

Einführung in die

EEG-Analyse mit „Biosig for Octave and Matlab“

Version 1.02

19. Apr 2010

Doz. Dr. Alois Schlögl
e-mail: alois.schloegl@biosig-consulting.com>

This resource is provided to you under a Creative Commons Attribution-Share Alike 3.0 license. Please feel free to distribute and share it with your colleagues.

The file is available from
<http://biosig-consulting.com/biosig/doc/IntroBiosig.doc>
<http://biosig-consulting.com/biosig/doc/IntroBiosig.pdf>

Table of Contents

EEG-Analyse mit „Biosig for Octave and Matlab“	1
)1 Einleitung.....	3
)2 Voraussetzungen	3
)3 Installation	5
)4 Daten in den Matlab-Workspace laden.....	6
)5 Trigger, Ereignisse, Marker und Annotationen	7
)6 Evozierte Potentiale.....	8
)7 Statistik, t-test	9
)8 Overflow/Sättigungsartefakte	10
)9 Räumliche (spatial) Filter (Mono, CAR, Laplace, PCA, CSP, ...)	11
)10 Korrektur von EOG-Artefakten	12
)11 Detektion von Muskelartefakten.....	13
)12 Missing Values	14
)13 Single Trial Klassifikation	15
)14 Kopplungsanalyse	16
)15 Anhang:	17

1) Einleitung

Biosig ist eine „Freie“ Softwarebibliothek zur Verarbeitung von biomedizinischer Signalen (kurz: biosignale), wie z.B. das Elektroenzephalogramm (EEG) und das Elektrokardiogramm (EKG). „Frei“ bezieht sich nicht auf den Preis, sondern auf folgende vier Freiheiten (0) Das Programm zu jedem Zweck auszuführen, (1) Das Programm zu studieren und zu verändern, (2) Das Programm zu verbreiten, (3) Das Programm zu verbessern und zu verbreiten, um damit einen Nutzen für die Gemeinschaft zu erzeugen. Um diese Freiheit auch für die Zukunft zu gewährleisten, ist Biosig mit der „GNU General Public Licence“ (GPL) lizenziert.

Biosig besteht aus mehreren Teilen, wie z.B. Biosig4OctMat, Biosig4C++, SigViewer, u.a. „Biosig4OctMat“ unterstützt die Biosignalverarbeitung mit Matlab und auch Octave. Octave ist eine Freie Alternative zu Matlab welche weitgehende Kompatibilität zu Matlab hat. Biosig kann auch mit Octave verwendet werden.

Im folgenden werden insbesondere die wichtigsten Funktionen von Biosig4OctMat beschrieben. Die Beschreibung ist keineswegs vollständig sondern hat den Zweck den Einstieg in die Verwendung von Biosig zu erleichtern.

2) Voraussetzungen

Tabelle 1: Voraussetzungen für die verschiedenen Biosig-Module.

	Beschreibung	Voraussetzungen
Biosig4OctMat	Biosignalanalyse mit Octave oder Matlab	Matlab (v6.5 oder höher) oder Octave (v2.9 oder höher)
Biosig4C++	libbiosig, mexSLOAD, Datenkonverter	MS Windows oder Linux oder MacOSX
SigViewer	Viewing und Scoring Software	MS Windows oder Linux oder MacOSX
rtsBCI	Echtzeitdatenaufzeichnung	Matlab, Simulink, und RTW-toolbox, Biosig4OctMat
Biosig4Python	Importfilter für Python	

Grundkenntnisse in Matrizenrechnung (Matrizenmultiplikation) werden vorausgesetzt.

Die einzelnen Biosig-Module haben unterschiedliche Voraussetzungen (siehe Tabelle 1). Für die Verwendung von „Biosig4OctMat“ wird entweder Matlab oder Octave benötigt. Falls keine Matlab-

Lizenz zur Verfügung, ist Octave eine gute Alternative. Octave für Windows ist auf <http://octave.sourceforge.net/> verfügbar oder kann auch über cygwin <http://www.cygwin.com/> installiert werden. Octave ist auch für alle gängigen Linux-Systemen vorhanden.

Für eine Einführung in Octave/Matlab sind eine Reihe von Einführungen im WWW verfügbar. Hier ist eine kurze Auswahl.

Matlab/Octave Primer

<http://people.sc.fsu.edu/~sachins/matlab.primer.pdf>

Getting Started with Matlab

http://www.mathworks.com/access/helpdesk/help/pdf_doc/matlab/getstart.pdf

A beginners Guide to Matlab

http://www.evergreen.loyola.edu/~cxenophontos/pubs/other/Beginners_guide_to_MATLAB.pdf

Introduction to GNU Octave

<http://math.jacobs-university.de/oliver/teaching/iub/resources/octave/octave-intro/octave-intro.html>

GNU Octave - An Introduction

<http://linuxgazette.net/109/odonovan.html>

3) Installation

1. Download "biosig4octmat-2.39.zip" von <http://biosig.sourceforge.net/download.html> !
2. Entpacke die Datei biosig4octmat-2.39.zip in das Verzeichnis c:\biosig\ oder ein beliebiges anderes Verzeichnis! Im folgenden wird dieses Verzeichnis mit \$BIOSIGDIR\$ bezeichnen.
3. Starte Matlab und wechsle mit dem Befehl "cd" in das Verzeichnis \$BIOSIGDIR\$!
4. [OPTIONAL] Konfiguriere den Mex-Compiler mit dem Befehl

```
mex -setup
```
5. Führe den Befehl "biosig_installer" aus! Dabei werden die verschiedenen Unterverzeichnisse in den Pfad eingebunden (siehe Tabelle 1). Des weiteren wird versucht einzelne mex-Funktionen neu zu kompilieren, um Geschwindigkeitsvorteile zu erhalten. Falls mex nicht konfiguriert ist, werden langsamere m-scripts verwendet, und Biosig4OctMat ist ebenfalls funktionsfähig.
6. Biosig4OctMat ist korrekt installiert.

Tabelle 2: Verzeichnisse welche von Biosig4OctMat verwendet werden.

Verzeichnis	Funktionsbezeichnung
biosig/demo	Einige demo-Funktionen
biosig/doc	Dateien zur Dokumentation
biosig/t200_FileAccess	Importfilter zum Laden von verschiedenen Datenformaten
biosig/t250_ArtifactPreProcessingQuality Control	Funktionen zur Qualitätskontrolle der Daten, Artefaktbearbeitung, und einige Vorverarbeitungsfunktionen (z.B. Triggerung)
biosig/t300_FeatureExtraction	Signalverarbeitung und Merkmalsextraktion
biosig/t310_ERDSMaps	ERD Analyse
biosig/t400_Classification	Single trial Klassifikation
[biosig/t450_MultipleTestStatistic]	Multiple Test Statistik
biosig/t490_EvaluationCriteria	Evaluierungskriterien
biosig/t500_Visualization	Visualisierung
biosig/t501_VisualizeCoupling	Visualisierung von Kopplungsanalysen
NaN/inst	Statistik- und Klassifikationsfunktionen für Daten mit und ohne "Missing Values"
NaN/src	Verschiedene mex-Dateien für die NaN-toolbox
tsa	Toolbox zur Zeitreihen Analyse (TSA = Time Series Analysis)
freetb4matlab/signal	Toolbox für Signalverarbeitung
freetb4matlab/statistics/distributions	Toolbox für Statistische Verteilungen
freetb4matlab/statistics/tests	Toolbox für Statistische Tests

4) *Daten in den Matlab-Workspace laden*

Biosig unterstützt an die 40 verschiedenen Datenformate, das ist mehr als jede andere EEG-Software. Dabei werden die verschiedenen Formate automatisch erkannt. Dem Benutzer wird eine einheitliche Schnittstelle zur Verfügung gestellt. Der aktuelle Stand für die Unterstützung verschiedener Formate ist hier <http://biosig-consulting.com/biosig/TESTED> zu finden.

Es gibt zwei Möglichkeiten um Daten in den Arbeitsspeicher (workspace) zu laden. `sload` lädt die gesamte Datei. Nach Möglichkeit greift `sload` auf die die mex-Funktion `mexSLOAD` zurück.

```
[d,HDR]=sload(dateiname);
```

Mit `sopen/sread/sclose` ist es möglich einzelne Abschnitte von Daten zu laden. Dies hat den Vorteil, dass auch sehr große Dateien, welche nicht als Ganzes in den Arbeitsspeicher passen, bearbeitet werden können.

```
HDR=sopen(dateiname,'r');  
[d,HDR]=sread(dateiname, NumberOfSeconds, StartPosition);  
HDR = sclose(HDR);
```

In beiden Fällen wird die Headerstruktur `HDR` und eine Datenmatrix `d` zurückgeliefert. `HDR` enthält verschiedene Felder wie Abtastrate (`HDR.SampleRate`), Kanalbezeichnungen (`HDR.Label`), Skalierungsfaktoren, Informationen über Filtereinstellungen (`HDR.Filter`), sowie Ereignisse und Annotationen (`HDR.EVENT`). Eine umfassende Dokumentation ist in der Datei `biosig/doc/header.txt` zu finden.

Die Datenmatrix `d` enthält `HDR.SPR*HDR.NRec` Abtastpunkte (Zeilen) und `HDR.NS` Kanäle (Spalten).

5) *Trigger, Ereignisse, Marker und Annotationen*

Ereignisse sind in `HDR.EVENT` gespeichert. Dabei enthält `HDR.EVENT.POS` die (Start-)Position der einzelnen Marker enthalten. `HDR.EVENT.TYP` enthält die Kodierung des Ereignistyps. Optional können noch die Felder `HDR.EVENT.DUR` und `HDR.EVENT.CHN` vorhanden sein, welche die Dauer eines Ereignisses und eine Kanalzuordnung bezeichnen.

Eine Liste von vordefinierten Ereignistypen für die Codes in `HDR.EVENT.TYP` ist in der Datei `biosig/doc/eventtypes.txt` zu finden. Dabei ist er Bereich von `TYP=1..255` für benutzerspezifische Ereignisse, welche nicht durch andere Typen beschrieben werden kann, vorgesehen.

Häufig sind die Ereignisse von 2 oder mehreren Versuchsbedingungen zu extrahieren, und die Daten in einzelne Trials zu segmentieren. Dies kann folgendermassen durchgeführt werden:

```
t1 = HDR.EVENT.POS(HDR.EVENT.TYP == hex2dec('0001'));
    % Triggerpunkte von Versuchsbedingung 1
t2 = HDR.EVENT.POS(HDR.EVENT.TYP == hex2dec('0002'));
    % Triggerpunkte von Versuchsbedingung 2
```

In manchen Fällen müssen die Trigger aus einem eigenen Triggerkanal extrahiert werden.

```
t0 = gettrigger(d(:, triggerchannel));
```

Die Triggerung (Segmentierung) der Daten kann dann mit folgendem Befehl durchgeführt werden:

```
[X1, sx1] = trigg(d, t1, PRE, PST [, GAP])
```

X hat `HDR.NS` Zeilen (!) und `length(t1) * (PST-PR+1+GAP)` Spalten. Mit dem Befehl

```
reshape(X1, sx1)
```

werden die Daten in eine 3-dimonsionale Matrix mit den Dimensions `[NS, (PST-PR+1+GAP), length(t1)]` umgewandelt.

6) *Evozierte Potentiale*

Nachdem die Daten getriggert sind, kann eine Mittelwertbildung über alle Trials durchgeführt werden um die Evozierten Potentiale zu erhalten.

```
EP = mean(reshape(X1, sx1), 3);

[Y.SEM, Y.MEAN] = sem(permute(reshape(X1, sx1), [2, 1, 3]), 3);
Y.T = [PRE:PST]/HDR.SampleRate;
Y.datatype = 'MEAN+STD';
Y.Label = HDR.Label;
plota(Y);
```

Damit wird das evozierte Potential von allen Kanäle für den Ereignistyp 1 dargestellt. Die Funktion `evoked_potential.m` führt alle diese Arbeitsschritte durch.

```
R = evoked_potential(filename, CHAN, t_start, t_end, EventType);
plota(R);
```

7) Statistik, t-test

Um einen statistischen t-test durchzuführen, stehen die beiden Funktionen `t_test` und `t_test_2` zur Verfügung.

```
[PVAL, T, DF] = t_test (X, M, ALT)      % gepaarter T-Test  
[PVAL, T, DF] = t_test_2 (X, Y, ALT)   % ungepaarter T-Test
```

Um die evozierten Potentiale von zwei Versuchsgruppen auf unterschiede zu testen, könnte man folgenden Code verwenden

```
Y1 = reshape(X1, sx1);  
Y2 = reshape(X2, sx2);  
p = repmat(NaN, sx(1:2));  
for k1=1:sx1(1),  
for k2=1:sx1(2),  
    p(k1,k2) = t_test_2 (Y1(k1,k2,:), Y2(k1,k2,:), '<>');  
end;  
end;
```

8) **Overflow/Sättigungsartefakte**

Viele Datenformate unterstützen keine automatische Overflowdetektion, da die entsprechenden Sättigungswerte nicht im Datenformat gespeichert werden. Um die entsprechenden Sättigungswerte effizient zu bestimmen, kann die eine Histogramm-Methode (Schlögl et al 1999¹) verwendet werden.

```
Q = eeg2hist(filename, 'manual'); % manual selection of threshold  
  
clear H;  
  
H.FileName = Q.FileName;  
  
H.THRESHOLD = Q.THRESHOLD;  
  
[HDR]=sopen(H, 'r', 0); [s, HDR]=sread(HDR); HDR=sclose(HDR);
```

1 http://www.biosig-consulting.com/publications/neurophys1999_2165.pdf

9) *Räumliche (spatial) Filter (Mono, CAR, Laplace, PCA, CSP, ...)*

Räumliche Filter sind ein wichtiges Werkzeug zur Vorverarbeitung von EEG-Daten.

```
W = spatialfilter(...);
```

Bekannte Räumliche Filter sind “bipolare” Ableitungen, “common average reference” (CAR), und Laplace filter. Aber auch Hauptkomponentenanalyse (PCA) und “Independent Component Analysis” (ICA) zählen dazu. Sobald die Koeffizienten des Räumlichen Filters W ermittelt sind, kann das Filter durch eine einfache Matrizenmultiplikation auf die Daten angewandt werden.

```
D = d*W;
```

10) Korrektur von EOG-Artefakten

Ein Validierung durch Experten konnte zeigen dass mit multipler Regression bei zwei bipolaren EOG-Kanälen etwa 80% der EOG Artefakte entfernt werden können (Schlögl et al. 2007)². Neuere Ergebnisse zeigen keine Vorteile von ICA gegenüber dieser Regressionsmethode.

```
eogchan=identify_eog_channels(filename);  
  
eegchan=find(HDR.CHANTYP=='E'); % exclude any non-eeg channel.  
  
R = regress_eog(s,eegchan,eogchan);  
  
s = s*R.r0; % reduce EOG artifacts
```

² <http://www.biosig-consulting.com/publications/schloegl2007eog.pdf>

11) *Detektion von Muskelartefakten*

```
[INI,S,E] = detectmuscle(d , Fs, Mode);
```

Es sind vier Methoden implementiert. Allerdings wurde die Genauigkeit dieser Methoden noch nicht evaluiert. Daher wird empfohlen, Muskelartefakte manuell (z.B. mit Sigviewer) zu markieren, und diese Markierungen

12) Missing Values

Artefakte können ungültige Werte verursachen, welche in der Analyse nicht berücksichtigt werden sollen. Biosig verfolgt den Ansatz solche Werte als “missing” zu markieren, Dabei wird die Kodierung für “not-a-number” (NaN) des IEEE754 Standards für Fließkommazahlen verwendet.

Die meisten Standardfunktionen von Matlab haben als Ergebnis ein NaN sobald ein Wert in den Eingangsdaten NaN (missing) ist. Biosig enthält eine Reihe von Funktionen, welche die “missing” Werte ignorieren, und das Ergebnis aus den verbleibenden gültigen Werten ermittelt. Dazu werden insbesondere die Statistik-funktionen der NaN-toolbox verwendet, aber auch einige andere Funktionen (z.B.) MVAR.M können Daten mit “missing values” verarbeiten.

NaNs können auch hilfreich sein um unerwünschte Randeefekte zu vermeiden, wenn Datensegmente zusammengesetzt werden. Trigg.m und SLOAD.M werden diese Technik.

Beim Laden der Daten wird auch (nach Möglichkeit) eine automatische Sättigungsdetektion durchgeführt. Werte welche zu einer Übersteuerung des Aufnahmesystems führen, werden ebenfalls als “missing values” markiert.

13) Single Trial Klassifikation

Ein Demo-Analyse für die Single-Trial Klassifikation befindet sich in

```
biosig/demo/demo2.m
```

In einem ersten Schritt sind charakteristische Merkmale zu extrahieren. Da Single-trial Klassifikation häufig mit der Zielrichtung einer Online/Echtzeitanalyse eingesetzt wird, sind adaptive Methoden von Bedeutung.

```
% Time Domain Parameters
f = tdp(S,p,UC);

% Adaptive Autoregressive Parameter
for ch = 1:length(eegchan),

    X = tvaar(s(:,eegchan(ch)),MODE.MOP,MODE.UC);

    X = tvaar(s(:,eegchan(ch)),X);
    f2 = [f2,X.AAR,log(X.PE)];

end;

% Bandleistung
log10bp = bandpower(X, Fs, bands, smoothing, mode);
```

Aus den extrahierten Merkmale werden in der Folge ein Klassifikator berechnet.

```
C = train_sc(D,classlabel,TYPE)
```

Dieser Klassifikator kann dann wieder an neuen Daten getestet werden.

```
R = test_sc(CC,D,TYPE [,target_Classlabel])
```

Um die Performanz eines Klassifikators und die Trennbarkeit der Daten zu ermitteln, sollte eine Kreuzvalidierung der Daten durchgeführt werden. Damit wird aus "Overfitting" Problem vermieden.

```
[R,CC] = xval(D,classlabel)
```

Um die Performanz zu quantifizieren, gibt es verschiedene Evaluierungskriterien, wie z.B. Fehlerrate, Cohen's Kappa Koeffizient, Mutual information und Informationstransferraten.

```
[kap, sd, H, z, ACC, sACC, MI] = kappa(d1, d2);
```

14) Kopplungsanalyse

```
demo7.m
```

```
[AR, RC, PE] = mvar(Y, P);
```

```
M = size(AR, 1); % number of channels
```

```
A = [eye(M), -AR];
```

```
B = eye(M);
```

```
C = PE(:, M*P+1:M*(P+1));
```

```
[S, h, PDC, COH, DTF, DC, pCOH, dDTF, ffDTF, pCOH2, PDCF, coh, GGC, Af, GPDC]  
= mvfreqz(B, A, C, f, Fs)
```

```
R = tfmvar(s, TRIG, T, MOP, f, Fs)
```

```
plota(R);
```

```
plot_coupling
```

```
t501/main.m % demo
```

```
elpos compute and show XY position of electrodes
```

```
elpos3 show 3d electrode positions
```

15) Anhang:

```
a=1:4  
b=[5:8]'  
c=[1,2;3,4]  
b'  
c(1,:)   
c(:,2)
```

```
help  
help help  
plot  
size  
length  
whos
```