Computational Linear Algebra 2020/21: Mastery Coursework

Prof. Colin Cotter, Department of Mathematics, Imperial College London

This coursework is the mastery component for MSc/MRes/4th year MSci students only. Do not attempt it if you are a 3rd year student. Your submission, which must arrive before the deadline specified on Blackboard, will consist of three components for the submission.

1. A pdf submitted to the Mastery dropbox, containing written answers to the questions in this coursework.
2. A SHA tagging the revision of your `clacourse-2020` repository, containing your attempts at the exercises so far.
3. A SHA tagging the revision of your `clacourse-2020-mc` repository, containing your code used to answer questions in this coursework. Where possible you should use the functions that you have completed in your `clacourse-2020` repository.

If you have any questions about this please ask them in the Piazza Discussion Forum.

How to create your code repository for this submission

1. Go to this link [`https://classroom.github.com/a/hNGoEX42`] to create your Github classroom assignment repository for this coursework (`clacourse-2020-mc`).
2. Clone the repository on your computer and start working on the questions; in this coursework there is no "skeleton" code and you should just add any functions that you need.
3. Your functions should make use of your additions to the `cla_utils` library from the exercises so far in the course. To make use of these functions, make sure that you activate the virtual environment just like you had to do to attempt the exercises. Then you will be able to use the library after writing `from cla_utils import *`.
4. Don't forget to commit your changes, push them to Github classroom, and record the revisions of both repositories that should be assessed when you submit the coursework on Blackboard.

The coursework marks will be assigned according to:

- 70% for your written work, assessing your understanding, clear communication, and creative extension of examples.
- 30% for code quality: correct use of Git, clear and correct code, sensible use of numpy vector operations, appropriate documentation and comments. Where sensible, organise your code into functions in separate module files (just add your own) with suitable tests of your own devising (just keep everything in the root directory of your cloned `clacourse-2020-mc` repository.

Please be aware that all three components of the coursework (the PDF report, the exercises repository and the coursework repository) may be checked for plagiarism.

Mastery exercise Watch this video (a recording of a talk by Prof Nick Trefethen given to the Electronic Numerical Linear Algebra seminar in summer 2020) `https://www.youtube.com/watch?v=k68Mf8VAQz8`. In 10 pages or less, describe the Vandermonde+Arnoldi idea, filling in details and background where appropriate, making additional references, and using your own computational explorations of one or two of the examples (not the lightning Laplace/Stokes solvers which are too time consuming to investigate here), which you can also adapt or extend as you see fit. You should also provide references to relevant related work. If your document exceeds 10 pages then only the first 10 pages will be marked.

Marking guidance

1. 90-100 Excellent explanation of the fundamental idea, with creative and pertinent extension and investigation of numerical examples, and carefully selected relevant references to published research. Clear, complete code using Numpy vector operations appropriately, with well-designed tests and making use of programming exercises from the course where appropriate. Excellent presentation.
2. 80-89 Good explanation of the fundamental idea, with good extension and investigation of numerical examples, and relevant references to published research. Clear, complete code using Numpy vector operations appropriately, with well-designed tests and making use of programming exercises from the course where appropriate. Very good presentation.
3. 70-79 Good explanation of the fundamental idea, with some extension and investigation of numerical examples, and relevant references to published research. Well-written code using Numpy vector operations appropriately, with testing, making use of programming exercises from the course where appropriate. Very good presentation.
4. 65-69 Good explanation of the fundamental idea, with partial extension and investigation of numerical examples, and references to published research. Readable code using Numpy vector operations in most appropriate places, with testing, making use of programming exercises from the course where appropriate. Very good presentation.
5. 60-64 Good explanation of the fundamental idea, with some good ideas for extension and investigation of numerical examples but not developed far enough to draw useful conclusions, some account of references to published research. Readable code using Numpy vector operations in places, with partial testing, making use of some programming exercises from the course. Reasonable presentation.
6. 55-59 An explanation of the problem and the work of others on it, but without much independent implementation of the candidate's own. Some code is present with partial testing, making use of some programming exercises from the course, but also relying on Numpy/Scipy implementations where exercises were incomplete.
7. 50-54 As above but in some way defective. For example, one of: few references, some unclear text, poorly presented, code untested, Numpy vector operations not used; however still showing some understanding.
8. 40-49 Poor understanding; several of - few references, some unclear text, poorly presented, code untested, Numpy vector operations not used.
9. 20-39 Very poor understanding; no code present, few references, unclear text, poorly presented.
10. 0-19 No evidence of understanding; no code present, scant references, unclear text, very poor presentation.