EEGLAB, MPT, NetSIFT, NFT, BCILAB, and ERICA: New tools for advanced EEG/MEG processing

Arnaud Delorme\textsuperscript{1,2,3}, Tim Mullen\textsuperscript{1,4}, Christian Kothe\textsuperscript{1}, Nima Bigdely-Shamlo\textsuperscript{1}, Zeynep Akalin Acar\textsuperscript{1}, Andrey Vankov\textsuperscript{1}, Scott Makeig\textsuperscript{1}

1. Swartz Center for Computational Neuroscience, Institute for Neural Computation, University of California San Diego, La Jolla CA, USA
2. Université de Toulouse, UPS, Centre de Recherche Cerveau et Cognition, Toulouse, France
3. CNRS, CerCo, Toulouse, France
4. Department of Cognitive Science, UCSD, La Jolla, CA, USA

Abstract

We review a set of complementary EEG data collection and processing tools recently developed at the Swartz Center for Computational Neuroscience and connecting to the EEGLAB software environment (sccn.ucsd.edu/eeglab), a freely available and readily extensible processing environment running under Matlab (The Mathworks, Inc.). These new tools include: (1) a new and flexible EEGLAB STUDY design facility for framing and performing statistical analyses on data from multiple subjects; (2) a Measure Projection Toolbox (MPT) for aggregating independent component processes across subjects; (3) a Network Source Information Flow Toolbox (NetSIFT) for modeling event-related information flow between cortical areas; (4) a Neuroelectromagnetic Forward head modeling Toolbox (NFT) for building realistic electrical head models from available data; (5) a BCILAB toolbox for building online brain-computer interface (BCI) models from available data, and; (6) an Experimental Real-time Interactive Control and Analysis (ERICA) environment for real-time production and interactive coordination of multimodal experiments.

A variety of new signal processing methods have been applied to EEG signal processing over the past 15 years [1]. These new methods require new tools to allow routine processing of EEG data. Here we present a collection of recent advances implemented in or connecting to the EEGLAB software environment [2]. EEGLAB is an interactive, GUI menu and commandline script-based environment for processing electrophysiological data. Since its introduction in 2001, it has become a widely used platform for time series processing of biophysical data and for sharing of new signal processing approaches. A number of new EEGLAB-associated tools have been introduced that connect to the EEGLAB environment: NFT [3] is a new toolbox for electrical head modeling, an essential first step in electrophysiological source localization. NetSIFT allows users to implement a wide range of recently published methods for assessing effective connectivity between EEG signal processes including independent component processes. MPT is an EEGLAB plug-in for aggregating and comparing ICA components across subjects. Finally the ERICA framework, composed of the Datariver, Matriver, Enactor, and Producer toolboxes, as well as the BCILAB toolbox manage real time processing of EEG data and delivery of sensory feedback to the subject(s), system, and/or experimenters [4]. Figure 1 depicts how these toolboxes interact in connection with distributed data storage solutions such as the proposed HeadIT data and tools resource [5].
Figure 1. Complete electrophysiological experiment control, data collection, analysis, archiving, and meta-analysis suite (the ERICA framework for data recording, online analysis, and stimulus control environment; EEGLAB for data analysis; BCILAB for online and offline classification and BCI; HeadIT, an archival data and tools resource for laboratory or archival data storage, retrieval and meta-analysis; NFT for source localization; NetSIFT for information flow modeling; MPT for independent component process comparison).

EEGLAB

EEGLAB is an interactive menu based and scripting software for processing electrophysiological data based under the interpreted programming language Matlab [2]. EEGLAB provides an interactive graphic user interface (GUI) allowing users to flexibly and interactively process their high-density electrophysiological data (from up to several hundreds of electrodes) and/or other dynamic brain time series data. EEGLAB implements common methods of electroencephalographic data analysis including independent component analysis (ICA) and time/frequency analysis. EEGLAB has become a widely used platform for time series processing biophysical and sharing new techniques. At least 28 plug-in functions have been implemented and released by groups of users. Here we describe recent developments in EEG software inter-operative with EEGLAB. Several new tools are Matlab applications that conveniently plug in to the EEGLAB menu (or may also be run as stand-alone applications).

Key EEGLAB features include:
(1) EEGLAB’s event structure and functions for importing, editing and manipulating event information. In particular, users can select (sub)epochs time locked to classes of events, and can sort trials for visualization based on values in any event field (for instance subjects’ reaction time).

(2) EEGLAB’s pioneering software for independent component analysis (ICA) decomposition of electroencephalographic data [6]. Though ICA data analysis methods have been since incorporated into most commercial software processing EEG data (BrainVision, Neuroscan, BESA), EEGLAB has the most extensive repertoire of processing and data evaluation tools for ICA-based data analysis.

(3) EEGLAB’s adaptability to different levels of users and its unique “history” features that build scripts as users navigate through menus, allowing users to ‘replay’ or extend their data processing in simple-to-use Matlab scripts. Depending on their level of Matlab expertise, users can either interact only with the EEGLAB graphic interface (GUI), else they can call EEGLAB functions directly from the Matlab command line or write their own Matlab scripts using modular EEGLAB functions and structures.

(4) EEGLAB’s truly open source philosophy, allowing any researcher to contribute (and distribute) plug-ins to EEGLAB that appear automatically in the EEGLAB menu of their users. This structure ensures stability of the core code that currently only a handful of expert users modify while, at the same time, allowing free contribution of new algorithms and methods by other users.

EEGLAB comprises more than 400 Matlab functions for a total of more than 50,000 program lines. First developed under Matlab 5.3 running on Linux, EEGLAB currently runs under Matlab v7 on Linux, Unix, Windows, and Mac OS. Since the Matlab program is not free itself, we have also compiled the code for the EEGLAB software using the Matlab compiler for those users who do not have access to Matlab. To our knowledge, 28 user-initiated plugins have been developed for EEGLAB. The EEGLAB tutorial comprises more than 300 pages of documentation. In addition, each of the 400 stand-alone modular EEGLAB functions contains its own documentation. EEGLAB has been downloaded more than 65,000 times from 88 country domains since 2003; 8,392 unique opt-in users are currently on the EEGLAB mailing lists.

**STUDY.design**

The STUDY.design concept was introduced in June, 2010 in EEGLAB v9. Complex experiments require researchers to be able to define custom sets of independent variables within an experiment. The new STUDY.design structure in EEGLAB allows users to freely define independent and dependent variables and to analyze data channel as well as independent component activities or clusters using ERP, power spectrum, ERSP [7] and ITC [8] measures.
For example, a STUDY might contain data sets for two sets of conditions from two groups of subjects (a 2x2 [condition, group] design). Statistical comparisons might be targeted to look at main effects and interactions in this design, or separately at contrasts between selected (1x2) pairs of conditions or groups. Figure 2 shows the EEGLAB STUDY.design graphic interface by means of which users can create new designs and select their independent variables.

Building a STUDY.design involves multiple steps. Users begin by pre-processing binary EEG data sets for each subject; this involves importing raw data, re-referencing, filtering and removing artifacts. Once these data sets have been pre-processed, users then have to import all subject data sets into a STUDY. Selecting a STUDY.design for analysis then allows processing of data trials time locked specific event types for each subject. For example, in an oddball paradigm comprised of trials time locked to target, distractor, and standard stimuli, users might want to contrast these three types of trials using a 3x1 design. Alternatively, they might want to contrast distractor and target stimulus-locked trials, considered together, with responses to standard stimuli. The STUDY.design feature of EEGLAB allows users to investigate such contrasts in a natural way. In a STUDY with N subject groups, the STUDY design scheme also allows users to look at group effects for each type of stimuli using a 2XN design.

All of the above design concepts and more may be implemented within a single STUDY using multiple STUDY.design specifications. Finally, use of multiple designs may also be useful for testing different signal processing options. For instance, one might create two identical STUDY designs, one computing time/frequency measures using FFTs and the other using wavelets. Once computed, the user would be able to toggle between designs to compare the two types of time/frequency decomposition.

Figure 2. EEGLAB STUDY design interface using the tutorial STUDY available on the EEGLAB wiki page (sccn.ucsd.edu/wiki/eeclab). The three push buttons at the top may be used to add a
new design ("Add design"), rename a design ("Rename design"), or delete a design ("Delete design"). The "Independent variable 1" list helps define independent variables. The list of independent variables is automatically generated based on the STUDY definition information and individual data set event types. For a given independent variable, it is also possible to select a subset of its values or to combine some of its values. For instance, in this example the user has selected ‘ignore’ and ‘memorize’ stimuli as values for the independent variable ‘condition’. The "Subject" list contains the subjects to include in a specific design. Un-selecting a given subject from the list excludes him/her from further data analysis within the design. Once a design is selected, measures including ERPs, mean spectra or event-related spectral perturbations (ERSPs) may be plotted. Here, we have plotted the ERSP of an independent component cluster in the selected STUDY.design. In the top right panel, the scalp maps of one independent component cluster are shown – the large map representing the average scalp map. In the bottom right panel, mean cluster ERSPs are shown for Ignore versus Memorize letter trials, and significant differences are assessed using permutation-based statistical methods and a false discovery rate (FDR) method to correct for multiple comparisons.

EEGLAB uses statistical tools such as surrogate and parametric statistics to perform hypothesis testing on STUDY designs. Surrogate tests involve bootstrap and permutation methods. Depending on the design type, statistical hypothesis testing using t-test, one-way anova or two-way anova - or their surrogate-data equivalents - are performed for paired data or unpaired data designs. Finally, the FDR (False Discovery Rate) algorithm is applied to correct for multiple comparisons [9]. Using these simple yet powerful statistical tools, EEGLAB allows comparison of multiple experimental designs applied to a given data STUDY.

The NFT toolbox for electrical head modeling and source localization

Our previous work has shown that some ICA component processes are highly compatible with activity generating compact cortical domains of local field synchrony that may be localized in the brain [1, 10, 11] using a four-shell spherical model or the standard boundary element method (BEM) head model included in the EEGLAB Dipfit plugin [12]. Analysing electrical activity from independent component processes, however, allows more precise localization methods. For accurate source localization one needs to use a realistic head model that represents the electrical and geometric properties of the head correctly. NFT adds a realistic head modeling framework to the spherical and MNI head models already provided by EEGLAB. The NFT framework automates most of the tasks needed to generate a realistic head model from MRI images, or EEG sensor coordinates, and provides advanced Boundary Element Method (BEM) and Finite Element Method (FEM) solvers for computing fields for a given source distribution (i.e., the forward problem) [3].

NFT is accessible from the EEGLAB graphic interface configured as an EEGLAB plug-in. The toolbox provides both a Matlab commandline and graphical user interface for generating realistic head models from available subject information, and for solving the forward problem numerically to provide a lead-field-matrix for a given source space and sensor distribution. This
makes it easy to integrate a forward head model produced by NFT into any inverse source localization approach.

NFT performs the following steps:

1. **Segmentation of MR images:** If a 3-D whole-head structural T-1 MR image of the subject's head is available, the toolbox can segment the scalp, skull, CSF, and brain tissues.

2. **High-quality head models:** The accuracy of numerical solutions to an inverse source localization problem depends on the quality of the underlying meshes that model tissue conductance-change boundaries. NFT can create high-quality surface meshes from segmented MR images for use in BEM head model. FEM meshes may be generated from the BEM surface meshes using Tetgen tool [13]. Two examples of FEM and BEM meshes generated using NFT are shown in Figure 4.

3. **Warping a template head model:** While use of a subject whole-head MR image is the preferred way to generate a realistic head model, such an image may not be available for many subjects. NFT can generate a semi-realistic head model of the subjects' head by warping a standard template head model to the digitized 3-D electrode coordinates, when these are available.

4. **Registration of electrode positions with the mesh:** NFT has a two-step (manual and automatic) registration function for aligning the digitized electrode locations to the scalp mesh.

5. **Accurate high-performance forward problem solution:** The NFT uses high-performance BEM and FEM implementations from the open source METU-FP Toolkit (eee.metu.edu.tr/metu-fp) [14, 15] for bioelectromagnetic field computations.

![Figure 4: Two examples of (a) a BEM mesh and (b) a FEM mesh.](image)
FEM modeling is a recent addition to NFT that will be present in the next release. By converting the high-quality BEM surface meshes into volume meshes, it is possible to seamlessly integrate FEM into NFT as an alternative numerical solver. In the future, the NFT model will be able to incorporate current anisotropy based on white-matter distribution information in diffusion weighted imaging (DWI) head images co-registered with structural MR head images.

We have successfully used NFT to model realistic cortical source spaces that included a large number of dipolar elements that we assume to be oriented perpendicular to the local cortical surface. The cortical surface was extracted from subject MR head images using the tessellated FreeSurfer gray and white matter surfaces [16]. We created a multi-scale cortical patch basis on this surface by selecting seed points (single voxel dipoles), then extend each patch conformally to a set of gaussian-tapered patches with areas in the range ~50-200 mm² [17]. The patch-based source model generation will be integrated into NFT in future releases. NFT thus allows performance of precise source localization of independent component processes based on accurate electrical current flow models consistent with the individual subject head anatomy.

**A measure projection toolbox for source-level data analysis**

Once independent component (IC) processes have been isolated from high-dimensional EEG data using ICA methods, their activities characterized, and their positions localized via equivalent dipole localization, comparing ICs across subjects may begin with component clustering methods to find equivalence classes of corresponding component processes across subjects (as shown in Figure 2). By default, EEGLAB implements a k-means clustering method to group ICs across subjects and/or sessions based on similarities in their scalp maps, equivalent dipole locations and/or time-domain and/or time/frequency-domain activities.

Measure Projection is a promising alternative to independent component clustering that may be more powerful than classical k-means data reduction. MP defines a high density grid over the brain volume in standardized coordinates (e.g., those of the standard MNI brain) (Montreal Neurological Institute) and searches for regions (neighborhoods) in which component processes, across subjects and/or sessions, have consistent event-related dynamics. A local-mean projected EEG measure (e.g. ERP, ERSP, etc.) is assigned to each grid point that has significantly high measure consistency across subjects in its (Gaussian) neighborhood. Projected EEG measures represent best estimates of the expected response or other dynamics at given cortical locations. In essence, Measure Projection may be understood as local 3-D spatial averaging of EEG source measures relying on a Gaussian representation of the independent component (IC) equivalent dipole densities in the MNI brain space, accompanied by statistical significance analysis. Figure 3 shows an example of projected mean ERSPs for data epochs containing a transient increase in theta band power (4-7 Hz) in a Rapid Serial Visual Presentation paradigm [18]. Here, the equivalent dipole locations are grouped into a schematic 2-D representation not reflecting their actual 3-D locations.
Clusters of ICs associated with multiple subjects conceptually have a spatially discontinuous distribution. Their membership is highly dependent on clustering parameters such as the number of clusters or the relative weights of the various EEG measures that may be used in the clustering. By contrast, Measure Projection uses a common continuous brain space in which subject EEG processes, and their statistical perturbations around classes of experimental events are represented. The continuous nature of brain space makes it possible to apply new computational methods to EEG data organized using EEGLAB at the STUDY level.

For example, event-related dynamics from multiple subjects can be compared for a brain region of interest (ROI) (e.g., right occipital cortex). The Measure Projection toolbox facilitates the use of this method in a user-friendly and structured manner. It employs current EEGLAB STUDY measure pre-computation, statistics, and plotting functions, and offers new visualization and analysis methods. Currently we are comparing Measure Projection results with those of different component clustering methods, and will present results in a future report.

Figure 3. A projected mean ERSP (right panel) is a weighted sum of mean ERSP transforms of independent component epochs from multiple subjects. It reveals a post-event 4-7 Hz theta band power increase (centered near 400 ms) following detected target image presentations in a Rapid Serial Visual Presentation (RSVP) paradigm.
An electrophysiological information flow toolbox

Once activity in specific brain areas have been identified using source separation (e.g., ICA), localized (using NFT) and clustered across subjects (using MPT), it is possible to look for transient changes in the independence of these different brain source processes. Advanced methods for non-invasively detecting and modeling distributed network events contained in high-density scalp EEG data are highly desirable for basic and clinical studies of distributed brain activity supporting behavior and experience. In recent years, Granger Causality (GC) and its extensions have increasingly been used to explore ‘effective’ connectivity (directed information flow, or causality) in the brain based primarily on observed ongoing or event-related relationships between channel waveforms. Based on the prediction error of autoregressive (AR) models, a process (A) is said to granger-cause another process (B) if past values of process A, in addition to past values of process B, help to predict future values of process B beyond what can be achieved by using past values of process B alone [19]. While many landmark studies have applied GC to invasively recorded local field potentials and spike trains, a growing number of studies have applied GC to non-invasively recorded human EEG and MEG data (as reviewed by Bressler [20]).

The GC concept has also been extended to an arbitrary number of signals by the use of multivariate AR (MVAR) models. Using this approach, from the MVAR coefficient matrices, we can derive the transfer and spectral matrices, and ordinary, multiple, and partial coherences, where the latter quantity expresses the amount phase coherence between two channels after subtracting out the part of the interaction which can be explained by a linear combination of all other channels. We can then derive useful measures of directed interdependence closely related to Granger’s definition of causality such as the directed transfer function (DTF) and partial directed coherence (PDC). These and related estimators can describe different aspects of network dynamics and thus comprise a complementary set of tools for MVAR-based connectivity analysis within the well-established and interpretable framework of GC [21]. To study transient causal dynamics of non-stationary phenomena, an adaptive MVAR (AMVAR) approach may be applied over locally-stationary sliding windows [22]. This approach can be used to explore finely-resolved time- and frequency-dependent dynamics of directed information flow or causality between neuronal sources during cognitive information processing. Baseline significance levels for causal influence are typically obtained by a modification of a surrogate ‘phase randomization’ algorithm [23]. This and other bootstrap, permutation, and analytical tests can be used to establish rigorous confidence intervals on estimated connectivity. For additional details, see Kaminski [21].

NetSIFT is a collection of tools for modeling and visualizing information flow dynamics between identified sources of EEG data, generally after separating the data into (instantaneously) maximally independent (IC) processes. The toolbox currently consists of four modules, (1) data pre-processing, (2) model fitting and connectivity estimation, (3) statistical analysis, (4) visualization. The second module currently includes support for parametric (AMVAR) and non-parametric (spectral matrix factorization) estimation of a wide range of coherence and granger-
causal estimators published to date. The third module includes routines for parametric and non-parametric significance testing (phase-randomization, bootstrap statistics, permutation tests, and analytic statistics for PDC and DTF routines). The fourth module contains novel routines for interactive visualization of information flow dynamics across time, frequency and anatomical source location. Group analysis in the source domain can be performed by clustering sources to obtain regions of interest (ROIs) and then estimating average causal influence between ROIs across a population (Figure 5). We are currently developing a novel Bayesian approach for joint estimation of the most likely source locations and connectivity graph -- with robust confidence intervals -- across a subject population [24]. A GUI allows easy access to the NetSIFT data processing pipeline.

A key aspect of NetSIFT is that it focuses on estimating and visualizing effective connectivity in the source domain rather than directly between scalp electrode signals. This should allow us to achieve finer spatial localization of the network components while minimizing the challenging signal processing confounds produced by broad volume conduction from cortical sources (as well as non-brain sources) to the scalp electrodes. We are currently evaluating the relative suitability of various source estimation algorithms when combined with MVAR-based connectivity algorithms, and will further develop the toolbox accordingly.

NetSIFT may help find transient, dynamic network events that link spatially static component processes (Figure 5). We will soon incorporate additional dual Kalman filter-based routines for detecting transient granger-causal couplings in non-stationary processes. The modular architecture of the toolbox is also designed to allow easy addition of new methods from the user community. The toolbox may also be used for effective connectivity analysis and visualization of phenomena in electrocorticographic (ECoG) data, e.g. to identify sources and directions of information flow at onsets of and during epileptic seizures.
Figure 5. EEG-based brain connectivity measures using NetSIFT. (A): An interactive time-frequency grid demonstrating transient bursts of theta information flow (estimated using the direct DTF (dDTF)) between 12 clusters of IC sources calculated across 24 subjects during error commission. (B) One frame of an interactive brain movie showing an event-related causal relationship in the theta band between these clusters (250 ms after an erroneous button press is made). (C) Similar to (B) but using a different visualization mode showing the net causal inflow (green) and causal outflow (red) in the theta band at each brain location. This does not require source clustering but uses a similar approach to the Measure Projection concept described in the previous section.
The Experimental Real-time Interactive Control and Analysis (ERICA) framework

The analysis framework described above allows exploration of source-based methods for advanced brain-machine interfaces and real-time EEG processing and feedback methods. For this purpose, we have developed an online EEG and multimodal data collection, processing, and interactive feedback environment, ERICA. Processing of EEG data in real-time software applications first and foremost requires access to data. Once the data are acquired, they may then be streamed into online data processing processes (e.g., BCI) whose output, combined into the captured data stream (or ‘river’) can be used to control or adapt ongoing stimulation processes.

The ERICA framework is a distributed data acquisition, synchronization, online processing, and stimulus delivery environment based on a unique streaming data management and real-time cross-platform synchronization engine called DataRiver developed from an ADAPT data acquisition and stimulation control language [25]. Producer is an ERICA application that interacts with DataRiver to control stimulus presentation in a flexible way using an original scripting language. MatRiver, another ERICA component and DataRiver client, allows direct read/write access to and processing on DataRiver data streams from within Matlab processes.

The central application driving development of ERICA is our development of Mobile Brain/Body Imaging (MoBI) data acquisition and analysis [26] -- the simultaneous study of what the brain is doing (via EEG), what the brain is sensing (via audiovisual recording), and what the brain is controlling (the totality of our behavior, recorded via body motion capture and eye tracking) in natural 3-D task environments.

To allow real-time analysis, data streams acquired by separate devices first need to be synchronized. Such streams are, by definition, asynchronous, even when they are acquired at the same nominal sampling frequency. This is because, typically, independent clocks are used for data acquisition in each device. In addition, the sampling rates for different data sources may differ significantly. For example, while EEG is usually sampled between 250 Hz and 2,000 Hz, body motion capture or subject behavioral responses may be acquired at much lower rate, and video and audio data streams at still higher rates. For synchronization purposes, another important challenge is dealing with sporadic delays introduced by equipment acquisition, network, and operating system buffers that ensure overall regularity of data samples at the cost of ms-level predictability. For data acquired through an IP socket connection, network delays may be significant and constantly varying. Finally, Windows or any other multitasking operating system introduces variable delays in the processing of asynchronous flows – in a multitasking system, data are most often processed only when the corresponding thread is activated and not when the data first become available.

DataRiver was developed in an attempt to solve these synchronization problems. DataRiver is a flexible and universal high-precision synchronization engine, providing a strong and near real-
time synchronization of simultaneous data streams. It has been designed and tested with accuracy of better than 2 ms, even when synchronizing data acquisition streams from different computers (running Windows, Unix, Linux, or MacOS) over a Local Area Network or Internet subnet.

![Diagram of ERICA data flow](image)

**Figure 6.** ERICA data flow involving two separate computers each running an instance of DataRiver. Dashed lines indicate control signals. Computer visualization is performed using the MatRiver DataRiver client under Matlab.

The flexibility of the ERICA framework stems from its modular design - data output from a variety of devices are managed by specialized device drivers that convert each data stream into a device-independent stream. These streams are then merged into real-time as in a "river" (hence the name DataRiver). DataRiver device drivers are currently available for several types of input devices and data systems including Biosemi EEG, PhaseSpace and Optitrak motion capture systems, eye trackers, and the Wii remote (Nintendo, Inc.). This enables the rapid development of a wide range of experimental paradigms that can be tailored for a variety of multimodal experimental or application environments. Data from incoming DataRiver data streams may be used in real time by clients for recording, online data processing, and/or to provide feedback to the subject(s) being monitored. DataRiver has integrated support for data exchange in real time between one or more remote computers connected to a local area network (LAN), enabling distributed and cooperative experiments (Figure 6). New drivers and online data processing applications can easily be added to DataRiver to meet evolving research needs.

MatRiver is a MATLAB DataRiver client optimized for real time EEG data processing, buffering and visualization using the OpenGL-based 3-D Simulink environment (The MathWorks, Inc.).
MatRiver communicates with DataRiver by calling a binary library of functions (currently running under Windows OS). MatRiver allows online performance of common EEG preprocessing steps such as channel selection, channel re-referencing, frequency filtering and linear spatial filtering using a pre-defined ICA source signal unmixing matrix [6]. Most often, these steps may be accomplished in near real time by directly calling relevant EEGLAB functions. MatRiver also includes routines to dynamically detect "bad" channels and compensate for them by taking into account a linear ICA source propagation model. Preprocessed channel or independent component (IC) signals are accumulated and can subsequently be used for classification using MATLAB tools such as BCILAB (see following). MatRiver uses Matlab “timers” to run in the background allowing real-time processing in a non-blocking manner, even including near real-time interactive exploration of the incoming data from the Matlab command line. Continuous visualizations of data characteristics such as alpha band energy are also possible. In short, Matriver functions provide an elegant and straightforward pipeline for EEG pre-processing and classification using the rich tool set and programming simplicity of MATLAB.

**Designing brain-computer interfaces with BCILAB**

After results of data stream synchronization and preprocessing have been accomplished within the ERICA framework, one may use BCILAB, an open-source MATLAB toolbox and EEGLAB [2] plug-in, to support Brain-Computer Interface (BCI) research, and more generally, the design, learning (or adaptation), use, and evaluation of real-time predictive models operating on signals. The main objects of study in BCILAB are Brain-Computer Interface (BCI) models [27], generally defined as systems that take human bio-signals as input, and output estimations of some aspect of the person's cognitive state. The signals processed by BCIs are traditionally restricted to EEG signals, but may include other modalities, such as motion-capture data or skin conductance (plus context parameters such as vehicle state, previous events, etc.). These data can be processed either using BCILAB running as a data processing node in a real-time experimentation environment (e.g., ERICA), or offline in offline or simulated real-time applications to existing data. The classifier outputs of a BCI can be streamed to a real-time application to effect stimulus or prosthetic control, or may be derived post-hoc from recorded data, for example for statistical analysis of the model’s prediction accuracy when applied to a database of previously recorded data. BCILAB imposes no restrictions on the outputs, so that most accessible cognitive states can be investigated, for example imagined movements (affecting in sensorimotor mu rhythms), surprise (provoking, e.g., the oddball P3), or indicators of drowsiness.

The tools provided by BCILAB facilitate most steps in BCI research, including the design, implementation, learning, evaluation, and on- or off-line application of BCI (or other) models. Further tasks, including the exploration of recorded data and visualization of model parameters may be supported using EEGLAB tools. BCILAB has several layers, the top layer including a GUI (under development), a scripting interface, and a real-time application interface, with a second layer including core model learning, model execution, and model evaluation functions. These core facilities in turn rely on a framework of "BCI paradigms", which can be understood as prototypical template-like approaches to designing a BCI model. Pre-defined paradigms
include Common Spatial Patterns (CSP), logarithmic Band-power estimates (log-BP), and the approach proposed in the Dual Augmented Lagrange framework [28]. A BCI 'paradigm' defines the entire approach as it would be described in a publication, from raw data to final outputs, and usually involves both a learning and a prediction stage, because sufficient performance can often only be achieved after a model is learned (or calibrated) based on sample data from a given session, subject, or task. BCI paradigms can be fully customized by the user, including removal or addition of entire components, but come with defaults for all their parameters, both to keep the learning curve gentle as well as to minimize the amount of information that must be specified.

Table 1. Signal processing, feature extraction, and machine learning algorithms included in the BCILAB/EEGLAB framework.

<table>
<thead>
<tr>
<th>Signal processing</th>
<th>Feature extraction</th>
<th>Machine Learning Algorithms</th>
</tr>
</thead>
<tbody>
<tr>
<td>· Channel selection</td>
<td>· Multi-window averages [33, 34]</td>
<td>· Linear Discriminant Analysis (LDA) [40]</td>
</tr>
<tr>
<td>· Resampling</td>
<td>· Common Spatial Patterns (CSP) [35]</td>
<td>· Quadratic Discriminant Analysis (QDA) [41]</td>
</tr>
<tr>
<td>· Artifact Rejection (Spike detection, bad window detection, bad channel detection, local peak detection)</td>
<td>· Spectrally-weighted Common Spatial Patterns [36]</td>
<td>· Regularized and analytically regularized LDA and QDA [41, 42]</td>
</tr>
<tr>
<td>· Envelope extraction</td>
<td>· Adaptive Autoregressive Modeling, from BioSig [37]</td>
<td>· Linear SVM [43] (LIBLINEAR/CVX)</td>
</tr>
<tr>
<td>· Epoch extraction</td>
<td>· Dual-Augmented Lagrange (DAL) [28]</td>
<td>· Kernel SVM [43]</td>
</tr>
<tr>
<td>- Time-Frequency window selection</td>
<td>· Frequency-Domain DAL (FDAL)</td>
<td>· Gaussian Mixture Models (GMM), 9 methods [44-46]</td>
</tr>
<tr>
<td>- Spectral transformation</td>
<td>· Independent Modulators [38]</td>
<td>· Regularized and variational Bayesian logistic regression and sparse Bayesian logistic regression [47, 48]</td>
</tr>
<tr>
<td>- Re-referencing</td>
<td>· Multi-Model Independent Component features</td>
<td>- Relevance Vector Machines (RVM) [50]</td>
</tr>
<tr>
<td>· Surface Laplacian filtering [29]</td>
<td></td>
<td>- group-sparse/rank-sparse linear and logistic regression [28]</td>
</tr>
<tr>
<td>· ICA methods (Infomax, FastICA, AMICA) [6, 30]</td>
<td></td>
<td>- high-dimensional gaussian Bayes density estimator/classifier</td>
</tr>
<tr>
<td>· Spectral filters (FIR, IIR)</td>
<td></td>
<td>- Voting meta-learner</td>
</tr>
<tr>
<td>· Spherical spline interpolation [31]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>- Signal Normalization</td>
<td></td>
<td></td>
</tr>
<tr>
<td>- Sparse signal reconstruction (NESTA, SBL [32], FOCUSS, l1; currently offline only)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>- Linear projection</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

At lower levels, BCILAB provides additional frameworks that are designed to be extensible, flexible and have low implementation overhead. In particular, most BCI paradigms are defined within a "data flow" scheme wherein information is passed through several stages that are themselves plug-in frameworks: for filters (signal processing), feature maps (feature extraction), and model learners as well as predictors/estimators (using machine learning). These frameworks are general enough to cover a wealth of implementations, such as adaptive/statistical epoched-signal processing, adaptive feature extraction, and classification/regression/density estimation, with general (discrete/continuous, multivariate, point-estimate/full-posterior) outputs. We are currently introducing additional concepts, such as hierarchical Bayesian models spanning sessions, subjects and (related) tasks.
A typical use case of BCILAB is for the offline analysis of a BCI study. For example, given a collection of data sets, one per subject, containing imagined movements of either the left or the right hand in random order, with events SL / SR indicating the timing and type of the respective cue stimuli, a user of the BCILAB scripting interface may proceed as follows: For each subject,

1. Load a data set
   
   ```
   eeg = io_loadset('Projects/Imag/Subject5/session1.eeg');
   ```

2. Define an analysis approach (customizing parts of a standard paradigm)
   
   ```
   approach = {'SpecCSP', 'events', {'SL', 'SR'}, learner', 'logreg'};
   ```

3. Apply the approach to the data, to get an estimate of its performance on the given data
   
   ```
   [performance, model, statistics] = bci_train({'data', eeg, ...
       'approach', approach});
   ```

This analysis gives the prediction accuracy results that are the key ingredient of most BCI publications (along with visualizations). Step 3 above also produces a calibrated predictive model which can be loaded into one of the provided real-time plugins (for the real-time ERICA, BCI2000 [51], and OpenViBE [52] real-time environments, with others forthcoming) for online testing. A major focus of the BCILAB toolbox is to allow, as much as possible, that competitive BCI estimation performance may be obtained using simply stated procedures (as above). For this purpose, a large collection of state-of-the-art methods have been provided and are listed in Table 1. A second, complementary focus is to provide rigorous analyses (e.g., for performance estimation) by default. For this purpose, a framework for automated cross-validation, systematic parameter search, and nested cross-validation is provided, and a suitable evaluation method is automatically chosen depending on the supplied data (though the evaluation method may also be customized). For example, if a single data set and at least one unknown parameter is provided by the user, nested block-wise cross-validation with safety margins is chosen be default. In a similar vein, to rule out common BCI research errors such as accidental non-causal signal processing, offline and online processing uses identical code.

BCILAB aims to be not just a collection of off-the-shelf tools to enable BCI experiments, but is designed to be a development platform for new BCI technology, facilitating the creation of new methods, approaches (e.g. combining methods), and paradigms. For this purpose, the toolbox provides extensive infrastructure, including, among others, the frameworks mentioned above, a small Mathematica-inspired symbolic expression system, an Adobe ASL-inspired declarative GUI property model, a decentralized distributed computing infrastructure (not dependent on MATLAB toolboxes), a generic dependency loader, a transparent multi-level cache for results, as well as bundled toolboxes for convenience. All BCILAB code is thoroughly documented, with additional citation-rich documentation for user-facing functions. Backwards compatibility to MATLAB 7.1 is attempted (and reached for most functionality except the graphic interface, which requires Matlab 2008a+, due to the use of objects).
Conclusion

The comprehensive ERICA software suite is an ongoing product of a coordinated effort to develop and test new methods for observing and modeling the dynamics of non-invasively observed electrophysiological connectivity in human cortex during a wide range of behavioral task performance, both post hoc and in real time. The aim is to build, test, and demonstrate methods for modeling task-event related electrophysiological signals and information flow dynamics in anatomically and statistically validated functional cortical networks and use this information in a range of real time applications. We plan to continue to extend and further coordinate the modular ERICA tools for online streaming and data storage, advanced offline and online EEG analysis, source localization, and multivariate connectivity analysis, hoping they will facilitate research leading to novel EEG analysis and data mining techniques of the 21st century.

Contributions and acknowledgments

EEGLAB was mainly developed by A. Delorme and S. Makeig [2] from the ICA Electrophysiology Toolbox of Makeig and colleagues, with functions and design input from many dozens of colleagues and EEGLAB users. The Neuroelectromagnetic Forward head modeling Toolbox (NFT) was developed by Z. Akalin Acar [3]. The Measure Projection toolbox (MPT) was developed by N. Bigdely-Shamlo. NetSIFT (Network Source Information Flow Toolbox) was developed by Tim Mullen. BCILAB has been developed by C. Kothe inspired by the preceding PhyPA BCI toolbox created by C. Kothe and T. Zander at the Berlin Technical University. The ERICA framework was mainly developed by A. Vankov; its Matlab elements were developed by N. Bigdely-Shamlo. Gift support from The Swartz Foundation (Old Field NY), and grants from the National Institutes of Health (USA), the National Science Foundation (USA), and the Office of Naval Research (US) are gratefully acknowledged.

References


