Computational Linear Algebra 2021/22: First Coursework

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

This coursework is the first of three courseworks (plus a mastery component for MSc/MRes/4th year MSci students). Your submission, which must arrive before the deadline specified on Blackboard, will consist of two components for the submission.

1. A pdf submitted to the Coursework 1 dropbox, containing written answers to the questions in this coursework.
2. A SHA tagging the revision of your repository, i.e. the repository for your weekly exercises that you have attempted so far. The SHA is the (nearly) unique hexadecimal code tagging each git revision. To get a list of git revisions, type "git log". You will probably want the most recent revision, which will be at the top. You should enter this SHA as the title of your submission on Turnitin.

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

The coursework marks will be assigned according to:

- 35% for the code for the weekly exercises from the beginning until the end of Section 2.7.
- 65% for the code and written answers to the project work here.

In answering the project work here, you will need to write additional code. This code should be added to your git repository in the cw1 directory. It is expected that it will just be run from that directory, so no need to deal with making it accessible through the installed module.

Some dos and don'ts for the coursework:

- **don't** attach a declaration: it is assumed that it is all of your own work unless indicated otherwise, and you commit to this by enrolling on our degree programmes.
- **do** type the report and upload it as a machine-readable pdf (i.e. the text can be copied and pasted). This can be done by LaTeX or by exporting a PDF from Microsoft Word (if you really must). This is necessary to enable automated plagiarism checks.
- **don't** post-process the pdf (e.g. by merging pdfs together) as this causes problems for the automated checks, and makes the resulting documents very large.
- **don't** write anything in the document about the weekly exercises, we will just be checking the code.
- **don't** include code in the report (we will access it from your repository).
- **do** tell us which git commit is the one you want us to mark. This is done by pasting the SHA: this is a (nearly) unique hexadecimal code for each commit, which you can obtain by typing "git log" and copying it. You should then paste it as the title for your report submission, so that we can easily link the code and the report.
- **do** make regular commits and pushes, so that you have a good record of your changes.
- **don't** submit Jupyter notebooks as code submissions. Instead, **do** submit your code as .py modules and scripts.
- **do** remember to "git add" any new files that you add.
- **don't** forget to git push your final commit!
- **don't** use "git add ." or add files that are reproducible from running your code (such as stored matrices, or .pyc files, etc.)
- **don't** use screenshots of code output. Instead, paste and format it as text.

- **do** document functions using docstrings including function arguments.
- **do** add tests for your code, executable using pytest, to help you verify that your code is a correct implementation of the maths.
- **do** write your report as clearly and succinctly as possible. You do not need to write it as a formal report with introduction/conclusions, just address the questions and tasks in this document.
- **do** label and caption all of your figures and tables, and refer to them from the text by label (e.g. Figure 23) rather than relying on their position within the text (don't e.g. write "in the figure below). This is a good habit as this is a standard requirement for scientific writing.
- **don't** hide your answers to the questions in the code. The code is just there to show how you got your answers. Write everything in the report, assuming that the marker will only run your code to check that things are working.
- If you have any personal difficulties with your work on this course please **do** raise them with the course lecturer by email as soon as possible.

Please be aware that both components of the coursework may be checked for plagiarism. It is fine to work together on the exercises and discuss your answers to project questions but you should write your own code and text, answering the project questions yourself.

Coursework questions

1. (35% of the marks)
   Complete the weekly exercises until the end of Section 2.7 and make sure that the code is committed your git repository and pushed to Github Classroom.

2. (17% of the marks)
   This question concerns the data stored as "C.dat" in the Coursework 1 section on Blackboard. You should download this file, and then it can be loaded using `C = loadtxt('C.dat', delimiter=',')`
   You should verify that you now have a numpy array of shape `(1000, 100)`.
   Each row of C contains a time series for the electrical signal measured on a sample of a particular type of plant tissue when suddenly exposed to light, measured at 100 equispaced times. Each of the 1000 rows represents a different sample.

   (a) Using your `cla_utils` routines, compute the QR factorisation. What do you observe about the entries in R? What does this tell you about the possible variations in the sample? Justify your answer.

   (b) These observations suggest a way to compress the data stored in C, so that it takes up less space. Devise and describe a compression method based upon this QR factorisation. Implement this method as code, and verify that it works (using an automated test if possible). Describe where to find your code in the report.

   (c) How might C have been transformed to make these observations more obvious? How would this affect the compression method (you do not need to implement it again).

3. (16% of the marks) Construct an equispaced 1 dimensional array of length m of values from 0 to 1 using `arange(0., 1.0001, 1./51)`

   (a) Write code to construct the degree 12 polynomial interpolating the points above with the values $f_i = 1$ when $i = 0$ and $i = 50$ and $f_i = 0$ otherwise. The code should use a QR factorisation of a Vandermonde matrix, using your `cla_utils` routines, using Gram-Schmidt, modified Gram-Schmidt and the Householder method. Describe where to find your code in the report.

   (b) Compare the coefficients that you get from the three methods. Investigate the differences by inspecting the Q and R matrices made by each of the three methods.

4. (20% of the marks)

(a) In the case where $m \gg n$ for an $m \times n$ matrix and we only want the reduced QR factorisation, the Householder method described in the exercises for obtaining Q is rather inefficient. Explain why.

(b) All the information required to build Q is contained in the set of v vectors used to construct R. Explain how these v vectors can be stored in the array used to store R (by using the places where R is zero).

(c) Make a new Python function that transforms the input array A into an array containing the entries of R and the v vectors using your method above (we will call these arrays R-v arrays), and verify that it works (using an automated test if possible). Describe where to find your code in the report. Compare the results of the new algorithm with the one that we worked on in the weekly exercises.

(d) Devise an algorithm for computing Q^*b using a provided R-v array, without explicitly forming Q. Describe this algorithm in your report. Implement this algorithm as code and verify that it works (using an automated test if possible). Describe where to find your code in the report. Compare the results of the new algorithm with the one that we worked on in the weekly exercises.

(e) Devise an algorithm for solving least squares problems, using the R-v array, without explicitly forming Q. Describe this algorithm in your report. Implement this algorithm as code and verify that it works (using an automated test if possible). Describe where to find your code in the report. Compare the results of the new algorithm with the one that we worked on in the weekly exercises.

5. (12% of the marks) This question concerns the problem of minimising $\|Ax - b\|^2$ under the requirement that $\|x\| = 1$.

(a) By introducing a Lagrange multiplier $\lambda \in \mathbb{R}$, reformulate this problem as finding the stationary point of an objective function $\phi(x, \lambda)$.

(b) By computing $\frac{\partial \phi}{\partial x}$, find an equation for x (assuming the correct value of λ is known.

(c) By using the QR factorisation of A, and applying the Woodbury matrix identity

$$(B + UCV)^{-1} = B^{-1} - B^{-1}U(C^{-1} + VB^{-1}U)^{-1}VB^{-1}, \tag{1}$$

where B is $m \times m$, C is $n \times n$, U is $n \times m$ and V is $m \times n$, find an alternative formula for the solution of the equation from the previous part.
Under what circumstances for the original minimisation problem is this formula advantageous?
Under these circumstances, devise and describe an efficient algorithm for using this formula to solve for x given A, b and λ. Implement this algorithm as code and verify that it works (using an automated test if possible).

(d) Now we can compute an entire family of vectors x that depend on the value of λ. We need to find the value of λ such that $\|x\| = 1$. Describe a binary search algorithm to iteratively approximate this correct value of λ. Implement this algorithm as code and verify that it works (using an automated test if possible).

(e) Explore this algorithm for various A, commenting on the results.