Coding Standards and Recommendations
EECE 2150 Fall 2016

Why coding standards? In this course you will write small Matlab programs that only you and your lab partner are likely to use, so in principle you could write these programs using any style you like. However in modern programming, good programming practices are essential so that large complex programs written by many people have a reasonable chance of working, working over extended periods of time, and being maintainable and extensible. Moreover we hear frequently from industry that ECE students do not learn enough about how to program in a modern programming environment. Thus, at the risk of overkill, but in the spirit of learning good habits by practicing them, we want you to adhere to the following principles and practices in any programs you write in EECE 2150 this semester. Note that we will try to follow these practices ourselves and if you think that any code we distribute does not follow these practices you are encouraged to discuss this with us.

Basic principle: Code should be easily understandable at an appropriate level by both users and other programmers.

What we expect:

1. All script names, function names, and variable names should be descriptive of what they do / represent. No *x, y, t, n, f* !!!.

2. All function names should be the same as the name of the m-file that contains them.

3. All scripts or functions should contain a *preamble* so that running "help scriptname" or "help functionname" will type the preamble to the screen.

    a. The preamble should describe the purpose of the script or function.

    b. For functions it should describe input and output variables (type, eg scalar, vector, matrix, character) and meaning, as well as any restrictions on those variables and any use of optional variables (or in general optional ways to call the function).

    c. If your script or function calls other functions that are not part of the standard Matlab distribution that should also be described here.

    d. It should indicate the authors(s), revs, and date, with some description of any substantive revisions

4. All scripts or functions should contain sufficient, and sufficiently clear, internal documentation that a reader who is reasonably educated about the problem being addressed by the script or function can understand every step. If you use standard Matlab functions in a standard way you can comment at a somewhat higher level and somewhat less frequently. Any place you do something particularly clever or unusual you should put more comments in so that the code is clear to the reader!

5. If you can break the entire code down into modules of reasonable complexity you should do so, and put each module into its own m-file so that it can be more easily understood and reused.

6. Although Matlab allows you to put “subfunctions” in the same m-file as a “main” function, you should not do this unless you have a very good reason to do so. In general it just makes the structure more complex and also limits your ability to re-use your subfunction for some other purpose later on.

7. Use indentation as appropriate to indicate control flow, for example with *if* statements or loops. Always use comments to indicate which *end* statements correspond to which control structure.

Other good practices we are not insisting on or which are beyond the scope of the programming in this course: There are many other features of good programming practice / software engineering we are not asking for in this course. Of course if you want to use any of them you are *enthusiastically* encouraged to do so. They include checking input arguments (essential for any program to be used by real users), using range and scope restrictions, defining data structures such as structs, using version control, code repositories (you can look up *git, bitbucket, and github* on-line for some examples of modern open-source solutions) and organized code review, defining unit tests for each code module, doing regression testing and coverage testing, and many other practices.