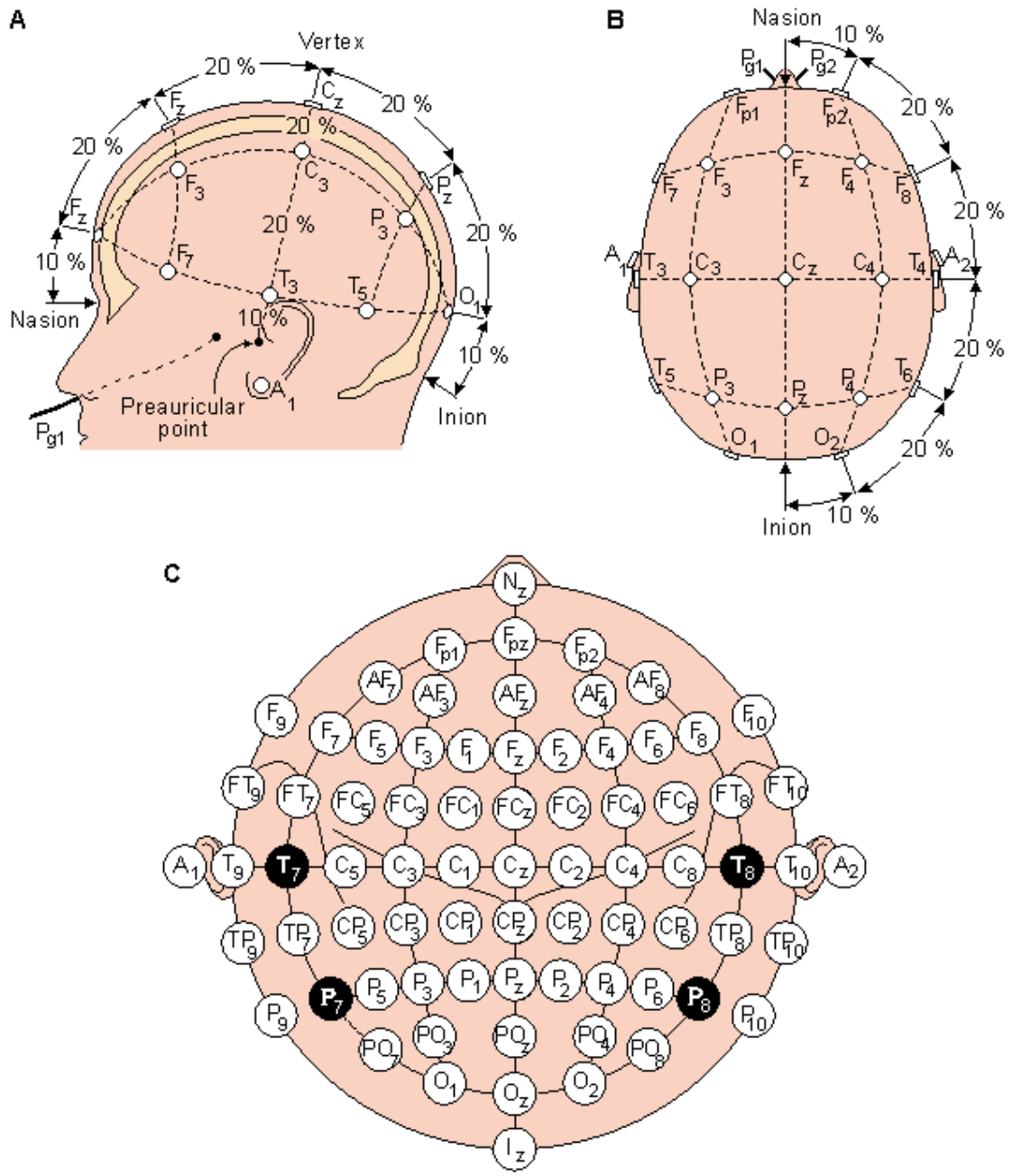
EEG ARTIFACTS **10**

BLINK ARTIFACTS 10
ELECTROOCULOGRAPHIC ARTIFACTS 11
ELECTROMYOGRAPHIC ARTIFACTS 11

The 10-20 International System

The 10-20 international system is the standard naming and positioning procedure for EEG applications. It is based on the iterative subdivision of arcs determined on the scalp starting from craniometric reference points: Nasion (Ns), Inion (In), Left (PAL) and Right (PAR) pre-auricular points. The intersection of the longitudinal (Ns-In) and lateral (PAL-PAR) is named the Vertex.

The figure on the next page describes the definition of all other positions, and their names.



The original 10-20 system included only 19 electrodes (see panel B of the figure). Later on, extensions were proposed so that now you can place over 70 electrodes in standard positions (see panel C of the figure). This extension also renamed four electrodes (marked in black in the figure); the original names were: T3, T5, T4, and T6 for T7, P7, T8, and P8, respectively.

Sometimes, one of the electrodes mounted in these positions is used as reference channel. More often, ear lobe or mastoid (i.e. bony outgrowth behind the ear) are used.

Data Formats

Mario Montage File

The current version of `mmf` files only allows the list of standard labels to be specified, together with the indication that irrelevant (or bad) channels are present in the data.

The `mmf` file is a text file. The following is the montage used throughout this tutorial:

```
list
Fp1  1
Fp2  2
F3   3
F4   4
C3   5
C4   6
P3   7
P4   8
T7   9
T8  10
CP3 11
CP4 12
Fz  13
Pz  14
Cz  15
Oz  16
```

On the first line the keyword `list` must appear.

Then, each line describes a different data channel: the label appears first, then the ordinal number, separated by a space.

If you want to exclude P3 from further computations, the corresponding line must report the keyword badch, as in the following example:

```
list
Fp1  1
Fp2  2
F3   3
F4   4
```

```
C3    5
C4    6
badch
P4    7
T7    8
T8    9
CP3  10
CP4  11
Fz   12
Pz   13
Cz   14
Oz   15
```

In the same way, it is possible to describe that the seventh channel in the dat file contains polygraphic signals that thus must not be considered.

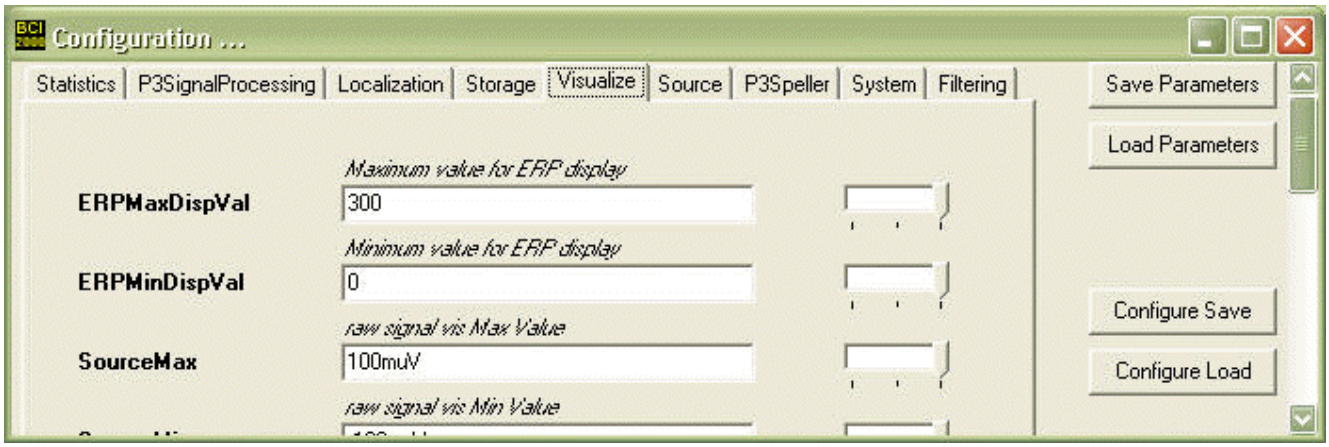
Future versions of this file will allow the definition of electrode positions.

BCI2000 parameter file

This topic is specifically covered in the BCI2000 documentation. Here we only report information relevant to creation of `prm` files that can work around difficulties with data analysis due to inappropriate settings of default parameters, and of prm files created by the Mario that can be fed back into BCI2000.

`prm` files are plain text. Every line contains the description of a different parameter. The format of the line depends on the datatype of the parameter.

You can load and save parameter files using the buttons at the top right corner of the Configuration window. Advanced load/save modalities can be activated by using the Configure Save and Configure Load buttons



The most important feature of `prm` files is that BCI2000 supports partial parameter files, meaning that you can prepare several files that only affect the value of one part of the setup. This is particularly useful if, for instance, you

want to use a setup with a different EEG Source. In that case you simply load the usual prm file, and then you load a second prm that only contains the settings for the new source module.
Another importan case in which you want to use partial prm files is when you want to change EEG features for cursor control or spelling applications: the Mario analysis program allows you to save a partial prm file for this purpose.

Syntax

The general syntax of the line is:

Section	Type	Name=	Value	// description of the parameter

The Value field is itself made of four elements:

Actual	Default	Minimum	Maximum

The Actual Value part of the line can have very different contents, according to the datatype. Examples are reported below

Integer:

```
UsrTask int FeedbackDuration= 20000 0 0 0 // Max Trial Duration
```

Float:

```
Targets float StartCursorX= 0.0 0 0 100.0 // Horizontal Start of
Cursor
```

Matrix (no labels):

```
Statistics matrix WeightControl= 4 1 Xadapt Yadapt AdaptCode
ResultCode 0 0 0 // State Names controlling Adaptation
```

Matrix (row and column labels):

```
Targets matrix TargetPos= { x%20start y%20start x%20length
y%20height x%20code y%20code adapt%20code } { target%201
target%202 } 90 90 0 51 10 10 50 50 0 0 -1 1 3 3 0 0 0 // Target
Position Matrix - Values are 0-100
```

Other datatypes are described usign primary datatypes, adding the necessary information in parentheses at the end of the comment field.

Booleans:

```
Storage int SavePrmFile= 0 1 0 1 // save additional parameter file
(0=no, 1=yes) (boolean)
```

File paths:

```
Storage string FileInitials= ..\tutorial\data\D2box a z 0 // path
to top level data directory (directory)
```

Enumerations:

```
Source int SignalType= 0 0 0 3 // numeric type of output signal:
0: int16, 1: float24, 2: float32, 3: int32 (enumeration)
```

For more information, please consult the BCI2000 Project Outline.

Montage Instructions

Following the flow of information, the biological (ionic) currents must be transduced into electronic currents. This task is carried on by the electrode-electrolyte interface. In general, the electrode picks up the scalp potential, but if you go a little deeper, you will realize the importance of the gel and how it can be source of artifacts.

For the reasons that we will cover later on (see determining the most relevant channel, artifact detection and spatial filtering) it is mosten often not sufficient to only acquire one channel. In this tutorial we will acquire from 16 channels, but numbers between four and 64 are all reasonable. Two more electrodes (named Reference (Ref) – and Ground (Gnd)) are needed for the electronics of the amplifiers to work properly (i.e with high CMRR)

Sticking 16 (or worse 64) individual electrodes on the scalp of a subject is not a simple task. And doing it with high geometrical accuracy is a task for very experienced technicians. This is why we use EEG caps. They are made of elastic fabric (available in different sizes), and electrodes are already fixed in the proper configuration. One proven technique to perform an accurate montage is the following:

- find the vertex on the scalp of the subject and mark it
- find the position for Fpz and Oz and mark them
- find the Cz electrode on the EEG cap and place on the vertex
- keep Cz still, and slip the cap onto the head.
- paying attention that Cz does not shift, adjust the cap so that:
    - the Fz, Cz Pz line is on the midline
    - the Fp1-Fp2 line is horizontal and at the level of the Fpz mark
    - the O1-O2 line is horizontal and at the level of the Oz mark

You can now fix the Ref and Gnd electrodes (usually on the earlobes, with a specific electrode or on the mastoid, with cup electrode and collodion; some caps have electrodes for Gnd embedded on a scalp position).

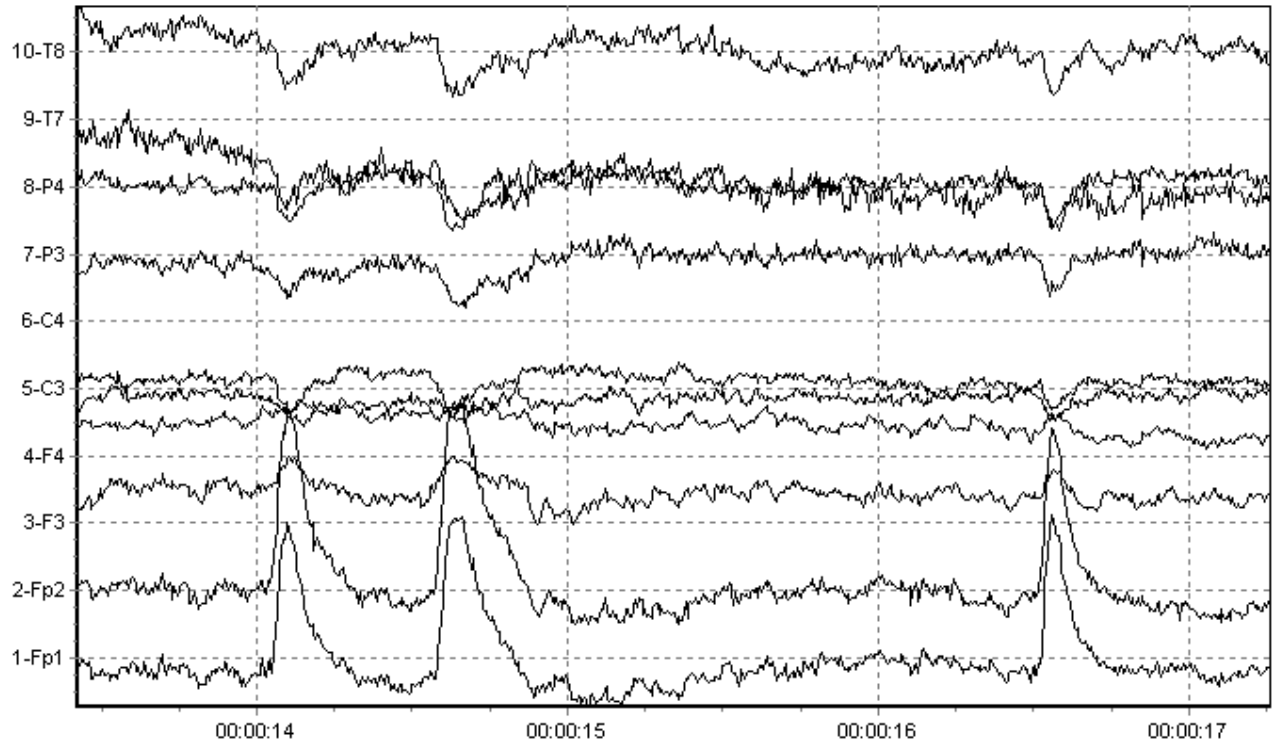
Reasonable EEG traces should now be visible on the screen as soon as gel is injected. Visual inspection can detect the effetct of poor contact - excessive mains disturbance, frequent loss of contact, etc.). You may also want to use an impedance meter to make sure that electrodes have good contact (i.e. that their impedance is low).

EEG Artifacts

Blink artifacts

Blink artifacts are generated by the dipole layer that is physiologically present on the cornea and on the retina. Closing the eyelids produces a fast movement that is detected as a positive peak lasting a few tenths of a second, mainly visible on the frontopolar derivations, but often perceivable on all the electrodes of the montage.

Their frequency content is negligible in the alpha band, so they produce no apparent effect on the Mu data analysis (if their number is reasonable). At the same time their amplitude is quite large so that time domain analyses (as P300 detection) can be strongly influenced by the presence of this kind of artifact.

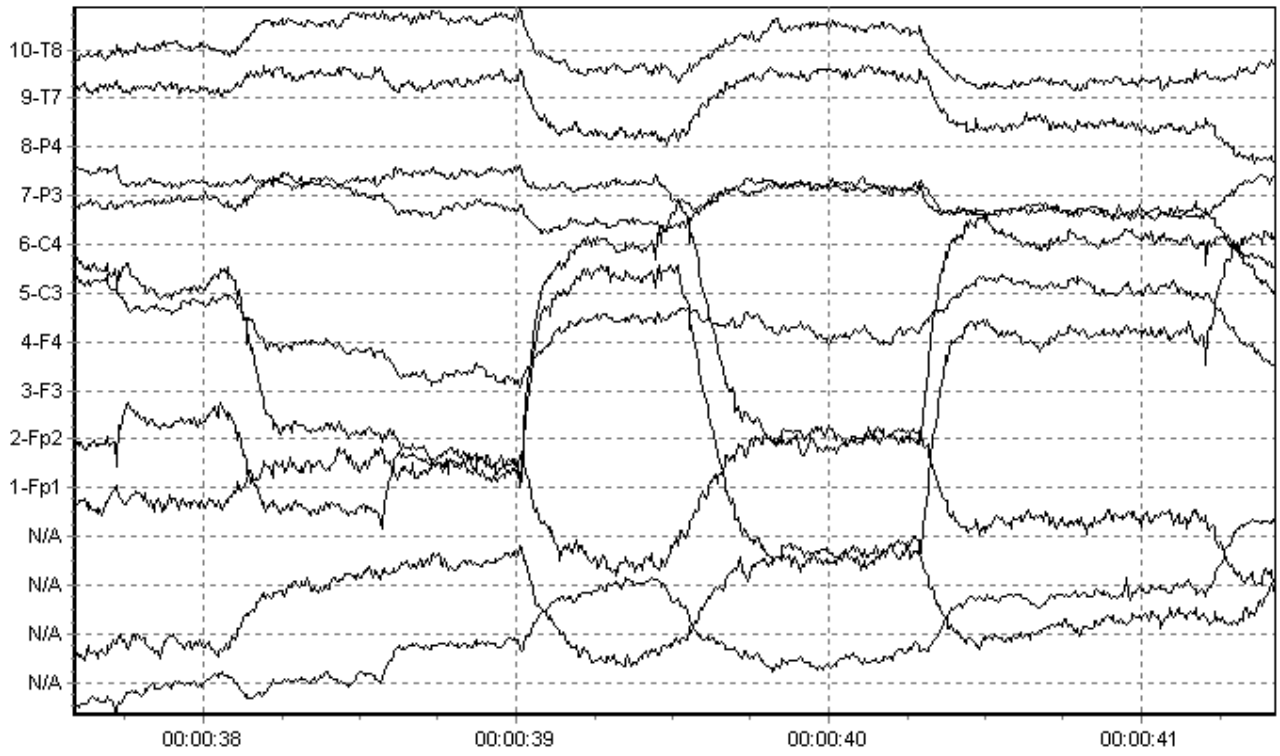


Electrooculographic artifacts

EOG artifacts are generated with similar mechanism to those underlying blink artifacts and are typically produced by eye movements.

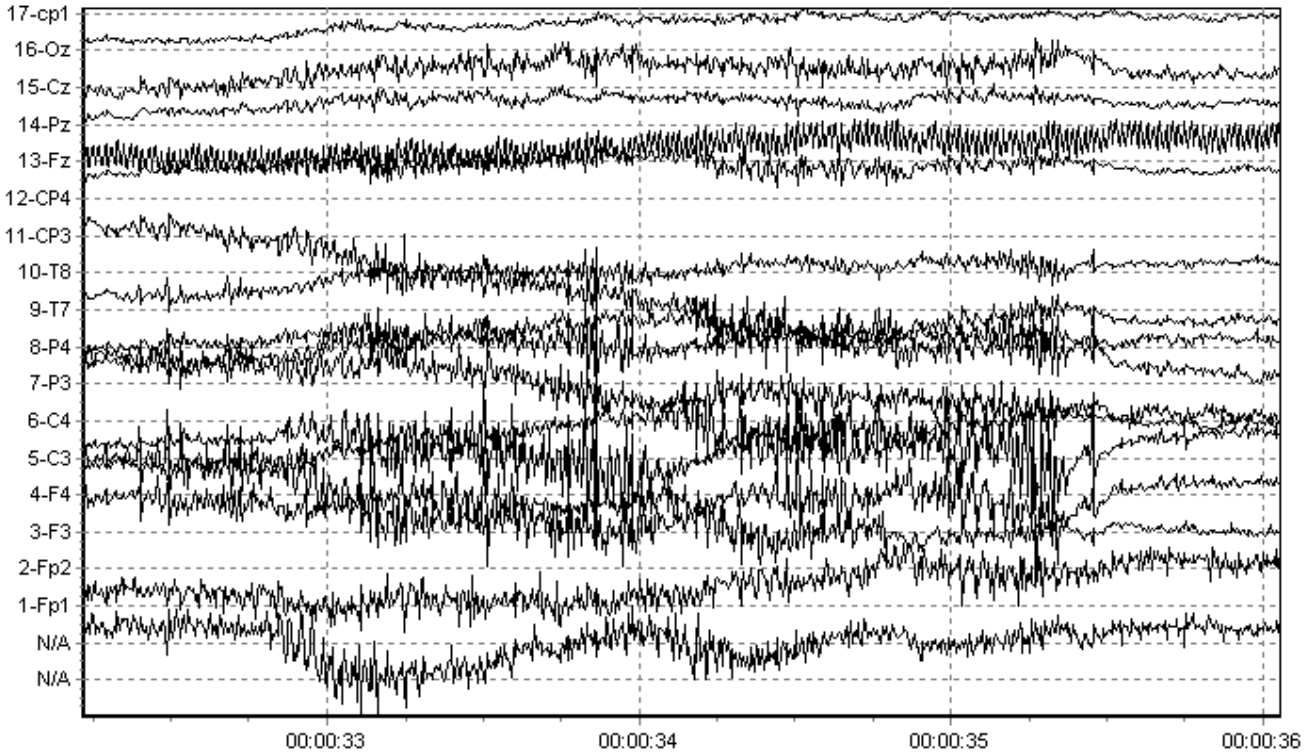
The effect on frontopolar and frontotemporal electrodes can be symmetric or antisymmetric, depending whether the movement was vertical or horizontal, respectively.

Their effect on frequency- or time-domain analysis is of the same kind of the effect of blink artifact except that the affected frequency band is at even lower frequencies, while amplitude can even be higher.



Electromyographic (EMG) artifacts

EMG activity must be carefully checked at the beginning of each recording, and verified throughout the recording. Its effect can completely obscure any frequency analysis. The most common sources of EMG are the muscles that lift the eye lids and those which close the jaw. Both groups are inadvertently contracted as a consequence of a psychological effort. Keeping the mouth slightly open (or the tip of the toungue between the foreteeth) is a good strategy to avoid jaw-generated EMG.


17-cp1
16-Oz
15-Cz
14-Pz
13-Fz
12-CP4
11-CP3
10-T8
9-T7
8-P4
7-P3
6-C4
5-C3
4-F4
3-F3
2-Fp2
1-Fp1
N/A
N/A
00:00:33
00:00:34
00:00:35
00:00:36