Computational Linear Algebra 2020/21: First Coursework

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

This coursework is the first of two 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 three 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.
3. A SHA tagging the revision of your `clacourse-2020-cw1` repository, containing any code used to answer questions in this coursework. Where possible you should use the functions that you have completed in your `comp-lin-alg` 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/wGaCbzCx`] to create your Github classroom assignment repository for this coursework (`clacourse-2020-cw1`).
2. Clone the repository on your computer and start working on the questions (adding functions to the files as requested).
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:

- 80% for answers to the questions.
- 10% for overall clarity and succinctness of the written submission. You do not need to write the work as a formal report with introduction/conclusions etc, just answer the questions in order as appropriate. Your answers should all appear in the document, rather than running the code, which is submitted to demonstrate how you answered the questions.
- 10% for code quality in both `comp-lin-alg-course` and `clacourse-2020-cw1`: 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-cw1` 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.

1. (25% of the marks) Any code used to answer this question (not including functions that you include from `cla_utils`) should be added to `q1.py`, which starts by loading up a dataset as a 2-dimensional array A. Each row of the array contains data from a realisation of a random function, with each column corresponding to the values of those functions at a particular point.

(a) Use your `cla_utils` to compute the QR factorisation of A.

(b) What do you observe about the contents of the Q and R matrices?

(c) What does this tell you about the matrix A?

(d) What does this tell you about the dataset?

2. (25% of the marks) Any code used to answer this question (not including functions that you include from `cla_utils`) should be added to `q2.py`, which starts by setting up some point values that we will interpolate.

   (a) Take the heuristic explanation given in lectures of the QR factorisation approach to solving least squares systems of the form
   $$\min_x \|Ax - b\|^2,$$
   and rewrite it as a rigorous proof that the least square solution is achieved.

   (b) The file `q2.py` provides an array of points x and and array of function values f, so that $f_i = F(x_i)$ for some function $F : \mathbb{R} \to \mathbb{R}$ which we have only observed at points. We will approximate F by polynomial interpolation. For some polynomial degree $m < 11$, we will assume that
   $$F(x) = \sum_{k=0}^{m} a_k x^k,$$
   for unknown polynomial coefficients $a_0, a_1, \ldots, a_m$. By evaluating F at each point x_i, $i = 1, \ldots, 11$, formulate a linear system relating polynomial coefficients and entries in the vector f.

   (c) Using the QR method to solve this system in the least squares sense, compute the polynomial coefficients for $m = 10$, and plot the resulting polynomial showing the values at x_i, $i = 1, \ldots, 11$ as well as for points in between. What do you notice? Make tests for your implementation.

   (d) Using your code implementation, investigate the sensitivity of the polynomial coefficients and the resulting polynomial plots under small perturbations in the values of f_i, $i = 1, \ldots, 11$.

   (e) Using the same method, compute the polynomial coefficients for $m = 7$, plotting the resulting polynomial. What do you notice in comparison with the $m = 10$ case?

   (f) Investigate the sensitivity of the polynomial coefficients and polynomial plots in the $m = 7$ case, comparing with the $m = 10$ case.

3. (25% of the marks) For the equispaced set of points $x_i = (i + 1/2)/M$, $i = -M, -M+1, \ldots, M-2, M-1$, we consider the Vandermonde matrix,
   $$V = \begin{pmatrix} {x_{-M}}^0 & {x_{-M}}^1 & \ldots & {x_{-M}}^p \\ {x_{-M+1}}^0 & {x_{-M+1}}^1 & \ldots & {x_{-M+1}}^p \\ \vdots & \vdots & \ddots & \vdots \\ x_0^0 & x_0^1 & \ldots & x_0^p \\ x_1^0 & x_1^1 & \ldots & x_1^p \\ \vdots & \vdots & \ddots & \vdots \\ x_{M-1}^0 & x_{M-1}^1 & \ldots & x_{M-1}^p \end{pmatrix},$$
   for $p > 0$.

   (a) In the case $p = 5$, compute the QR factorisation using Householder. Plot the data of each of the columns of Q. Describe what you see, and explain it using your knowledge of QR factorisation.

   (b) For large p, compute the QR factorisation using each of your three implementations (classical and modified Gram Schmidt, and Householder), plotting the last 5 columns. Using your knowledge of these algorithms, explain the differences between the plots for the three methods.

4. (25% of the marks) Any code used to answer this question (not including functions that you include from `cla_utils`) should be added to `q4.py`, which starts by loading up a dataset as a 2-dimensional array A. The first column represents depth in a non-dimensional unit, and the second column represents pressure measured in porous rock at that depth below ground, also in a non-dimensional unit. We know that the

geology is such that the type of rock changes below a depth of 1, so the relationship between depth and pressure is different there. However, physical laws tell us that the pressure should be continuous across the boundary at depth 1. We will also assume that the derivative of the pressure should jump up by 5 units from just above the depth of 1 to just below it. We will formulate a piecewise polynomial approximation for the pressure-depth relationship, so that we fit a degree-p polynomial separately for the two rocks, and impose that the pressure continuity conditions across the rock-rock boundary.

(a) Formulate this problem as a least squares problem of the form

$$\min_{Bx=d} \|Ax-b\|^2,$$

i.e. we minimise $\|Ax-b\|$ whilst requiring $Bx=d$, giving the specific matrices A, B and the vectors d and b.

(b) Consider a change of variables such that

$$Q^T x = \begin{pmatrix} y_1 \\ y_2 \end{pmatrix},$$

where Q is an orthogonal matrix, $y_1 \in \mathbb{R}^2$, and $y_2 \in \mathbb{R}^{2(p+1)-2}$. For suitable choice of Q, show how to transform the problem to become

$$\min_{Cy_1=d,\, y_2} \|A_1 y_1 + A_2 y_2 - b\|^2,$$

for some matrix C which you should give an explicit form of (hint: consider QR factorisations).

(c) Show that this problem can be solved by first finding y_1 and then y_2, using QR factorisations. Implement this algorithm as code, providing suitable tests. Present your results for various p, commenting about which might be best.

(d) Consider a smooth, closed curve $C : S^1 \to \mathbb{R}^2$ (S^1 is the unit circle). We will look at constructing approximations to the curve given a set of point values from S^1, i.e. we are given

$$0 \le \theta_1 < \theta_2 < \ldots < \theta_N < 2\pi,$$

and curve points $x_1, \ldots, x_N \in \mathbb{R}^2$ such that $x_i = C(\theta_i)$, $i = 1, \ldots, N$. The goal is to find an approximation $\hat{C}$ to C as follows. We fix an integer $M > 0$ and a polynomial degree $p > 0$, and assume that $\hat{C}(\theta)$ is a degree p polynomial when restricted to each interval $2\pi m/M \le \theta \le 2\pi(m+1)/M$ for $m = 0, 1, \ldots, M-1$. We also assume that $\hat{C}(\theta)$ is continuous and has continuous derivative at the points $\theta = 2\pi(m+1)/M$ for $m = 0, 1, \ldots, M-1$. Our goal is then to minimise the squares error

$$\sum_{i=1}^{N} \|\hat{C}(\theta_i) - x_i\|^2,$$

over the polynomial coefficients for each interval.
Formulate this problem as a constrained optimisation problem again, and extend your approach used in the first part of this question to this curve fitting problem, provide a code implementation and demonstrate and test your results.