User Guide: *MEPFeatX* – Automated Motor-Evoked Potentials Feature Extraction in Transcranial Magnetic Stimulation

Introduction

MEPFeatX is a comprehensive MATLAB package designed to extract features from motor-evoked potentials (MEPs) in transcranial magnetic stimulation (TMS) experiments. This user guide provides detailed instructions on verification of the package functionality, perform feature extractions, and utilizing MEPFeatX in several use cases.

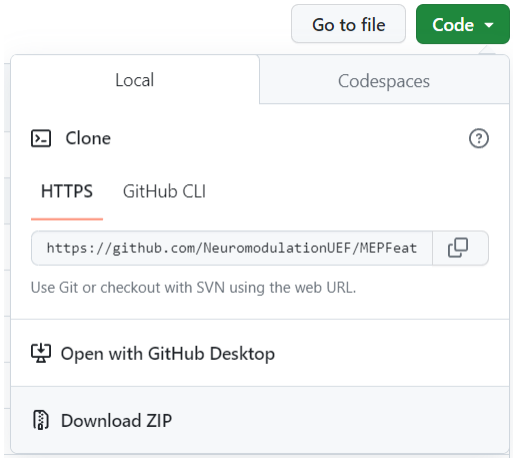
Dependencies and System requirements

The package requires three MathWorks products: MATLAB, Signal Processing Toolbox, and Statistics and Machine learning Toolbox. In addition, running this package requires a system with Intel core i3 or higher, and with the minimum amount of RAM 8GB.

The current version of the package is written in MATLAB R2022b, and verified in R2018b, R2019b, R2021b, R2022b. No error is expected in other MATLAB versions between R2018b to R2022b, as no change expected in the required dependencies in these MATLAB versions.

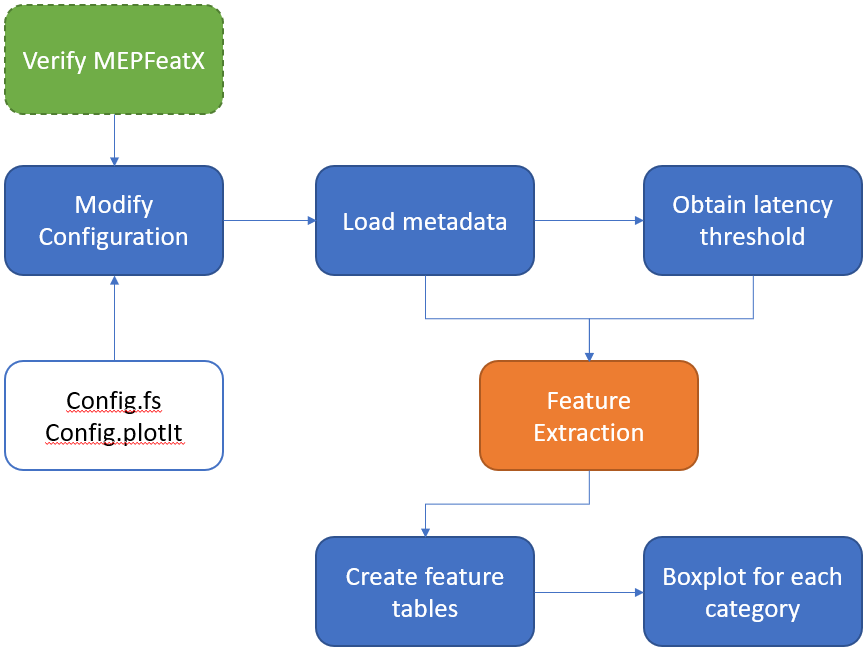
Download the package

The package is available from GitHub: https://github.com/NeuromodulationUEF/MEPFeatX. Click on *Code* to clone the package via Git or download ZIP file of the package.



Workflow

Below is a suggested workflow to utilize *MEPFeatX* for feature extraction and is demonstrated in the *main.m* script.



Verify MEPFeatX

After downloading, change the variable *dir_root* in *main.m* to the current folder.

```
28      % Define working folder
29 -    dir_root = 'F:\MEPFeatX\';
```

The package should be verified before first use, to confirm the newly extracted features have the same value as the referenced features. This can be done by running the script *verify_functionality.m*.

```
38      %% Verify the toolbox before the first use
39 -    disp(repmat('=', 1, 100))
40 -    diary([config.path_ref 'verifying_MEPFeatX_' time_now '.txt'])
41 -    verify_functionality
42 -    diary off
```

The logs of the verification are saved to the *reference/* folder.

```
92  ID01_features.mat
93  ===> Difference in feature value: 0
94  ID02_features.mat
95  ===> Difference in feature value: 0
96  ID03_features.mat
97  ===> Difference in feature value: 0
98  ID04_features.mat
99  ===> Difference in feature value: 0
100 ID05_features.mat
101 ===> Difference in feature value: 0
102 ID06_features.mat
103 ===> Difference in feature value: 0
104 ID07_features.mat
105 ===> Difference in feature value: 0
106 ID10_1_features.mat
107 ===> Difference in feature value: 0
108 ID10_2_features.mat
109 ===> Difference in feature value: 0
110 -v- The total difference of the new feature sets to the reference is near zero.
111 -v- Package functions got verified at 2023-05-16 21:39:20
```

Customize configuration file

The configuration parameters should be customized properly in *make_config.m* before running *MEPFeatX*. The most important parameter is the sampling frequency, which was set 3 kHz by default.

```
41 -    config.fs = 3; % sampling frequency in kHz
```

Note: The sample datasets are recorded at 3 kHz sampling frequency. Therefore, the configuration running for sampling frequency for the package verification must be at 3 kHz.

The other is the visualization plotting options for the main figures, which can be found also in *make_config.m.*

Deployment

MEPFeatX utilizes predefined threshold values in a look-up table for latency threshold based on the subject's age group (either *Child* or *Adult*) and target muscle, which are listed in the metadata table. Both tables are provided along with the sample dataset.

- Onset_table reads on the latency_threshold.xlsx in the *data/* folder, and outputs the thresholding value based on the information of the subject age group, targeted muscle, and the sampling frequency.
- Metadata table reads on the metadata_table.xlsx in the *data/* folder. This table contains the metadata of each dataset.

```
44
45 -    %% Load onset threshold table and metadata table
46 -    onset_table = readtable(config.latency_threshold, 'ReadRowNames',true);
        metadata = readtable(config.metadata);
```

Note: In the actual use of MEPFeatX, the user should modify these two tables according to the study protocol.

After the *onset_table* and *metadata* are loaded, you can load the data into *extract_features* or *extract_features_[protocol]*. The data should be in .mat file, and ensure that its rows stores samples, and columns as trials. In the *main.m* script, you can find both templates to run one dataset or all datasets that listed in the metadata table.

Create feature table and further analysis

The script *create_feature_table.m* combines the metadata table and all the feature files in analysis_xxxxxx/features folders into a comprehensive table, which can be used later for further analysis, such as boxplots for each subcategory.

```
%% Create feature table for further analysis
disp(repmat('=', 1, 100))
disp('Create a table for MEP features')
create_feature_table

disp('Plot feature boxplots')
plot_feature_boxplots(config, 'Session')
plot_feature_boxplots(config, 'Muscle')
plot_feature_boxplots(config, 'SequenceType')
```

Conclusion

MEPFeatX offers a compact tool to extract MEP features in TMS experiments. By following this user guide, we hope you can utilize the package in your MEP analysis pipeline. For additional support and detailed documentation, refer to the MEPFeatX GitHub webpage. Don't hesitate to address any issue of *MEPFeatX* and enjoy using *MEPFeatX* for your TMS research!