ENGR 132 Programming Standards

Programming standards are the guidelines you will use to write code that is understandable by others, well documented, and develops good programming habits.

The standards in this document are required for all MATLAB code submissions in this course.

Why Follow Programming Standards?

- **Make code easier to understand**. Your code will be read by team members, graders, instructors, and eventually co-workers and supervisors. You will read code written by others. Well-written code makes its purpose and logic clear, and it is efficient for others to use.
- **Help minimize errors**. Readable, easy-to-follow code helps minimize errors and allows for easier debugging when errors do occur.
- **Develop good habits**. Learning these standards now will help you be an efficient programmer and help you resist developing bad programming habits.

The course materials will attempt to abide by these standards as closely as possible to serve as an example. However, sometimes headers and comments may be excluded to save space in the course materials; this is not an option for your assignments.

Use the Appropriate Header Template

You must use an ENGR 132 course header template for most assignments. There are two templates, one for script files and one for function files. Use the correct header for the type of MATLAB code you are developing (see pages 4-5). If you are provided with a template for a problem, use that template. Otherwise, you can download a generic template from Blackboard.

All m-files, regardless of type, need to follow these guidelines:

- Other than adding the requested information to the header, do not modify the header in any way. The header in the template must not interfere with the execution of your code.
- Use only your Purdue email address and your official login (no aliases).
- Record the full name and email address of every student with whom you collaborate on your code's development. This is especially important in paired programming and team activities.
- Do not delete or modify the academic integrity statement at the bottom of the template.
- Provide a description of the script/function. Anyone looking at your code, including your grader, should be able to determine from your description of the algorithm what you were attempting to implement.

Function files have a modified header and need to follow these additional guidelines:

- The header in a function file starts on the second line of the code; keep the function definition line as the first line in the code.

- Give a short description and specify the units for each input and output argument variable in the header. This is particularly important for inputs because there is no place to describe them in the body of the code.

Organize Your Script/Function

Organized code is easier to read and understand. Use the following sections as guidelines when developing your code:

Initialization	Assign constants or variables, import data, and receive input from a user
Calculations	Perform calculations and manipulate inputs
Formatted Text & Figure Displays	Generate any final results, including figures and print statements
Command Window Outputs	Paste command window outputs; use only when the assignment requires it
Analysis	Answer assigned questions for a given problem. Present in problem-specific templates

These sections should be used whenever possible. When you are provided with a problem-specific template, then you must use the sections provided in that template.

Use Variables and Hardcoded Values Appropriately

- Use descriptive variable names that make the variable's purpose clear without having to look at any more code.

  Consider these examples:

  Code 1 *vs* Code 2

```
25  % --- INITIALIZATION ---
26  x = 100
27  y = 88
```

```
25  % --- INITIALIZATION ---
26  numStudents = 100
27  avgGrade = 88
```

  The variables x and y are vague and non-descriptive while numStudents and avgGrade clearly describe what they represent.

- Assign only one variable per line.
- Assign all calculations in your code to variables.
- Assign hardcoded values to variables, unless the values are used for array indexing or as equation constants (e.g. `circumference = 2*pi*radius`).
- Do not hardcode values for printing or for use in intermediate steps within the code.

Comment Your Code

Comments provide information to help users understand the code. MATLAB uses % to define a comment.

- Use comments to describe the purpose of all variables and constants. Include units where applicable.

- Use comments to inform others (particularly the grader) what exactly you are attempting to implement. Comments at the top of code blocks should explain the purpose of the next few lines of code.
- Use only single-line comments. Do not use %{ for multi-line comments.

Formatting

Code formatting shows how the code is structured and makes it more readable to other programmers. Good formatting uses whitespace and indentation to organize the code.

- Indent selection structures and loops. See examples 1 and 2 below.

```
% Example 1: Selection Structure with indentation
if r == 1                   % Location is left end of array
    new_temp = (temp_matrix(r+1) + temp_matrix(r))/2;
else                        % Location is not the left end of array
    new_temp = (temp_matrix(r-1) + temp_matrix(r))/2;
end
```

```
% Example 2: Selection Structure/Nested Loops with indentation
if n < 2 || m < 2           % Checks that matrix is at least 2x2
    error('Please enter a matrix with dimensions of at least 2x2')
else
    for x = 1:n             % Steps through the rows of the matrix
        for y = 1:m         % Steps through the columns of the matrix
            new_temp(x,y) = temperature_update(temp_matrix,x,y);
        end
    end
end
```

- Place an additional line between sections or code blocks to help “group” the code.
- Place a space between operators and operands to improve readability. For example, y = y - g is better than y=y-g.

(These standards were originally adapted in 2012 from CS 15900 Documentation, Programming, and Course Standards.)

General Script Header Template

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% ENGR 132
% Program Description
%   ...
%
% Assigment Information
%   Assignment:     PS ##, Problem #
%   Author:         Name, login@purdue.edu
%   Team ID:        ###-##
%   Contributor:    Name, login@purdue [repeat for each]
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%% _____________________
%% INITIALIZATION


%% _____________________
%% CALCULATIONS


%% _____________________
%% FORMATTED TEXT & FIGURE DISPLAYS


%% _____________________
%% ACADEMIC INTEGRITY STATEMENT
% I/We have not used source code obtained from any other unauthorized
% source, either modified or unmodified.  Neither have I/we provided
% access to my/our code to another. The project I/we am/are submitting
% is my/our own original work.
```

General Function Header Template

```
your function definition line goes here
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% ENGR 132
% Program Description
%   ...
%
% Function Call
%   ...
%
% Input Arguments
%   1. None
%
% Output Arguments
%   1. None
%
% Assignment Information
%   Assignment:         PS ##, Problem #
%   Author:             Name, login@purdue.edu
%   Team ID:            ###-##
%   Contributor:        Name, login@purdue [repeat for each]
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%% _____________________
%% INITIALIZATION


%% _____________________
%% CALCULATIONS


%% _____________________
%% FORMATTED TEXT & FIGURE DISPLAYS


%% _____________________
%% COMMAND WINDOW OUTPUT


%% _____________________
%% ACADEMIC INTEGRITY STATEMENT
% I/We have not used source code obtained from any other unauthorized
% source, either modified or unmodified.  Neither have I/we provided
% access to my/our code to another. The project I/we am/are submitting
% is my/our own original work.
```