Mathematical Programming MATH20014: Group projects

February 20, 2023

Assessment criteria

The projects contain a core part (marked **[core]** or **(core)** below) which you need to satisfactorily complete to obtain a pass mark. The other parts are extensions that you can pick and choose. The grading criteria fall under headings 1-5 below.

1. **Correctness.**
   - Does your code do what it is supposed to do: in particular does it perform the computations required in the core parts of the project?
   - Is your code suitably efficient? In other words your code should not use excessive amounts of memory space or time unnecessarily. (Credit will be given for nice space and time optimisations of the code and for explanation of this.)
   - Is your code designed to use appropriate types of data and functions (again with efficiency in mind)? Here are some typical questions that might apply to your code.
     - Have you used a list where it would be better/more correct to simply use a for or while loop, or otherwise to use a numpy array, or a dictionary.
     - Have you written a recursive function for a task when it would have been better to write an iterative function?
     - In designing a recursive function have you used appropriate memoization (if applicable)?
   - Is your code syntactically correct? (This includes using correct naming conventions of variables.)
   - Is the mathematics that you implement in your code correct?
2. **Quality: structure and design.**
   - Is the structure of your code clear and easy to maintain (i.e. understand/adapt/modify at a later date)? This involves a number of good design choices. For example you should keep the following in mind.
     - Have you repeated the same piece of code in two or more places? If so, should not this code be wrapped up as a function?
     - Is your code written so that the different tasks achieved by it are suitably broken up and organised, for example into separate functions?
     - Have you written two or more functions in different parts of the project that achieve the same thing? If so should they not be redesigned to be the same function, or using the same function?
     - Is the way that your code handles functions suitably designed? For example should you be making more use of the fact that you can pass functions in as inputs to other functions?
     - Are you making the right choice for the use of local versus global variables and, if you are using global variables, is their use fully transparent to the user?
     - Should you organise the functions that you design into one or more modules to easily use them elsewhere? (Creating modules is not a necessary part of the project but may be appropriate in certain cases.)

- Is your code designed in a user friendly way? Here are some points to keep in mind.
  - Is any graphical output clear and suitably embellished?
  - Have you designed functions to give the user appropriate printouts? Should you supply a verbose parameter/switch to the user to inspect intermediate output?
  - Have you given easy tools to the user for testing your functions? For example using randomly generated input or input supplied by the user interactively.

3. **Documentation and Clarity.**

   Is your code written in such a way that the interested programmer can easily understand what is going on? (This may include yourself if you want to modify your code in several months or years time.) For example, you should keep the following points in mind.

   - Is your code suitably commented? Comments should be meaningful (i.e. explaining what is really happening instead of just repeating what the code itself says), concise and to the point.
   - Have you used meaningful names for your variables and functions? This can considerably improve the clarity of your code.
   - Python has many slick constructs/built-in functions − for example to manipulate lists − that you should not expect the reader/programmer (e.g. someone used to another programming language) to be conversant with. If you have used such constructs, have you always taken sufficient care in explaining their use?
   - For more complicated code, comments inside the code may not be enough for the reader/programmer to easily understand what is going on. Have you added further (syntactic or semantic) explanations into the text of your project in such cases?

4. **Scope.**

   There are two questions here.

   - How well have you developed the core parts of the project in terms of design, insight and explanation?
   - Have you sufficiently developed one or more (depending on the project) of the suggested extensions/non core parts of the project?

   Note that, in terms of scope, we are above all looking for quality rather than quantity: how well thought out are the underlying ideas, how interesting is the mathematical content and/or the computational aspect of your work, how well designed are your algorithms?

5. **Project Writing.**

   How good is the surrounding text of the project itself? Here are some points to keep in mind.

   - The main purpose of the surrounding text is to provide explanation and analysis of your code. How thorough and insightful is your project in this respect?
   - Is your project well structured? For example does it have a well designed introduction, appropriate chapters and a conclusion. (A table of contents is a bonus.)
   - Is there a consistent narrative throughout the project that ties it together? For example are there appropriate introductory and/or concluding remarks in each chapter and are the chapters appropriately cross-referenced?
   - Have you cited your sources and included your references in a bibliography at the end of the project?
   - Is the typesetting of your text well designed and is the English writing correct/free of typos and of good quality?