Computational Modeling & Simulation I (Interdisciplinary Science)
Colby College CS346, Spring 2020

CS346 HW Notes & Guidelines

Here are some notes and guidelines regarding the preparation, presentation, and submission of assignments in CS346. Please apply them to all HW assignments in the course. Many of them are repeated from the guidelines included on HW1, but some are new to provide additional guidance.

Please note that some of the CS346 Matlab Programming Style Notes, presented in a separate document, also apply to presenting HWs in CS346.

As always, feel free to ask me any questions about them. I hope you find them helpful!

General HW Submission Instructions; Naming Conventions for HW Folders

For each exercise, please electronically submit your code in `.m` files. Your work for these exercises should be placed in a folder called `HW<n>_<userid>` (e.g., for my HW2, the folder would be called `HW2_eaaron`); if HW is done in a team, the folder name should include the userid of each person on the team (e.g., a folder named `HW3_eaaron_bmaxwell` from the team of Eric Aaron and Bruce Maxwell).

As a general rule, in that folder should be one Matlab `.m` file per exercise—e.g., for HW1, because there are three exercises, there would be three files of Matlab code in the folder—along with one PDF document containing your write-up for that HW.

Each homework exercise should be submitted in its folder to the submitter's `Private` directory in their CS346 filespace: Go to `filer.colby.edu/Courses/CS346` to find your folder in the course's filespace; you should find a sub-folder called `Private` in your filespace; submit your work by creating a sub-folder of `Private` to prevent unauthorized access to your work. **Please make sure all submissions are to your `Private` folders!**

In cases where work is done in teams, only one member of each team needs to submit the team's work to their filespace. As noted above, however, please make sure the directory name (and all filenames) contain identifiers for every team member.

In general, please submit one file per exercise (not per HW, but per exercise on an HW) unless explicitly instructed otherwise. If you feel more than one file is needed—say, because some long function is used repeatedly and it would impair readability to inline it every time it's needed—then please explain that in your write-up.

Naming Conventions for Files

As part of file naming conventions for the course, please make sure an identifier—such as your name, your userid, or your initials—is part of every file you submit. (Comparing multiple files in a multi-tab editor is very difficult when filenames do not immediately identify their sources!)

For exercises in CS346 done in teams, each filename (including that of the folder in which the exercises are submitted; see above) should contain an identifier for each person on the team. For example, a file named `eaaron_djskrien_cs346_hw3_ex2.m` or `cs346_ex2_ds_ea` or something similar would be submitted for exercise 2 on CS346 HW3, done by the team of Eric Aaron and Dale Skrien.

Write-ups

As mentioned in a previous section, you'll submit one PDF write-up with each HW. (Please follow the above naming conventions!) Your write-up should contain an English description of the code written for the exercises, along with answers to questions specified to be included in the write-up. The write-up can be helpful in documenting your work and ensuring that readers of your code can easily understand how your code works and what its inputs (if appropriate) and outputs are. As usual, code or write-ups that cannot be easily understood may not receive full credit; please feel free to ask your Prof. any questions you have about presentation and explanations in CS346!

In general, explain your answers to all exercises (especially exercises to be answered in write-ups for your assignments!)—that is, explain what your answers mean, and also explain your work or ideas that led to those answers. As part of this, when a question asks for a conclusion based on data, please explicitly show the data used to arrive at that conclusion (but, see below!). In some cases, it might also be important how the data are presented (e.g., in a table, with a graph, with verbal descriptions).

As part of this, if multiple tests or multiple trials are used to arrive at conclusions, be sure to mention that in your write-up, and present (at least summarize—see below!) what those trials show.

For exercises that require plotting values or other visualization of data, it can significantly improve presentation to include images from your Matlab plots in your write-up document.

Please make decisions about what to include in write-ups based on whether it contributes to a *clear presentation of the information you intend to convey.* This means presenting *enough* information but not *too much* information: Show output from enough tests to convey command of the relevant concepts, making clear to readers that your conclusions are supported by your work, but not so much as to be overwhelming. If you've run multiple tests or experiments to support your conclusion, for instance, you may not need to include output from every test; sometimes, a subset of the output is sufficient to convey the ideas. For example, if you did 200 tests, you might consider showing only 4–6 of the visualizations, if that's sufficient to support your conclusions... if you were a reader, would you learn more from 200 visualizations than from 5 that convincingly conveyed the key points, or would those 200 visualizations feel like an overload? In your write-up, clear communication *is* the point; please choose what to include accordingly!

When you do provide only a subset of the data you generated, however, please document your process to enable full replicability. Give a full list of the tests run (e.g., the parameter values tested) and indicate that the output included in the write-up is a representative sample to illustrate the key points. Also, please give readers enough information so that they could fully replicate your work by running all of the same tests you did.

Ideally, code would be so well documented and write-ups would be so clearly presented that a grader could fully evaluate an assignment without even running the code! Please strive for that in your presentation.

Code Output

For every exercise (or part of an exercise) that calls for calculation of a value, be sure to include Matlab code that calculates and displays the specified values, so that when your submitted code is run according to the instructions you provide, the relevant code is executed and the specified values displayed. That is, code should display answers to the relevant questions to the screen when it is run, not just simply store answers in variables that could be examined in the Matlab *Workspace*. In addition, please do not simply include commented-out (hence, not executed when run) blocks of the relevant code, which would require code modification to result in complete answers for an exercise.

In addition, please make sure your code outputs *only* the values required for the exercise, without clutter that could impede readability. In general, please make sure your work matches the given specifications, for programming exercises as well as for verbal answers in write-ups! As part of this, please express answers in the terms requested by an exercise. For example, if asked to express how data varied with some quantity X, a response should express how data vary with X in particular, not some other quantity or variable.

Moreover, please do not output anything without a descriptive label! Easily readable and understandable output is essential for good code (and good code testing). Please use appropriate output statements such as `fprintf` and `disp`, both of which can be used to display information to the screen, to label output appropriately. (You may often find that `fprintf` yields cleaner output—please use `fprintf` when appropriate!)

Code that, when run, does not produce easily readability and understandable answers to all the relevant computational exercises is not a full-credit response to a homework assignment. As always, please let me know if there are questions about this!

Code Efficiency

Although the speed of your code is not the dominant concern in this course, efficiency counts to some extent: If code submitted for an exercise is obviously very inefficient, it may not receive full credit. Similarly, excessively inelegant answers—which may give correct output but be extremely lengthy or convoluted—may not receive full credit.

(As an example of this, if code uses lengthy matrix operations when fast scalar operations would suffice, and code becomes excessively slow because of this, it may not be a full credit solution for an exercise.)

Please note that overloading or abusing Matlab syntax by using keywords or built-in function names as variables can be very, very inefficient. Avoid, e.g., using `sum` as a variable name.

In addition, although memory keeps becoming cheaper and more plentiful, memory efficiency is relevant. Algorithms that have needlessly memory-intensive data structures or algorithms may not scale well to very large applications. Please try to write memory-efficient code! As with time efficiency, code that is excessively inefficient regarding memory usage may not merit full credit.

As always, please feel free to ask me any questions you might have about these HW notes or guidelines for CS346!