Friedrich-Alexander-Universität
Neuromuscular Physiology and
Neural Interfacing Laboratory

Exercise 2 **13.12.2023**

General Information

This exercise will guide you to various surface EMG data postprocessing and analysis steps. We suggest using MATLAB to visualize, quantify and process the EMG data and corresponding force signal during the exercises. Note that using Python is also possible, although we will use MATLAB for the exemplary solution. If you have not installed MATLAB yet, click on the link below and take advantage of FAU's Campus-Wide License to install and use MATLAB on your computer: www.mathworks.com/products/matlab/student.html

In addition to the base MATLAB version, you might need the MATLAB signal processing toolbox (depends on the functions you want to use). You can install it in MATLAB under Home -> Add-Ons.

MATLAB provides a lot of built-in functions that can make your life easier during these exercises. It is always a good idea to check the **MATLAB documentation**. Don´t hesitate to also use Google or **write in the StudOn forum**. Also check the **Tips** section on the last page.

Submission:

1. Your code as a .m (or .py) file (Your main script as well as any potential self-defined functions).
2. A .pdf file with your results. The results that should be included in the report are **written in this colour** (You can save the plots as a .png with *save as* in the plot window). For questions that require written text as an answer, 1-3 sentences or bullet points should usually be enough. Plots need to contain labelled axes and suitable units.

!
- Please **do not** submit a document with both the code **and** the plots & answers! Only submissions with two **separate** files, particularly a .pdf with the plots & answers, and a separate .m/.py/.ipynb file containing your code will be accepted.
- This is an individual exercise. If the submitted code and/or .pdf file is identical to another submission, both submissions will be marked as failed.

Submission Deadline: 10.01.2024, 23:55 pm

Note that the interpretation of the exercise results and the physiological concepts behind these **are relevant for the exam. Additionally, there will be bonus points for submitting both exercises.** Note the bonus points will only be granted if you submitted adequate results.

The Dataset

For this exercise, you will again use the *Slow_Contraction.mat* dataset from exercise 1. Check there for a detailed description of the data acquisition.

In addition to the variables you used in exercise 1, in this exercise you will need the motor unit action potential discharge timings (`MUPulses`). These timings, also called motor unit spikes, were obtained from the **decomposition** of the EMG data. As you know from the lectures, decomposition is based on blind source separation approaches to extract the single motor unit firings from the high-density surface EMG signals.

The variable `MUPulses` contains the samples of the action potential discharge timings of each motor unit in a cell array (In this dataset, 32 motor units were found by the decomposition algorithm). That means for a motor unit (MU) with action potential discharges at samples 5, 8 and 17: `MUPulses{1,1}=[5 8 17]`

The folder also contains the MATLAB functions `plotSpikeRaster.m` and `spikeTriggeredAveraging.m` that you will need during this exercise. If these function files are located in your current working directory, you can call them just like any native MATLAB function.

Task 1. Spike trains

Plotting the MU action potential discharge timings as so-called **spike trains** is a good way of visualizing the firing patterns and discharge rates of different motor units during isometric or kinetic contractions (see Fig.1).

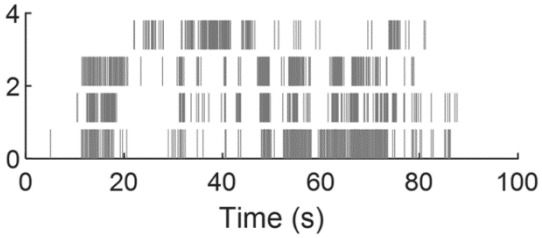

Figure 1: Exemplary MU Spike Trains. From: Del Vecchio et al. (2020).

For plotting the MU spike trains, you can use the provided MATLAB function `plotSpikeRaster(firingMatrix,'PlotType','vertline','VertSpikeHeight',0.9)`. The function requires a matrix `firingMatrix` with the so-called binary code of MU firings. Rows correspond to the MUs and columns to the samples (E.q. For motor unit data containing 5 MUs with a signal length of 2000 samples, the function requires a matrix of shape 5x2000), with zeros at the positions where the MU was not active, and ones where the MU discharged an action potential.
E.g., for an exemplary MU1 with action potentials at samples 5, 8 and 17 and MU2 at samples 3 and 12:

```
firingMatrix=[0 0 0 0 1 0 0 1 0 0 0 0 0 0 0 0 1;
              0 0 1 0 0 0 0 0 0 0 0 1 0 0 0 0 0]
```

This matrix needs to be of data type **`logical`** (In MATLAB, you can convert an array or matrix of type `double` to an array of type `logical` with the command `logical(matrix)`).

1.1. **Create a figure with the spike trains of all motor units and the reference force signal in N in the same plot. Can you notice any special relation between the spike trains and the force signal?**
1.2. Looking at the MU spike trains, **what differences can you see between the MUs? What parameters/properties could be extracted from this data that characterizes the MUs and their behaviour?**

Task 2. Spike triggered averaging

Spike triggered averaging (STA) is a useful event detection technique that is often used in neural signal processing. STA is a measure to relate the spike trains to a continuous signal, in our case the raw EMG signal. It represents the average EMG signal present inside a window around the times of spike occurrences and is therefore equivalent to the cross-correlation between the EMG signal and the spike train. As a result, the STA gives us the **average motor unit action potential (MUAP) shape** of a motor unit for each channel of our EMG data. Figure 2 illustrates the concept of STA with the spike train of an exemplary Motor Unit #12.

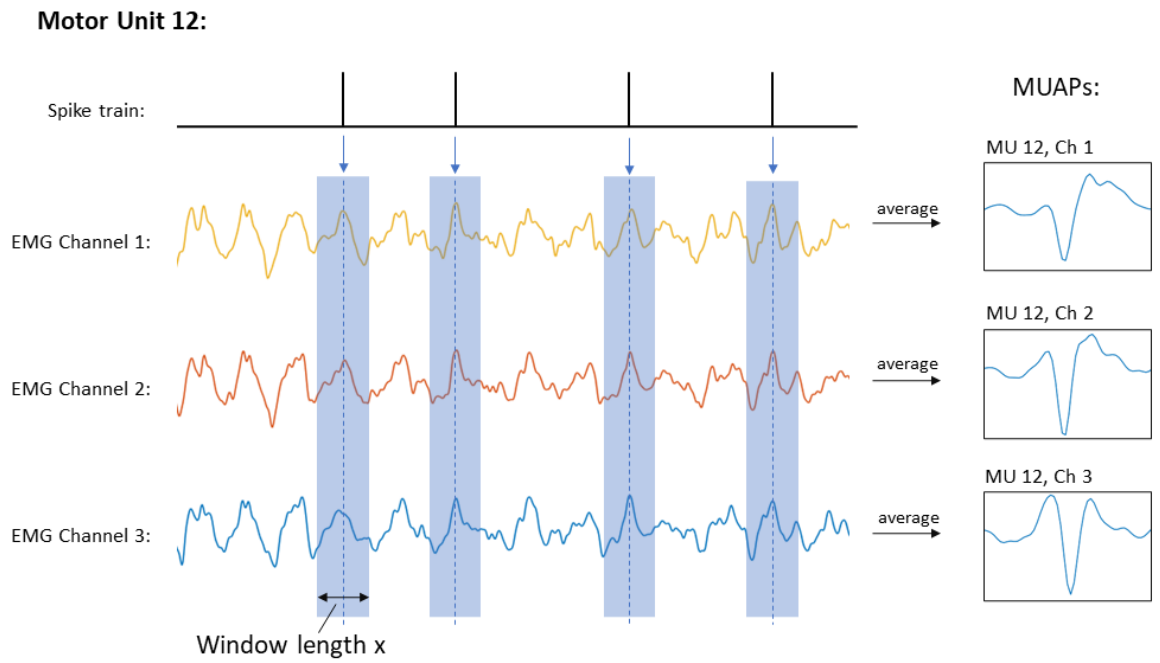

Figure 2. Concept of Spike triggered averaging. STA results in the MUAPs for each channel of the respective motor unit.

2.1. The provided `spikeTriggeredAveraging(SIG,MUPulses,STA_window,fsamp)` returns you the MUAP shapes of all motor units for each channel. Check the function code for a description of the input/output parameters. **Plot the MUAP shapes of all 64 channels of one exemplary MU of your choice. Arange the channel plots in the shape of the original EMG grid.** (In MATLAB, you can use the `subplot(m,n,p)` command).

Task 3. Motor Unit Action Potential Amplitudes

From the lectures you should know that smaller motor units are recruited at lower forces, and bigger motor units at higher forces. Also, larger motor units produce larger action potentials than smaller motor units. Therefore, the recruitment threshold should show a positive correlation to the motor unit action potential amplitude.

3.1. Compute the maximum peak-to-peak value of the MUAP shapes of each motor unit to obtain the amplitude of the motor unit action potentials.

3.2. **Visualize, if there is a dependency between the amplitude of the MUAP and the order of recruitment of the motor units.** There are multiple possibilities of visualization.

Task 4. Motor Unit locations

Using the MUAP shapes obtained from the STA, we can determine the spatial location of the motor units with respect to the EMG electrode grid (see Fig. 3). The closer the motor unit is located to an electrode, the higher the amplitude of the MUAP in that specific channel will be.

4.1. Compute the root-mean-square (RMS) of each MUAP of one MU of your choice, resulting in 64 RMS values, one for each channel. (No moving average needed here, just the root-mean-square of each MUAP)

4.2. Arange the RMS values in a matrix according to the position in the electrode grid, and **plot the resulting matrix as a heatmap** (MATLAB: `imagesc()`). See Fig. 3 for an exemplary heatmap. This way you can see the MUAP amplitude colorcoded in the shape of the EMG electrode grid, visualizing the corresponding motor unit location in the underlying muscle.

4.3. **Repeat this for two more motor units**. Try to find MUs that show distinct heatmaps. On a physiological level, **where could the differences between the three MU MUAP RMS heatmaps originate from? What does that tell us about the motor units?**

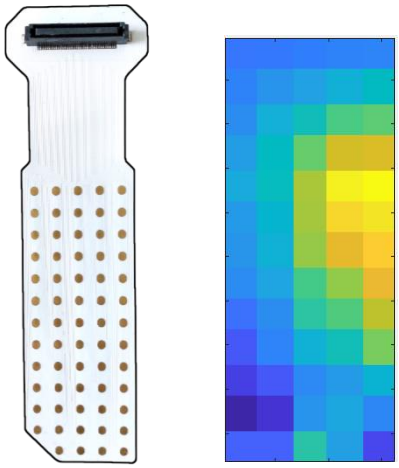

Figure 3: Left: sEMG electrode grid used for the acquisition of this data. Electrodes are aranged in a 13 x 5 matrix with the bottom left electrode missing. Right: Visualization of spatial location of an exemplary motor unit in a 13x5 electrode grid. RMS values of MUAPs colorcoded according to its amplitude.

Tips

- **Script Structure:** To keep your script well-structured and clear, you can create script sections by writing `%%`, followed by a blank space and a section title. You can execute single script sections with the `Run Section` button in the toolbar or `Ctrl+Enter`, instead of having to execute the entire script. (https://www.mathworks.com/help/matlab/matlab_prog/create-and-run-sections.html)

- **Data Storage:** When loading the dataset into the workspace using `datasetX = load(filepath)`, the variables are combined into a `structure array (struct)`, which is a very convenient way of storing variables that belong to the same dataset, as they can be easily accessed via dot notation as in e.q.: `datasetX.SIG{row,col}`. This keeps the workspace clean when working with multiple datasets, and also allows to load datasets that contain variables with the same name, as those variables would overwrite each other otherwise. You can add more fields to the `struct` by using the same dot notation, e.q. if you want to add the a new data variable *SIGfilt*: `datasetX.SIGfilt= yourFilterfunction(datasetX.SIG,…)`
(https://www.mathworks.com/help/matlab/ref/struct.html)

- **Custom Functions:** If you need to apply the same data processing steps to multiple sets of data, is it advised to write a custom function for these steps, that you can then pass the datasets to. A function can either be saved in a separate .m file or defined at the bottom of your main script. Check https://www.mathworks.com/help/matlab/ref/function.html on how to define functions.

- **Data Plotting:** If you want to plot multiple related figures into one figure window, check out the `subplot` command. If you want to plot multiple plots into the same figure, use the `hold on` command. `yyaxis right` lets you plot multiple data series into one plot with independent y axes.
(https://www.mathworks.com/help/matlab/ref/subplot.html)
(https://www.mathworks.com/help/matlab/ref/hold.html)
(https://www.mathworks.com/help/matlab/ref/yyaxis.html)
Important: Don´t forget to label your axes in your plots and to give them informative titles.

- If you do not know how to code a certain task, there is probably a **built-in MATLAB function** for you. Check the documentation, Google or the AI language model of your choice.