Assignment 5 - cheat sheet

Struct, cell, array

% Load the struct named 'data' from the file named 'data.mat' into the workspace
load('data.mat', 'data');

After this, the data variable will be a 1x5 struct. A field can be extracted into cell elements like {data.force}, this will be a 1x5 cell array with each element containing a force vector belonging to that segment. Note, it would be very impractical to use a matrix instead of a cell array here, because the segments have different lengths. (The alternative to a cell array is having a separate variable for each segment.)
Cell arrays can be batch processed with cellfun, which will create a 1x5 vector result in which each element is the result of processing a single cell element.

% Calculate average force of each segment (1xN vector)
AF = cellfun(@part1, {data.force});

This syntax can then be used, and you will need to implement the part1 function to handle a single cell element, i.e. which calculates the average of an input vector in the first task. You can define part1 at the end of your main script or in a separate part1.m file. The function signature will be something like the following.

```
function af = part1(force)
    af = ... ;
end
```

Mean subtraction:

*Please note, that you **do not** need to subtract the mean value of the signal to calculate the required parameters. **The grader accepts the solution without the mean subtraction.***

Turn count rate

The TCR calculation task might be challenging. If you are confident in your algorithm, but do not get the expected result, you should plot your turn points and/or significant turn points and see the points for yourself.

```
turns = ... ;  % your algorithm to obtain indices of (significant) turn points
emg = data(5).EMG;
t = data(5).t;
figure; hold on;
plot(t, emg);  % the EMG signal
plot(t(turns), emg(turns), 'r*');  % the turn points in the signal
```

The above code snippet might be useful to visualize your turn points (you can zoom into the plot). If you have a boolean mask instead of indices for your turn points, you might want to check out the documentation for the find function to obtain indices. If you process your turn points in a (for) loop, you can also call the plot(..., ..., 'r*'); function at each iteration for a single point. The syntax might be a bit different, but you can also plot within your function calculating the turn count rate.

Polynom fitting

If you want to fit a polynom on the y(x) function, you can use the following syntax.

```
% x: values of independent variable
% y: values of dependent variable
polynom_coeffs = polyfit(x, y, degree)
y_estimate = polyval(polynom_coeffs, x)
plot(x, y_estimate)
```

Of course, you can solve the assignment in many (very) different ways, but this cheat sheet might be handy when obstacles show up.