Computational Linear Algebra 2020/2021 **(Due: 20/11/2020)**

Coursework #1

Name: George Hutchings *CID:* 01357062

This is my own unaided work unless stated otherwise.
Classwork SHA: 9f229e2b61374f7cc471a2af30de777ad7b37a42
Courseework SHA: 15823f8f4f9acc5100e4214c5bd1ca3c449e789d

Question 1: (25 %)

(a) I used my householder_qr function to QR factorise A, since analysing Q and R produced by Gram-Schmidt methods is problematic due to the inherent instabilities in the Gram-Schmidt algorithm. I checked the properties of a QR factorisation, by test 'test_QR()', that is:

- R is upper triangular
- Q is orthogonal
- $QR = A$

(b) Using the variable explorer I could see that the matrix R was sparse. However to confirm this preconception I created a contour plot of the reduced for of R, $\hat{R}$ as seen in Figure 1.

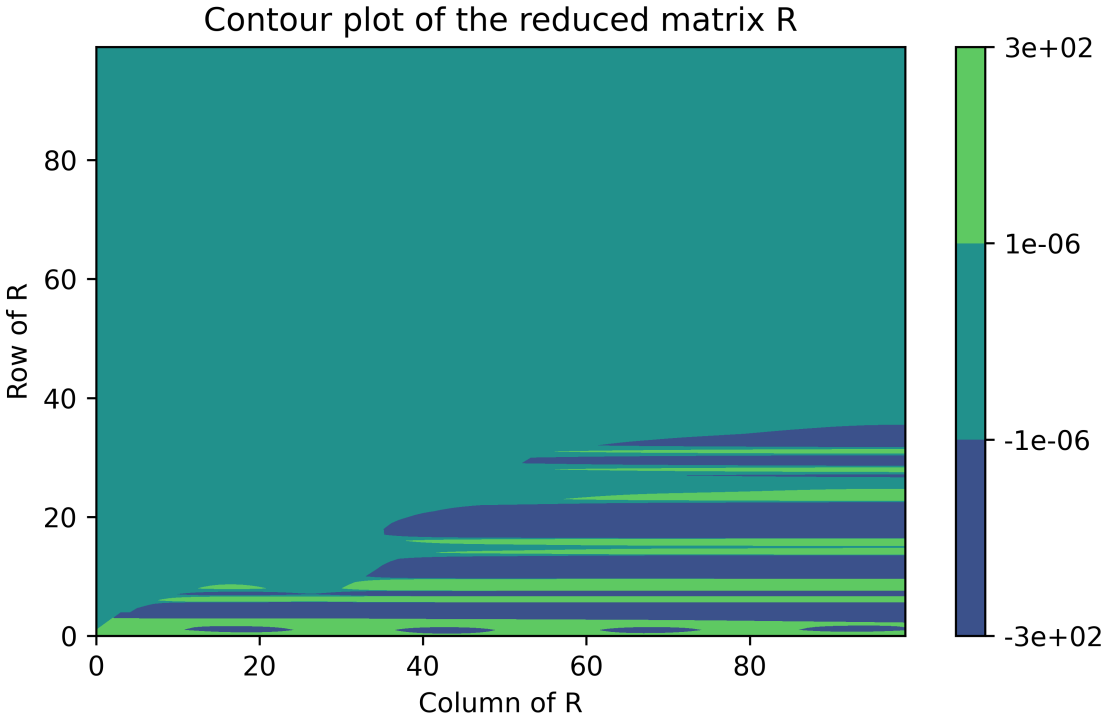

Figure 1: Contour plot of the reduced matrix R

We would expect the upper left part of Figure 1 to be zeros, since R is upper triangular, however we also see that aside from that it is incredibly sparsely populated. Particularly beyond about row 40 where all entries are within 10^{-6} of zero.

To look at this more rigorously see Figure 2, a semi-log plot of the rows of R against their the average magnitude of that row. Note that only the upper triangular (non-zero) part of R has been considered in this calculation.

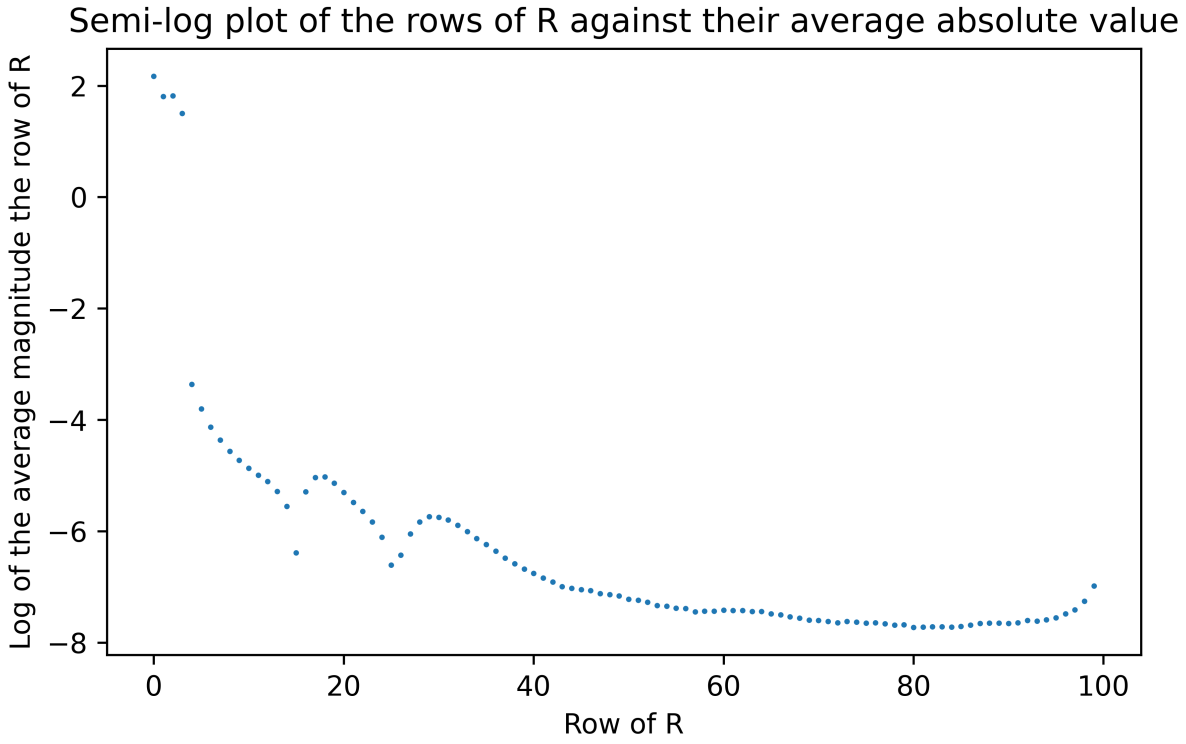

Figure 2: Semi-log plot of the rows of R against their average absolute value

This confirms my preconceptions: the entries in R do indeed become incredibly small the further down the matrix we go.

(c) Consider the (reduced) factorisation $A = \hat{Q}\hat{R}$ which we have calculated in part a.

$$\hat{Q} \coloneqq \begin{bmatrix} \boldsymbol{q_1} & \dots & \boldsymbol{q_k} & \boldsymbol{q_{k+1}} & \dots & \boldsymbol{q_n} \end{bmatrix} \in \mathbb{R}^{m \times n}$$

$$\hat{R} \coloneqq \begin{bmatrix} \boldsymbol{r_1} \\ \vdots \\ \boldsymbol{r_k} \\ \boldsymbol{r_{k+1}} \\ \vdots \\ \boldsymbol{r_k} \end{bmatrix} \in \mathbb{R}^{n \times n}$$

With $\boldsymbol{q_i} \in \mathbb{R}^{m \times 1}$ being the columns of $\hat{Q}$, and , $\boldsymbol{r_i} \in \mathbb{R}^{1 \times n}$ being the rows of $\hat{R}$. As per my observation in part b, I will examine the case when all rows of $\hat{R}$ below k are zero, with k being an integer between 0 and n .

$$A_k \coloneqq Q\hat{R}_k = \begin{bmatrix} \boldsymbol{q_1} & \dots & \boldsymbol{q_k} & \boldsymbol{q_{k+1}} & \dots & \boldsymbol{q_n} \end{bmatrix} \begin{bmatrix} \boldsymbol{r_1} \\ \vdots \\ \boldsymbol{r_k} \\ \boldsymbol{0} \\ \vdots \\ \boldsymbol{0} \end{bmatrix} = \boldsymbol{q_1 r_1} + \dots + \boldsymbol{q_k r_k}$$

But this is just a linear combination of the first k columns of Q (which are orthonormal and so linearly independent). Hence A constructed like this has rank k.

Applying this to the data from the question, and considering 2 cases:

- **Case i:** $k = 4$

  Looking at Figure 2 perhaps the obvious choice is $k = 4$, since after this there is a big drop in the average element size of the rows of R. I approximate A with only the first 4 columns of Q, lets call this approximation A_4 I then calculated (in q1.py):

  $$\|A - A_4\|_F = 0.006005092479951392$$

  Which intuitively seems very small, however, when using the allclose (and its default tolerances) function to compare A and A_4 it returns false, hence despite being very close to A, A_4 is not quite computationally equal. This is validated in my test 'test_sparseR4()'This motivated me to find a more accurate approximation of A.

- **Case ii:** $k = 45$

  I then estimated $k = 45$, since this is where the Figure 2 begins to plateau. I again approximate A but now with the first 45 columns of Q, calling this approximation A_{45} I then calculated (in q1.py):

  $$\|A - A_{45}\|_F = 2.037343305291692e - 06$$

  Which is very small, significantly smaller than $\|A - A_4\|_F$ and, when using the allclose function (and its default tolerances) to compare A and A_{45} it returns true, (This is validated in my test 'test_sparseR45()') hence A and A_{45} can be considered computationally equal.

To conclude A can be approximated very well by the first 4 columns of Q, and can be approximated almost perfectly by the first 45.

(d) We have shown the A can be approximated well by a rank 4 matrix and almost perfectly by a rank 45 matrix.

Consider the data, with $f_j : D \to \mathbb{R}$ our random functions and x_i the points at which its evaluated, $j = 0, 1 \cdots m-1$, $i = 0, 1 \cdots n-1$

$$A = \begin{bmatrix} f_0(x_0) & f_0(x_1) & \cdots & f_0(x_{n-1}) \\ f_1(x_0) & f_1(x_1) & \cdots & f_1(x_{n-1}) \\ \vdots & \vdots & \ddots & \vdots \\ f_{m-1}(x_0) & f_{m-1}(x_1) & \cdots & f_{m-1}(x_{n-1}) \end{bmatrix}$$

Let $\boldsymbol{f} : D \to \mathbb{R}^n$ correspond to the columns of A, that is:

$$\boldsymbol{f}(x_i) \coloneqq \begin{bmatrix} f_0(x_i) \\ f_1(x_i) \\ \vdots \\ f_m(x_i) \end{bmatrix}$$

Assuming that the the x_i are distinct, the hypothesis that the column space of A is spanned by 4 linearly independent vectors (it is of rank 4) means that there are only 4 linearly independent $\boldsymbol{f}(x_i)$, that is, for all i there exists distinct $i_k \in \{0, 1, \cdots, n-1\}$ such that

$$\boldsymbol{f}(x_i) = \sum_{k=1}^{4} a_k \boldsymbol{f}(x_{i_k})$$

Hence all functions $\boldsymbol{f}$ are a linear combination of the same 4 functions. The case where $k = 45$ is considered similarly, and draws the conclusion that: For all i there exists distinct $i_k \in \{0, 1, \cdots, n-1\}$ such that

$$\boldsymbol{f}(x_i) = \sum_{k=1}^{4} 5a_k \boldsymbol{f}(x_{i_k})$$

Hence all functions $\boldsymbol{f}$ are a linear combination of the same 45 functions.

Question 2: (25 %)

(a) For

$$\begin{aligned} A &\in \mathbb{R}^{m\times n} \\ Q &\in \mathbb{R}^{m\times m} \\ R &\in \mathbb{R}^{m\times n} \\ \boldsymbol{\alpha}, \boldsymbol{\beta} &\in \mathbb{R}^{m} \end{aligned}$$

Where:

$$\boldsymbol{\alpha} = \begin{bmatrix} (Q^T\boldsymbol{b})_1 \\ \vdots \\ (Q^T\boldsymbol{b})_n \\ 0 \\ \vdots \\ 0 \end{bmatrix} \implies \hat{\boldsymbol{\alpha}} = \begin{bmatrix} (Q^T\boldsymbol{b})_1 \\ \vdots \\ (Q^T\boldsymbol{b})_n \end{bmatrix}$$

$$\boldsymbol{\beta} = \begin{bmatrix} 0 \\ \vdots \\ 0 \\ (Q^T\boldsymbol{b})_{n+1} \\ \vdots \\ (Q^T\boldsymbol{b})_m \end{bmatrix} \implies \hat{\boldsymbol{\beta}} = \begin{bmatrix} (Q^T\boldsymbol{b})_{n+1} \\ \vdots \\ (Q^T\boldsymbol{b})_m \end{bmatrix}$$

$$\implies Q^T\boldsymbol{b} = \begin{bmatrix} (Q^T\boldsymbol{b})_1 \\ \vdots \\ (Q^T\boldsymbol{b})_n \\ 0 \\ \vdots \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ \vdots \\ 0 \\ (Q^T\boldsymbol{b})_{n+1} \\ \vdots \\ (Q^T\boldsymbol{b})_m \end{bmatrix} = \boldsymbol{\alpha} + \boldsymbol{\beta}$$

Claim: $\bar{\boldsymbol{x}}$, the solution to the upper triangular system below (1), minimises $\|A\boldsymbol{x} - \boldsymbol{b}\|^2$, where $\hat{Q}\hat{R}$ is the reduced QR factorisation of A.

$$\hat{R}\bar{\boldsymbol{x}} = \hat{Q}^T\boldsymbol{b} \tag{1}$$

Proof (which we did in last year's Numerical Analysis course with Prof. Schmid):

$$\begin{aligned} \|A\boldsymbol{x} - \boldsymbol{b}\|^2 &= \|Q^T(A\boldsymbol{x} - \boldsymbol{b})\|^2 \\ &= \|Q^TA\boldsymbol{x} - Q^T\boldsymbol{b}\|^2 \\ &= \|R\boldsymbol{x} - (\boldsymbol{\alpha} + \boldsymbol{\beta})\|^2 \\ &= \langle R\boldsymbol{x} - (\boldsymbol{\alpha} + \boldsymbol{\beta})\ ,\ R\boldsymbol{x} - (\boldsymbol{\alpha} + \boldsymbol{\beta})\rangle \\ &= \|R\boldsymbol{x} - \boldsymbol{\alpha}\|^2 - 2\langle R\boldsymbol{x} - \boldsymbol{\alpha}\ ,\ \boldsymbol{\beta}\rangle + \|\boldsymbol{\beta}\|^2 \\ &= \|R\boldsymbol{x} - \boldsymbol{\alpha}\|^2 + \|\boldsymbol{\beta}\|^2 \end{aligned}$$

Where $\langle R\boldsymbol{x} - \boldsymbol{\alpha}\ ,\ \boldsymbol{\beta}\rangle = 0$ because $R\boldsymbol{x} - \boldsymbol{\alpha}$ is zero below row n and $\boldsymbol{\beta}$ is zero above row $n+1$

$$\implies \min_x \|A\boldsymbol{x} - \boldsymbol{b}\|^2 = \min_x \left(\|R\boldsymbol{x} - \boldsymbol{\alpha}\|^2 + \|\boldsymbol{\beta}\|^2\right)$$

Hence the minimum is obtained by $\boldsymbol{x} \in \mathbb{R}^m$ such that $R\boldsymbol{x} = \boldsymbol{\alpha}$ However we can simplify this down, since only the top n rows of R and $\boldsymbol{\alpha}$ are non-zero:

$$\hat{R}\boldsymbol{x} = \hat{\boldsymbol{\alpha}} = \hat{Q}^T\boldsymbol{b}$$

which is exactly our claim. □

(b)

Consider the Vandermonde matrix A, that is,

$$A = \begin{bmatrix} | & & | & | \\ \mathbf{1} & \cdots & \boldsymbol{x^{m-1}} & \boldsymbol{x^m} \\ | & & | & | \end{bmatrix} \in \mathbb{R}^{n\times m} \tag{2}$$

This means we can construct the linear system

$$\begin{bmatrix} | & & | & | \\ \mathbf{1} & \cdots & \boldsymbol{x^{m-1}} & \boldsymbol{x^m} \\ | & & | & | \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_m \end{bmatrix} = \begin{bmatrix} f_0 \\ f_1 \\ \vdots \\ f_n \end{bmatrix} \tag{3}$$

$$\iff$$

$$\sum_{k=0}^{m} a_k \boldsymbol{x^k} = \boldsymbol{f} \tag{4}$$

Where:

$$\boldsymbol{f} := \begin{bmatrix} f_0 \\ f_1 \\ \vdots \\ f_n \end{bmatrix} \in \mathbb{R}^n$$

$$\boldsymbol{x^k} := \begin{bmatrix} x_0^k \\ x_1^k \\ \vdots \\ x_n^k \end{bmatrix} \in \mathbb{R}^n$$

And the a_i's are the coefficients to the polynomial as in (4).

(c) The linear system (3) can be solved by QR decomposition as explained in part a. This is implemented in q2.py , which gives the polynomial coefficients $\boldsymbol{a}$ as:

$$\boldsymbol{a} = \begin{bmatrix} 1. \\ -2.976 \\ -18.733 \\ 12.173 \\ 180.649 \\ -5.968 \\ -657.01 \\ -19.376 \\ 930.06 \\ 16.147 \\ -435.965 \end{bmatrix}$$

I have tested that this polynomial does in fact exactly interpolate the data with my test function 'test_polydegree10()'. The plot of the data and the interpolating polynomial can be seen in Figure 3, we can see that as Runge's phenomenon predicts, since we are interpolating equispaced points with a high degree polynomial we have large oscillations at the endpoints of our data, this can be easily seen with the overshoots at either end of the graph.

(d)

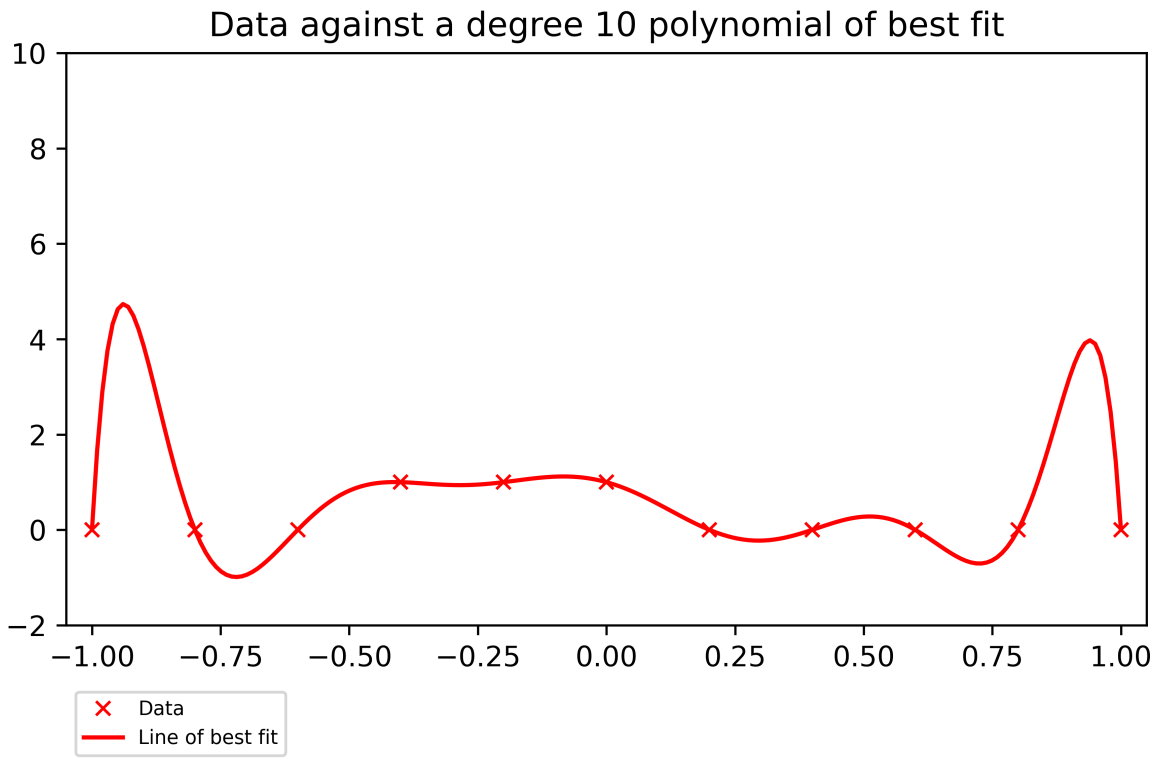

Figure 3: Data against degree 10 polynomial of best fit

We can calculate the relative condition number, that is

$$\begin{aligned} A(\boldsymbol{x}^* + \boldsymbol{\delta x}^*) &= \boldsymbol{f} + \boldsymbol{\delta f} \\ \implies A\boldsymbol{\delta x}^* &= \boldsymbol{\delta f} \\ \implies Q\hat{R}\boldsymbol{\delta x}^* &= \boldsymbol{\delta f} \\ \implies \boldsymbol{\delta x}^* &= \hat{R}^{-1}Q^T\boldsymbol{\delta f} \\ \implies \|\boldsymbol{\delta x}^*\| &= \|\hat{R}^{-1}Q^T\boldsymbol{\delta f}\| \leq \|\hat{R}^{-1}\|\|\boldsymbol{\delta f}\| \end{aligned} \tag{5}$$

Where (5) follows from properties of norms and that Q is orthogonal.
Hence our relative condition number:

$$\begin{aligned} \kappa &= \sup_{\delta \neq 0} \frac{\|\delta x\|\|b\|}{\|\delta b\|\|x\|} \\ &= \frac{\|\hat{R}^{-1}\|\|\delta b\|\|b\|}{\|\delta b\|\|x\|} \\ &= \frac{\|\hat{R}^{-1}\|\|b\|}{\|x\|} \end{aligned} \tag{6}$$

Using the same method for finding the operator norms of matrices as we did in the exercises, that is $\|A^{-1}\|_2 = \sqrt{\frac{1}{\lambda_{min}(A^TA)}}$. We calculate the relative condition number (as seen in q2.py) as 2726.1065637252846, for interpolation with a degree 10 polynomial, which is fairly high suggesting that this problem is ill conditioned for perturbations in $\boldsymbol{f}$.

This can further be seen in Figure 4, where the blue area shows the rough range that the polynomial will lie in when $\boldsymbol{f}$ is perturbed by a vector size 1. The area was found by finding the minimum and maximum points of the line of best fit by considering multiple (200) random perturbations. The blue area is quite large (particularly near the endpoints), hence this further provides evidence that the problem is ill conditioned.

(e) Finding the polynomial of best fit, with degree 7, I used the same method as described in part a and

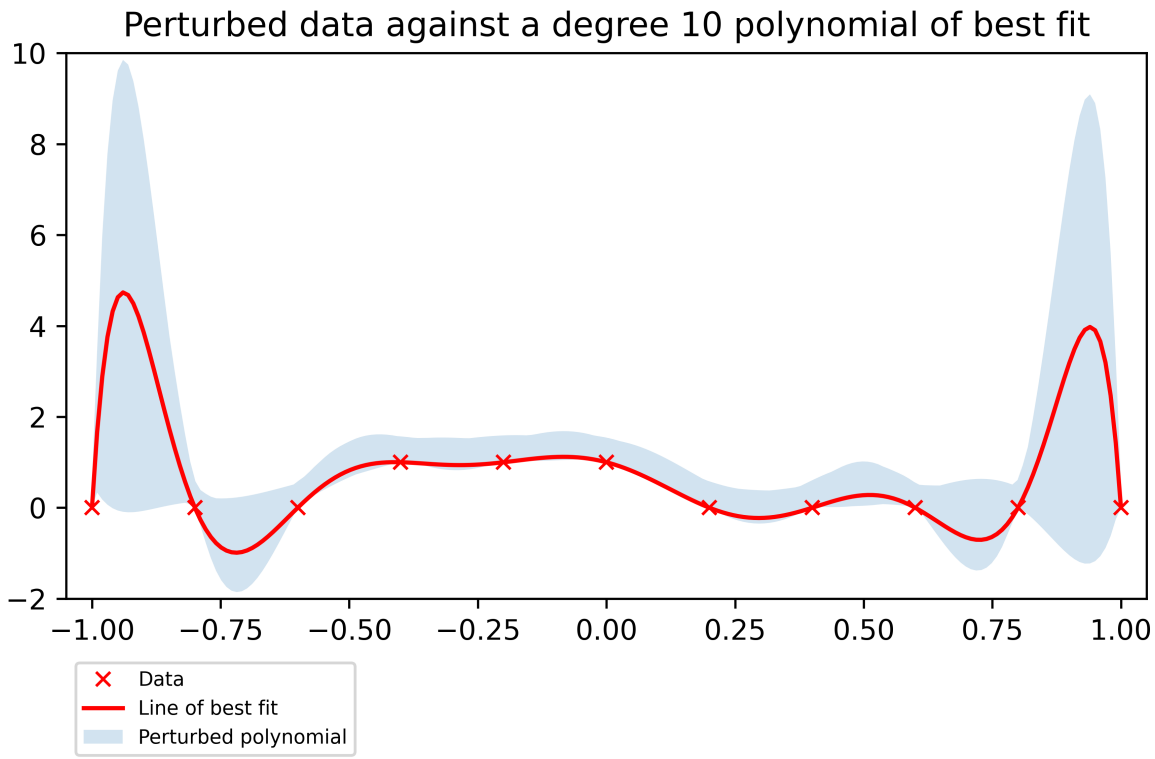

Figure 4: Perturbed data against degree 10 polynomial of best fit

implemented in part b (as in q2.py) and got the below coefficients and plot (Figure 5):

$$\begin{bmatrix} -3.273 \\ -3.486 \\ 17.169 \\ 4.721 \\ -27.574 \\ -2.042 \\ 13.677 \end{bmatrix}$$

We can see from Figure 5 that that the degree 7 polynomial of best fit does not have the same large oscillations at its endpoints as the degree 10 one does. And hence despite not going through all the data exactly; it can generally be considered as a better model of the data, since it would likely allow for more accurate date extrapolation and interpolation.

(f) Computing the relative condition number as we did in part d, but this time for the degree 7 polynomial we get: 56.98907587613343 , which is far less than the corresponding condition number for degree 10, and hence the degree 7 polynomial can be considered better conditioned to perturbations of $\boldsymbol{f}$ than the degree 10 polynomial. Figure 6, which has been produced similarly to Figure 4, also supports this up since the blue area, the region of which most perturbed polynomials will lie, is far smaller in Figure 6 than in Figure 4 .

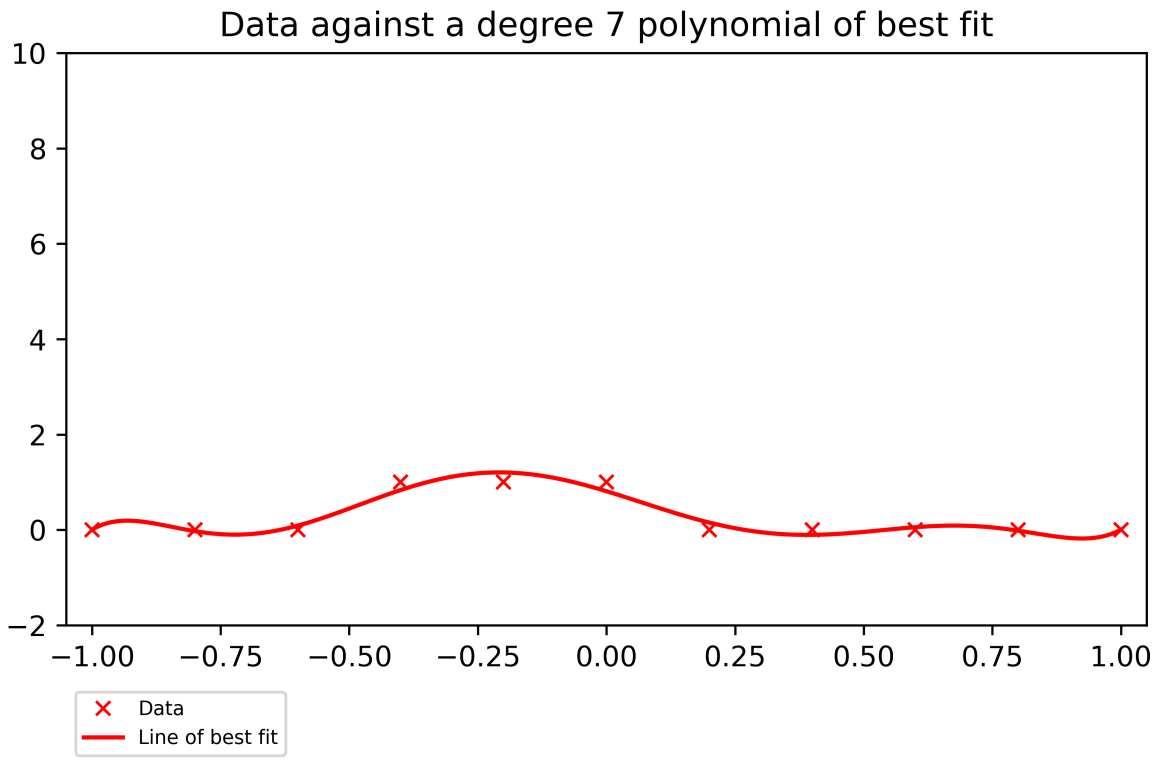

Figure 5: Data against degree 7 polynomial of best fit

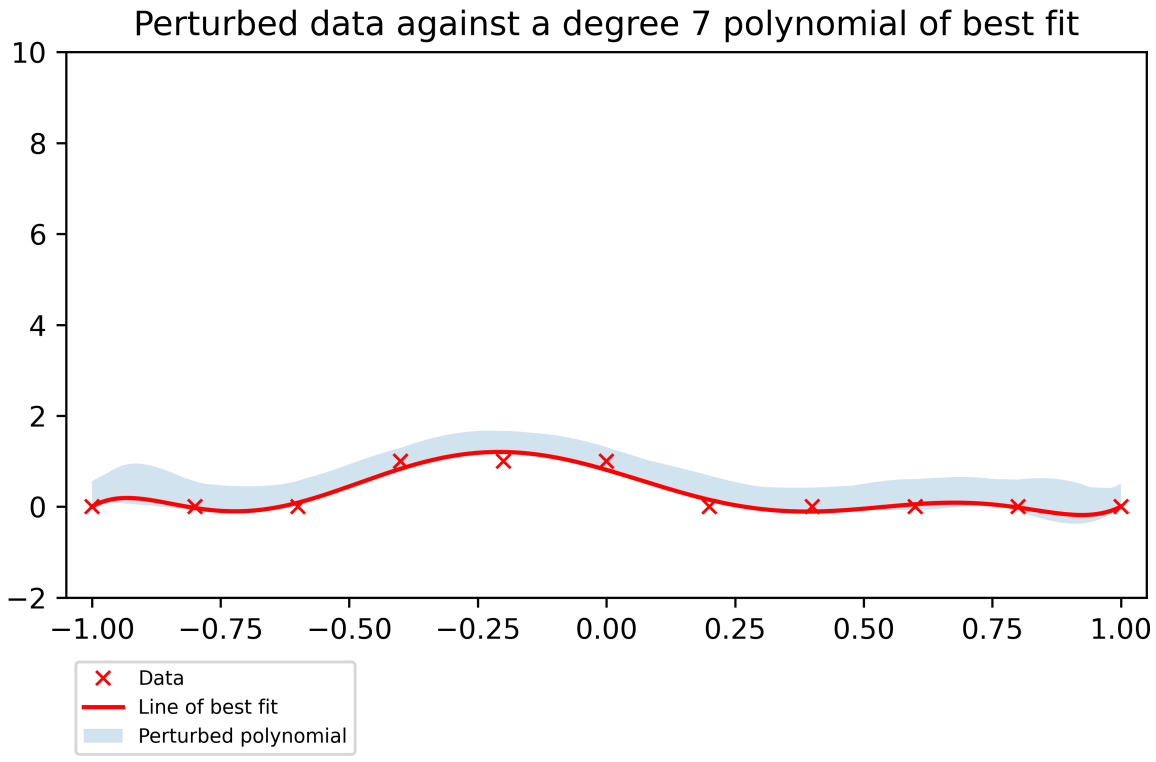

Figure 6: Perturbed data against degree 7 polynomial of best fit

Question 3: (25 %)

(a) When plotting the first 5 columns of Q, which has been produced via QR factorisation by householder on the data mentioned in the question (with M=1000, p = 5) we see Figure 7. Initially, I thought since the columns of Q are orthogonal and our x_i are approximately in $[-1, 1]$ that these could be the first 5 Legendre polynomials. Comparing Figure 7 and the first 5 Legendre polynomials (Figure 8), we can see that each column seems to be a multiple of the corresponding Legendre polynomial, with negative multiples likely down to the 'choice'of sign in the Householder algorithm to maximise stability.

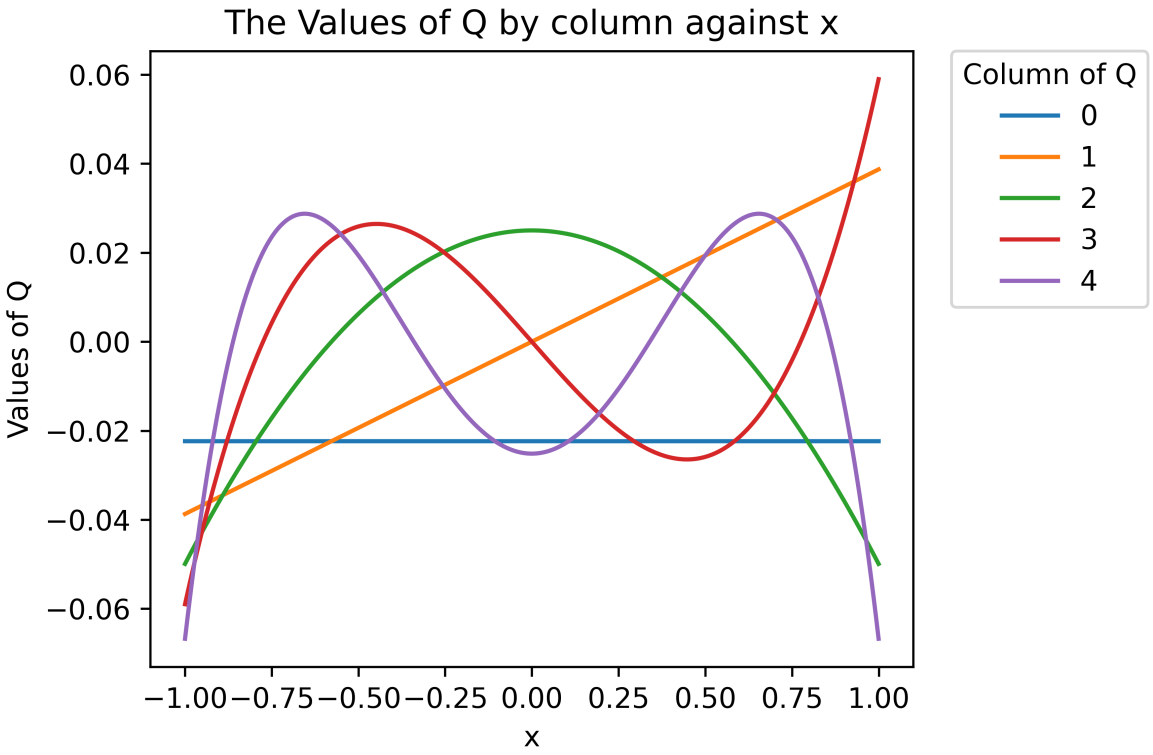

Figure 7: First 5 columns of Q against x

To provide some explanation to my claim: $A = \hat{Q}\hat{R}$ decomposition can be thought of as:

$$\begin{bmatrix} | & & | & | \\ \mathbf{1} & \cdots & \boldsymbol{x^{p-1}} & \boldsymbol{x^{p}} \\ | & & | & | \end{bmatrix} = \begin{bmatrix} | & & | & | \\ \boldsymbol{q_0(x)} & \cdots & \boldsymbol{q_{p-1}(x)} & \boldsymbol{q_p(x)} \\ | & & | & | \end{bmatrix} \hat{R}$$

Where $\boldsymbol{x} = \begin{bmatrix} x_{-M} \\ \vdots \\ x_M \end{bmatrix} \in [-1 + \frac{1}{2M}, 1 - \frac{1}{2M}]^{2M}$

We know due to the orthogonality of Q , the columns of Q are orthogonal, hence:

$$\langle \boldsymbol{q_i(x)} \, , \, \boldsymbol{q_j(x)} \rangle = \sum_{k=0}^{2M} q_i(x_{-M+k}) q_j(x_{-M+k}) = \delta_{ij} \tag{7}$$

Now (in an entirely non rigorous way) lets consider the continuous case, which can be seen as analogous to letting $M \to \infty$, hence instead of considering discrete vectors $\boldsymbol{x^i}$ which are the realisations of continuous functions $\boldsymbol{x}$ lets consider continuous functions themselves. Defining $dx \coloneqq x_{i+1} - x_i = \frac{1}{M}$ and noticing that as

$$M \to \infty, \;\; [-1 + \frac{1}{2M}, 1 - \frac{1}{2M}] \to [-1, 1]$$

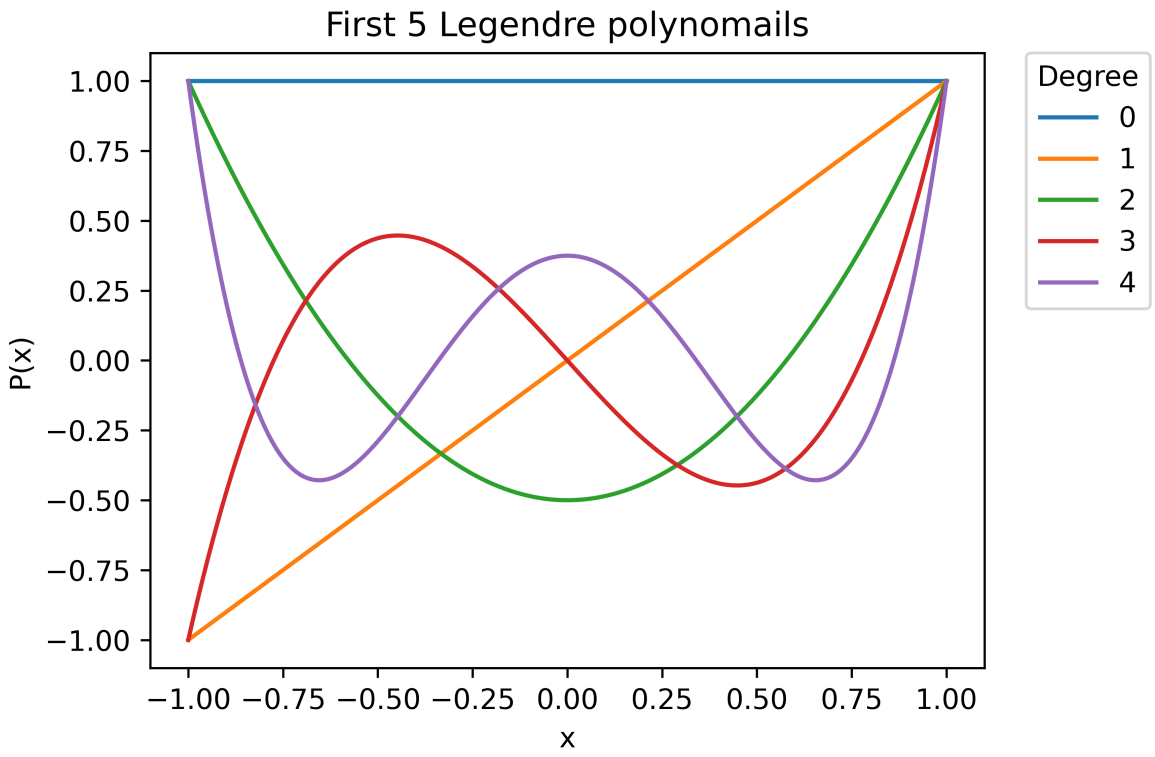

Figure 8: First 5 Legendre polynomials

$$(7) \cdot dx = \sum_{k=0}^{2M} q_i(x_{-M+k}) q_j(x_{-M+k}) dx = \delta_{ij} dx \tag{8}$$

$$M \to \infty,\ 8 \to \int_{-1}^{1} q_i(x) q_j(x) dx = \frac{\delta_{ij}}{M}$$

$$\implies \int_{-1}^{1} \bar{q}_i(x) \bar{q}_j(x) dx = \delta_{ij} \tag{9}$$

Where we have defined $\bar{q}_i(x) = \sqrt{M} q_i(x)$, that is some multiple of $q_i(x)$. We can notice that 9 is very similar to the definition of the Legendre polynomials. Hence this provides evidence to support our claim that each column of Q is equivalent (up to a scale factor) to the Legendre polynomials. We can further notice that like the Legendre polynomials they are likely equivalent to, each $\boldsymbol{q_i(x)}$ is of degree i. Since $A\hat{R}^{-1} = \hat{Q}$ and recalling that $\hat{R}$ is upper triangular $\implies$ $\hat{R}^{-1}$ is upper triangular.

To further support my claim I scaled the (realised) polynomials produced by QR factorisation to ensure that $\boldsymbol{q_i(1)} = 1$. That is I divided each column by its last entry (q3.py). This gave the plot seen in Figure 9 which looks identical to Figure 8 providing more support for my hypothesis.

(b) In figures 10 11 12 the last 5 columns of Q have been plotted (each colour line referring to a different polynomial) by Householder, modified Gram-Schmidt, and classical Gram-Schmidt respectively, for $M = 200$, and $p = 30$. Firstly comparing the Householder and Gram-Schmidt plots, the graph for Householder looks plausible for representing the appropriate Legendre polynomials, since all 5 columns appear distinct on the plot and so can be orthogonal, it also appears that their degrees differ by one which is what we would expect. However for the Gram-Schmidt plots it appears as if there are only two distinct polynomials represented (and hence only two orthogonal polynomials), therefore they cannot represent 5 Legendre polynomials as we know these are orthogonal. I suspect the reason why there are only two polynomials shown is that for such a large p the inherent instabilities with Gram-Schmidt, in that it produces a Q which is not orthogonal, has meant that it cannot produce higher degree orthogonal polynomials. It is interesting to note that instead it alternates between an even and odd polynomial.

Comparing the Modified Gram-Schmidt and the Classical Gram-Schmidt they are very similar as we would

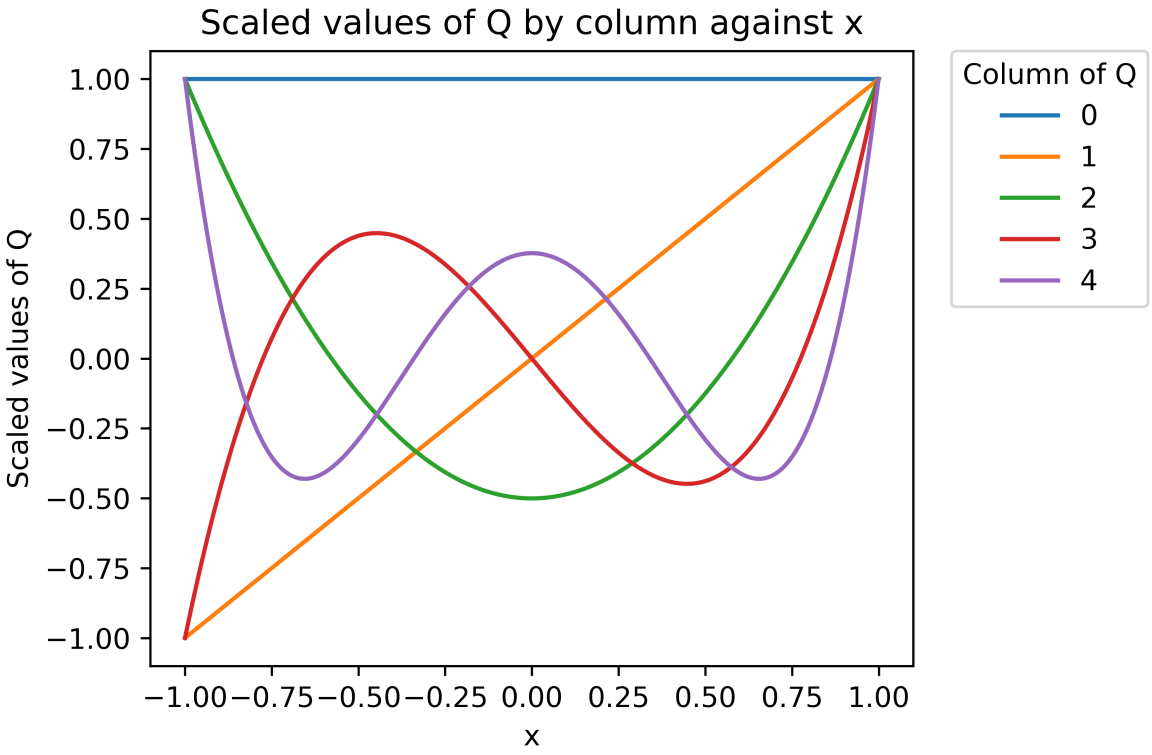

Figure 9: First 5 columns of Q (Scaled) against x

expect, however, looking carefully, one can note that the modified Gram-Schmidt produces a polynomial (Figure 11 the blue line) of two degrees higher than the classical, this is because of the differences to the modified algorithm which give it slightly more stability.

It is difficult to note for large p, however the Householder method will produce some polynomials that differ from those produced by Gram-Schmidt by a factor of minus one, this is because of the 'choice'of sign made during the Householder algorithm (to improve stability) which is not made in GS algorithms. Though the negative column in Q produced by Householder is cancelled by negative values in R as such both algorithms factor $A = QR$.

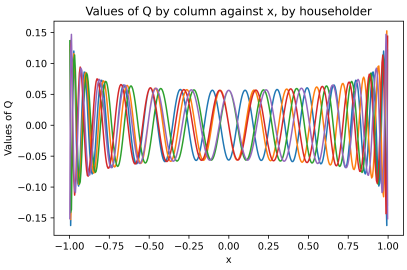

Figure 10: By Householder

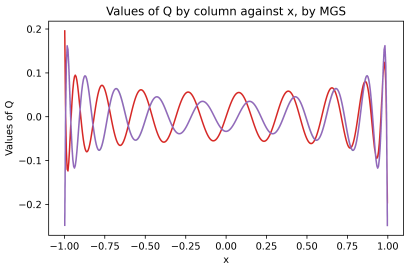

Figure 11: By MGS

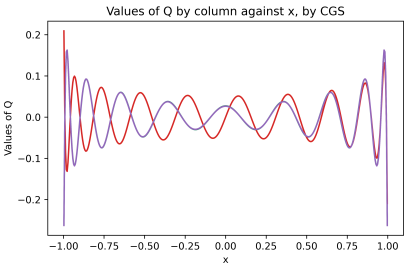

Figure 12: By CGS

Question 4: (25 %)

(a) For this problem we split it into two cases:

- **Case i** Depth above 1

  Let the pressure depth relationship for depth above 1 be approximated by polynomial $P_a(x)$ of degree p, that is:

  $$P_a(x) = \sum_{i=0}^{p} a_i x^i$$

  The coefficients can be calculated via a least squares problem, that is, by minimising:

  $$\|A_a \boldsymbol{x_a} - \boldsymbol{b_a}\|^2$$

  Where:

  - ⋆ $\boldsymbol{d_a}$ is the vector of depths above 1
  - ⋆ $\boldsymbol{b_a}$ is the corresponding vector of pressures for depths above 1
  - ⋆ $A_a = \begin{bmatrix} | & & | & | \\ \mathbf{1} & \cdots & \boldsymbol{d_a^{p-1}} & \boldsymbol{d_a^p} \\ | & & | & | \end{bmatrix}$
  - ⋆ $\boldsymbol{x_a} = \begin{bmatrix} a_0 \\ \vdots \\ a_p \end{bmatrix}$

- **Case ii** Depth below 1

  Similarly, let the pressure depth relationship for depth below 1 be approximated by a polynomial of degree p, that is:

  $$P_b(x) = \sum_{i=0}^{p} c_i x^i$$

  Again, the coefficients can be calculated via a least squares problem, that is, by minimising:

  $$\|A_b \boldsymbol{x_b} - \boldsymbol{b_b}\|^2$$

  Where:

  - ⋆ $\boldsymbol{d_b}$ is the vector of depths below 1
  - ⋆ $\boldsymbol{b_b}$ is the corresponding vector of pressures for depths below 1
  - ⋆ $A_a = \begin{bmatrix} | & & | & | \\ \mathbf{1} & \cdots & \boldsymbol{d_b^{p-1}} & \boldsymbol{d_b^p} \\ | & & | & | \end{bmatrix}$
  - ⋆ $\boldsymbol{x_b} = \begin{bmatrix} c_0 \\ \vdots \\ c_p \end{bmatrix}$

  We can combine these to form one least squares problem:

  $$\min_x \|A\boldsymbol{x} - b\|^2$$

  Where:

  - ⋆ $A = diag(A_a, A_b) = \begin{bmatrix} A_a & \\ & A_b \end{bmatrix}$

$$\star\ \boldsymbol{x} = \begin{bmatrix} a_0 \\ \vdots \\ a_p \\ b_0 \\ \vdots \\ b_p \end{bmatrix}$$

$$\star\ \boldsymbol{b} = \begin{bmatrix} \boldsymbol{b_a} \\ \boldsymbol{b_b} \end{bmatrix}$$

However we have yet to enforce the conditions mentioned in the question.

- Pressure should be continuous across the boundary at depth 1.

$$\begin{aligned} &\implies P_a(x)|_{x=1} = P_b(x)|_{x=1} \\ &\implies \sum_{i=0}^{p} a_i(1)^i - \sum_{i=0}^{p} c_i(1)^i = 0 \\ &\implies \boldsymbol{B_1 x} = 0 \end{aligned}$$

Where:

$$\boldsymbol{B_1} = \begin{bmatrix} 1 & \cdots & 1 & -1 & \cdots & -1 \end{bmatrix} \in \mathbb{R}^{1\times 2p+2}$$

- The derivative of the pressure should jump up by 5 units from just above the depth of 1 to just below it.

$$\begin{aligned} &\implies \left.\frac{dP_a(x)}{dx}\right|_{x=1} - \left.\frac{dP_b(x)}{dx}\right|_{x=1} = -5 \\ &\implies \sum_{i=0}^{p} i a_i(1)^{i-1} - \sum_{i=0}^{p} i b_i(1)^{i-1} = 0 \\ &\implies \boldsymbol{B_2 x} = -5 \end{aligned}$$

Where:

$$\boldsymbol{B_2} = \begin{bmatrix} 0 & 1 & \cdots & p & 0 & -1 & \cdots & -p \end{bmatrix} \in \mathbb{R}^{1\times 2p+2}$$

These can be combined into one condition, namely:

$$B\boldsymbol{x} = \boldsymbol{d}$$

Where:

- $B = \begin{bmatrix} \boldsymbol{B_1} \\ \boldsymbol{B_2} \end{bmatrix} \in \mathbb{R}^{2\times(2p+2)}$
- $\boldsymbol{d} = \begin{bmatrix} 0 \\ -5 \end{bmatrix} \in \mathbb{R}^2$

(b) Consider the QR factorisation of B^T,

$$\begin{aligned} B^T &= QR \\ \implies B &= R^T Q^T \\ \implies B\boldsymbol{x} &= R^T Q^T \boldsymbol{x} \\ &= R^T \boldsymbol{y} \\ &= \hat{R}^T \boldsymbol{y_1} \end{aligned} \tag{10}$$

Where $\hat{R}$ is the reduced form of R, and equality at (10) holds due to the sparsity of R (it is upper triangular). This means that for C as in the question, $C = \hat{R}^T$.

We can calculate the form of C explicitly since it only depends on p as

$$C = \begin{bmatrix} \sqrt{2p+2} & 0 \\ \frac{p\sqrt{2p+2}}{2} & \frac{p\sqrt{6p(p+1)(p+2)}}{6} \end{bmatrix}$$

However this is not insightful as we do not know the sign of the entries since they depend on the sign of the columns of Q, and we need to factorise to get Q to solve the problem so we naturally can calculate R (and so C) this way instead.

This gives the constraints on the problem as:

$$\hat{R}^T \boldsymbol{y_1} = \boldsymbol{d} \tag{11}$$

This is a lower triangular system, that can be easily be solved by back substitution, to give $\boldsymbol{y_1}$.

Considering:

$$AQ = \begin{bmatrix} A_1 & A_2 \end{bmatrix}$$

Where A_1 is the first two columns of AQ and A_2 is the subsequent columns.

$$A\boldsymbol{x} = AQ\boldsymbol{y} = \begin{bmatrix} A_1 & A_2 \end{bmatrix} \begin{bmatrix} \boldsymbol{y_1} \\ \boldsymbol{y_2} \end{bmatrix} = A_1\boldsymbol{y_1} + A_2\boldsymbol{y_2}$$

$$\implies A\boldsymbol{x} - \boldsymbol{b} = A_1\boldsymbol{y_1} + A_2\boldsymbol{y_2} - \boldsymbol{b}$$

$$\implies \min_{B\boldsymbol{x}=\boldsymbol{d}} \|A\boldsymbol{x} - \boldsymbol{b}\|^2 = \min_{C\boldsymbol{y_1}=d, \boldsymbol{y_2}} \|A_1\boldsymbol{y_1} + A_2\boldsymbol{y_2} - \boldsymbol{b}\|^2$$

As required.

(c) As briefly mentioned in part b $\boldsymbol{y_1}$ can be found easily via back substitution into the lower triangular system (11).

To find $\boldsymbol{y_2}$ we solve the least squares problem

$$\min_{\boldsymbol{y_2}} \|A_2\boldsymbol{y_2} - (-A_1\boldsymbol{y_1} + \boldsymbol{b})\|^2$$

Which we know how to do via QR factorisation, this is implemented via our least squares function and can be called by 'q4c(p)'. I tested this, by visually checking that the graphs behaved as expected (they appeared continuous, there was a slight gradient change at depth 1 and they interpolated the data well) for various p, this can be seen in Figures 13, 14, 15. $\boldsymbol{y}$ can be reconstructed by concatenating $\boldsymbol{y_1}, \boldsymbol{y_2}$ as per our definition of $\boldsymbol{y}$. Then:

$$\boldsymbol{x} = Q^T\boldsymbol{y}$$

which is exactly the coefficients of our polynomials.

Comparing Figures 13, 14, 15 which represent fitting the data with polynomials of degree 3, 5, 30 respectively:

- $p = 3$, At low p there is a surprisingly good fit for the data, however perhaps near the end points it lack the ideal fit.
- $p = 5$, At medium p there is a marginally better fit, especially near the endpoints, at the cost of being a more complicated (2 degrees greater) function.
- $p = 30$, At large p there Runge's phenomenon occurs and the data is fitted poorly.

I would say lower degrees say $p = 3$ are optimal as they provide a good fit at low computational cost, although fitting at the endpoints is marginally compromised.

(d)

We have, for $m = 0, 1, \cdots M-1$:

$$\hat{\boldsymbol{C}}(\theta) = \hat{\boldsymbol{C}}_m(\theta) \text{ for } \theta \in \left[\frac{2\pi m}{M}, \frac{2\pi(m+1)}{M}\right]$$

$$\hat{\boldsymbol{C}}_m(\theta) = \sum_{i=0}^{p} \boldsymbol{y_{m,i}}\theta^i$$

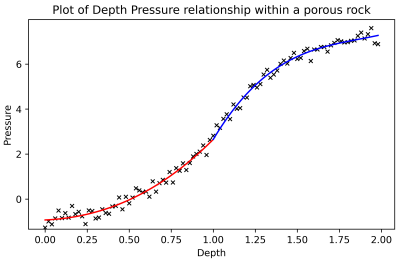

Figure 13: p = 3

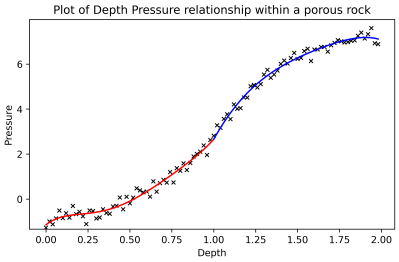

Figure 14: p = 5

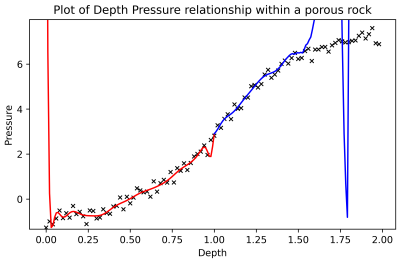

Figure 15: p = 30

Where: $\boldsymbol{y_{m,i}} = \begin{bmatrix} a_{m,i} \\ b_{m,i} \end{bmatrix} \in \mathbb{R}^2$ are the polynomial coefficients.

Applying the constraints:

- $\hat{\boldsymbol{C}}(\theta)$ to be continuous. Let $\varphi_m = \left(\frac{2\pi(m+1)}{M}\right)$

$$\implies \hat{\boldsymbol{C}}_m(\theta) \mid_{\theta=\varphi_m} = \hat{\boldsymbol{C}}_{m+1}(\theta) \mid_{\theta=\varphi_m}$$

$$\implies \sum_{i=0}^{p} \boldsymbol{y_{m,i}}\theta^i \mid_{\theta=\varphi_m} - \sum_{i=0}^{p} \boldsymbol{y_{m+1,i}}\theta^i \mid_{\theta=\varphi_m} = 0$$

$$\implies \begin{bmatrix} \varphi_m^0 & \cdots & \varphi_m^p & -(\varphi_{m+1})^0 & \cdots & -(\varphi_{m+1})^p \end{bmatrix} \begin{bmatrix} \boldsymbol{y_{m,0}} \\ \vdots \\ \boldsymbol{y_{m,p}} \\ \boldsymbol{y_{m+1,0}} \\ \vdots \\ \boldsymbol{y_{m+1,p}} \end{bmatrix} 0$$

Note that we want continuity at $\theta = 0 = \varphi_{-1}, \theta = 2\pi = \varphi_{M-1}$

$$\implies \begin{bmatrix} \varphi_{-1}^0 & \cdots & \varphi_{-1}^p & \cdots & -(\varphi_{M-1})^0 & \cdots & -(\varphi_{M-1})^p \end{bmatrix} \begin{bmatrix} \boldsymbol{y_{0,0}} \\ \vdots \\ \boldsymbol{y_{0,p}} \\ \boldsymbol{y_{M-1,0}} \\ \vdots \\ \boldsymbol{y_{M-1,p}} \end{bmatrix} = 0$$

Expressing this as a linear system:

$$\begin{bmatrix} \boldsymbol{B_{1,0}} & -\boldsymbol{B_{1,0}} & & \\ & \boldsymbol{B_{1,1}} & -\boldsymbol{B_{1,1}} & \\ & & \ddots & \\ -\boldsymbol{B_{1,-1}} & & & \boldsymbol{B_{1,M-1}} \end{bmatrix} \begin{bmatrix} \boldsymbol{y_{0,0}} \\ \vdots \\ \boldsymbol{y_{0,p}} \\ \boldsymbol{y_{1,0}} \\ \vdots \\ \boldsymbol{y_{1,p}} \\ \vdots \\ \boldsymbol{y_{M-1,0}} \\ \vdots \\ \boldsymbol{y_{M-1,p}} \end{bmatrix} = 0$$

Where,

$$\boldsymbol{B_{1,m}} = \begin{bmatrix} \varphi_m^0 & \cdots & \varphi_m^p \end{bmatrix} \in \mathbb{R}^{1\times p+1}$$

- The derivatives are continuous.

$$\implies \frac{d\hat{\boldsymbol{C}}_m(\theta)}{d\theta} \mid_{\theta=\varphi_m} = \frac{d\hat{\boldsymbol{C}}_{m+1}(\theta)}{d\theta} \mid_{\theta=\varphi_m}$$

$$\implies \sum_{i=0}^{p} i\boldsymbol{y_{m,i}}\theta^{i-1} \mid_{\theta=\varphi_m} - \sum_{i=0}^{p} i\boldsymbol{y_{m+1,i}}\theta^{i-1} \mid_{\theta=\varphi_m} = 0$$

Expressing this as a linear system (and similarly noting we want continuity of derivatives at $\theta = 0 = \varphi_{-1}, \theta = 2\pi = \varphi_{M-1}$):

$$\begin{bmatrix} \boldsymbol{B_{2,0}} & -\boldsymbol{B_{2,0}} & & \\ & \boldsymbol{B_{2,1}} & -\boldsymbol{B_{2,1}} & \\ & & \ddots & \\ -\boldsymbol{B_{2,M-1}} & & & \boldsymbol{B_{2,M-1}} \end{bmatrix} \begin{bmatrix} \boldsymbol{y_{0,0}} \\ \vdots \\ \boldsymbol{y_{0,p}} \\ \boldsymbol{y_{1,0}} \\ \vdots \\ \boldsymbol{y_{1,p}} \\ \vdots \\ \boldsymbol{y_{M-1,0}} \\ \vdots \\ \boldsymbol{y_{M-1,p}} \end{bmatrix} = 0$$

Where,

$$\boldsymbol{B_{2,m}} = \begin{bmatrix} 0 & 1 & \cdots & p\varphi_m^{p-1} \end{bmatrix} \in \mathbb{R}^{1\times p+1}$$

Combining these two constraints, where $B = \begin{bmatrix} \boldsymbol{B_1} \\ \boldsymbol{B_2} \end{bmatrix}$

$$\implies \begin{bmatrix} B_1 \\ B_2 \end{bmatrix} \begin{bmatrix} \boldsymbol{y_{0,0}} \\ \vdots \\ \boldsymbol{y_{0,p}} \\ \boldsymbol{y_{1,0}} \\ \vdots \\ \boldsymbol{y_{1,p}} \\ \vdots \\ \boldsymbol{y_{M-1,0}} \\ \vdots \\ \boldsymbol{y_{M-1,p}} \end{bmatrix} = 0$$

Let the integer i_m be the largest such that $\theta_{i_m} < \varphi_m \ m = 0, \cdots, M-1$ (where $\theta_{i_{M-1}} = \theta_N \ \theta_{i_{-1}} = \theta_1$) .

Hence our interpolating polynomial evaluated at these points can be written as $A\boldsymbol{y}$, where $\boldsymbol{y}$ are the polynomial coefficients and A is as defined below:

$$A_m = \begin{bmatrix} 1 & \theta_{i_{m-1}} & \dots & \theta_{i_{m-1}}^p \\ \vdots & \vdots & \ddots & \vdots \\ 1 & \theta_{i_m} & \dots & \theta_{i_m}^p \end{bmatrix}$$

$$A = diag(A_0, \cdots, A_{M-1}) = \begin{bmatrix} A_0 & & & \\ & A_1 & & \\ & & \ddots & \\ & & & A_{M-1} \end{bmatrix}$$

Reducing our problem to:

$$\min_{B\boldsymbol{y}=\boldsymbol{d}} \|A\boldsymbol{y} - \boldsymbol{b}\|^2$$

Where $\boldsymbol{d} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$ Which can be solved as described in part d ,(11), it can be noted that since $\hat{R}$ is full rank by construction, and considering $\boldsymbol{y_1}$ analogous to in (11):

$$\boldsymbol{d} = 0 \implies C\boldsymbol{y_1} = \hat{R}^T\boldsymbol{y_1} = 0 \implies y_1 = 0$$

I implemented my code (q4.py) on a heart shape parameterisation (Figures 16, 17, 18) and varied M, p to see the impact. Even with low M and p, this complicated shape is modelled very well, however increasing p has a large impact, both improving the general shape and fit as seen in Figure 17, which is expected as we are making the problem less over determined. Increasing M although isn't as profitable as increasing p. Although increasing M does give the graph a much better shape, even if it doesn't quite minimise the residuals as much as increasing p.

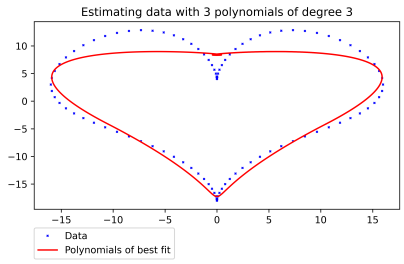

Figure 16: M = 3, p = 3

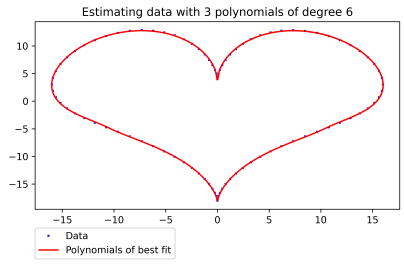

Figure 17: M = 3, p = 6

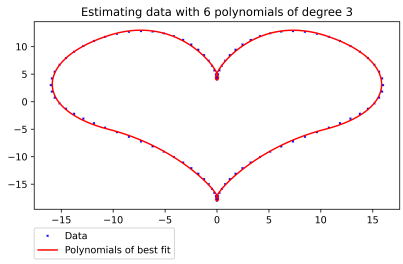

Figure 18: M = 6, p = 3

To test this function I visually checked to see if the plots were smooth and fit the data as expected. I did this with various functions and M, p values as seen in q4.py and in Figures 16, 17, 18, 19, 20, 21.

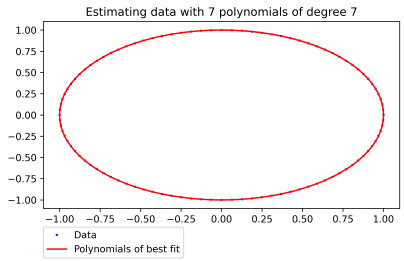

Figure 19: M = 7, p = 7

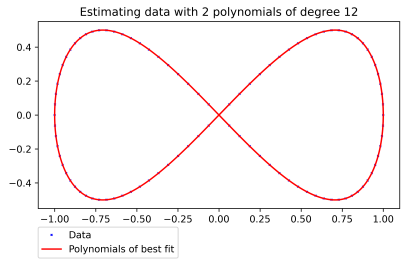

Figure 20: M = 2, p = 12

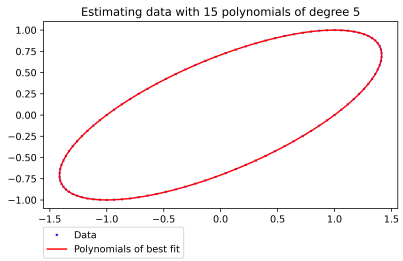

Figure 21: M = 15, p = 5