MATLAB Programming

(Module code: 03 36257)

2020

Structured Programming Exercise (SPE)

Dr Hyojin Park

The aim of the structured programming exercise is to

1) Understand brain imaging data structure, e.g., electrophysiological data
2) Analyse the data both at individual and group levels
3) Visualise the results

The main task is to analyse the **relationship** *between* **brain signals in each sensor of the MEG data** *and* **signals from audio & visual speech stimuli** that have been presented to the subjects during the experiment. The relationship between the brain and stimuli signals will be analysed using cross-correlation in MATLAB.

Detailed information for the data

1. subXX.mat: Magnetoencephalography (MEG) data for 5 subjects numbered 01 to 05

1) Subjects were presented with natural audiovisual speech for 154 seconds.

2) Data explained for each field in data_cont
- label: MEG sensor label
- trial: Pre-processed MEG signals in sensor x time (corresponds to the label and time field respectively)
- time: time information in second
- fsample: sampling rate in Hz

3) The number of MEG sensors (shown as label) can be different because data from some sensors were excluded due to some technical faults or unexplained artefacts during the experiment.

2. av_speech.mat: Speech signals that contain both auditory and visual speech that were presented to the subjects during the experiment. Each of auditory and visual speech signal and MEG singal in each sensor have the same length in time (154 s).

1) a_sound: auditory speech signal (signal from the acoustic envelope)
2) v_lip: auditory speech signal (signal from lip area over time)

3. dat_label_common.mat: MEG sensor labels for Exercise 3 (see below)

1) As described above, the number of sensors is varied in subjects' data.
2) dat_label_common contains a list of sensor label that is found in common in all subjects' data.
3) This data should be used for the grand-average in Exercise 3.

Note: Please be mindful

1) File name and variable name (when you open the file) can be different.

2) Variable name can be the same across subjects.

3) Do NOT use any other toolboxes, e.g., EEGLab, Fieldtrip etc. Use default functions in MATLAB only.

4) Use control statements (e.g., for-loops) wherever possible, rather than coding it manually e.g., for each subject/sensor.

5) Every Exercise should start with clean workspace ("clear all") and without any figures open ("close all").

6) To understand the role of the functions in MATLAB, sometimes it helps if you test a function with a simple dataset that you can understand and plotting the result (i.e., simulation) – If you did this to understand *the function for cross-correlation*, please include your simulation in the code.

Things you need to know about evaluation

1) I will copy your analysis scripts to the folder that contains data files that have been given to you.

2) I will NOT load any of the data myself – I will just run your main analysis code (SPE_XXXXXXX_main.m).

3) Read every line of instruction carefully, if you have any more questions on the SPE, please post your questions on the Discussion board on Canvas. I will answer them on the board. This is to be fair to all students since replying to individual emails could yield some biased outcome.

4) Please read "Overall Grade Descriptors" and "Detailed Marking Criteria for the SPE" in the SPE ASSESSMENT MARKING SCHEME section below.

Other useful information

1) Look into the data (variables) on Variables Window (or typing whos variablename) , e.g. variable type, size etc. In other words, use every information you can find in MATLAB. MATLAB already shows useful information that you might have overlooked.

2) Always practice good programming style and soft-coding.

3) Simple, fewer lines of code – with keeping good programming style – is always better.

Files you need to submit (in a zip: SPE_XXXXXXX.zip)

1) SPE_XXXXXXX_main.m (main analysis script)
2) SPE_XXXXXXX_GAplot.m (function script created in Exercise 4)
(XXXXXXX is your Student ID Number)

See HOW TO SUBMIT YOUR ASSESSMENT DATA section below.

Exercise 1. Cross-correlation

We are interested in the relationship between each of auditory and visual speech signal and brain signals measured using MEG. Thus, we would like to analyse cross-correlation between each speech signal and each of the MEG sensors.

Cross-correlation between
1) Each of MEG sensor data and auditory speech (A)
2) Each of MEG sensor data and visual speech (V)

- With the maximum lag (maxlag) for 2 s. Resulting time lag will be ranged from -2 s to 2 s.
- For each subject. Data dimension will be the number of sensor x number of sample points corresponding to -2 to 2 s.
- How to create the resulting data matrix is up to you; however, practice the most convenient way for loading data in Exercise 2.
- Also, write the code for saving the output data, but you do NOT need to submit the saved data.

Exercise 2. Plot individual results

Plot and save (both .fig and .png format) the figure for the cross-correlation result from the Exercise 1.

- For the cross-correlation results for A only in one sensor labelled ‘A210’ (one of the auditory sensors)
- No need to plot cross-correlation results for V
- For all subjects
- Write the code for saving figures, but you do NOT need to submit the saved figures.
- Plot the figure(s) as informative as possible.

Exercise 3. Grand-average

This time, we are curious about the group-level result, so we will average the result (from Exercise 1) over subjects (i.e., grand-average).

- Keep the dimension for sensor and time – average over subjects only.
- Do the grand-average only for common sensors that can be found in all subjects' data.
- For this, use dat_label_common.mat that contains sensor label found in common in all subjects' data.
- For the cross-correlation results for both A and V.
- Write the code for saving output data, but you do NOT need to submit saved data.

- Here I provided dat_label_common.mat. If you figured out yourself how to make dat_label_common.mat using individual data (subXX.mat), please write the code at the end of Exercise 3. It will be marked as an extra **BONUS** (+5 marks).

Exercise 4. Create a function returning a figure for the grand-average result

Now, we would like to make a function that returns a figure showing the grand-average result for a particular sensor, i.e., one input to the function should be a sensor label we would like to check.

- Function file name should be: SPE_XXXXXXX_GAplot.m

- Input data to the function for the figure: The grand-average result of either A or V from Exercise 3.

- One of the optional inputs should include:
    - The returning figure should be shown in one of
        - the warm colours (e.g., red, orange, etc.) that you like if the input data is the grand-average result for A
        - the cool colours (e.g., blue, green, etc.) that you like if the input data is the grand-average result for V

- Write line(s) of code in the main analysis script, so that I can evaluate it.
- For example, SPE_ XXXXXXX_GAplot('A210', other inputs you made...)

- Write the code for saving the returned figure, but you do NOT need to submit the saved figure.
- Return a figure as informative as possible.

General notes about the SPE

The aim of this assignment is for you to apply the knowledge and skills you have been acquiring since you started the course. Up until now, the applications of the skills have been on very carefully structured exercises. This time it is significantly more *open-ended, much more like how you would use MATLAB in real-life situations using a real dataset*. This is an opportunity to not just display your MATLAB skills but integrate them with your own ideas and reveal the true power of MATLAB.

There are a number of advantages to this open-ended programming exercise.

Firstly, it gives you a chance to be creative and solve a problem with the skills you know or want to extend. *Secondly*, this is a semirealistic exercise in programming that more likely matches what you will encounter in future work in psychology or computational neuroscience.

It is highly unlikely that you will be given a detailed plan of how to program a specific experiment or analyse the data you acquired. It is more likely you will be given a general framework with some pointers to help.

This programming exercise is structured in the sense that there are some common expectations as to what the program should achieve. *How you fulfil these criteria is up to you. You have the freedom to be as creative as possible* within the limits of what MATLAB allows you. At the absolute minimum, it is essential to make sure that your program basically does run without any substantial crashes.

If you wish to use skills which are beyond those mentioned in the workshop, you are free to do so, but it is not absolutely necessary. Marking of the submitted programs will be undertaken in an evaluative manner which assesses not just whether it runs or not, but how well thought through it is and how well-structured the code is. It is very rare that even an experienced programmer can make the perfect program in one go. It requires testing, debugging, modifications and extensions multiple times. Do not be afraid of making errors; it is only natural!

I strongly recommend you start small and work your way upwards in terms of coding. Try to keep your code modular by using separate functions that perform specific actions. Write these functions and then test them individually before putting them together overall.

Add comments in sections to help you (as well as a hypothetical user looking over your shoulder at your code) along the way. This will help remind you what certain lines or sections of code are. Make sure the comments make sense to you. If you cannot understand them, it is unlikely another person or assessor will either. You will also be marked on the appropriateness and intelligibility of the

comments you make, so basically if you comment appropriately, you will receive marks for reminding yourself about how your own program works.

If you have any questions on the SPE, you are very welcome to post your questions on the Discussion board of the module page. If it is about technical issues in MATLAB or other general queries, you can write an email to me (h.park@bham.ac.uk). However, I will not be able to write any program code for you nor provide direct feedback on any MATLAB code you have written for the SPE. Remember, this program is an assessment of your applied MATLAB skills, after all.

SPE ASSESSMENT MARKING SCHEME

Overall Grade Descriptors

It may be helpful to think of the expected overall quality of MATLAB code for the SPE in the following way:

A (Distinction): 74-100%
Programs at this standard should run efficiently with style and flair with appropriate safety checks built in to handle any predictable problems arising. The program should be user friendly, informative and intuitive to run by a lay user. There are clear innovation and originality which works effectively. The program will likely include one or more of the additional functionalities.

B (Merit): 61-73%
The program runs well fulfilling all basic requirements to a good and relatively efficient manner. There are some signs of originality in design, albeit not fully polished. There are some minor bugs or redundancies but without too much detriment to the overall performance and quality of program produced. The program will likely include one of the additional functionalities.

C (Pass): 50-60%
The program runs or nearly runs at a fairly basic level. It fulfils the basic minimal requirements set, although it may have errors or bugs. There are some problems with the program but not enough to produce major program crashes. Programs should run fairly well with some minor issues or problems which may make the functioning inefficient or not optimal in various ways. There are some comments and safety checks built in which help guide the lay user through the program. It could be run mainly by a conversant programmer.

D (Marginal Fail): 40-49%
The program contains some serious faults which affect usability and the capability to perform as specified, although it may perform some basic functions or some of the code may match the program specification. It lacks safety checks and/or comments.

F (Fail): less than 40%
The program fails to function in any meaningful way. It contains errors which severely affects its usability and capability. It is substantially incomplete. It cannot be run.

Detailed Marking Criteria for the SPE

The Structured Programming Exercise coursework will be scored separately on each of the following 5 factors for each Exercise:

1. Task Fulfilment (0-5 marks)
2. Range of Skills Used (0-5 marks)
3. Elegance of Code (0-5 marks)
4. Use of Functions and Variables (0-5 marks)
5. Comments and Safety Checks (0-5 marks)

These marks (25 marks per Exercise) when totalled form a base score in the range 0-100 (i.e., 100 marks for 4 Exercises).

Any late penalties are then deducted (5 mark penalty for every 24 hours).

This is the final score for the MATLAB Assessment.
The pass mark for the module is >= 50% overall.

NOTES ABOUT THE ASSESSMENT SUBMISSION

Remember this is a coursework assessment piece, not an exam. That means you are encouraged to take full opportunity to use all the lecture notes from the course, the exercises, the course materials and the help available inside MATLAB.

This is a 10 credit module which is expected to translate into something akin to 100 hours of study. Therefore even after the workshops and independent study, it is expected that you will need to spend 40-60 hours or more if needed working on this assessment to achieve a high score. This means you must make sufficient time available and plan your programming carefully. *This assessment should not be rushed or left to the last minute*.

PLAGIARISM

However, please do note that this is an independent piece of work much like everything you do in your academic studies and so standard plagiarism checks will be applied. You should not look for or download MATLAB code from the Web that implements any version of the task. Code available on the Web is easy for markers to spot. Equally, you should not copy or share code with any person or work collectively to implement code. Both will be checked for. Any identical or suspiciously similar code identified will be considered as an academic offence and referred to the PG Plagiarism Officer. Sanctions include resubmission of a new version of your coursework (to a Pass/Fail standard), a fail of the module and/or a referral to the College for serious academic misconduct.

HOW TO SUBMIT YOUR ASSESSMENT DATA

This assessment should be submitted via Canvas in a zip file using the following file name:

SPE_XXXXXXX.zip where XXXXXXX = Your Student ID Number
e.g. SPE_0123456.zip

Your zip file should contain both your MATLAB program files as requested above. Remember that your main programs and data files must be called

SPE_XXXXXXX_main.m
SPE_XXXXXXX_GAplot.m

Please do not assume I will have any files needed to run your programs. Everything must be in the zip file. This zip file should be uploaded to your Canvas account.

The program must run on the current version of MATLAB R2020a.

Note that the deadline for the SPE submission is **12 noon Monday 9th November 2020**.

Any submissions after this date will have an automatic 5-mark penalty for every 24 hours (excluding weekends and public holidays) after the deadline which is cumulative until receipt of submission. Even if the submission arrives less than 24 hours after, but still after the 12 noon deadline, 5 marks will be deducted from the SPE mark. In other words, the 5-mark deduction starts at 12 noon each day, so if you submit 10 minutes late, or 23 hours 50 minutes late, you still lose 5 marks.