Coursework 1 - Computational Linear Algebra

Marwan Riach, CID: 01349928

November 2020

To produce all figures in the report, simply run the respective $q_i.py$ files

1 Question One

The QR factorisation of A can be obtained by running the $q1.py$ file. Do be warned that this will take about 1 minute to run, owing to the sheer size of the $values.npy$ file.

The algorithm used here was the householder algorithm in cla_utils. When performing orthogonalisation on a computer, the householder transformation is usually preferred over the GS process since it is more numerically stable and its rounding errors tend to have less serious effects.

For the study of A's properties, we will work with the reduced QR factorisation, $\hat{Q}\hat{R}$ since its contents is much smaller and thus quicker to work with and easier to visualise.

Below is a plot of the rows of A, so as to give us a picture of each row's trajectory as time elapses. Immediately, we can see that all of the trajectories are continuous:

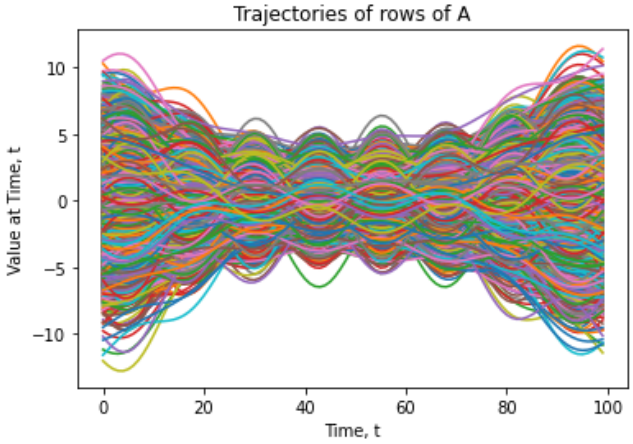

Figure 1: Trajectories of all rows of A

Looking at the reduced QR factorisation and viewing matrix multiplication as a linear combination of columns will help us understand what is going on in the trajectories.

Exporting $\hat{Q}$ to a csv file, we can see that each element of the matrix is moreorless of the same order of magnitude. Exporting $\hat{R}$ to a csv file, we can see that the entries in the lower triangle are of the order between 10^{-25} and 10^{-15}. These are supposed to be zero, and the reasons as to why they are not exactly zero can be attributed to the rounding errors in the algorithm and the rounding itself in the machine used to carry out the algorithm. The non-zero entries of the top 4 rows of $\hat{R}$ also appear to be significantly bigger than the non-zero entries in the subsequent rows. However, the non-zero entries in the subsequent rows are not small enough to be attributed to errors within the algorithm and thus we can say that the matrix A is of full rank (i.e. rank 100).

Below is a heat map of $\hat{R}$ to illustrate the size of each entry in the matrix relative to the other entries:

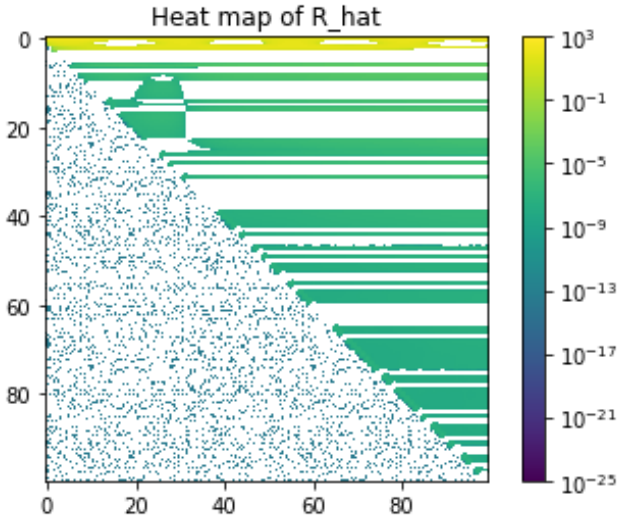

Figure 2: Heat map of $\hat{R}$

This map confirms the fact that the non-zero entries of the top 4 rows of $\hat{R}$ appear to be significantly bigger than the non-zero entries in the subsequent rows and that the matrix is upper triangular.

For a further insight into the origin of each trajectory, we will consider the trajectories as a linear combination of the rows/columns of our $\hat{Q}$ and $\hat{R}$. To see this, let us consider the transpose of A which we denote A^T:

$$A = QR,$$
$$\Rightarrow A^T = R^T Q^T$$

The columns of A^T are the trajectories of our data set and $\hat{R}^T$ is now lower-triangular. Using what we know about matrix multiplication, we can know interpret each realisation of our random function as a linear combination of the columns of $\hat{R}^T$ with the coefficients of the linear combinations being given in the ith column of $\hat{Q}^T$. The columns of $\hat{Q}^T$ can be seen to be the realisations of a random variable.

Since the coefficients are all of roughly the same order of magnitude, we turn to columns of $\hat{R}^T$. Given what we know about $\hat{R}$ and that the size of the entries first 4 rows is considerably bigger than the others, we can say that, up to some tolerance, every trajectory is simply a linear combination of the first 4 rows of $\hat{R}$ with some perturbation given by the subsequent rows of $\hat{R}$.

The last aspect we will investigate is the distribution of the coefficients that are used in the linear combinations of columns of $\hat{R}^T$ (i.e. the elements of the rows of $\hat{Q}^T$). The elements of the ith row of $\hat{Q}^T$ tell us all the coefficients used in the ith column of $\hat{R}^T$ across all 10,000 realisations. Plotting a histogram for two rows in $\hat{Q}^T$ will give us a general idea of the distribution of the coefficients.

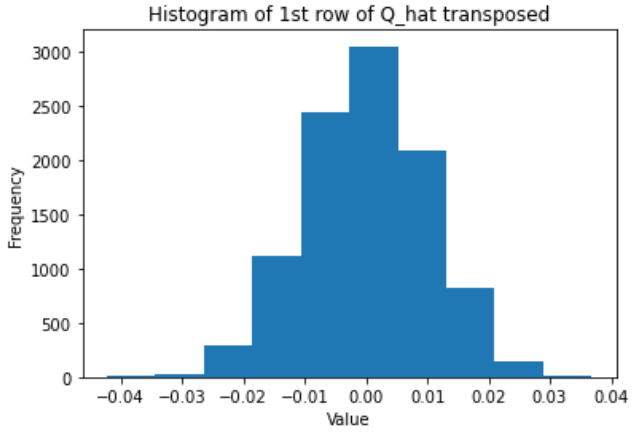

Figure 3: Histogram of 1st row of $\hat{Q}^T$

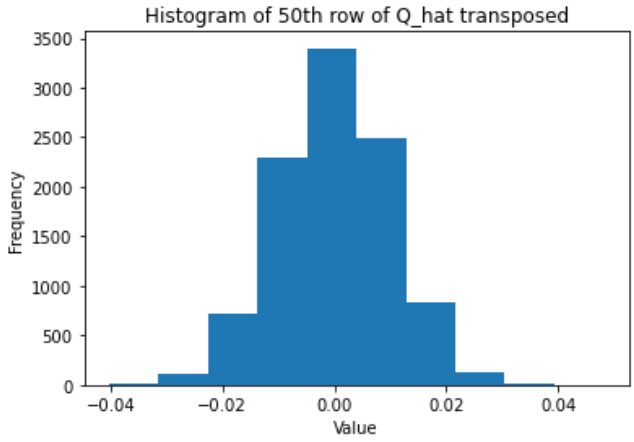

Figure 4: Histogram of 50th row of $\hat{Q}^T$

As can be seen from both figures above, the coefficients appear to be normally distributed with a mean close to zero.

2 Question Two

We seek $\hat{x}$ such that $||A\hat{x} - b||^2 \leq ||Ax - b||^2, \ \forall x \in \mathbb{R}^n$.

Define $f(x) = ||Ax - b||^2$ and consider its partial derivative and gradient:

$$\begin{aligned} f(x) = ||Ax - b||^2 = \sum_{i=1}^{m}(\sum_{j=1}^{n} A_{ij}x_j - b_i)^2, \\ \Rightarrow \frac{\partial f(x)}{\partial x_k} = 2\sum_{i=1}^{m} A_{ik}(\sum_{j=1}^{n} A_{ij}x_j - b_i) = 2(A^T(Ax - b))_k, \\ \Rightarrow \nabla f(x) = 2A^T(Ax - b) \end{aligned}$$

The x which minimises f is that which sends the gradient of f to zero:

$$\begin{aligned} f(\hat{x}) = 2A^T(A\hat{x} - b) = 0, \\ \Rightarrow A^T(A\hat{x} - b) = 0 \end{aligned}$$

This means that $A\hat{x} - b$ is orthogonal to $range(A)$ and thus is also orthogonal to $range(P)$, where P is the orthogonal projector onto the range of A.

From theorem 1.28, we know that $P = \hat{Q}\hat{Q}^T$ with $\hat{Q}$ being taken from the reduced QR factorisation of A. This now means that $A\hat{x} - b$ must be in the null space of P. This is equivalent to $A\hat{x} - b$ being in the range of the complementary projector, $I - P$, and setting $A\hat{x} = Pb$ achieves this.

Writing A in its reduced QR form yields:

$$\begin{aligned} A\hat{x} = Pb, \\ \Rightarrow \hat{Q}\hat{R}\hat{x} = \hat{Q}\hat{Q}^T b, \\ \Rightarrow \hat{R}\hat{x} = \hat{Q}^T b \end{aligned}$$

and $\hat{x}$ can be solved using backwards substitution. $\square$

The linear system relating polynomial coefficients and entries in the vector f can be written as follows:

$$\begin{pmatrix} 1 & x_1 & x_1^2 & \dots & x_1^m \\ 1 & x_2 & x_2^2 & \dots & x_2^m \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_{11} & x_{11}^2 & \dots & x_{11}^m \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ \vdots \\ a_m \end{pmatrix} = \begin{pmatrix} F(x_1) \\ F(x_2) \\ \vdots \\ F(x_{11}) \end{pmatrix}$$

The function LHS_Matrix in the $q2.py$ file constructs the above matrix on the left hand side.

The least squares solution for the a_i coefficients for $m = 10$ is given below in the form of a graph. The black dots represent the data points that were used to construct the polynomial. The coefficients themselves can be outputted by running the $q2.py$ file. The method here that was used to factorise the matrix A (the left-hand side matrix in this case) was the householder algorithm:

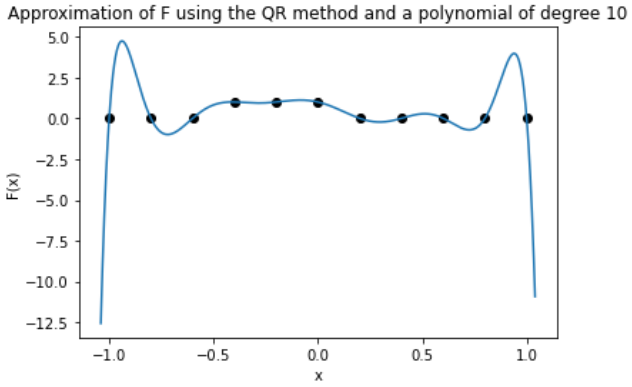

Figure 5: Approximation of F with data points and polynomial of degree 10

The coefficients a_i of the polynomial of degree 10 in ascending order are:

[1, -2.97619048 , -18.73313492 , 12.17344577, 180.64856151, -5.96788194, -657.00954861, -19.37624008, 930.05952381, 16.14686673, -435.96540179]

Here, we have eleven points in the form of a discrete square wave, represented by black dots in the above figure. While the curve fits through all eleven points, the fit is unsatisfactory. Near the ends of the interval, $F(x)$ exhibits large oscillations that are clearly a consequence of the interpolation process and not a reasonable reflection of the data.

This behaviour is typical of polynomial interpolation; the fits produced are seldom good and they get worse with the more data points that are used. To

avoid these problems, we could have used a non-uniform set of interpolation points such as Chebyshev nodes.

*test*1 in the submission tests to see if the polynomial produced in $m = 10$ passes through all data points that are used in the construction of this polynomial. To execute this, run the *testq*2.*py* file.

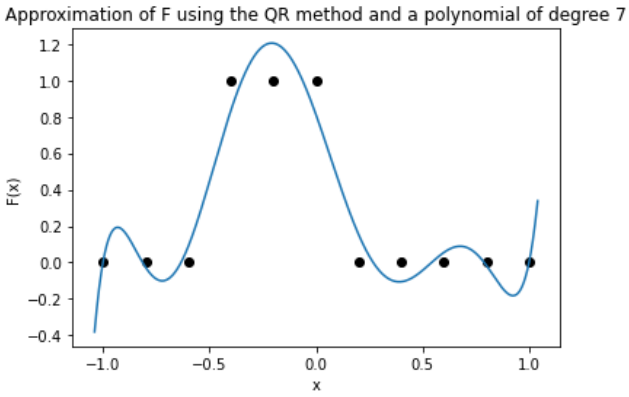

Figure 6: Approximation of F with data points and polynomial of degree 7

The coefficients a_i of the polynomial of degree 7 in ascending order are:

[0.80995475, -3.27307986, -3.48566548, 17.1686463, 4.72127954, -27.57352941, -2.04248366, 13.67734594]

The above polynomial ($m = 7$) does not appear to have wild oscillations in the extremes as compared with $m = 10$, suggesting that this is a superior fit. Furthermore, this polynomial appears to respect the position of the 11 data points relatively well, while not being overfitted, unlike with $m = 10$.

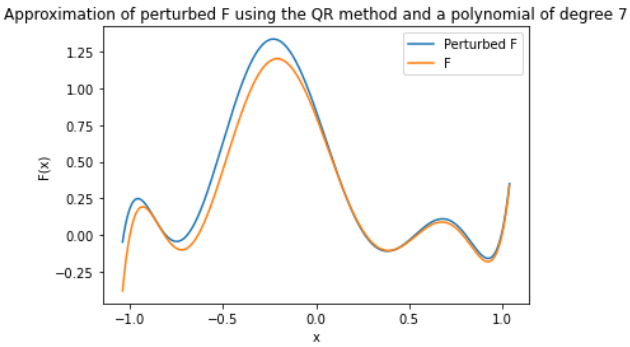

Figure 7: Perturbation of F with polynomial of degree 7

The coefficients of the perturbed polynomial can be outputted by running the *q2.py* file.

The perturbations are applied to the values of F and they are perturbed by adding a realisation of a normal distribution with mean 0 and variance 0.1 to each component.

What can be seen visually from the graphs above and below are that the polynomial of degree 10 is far more sensitive to a perturbation in the initial conditions than the polynomial of degree 7.

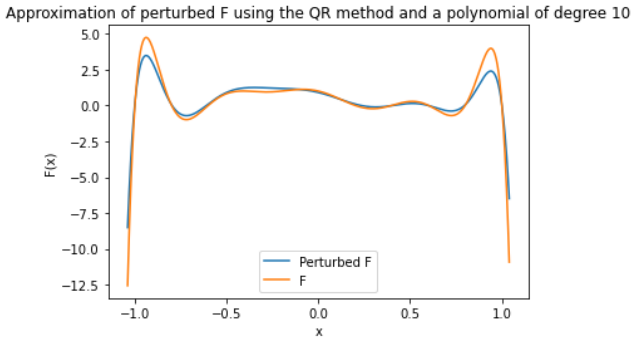

Figure 8: Perturbation of F with polynomial of degree 10

To investigate the sensitivity of the two polynomials when perturbed, the inner product was taken of the difference 10 random points' (in $[-1, 1]$) mappings according to the unperturbed function and perturbed function. The full output can be obtained by running the *q2.py* file.

The output showed that for $m = 7$, the size of the inner product was 0.04 and for $m = 10$, the size of the inner product was 2.5, suggesting very strongly that the polynomial of degree 10 is far more sensitive to the initial conditions than the one of degree 7.

This supports the idea that the polynomial of degree 7 is more preferable.

To investigate the sensitivity of the coefficients of the two polynomials when perturbed, the inner product of the difference between the two polynomial coefficients was taken. For $m = 7$, the result was 5.95 as compared with 248181 for $m = 10$, suggesting strongly that the $m = 10$ polynomial is much more sensitive to perturbations than the $m = 7$ polynomial.

3 Question Three

In this question, all QR factorisations that are used will be in reduced form since the full QR factorisation is superfluous to requirement. This means that in this section we will omit the hat notation and simply assume that any Q and R is reduced.

The columns of V represent a vector x raised to the power of $j-1$ where j is the index number of the column of V and $x_i = (i + 1/2)/M$.

For the first part of this question, we will consider $M = 100$ to see what the contents of Q and R appears to be using the householder algorithm. The choice of $M = 100$ is to ensure the plots of the columns represent some form of continuity.

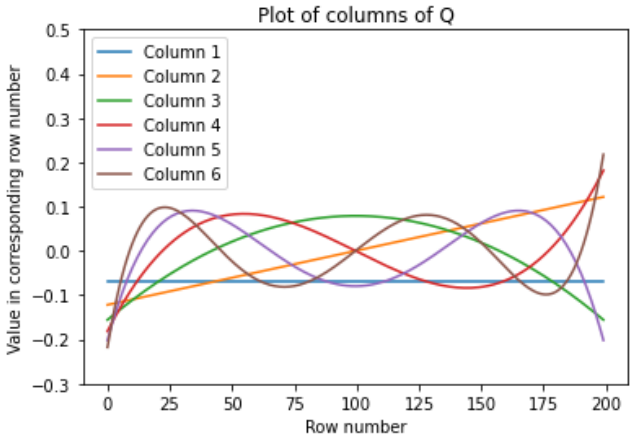

Figure 9: Plot of columns of Q

From this plot it would appear that the data in column i of Q yield data from a polynomial of degree $i-1$. Considering the QR factorisation, this makes sense since the ith column of V is a linear combination of the first i columns of Q. We know that the degree of the ith column of V is $i-1$ (by design) which means that the ith column of Q must also represent points from a polynomial of degree $i-1$.

For the next section we will consider $p = M = 100$ and see how the last 5 columns of Q appear when plotted. We will make use of two more algorithms, Classical Gram Schmidt and Modified Gram Schmidt:

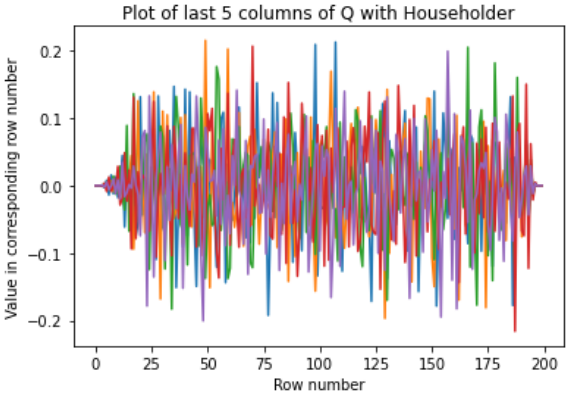

Figure 10: Plot of last columns of Q using Householder

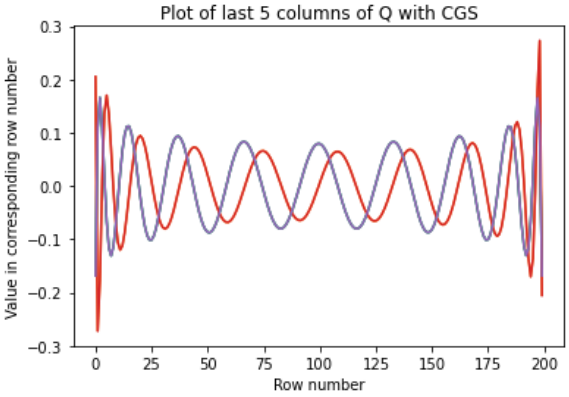

Figure 11: Plot of last columns of Q using CGS

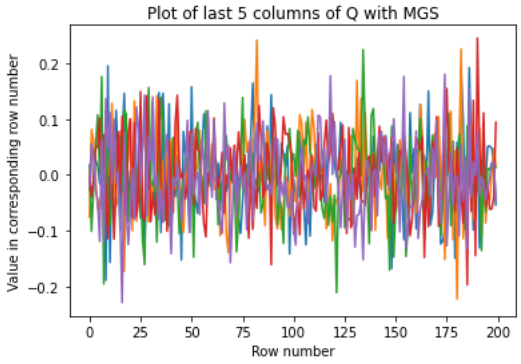

Figure 12: Plot of last columns of Q using MGS

From the above three figures, it can be seen that the Householder and MGS algorithms produce 5 linearly independent columns whereas the CGS only produces two linearly independent columns for the last 5 columns of Q. This is not a surprising result given that CGS is the most numerically unstable algorithm out of the three and the extent of the loss of orthogonality in the CGS algorithm has meant that the Q matrix is no longer of full rank.

To measure the loss of orthogonality of the last 5 columns of Q we make use of the infinity norm which is defined as follows:

$$\|A\|_\infty = \max_{1 \leq i \leq m} \sum_{j=1}^{n} |a_{ij}|$$

To measure the loss of orthogonality, we consider the infinity norm of the difference between $Q^T_{[:,-5:]}Q_{[:,-5:]}$ and I_5, since if the last 5 columns of Q were orthogonal, then $Q^T_{[:,-5:]}Q_{[:,-5:]} = I_5$ would be satisfied.

According to python's *norm* function, the orthogonality errors for each method are as follows:

$$Householder : 1.23 * 10^{-15}$$
$$CGS : 2.00$$
$$MGS : 3.67 * 10^{-14}$$

As can be seen from these numbers, the MGS and Householder algorithm show little to no loss of orthogonality which explains the almost identical plots that are produced; the Householder loses marginally less orthogonality than MGS. Given what we know about the three algorithms, this order makes sense since MGS is supposed to reduce the numerical instability of the CGS algorithm, and this can be seen by the significantly lower orthogonality error that has been produced.

4 Question Four

Let $P_1(x)$ be the polynomial of degree p that forms the relationship between depth, x, and pressure, $P(x)$, at depths less than 1 and $P_2(x)$ be the polynomial of degree p for depths greater than or equal to 1.

We write P_1 and P_2 as follows:

$$P_1(x) = \sum_{i=0}^{p} a_i x^i,$$
$$P_2(x) = \sum_{i=0}^{p} b_i x^i$$

The file in question has dimensions 100 x 2 with the first column pertaining to the depth and the second, the corresponding pressure.

We will denote A_i as being the ith entry of the *pressure.dat* file in the 1st column (with indexing here being 1-indexed) and $P(A_i)$ as being the ith entry of the *pressure.dat* file in the 2nd column of the *pressure.dat* file.

Below is the system of equations we wish to solve, with the constraints that $P_1(1) - P_2(1) = 0$ and $P_1'(1) - P_2'(1) = -5$:

$$\begin{pmatrix} 1 & A_1 & \dots & A_1^p & 0 & 0 & \dots & 0 \\ 1 & A_2 & \dots & A_2^p & 0 & 0 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & A_{50} & \dots & A_{50}^p & 0 & 0 & \dots & 0 \\ 0 & 0 & \dots & 0 & 1 & A_{51} & \dots & A_{51}^p \\ 0 & 0 & \dots & 0 & 1 & A_{52} & \dots & A_{52}^p \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 0 & 1 & A_{100} & \dots & A_{100}^p \end{pmatrix} \begin{pmatrix} a_0 \\ \vdots \\ a_p \\ b_0 \\ \vdots \\ b_p \end{pmatrix} = \begin{pmatrix} P(A_1) \\ \vdots \\ P(A_{50}) \\ P(A_{51}) \\ \vdots \\ P(A_{100}) \end{pmatrix} = Ax = b$$

We wish to minimise $||Ax - b||^2$ subject to:

$$\begin{pmatrix} 1 & 1 & \dots & 1 & -1 & -1 & \dots & -1 \\ 0 & 1 & \dots & p & 0 & -1 & \dots & -p \end{pmatrix} \begin{pmatrix} a_0 \\ \vdots \\ a_p \\ b_0 \\ \vdots \\ b_p \end{pmatrix} = \begin{pmatrix} 0 \\ -5 \end{pmatrix} = Bx = d$$

We now seek to rewrite this problem in such a way that it is easier to solve. This is achieved by finding a change of variables:

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

where Q is an orthogonal (and hence square) matrix, $y_1 \in \mathbb{R}^2$ and $y_2 \in \mathbb{R}^{2(p+1)-2}$. We want Q to be such that the constraints only apply to y_1 and we minimise with respect to y_2 (i.e. $Cy_1 = d$):

$$\min_{\{y_2 \in \mathbb{R}^{2(p+1)-2}\}} ||A_1 y_1 + A_2 y_2 - b||^2, \; Cy_1 = d$$

where $A_1 = (AQ)_{[:,1:2]}$ and $A_2 = (AQ)_{[:,3:2(p+1)]}$.

Applying the transformation to the constraint $Bx = d$ gives us:

$$(BQ)_{[:,1:2]} y_1 + (BQ)_{[:,3:2(p+1)]} y_2 = d$$

If we want this to only be dependent on y_1, then we require:

$$(BQ)_{[:,3:2(p+1)]} y_2 = 0$$

and that Q must have dimensions $2(p+1)$ by $2(p+1)$.

If we consider the QR factorisation of B^T, then the Q has dimensions $2(p+1)$ by $2(p+1)$ and the R has dimensions $2(p+1)$ by 2. i.e. $B^T = QR, \; B = R^T Q^T$ with Q being orthogonal and R being upper triangular.

Substituting this into the above yields:

$$(BQ)_{[:,3:2(p+1)]} y_2 = (R^T Q^T Q)_{[:,3:2(p+1)]} y_2 = (R^T I)_{[:,3:2(p+1)]} y_2 = 0,$$

since all elements of $R^T_{[:,3:2(p+1)]}$ are equal to zero. We have now found a suitable Q that simplifies the problem before us. The matrix C is simply the first two columns of R^T which is of the following form:

$$C = \begin{pmatrix} R_{1,1} & 0 \\ R_{1,2} & R_{2,2} \end{pmatrix},$$

with $R_{i,j}$ being the ij entry of R from the QR factorisation of B^T. Now that we know C, we can work out the unique value of y_1 that satisfies the constraint by using the $householder - solve$ function from $exercises3$.

This now leaves us with y_2 and the following problem (with y_1 now known):

$$\min_{\{y_2 \in \mathbb{R}^{2(p+1)-2}\}} ||A_2 y_2 - (b - A_1 y_1)||^2$$

which is in fact a least squares problem with no constraints and can be solved using QR factorisation as per the method in question two. Once y_2 is obtained using the least squares method, we can concatenate y_1 and y_2 and then left multiply by Q to obtain x, as required. The solution to this least squares problem is in the function called $LSSPolyCoeff$ in the $q4.py$ file.

The number of points we have for each of two polynomials is 50, so we will investigate p values less than 50.

Below is a plot illustrating the inner product of the difference between 10 fixed points' mappings according to the perturbed and unperturbed polynomials. This helps us gauge the sensitivity of the polynomial to perturbations in the initial conditions; the perturbations were the same as in question 2.

What can be clearly seen in this graph, is that the error increases exponentially (since the y-axis is logarithmic) as you increase p. This would suggest that a high p value will be a poor fit owing to its ultra-sensitivity to the initial conditions. This graph would suggest to me that any p value in excess of 25 is a poor fit as the difference/inner product is significant.

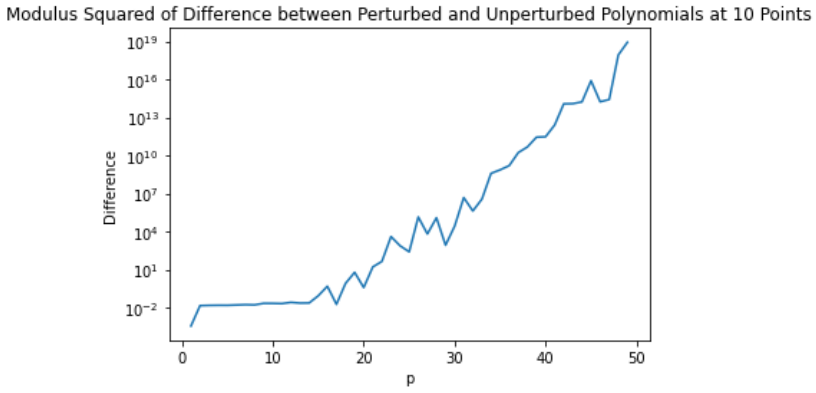

Figure 13: Inner product of the difference of the two polynomials at 10 fixed points

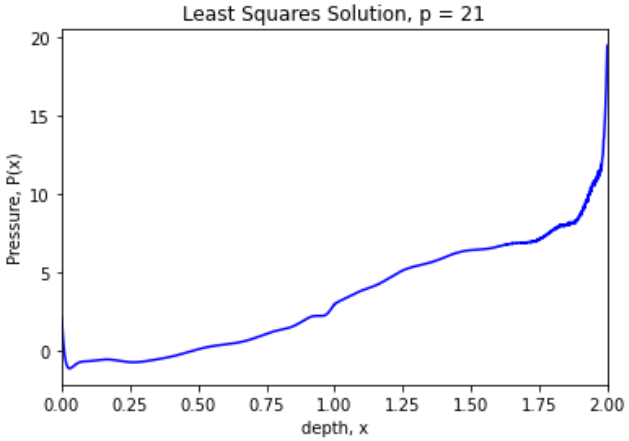

Figure 14: Least squares solution p = 21

As can be seen above, the p value of 21 appears to reflect the problem well, with the kink at $x = 1$ representing the jump in the derivative by 5 units. The below graph illustrates the woeful fit for when $p = 41$, since it begins to climb up at an exponential rate before accounting for the points between 1.8 and 2, which do not increase exponentially.

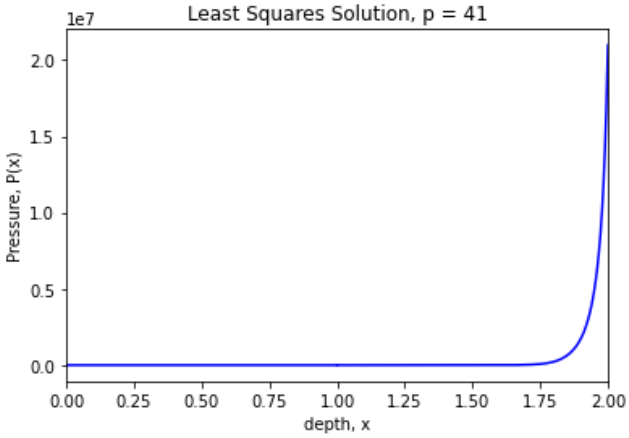

Figure 15: Least squares solution p = 41

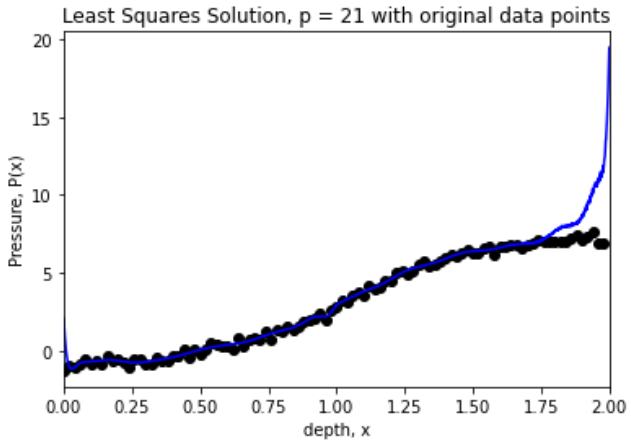

Figure 16: Least squares solution p = 21 with data points

As can be seen by both graphs, while $p = 21$ has its issues towards the depth at 2, its discrepancies are nowhere near as stark as with $p = 41$ which is demonstrably an appalling fit. Having said that, the jump in value of $p = 21$ as you approach 2 would suggest that outside the plotted range, the polynomial fits quite poorly.

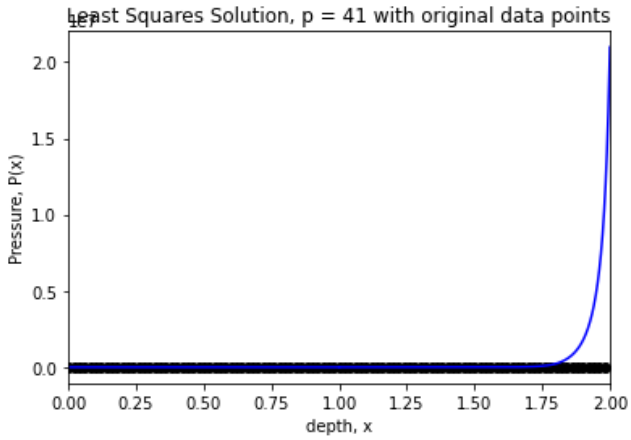

Figure 17: Least squares solution p = 41 with data points

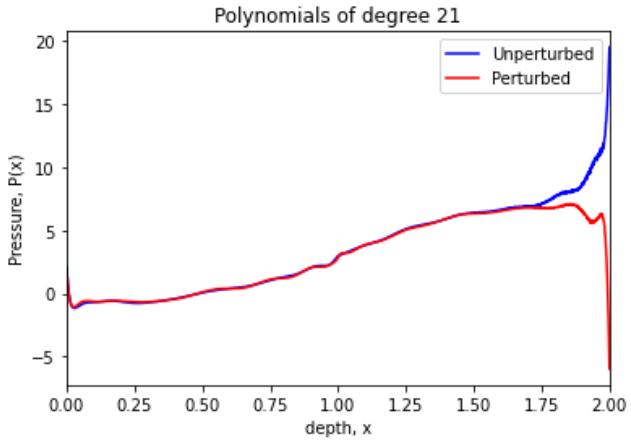

Figure 18: Perturbed and unperturbed polynomials of degree 21

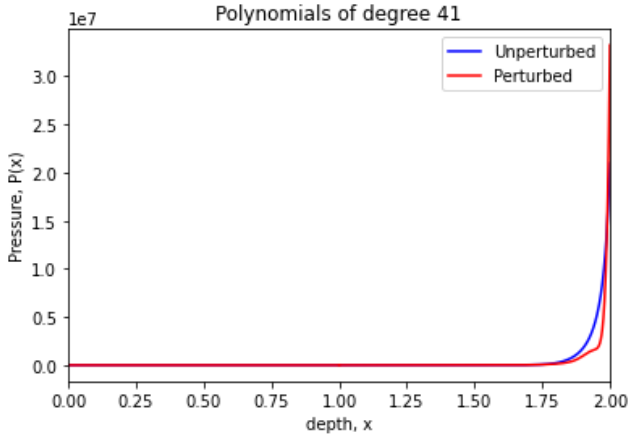

Figure 19: Perturbed and unperturbed polynomials of degree 41

The perturbed and unperturbed polynomials for $p = 21$ appear to agree up until the last points where the perturbed polynomial believes that the pressure decreases once you go beyond a certain depth, which is wrong. We also do not want the boundaries of our data points to behave in an extreme way. This could be mitigated by using a polynomial of lower degree. We know from figure 13 that low p values yield small perturbations which is ideal.

Like in question 2, the data points are all uniformly spaced out. It would be better if the data points with which we are working were spread out in a more random fashion so as to make the polynomial less sensitive.

Let us finally investigate with $p = 3$ and see how it compares with $p = 21$ and $p = 41$.

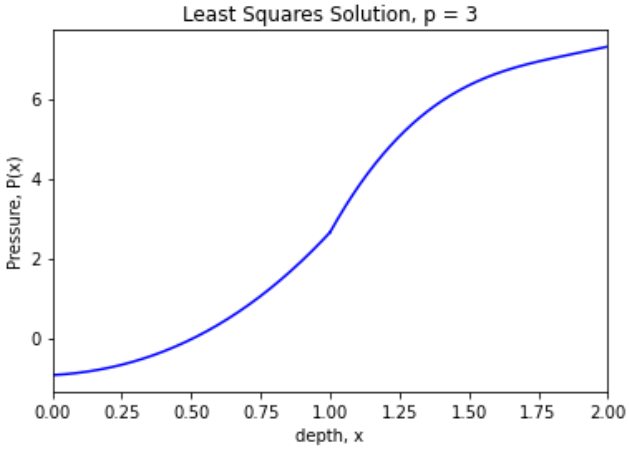

Figure 20: Least squares solution p = 3

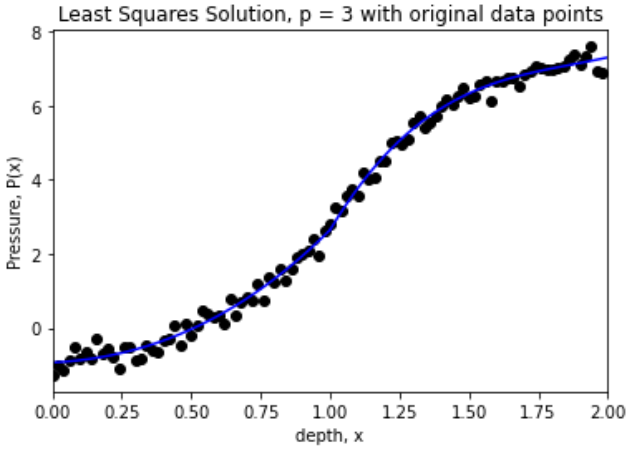

Figure 21: Least squares solution p = 3 with data points

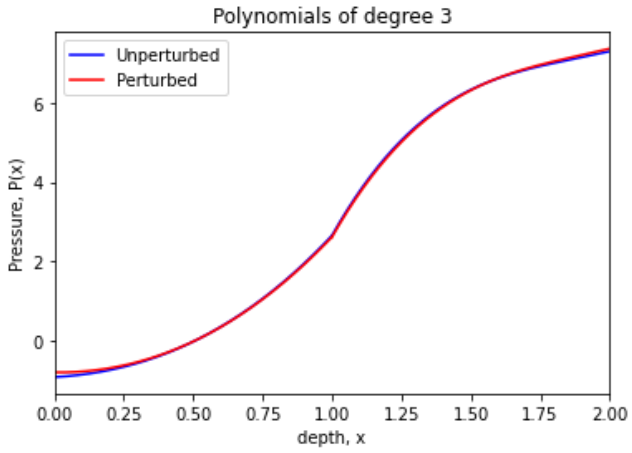

Figure 22: Perturbed and unperturbed polynomials of degree 3

As can be seen, from the above 3 plots, the fit of $p = 3$ is much better than the other p values. The low p value has successfully mitigated the sharp rise in pressure towards a depth of 2, which never agreed with the data and the low p value has also ensured that there is no discernible difference between the perturbed polynomial and the unperturbed polynomial. From these plots, we can now deduce that a p value of 3 is the best value to use.

To verify the above methods for finding x, there are two tests $test2$ and $test4$ which test to see if the constraints are met and that the solution is indeed the minimum.

$test2$ verifies if the constraints are met for $p = 10$ by computing the difference in Bx and d and seeing if they are equal to 0, up to a tolerance. $test4$ compares x with other vectors that satisfy the constraints and sees if x produces the smallest inner product of $Ax - b$ out of all other vectors which agree with the constraints. Here, we use $p = 1$, as it is easier to find vectors that satisfy the constraints.

Both of these tests can be found in the $testq4.py$ file.

For the final part of this question, we approximate C as a piecewise polynomial, $\hat{C}$, with M components and degree p. Each component pertains to a θ in the interval $2\pi m/M \leq \theta \leq 2\pi(m+1)/M$ for $m = 0, 1, ..., M-1$.

We also require that $\hat{C}$ is continuous and that its derivative is continuous at the boundaries of the individual polynomials in the piecewise polynomial; these from the constraints of our problem. Minimising the sum of the squares of the deviation from the data (which is the problem we face) is equivalent to minimising the square of the residual, $Ax - b$.

We therefore seek to write this problem in the following form:

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

Here, x represents the coefficients of the polynomials of $\hat{C}$. Since C returns a vector with 2 entries, we can consider a polynomial approximation as being of the following form:

$$\hat{C}_m(\theta) = \sum_{i=0}^{p} \theta^i \begin{pmatrix} a_{i,m} \\ b_{i,m} \end{pmatrix}$$

where $a_{i,m}$ and $b_{i,m}$ represent the coefficients of the ith degree term in the mth 'piece' of the polynomial for the first and second output respectively.

The constraints are as follows:

$$\hat{C}_0(0) = \hat{C}_{M-1}(2\pi),\ \hat{C'}_0(0) = \hat{C'}_{M-1}(2\pi),$$
$$\hat{C}_m(2\pi(m+1)/M) = \hat{C}_{m+1}(2\pi(m+1)/M),\ \hat{C'}_m(2\pi(m+1)/M) = \hat{C'}_{m+1}(2\pi(m+1)/M),$$
$$m = 0, 1, ..., M-2$$

Given that we have N data points (for θ_1 to θ_N), we can formulate this problem in the following way, using the above constraints, in matrix form:

$$\begin{pmatrix} 1 & 0 & \theta_1 & 0 & \dots & \theta_1^p & 0 & 0 & \dots & 0 \\ 0 & 1 & 0 & \theta_1 & \dots & 0 & \theta_1^p & 0 & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \dots & 0 & 0 & \dots & \theta_N^p & 0 \\ 0 & 0 & 0 & 0 & \dots & 0 & 0 & \dots & 0 & \theta_N^p \end{pmatrix} \begin{pmatrix} a_{0,0} \\ b_{0,0} \\ \vdots \\ a_{p,0} \\ b_{p,0} \\ \vdots \\ a_{0,M-1} \\ b_{0,M-1} \\ \vdots \\ a_{p,M-1} \\ b_{p,M-1} \end{pmatrix} = \begin{pmatrix} C(\theta_1) \\ C(\theta_2) \\ \vdots \\ C(\theta_N) \end{pmatrix} = Ax = b$$

where for the purposes of illustration, we have assumed that:

$$0 \leq \theta_1 \leq 2\pi/M,$$
$$2\pi(M-1)/M \leq \theta_N \leq 2\pi$$

The matrix, A, has dimensions $2N$ by $2(p+1)M$ with each set of two rows corresponding to $C(\theta_i)$ and the position in the matrix being chosen in such a way that it only interacts with the coefficients in $\hat{C}_j(\theta_i)$ where j corresponds to the region in which θ_i appears. x has dimensions $2(p+1)M$ by 1, and b has dimensions $2N$ by 1.

The constraints matrix is as follows. Here, we consider the difference between the constraints as being equal to zero. We also choose the position of the pairs of rows in such a way that they only interact with the coefficients in x as in the above algebraic constraints:

$$\begin{pmatrix} 1 & 0 & 0 & 0 & \dots & -1 & 0 & \dots & -(2\pi)^p & 0 \\ 0 & 1 & 0 & 0 & \dots & 0 & -1 & \dots & 0 & -(2\pi)^p \\ 0 & 0 & 1 & 0 & \dots & 0 & 0 & \dots & -p(2\pi)^{p-1} & 0 \\ 0 & 0 & 0 & 1 & \dots & 0 & 0 & \dots & 0 & -p(2\pi)^{p-1} \\ 1 & 0 & \frac{2\pi}{M} & 0 & \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & 1 & 0 & \frac{2\pi}{M} & \dots & \dots & \dots & \dots & \dots & \dots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \dots \end{pmatrix} \begin{pmatrix} a_{0,0} \\ b_{0,0} \\ \vdots \\ a_{p,0} \\ b_{p,0} \\ \vdots \\ a_{0,M-1} \\ b_{0,M-1} \\ \vdots \\ a_{p,M-1} \\ b_{p,M-1} \end{pmatrix} = \begin{pmatrix} 0 \\ \vdots \\ 0 \end{pmatrix} = Bx = d$$

B has dimensions $4M$ by $2M(p+1)$ and d has dimensions $4M$ by 1 with all entries being 0. An example of this in action will be given so as to help see the form of the matrix, B, above.

In a way very similar to the earlier part of the question, we can rewrite this as an unconstrained least squares problem.

The method in this case is to, again, consider the QR factorisation of B transposed and split x:

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

where Q is an orthogonal (and hence square) matrix, $y_1 \in \mathbb{R}^{4M}$ and $y_2 \in \mathbb{R}^{2Mp-2M}$.

Here is an example to illustrate the method and elucidate the construction of the above matrices. We shall work with $p = 2$, $M = 2$ and $N = 8$ and the following function, C:

$$C(\theta) = \begin{pmatrix} sin(\theta) \\ 2cos(\theta) + sin(\theta) \end{pmatrix}.$$

We know that this function is smooth since a linear combination of trigonometric functions is smooth. We will use the following curve points in the set up of our problem:

θ_i	$C(\theta_i)$
0	(0,2)
$\pi/4$	$(\frac{\sqrt{2}}{2}, \frac{3\sqrt{2}}{2})$
$\pi/2$	(1,1)
$3\pi/4$	$(\frac{\sqrt{2}}{2}, \frac{-\sqrt{2}}{2})$
π	(0,-2)
$3\pi/2$	(-1,-1)
$5\pi/3$	$(-\frac{\sqrt{3}}{2}, -\frac{\sqrt{3}}{2}+1)$
$7\pi/4$	$(-\frac{\sqrt{2}}{2}, \frac{\sqrt{2}}{2})$

With our eight data points, we can split the first 4 and the last 4 into two categories for each of the two piecewise polynomials.

The polynomial of degree 1 which we use to approximate C is:

$$\hat{C}(\theta) = \begin{cases} \hat{C}_0(\theta), \ 0 \leq \theta \leq \pi \\ \hat{C}_1(\theta), \ \pi \leq \theta \leq 2\pi \end{cases}$$

and the piecewise polynomials are of the following forms:

$$\hat{C}_0(\theta) = \begin{pmatrix} a_{0,0} + a_{1,0}\theta + a_{2,0}\theta^2 \\ b_{0,0} + b_{1,0}\theta + b_{2,0}\theta^2 \end{pmatrix}$$
$$\hat{C}_1(\theta) = \begin{pmatrix} a_{0,1} + a_{1,1}\theta + a_{2,1}\theta^2 \\ b_{0,1} + b_{1,1}\theta + b_{2,1}\theta^2 \end{pmatrix}.$$

The constraints of this problem are as follows:

$$\hat{C}_0(0) = \hat{C}_1(2\pi), \ \hat{C'}_0(0) = \hat{C'}_1(2\pi)$$
$$\hat{C}_0(\pi) = \hat{C}_1(\pi), \ \hat{C'}_0(\pi) = \hat{C'}_1(\pi).$$

The above information can be wrapped up in the following matrices:

$Ax = b$:

$$\begin{pmatrix}
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & \frac{\pi}{4} & 0 & (\frac{\pi}{4})^2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & \frac{\pi}{4} & 0 & (\frac{\pi}{4})^2 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & \frac{\pi}{2} & 0 & (\frac{\pi}{2})^2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & \frac{\pi}{2} & 0 & (\frac{\pi}{2})^2 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & \frac{3\pi}{4} & 0 & (\frac{3\pi}{4})^2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & \frac{3\pi}{4} & 0 & (\frac{3\pi}{4})^2 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & \pi & 0 & \pi^2 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & \pi & 0 & \pi^2 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & \frac{3\pi}{2} & 0 & (\frac{3\pi}{2})^2 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & \frac{3\pi}{2} & 0 & (\frac{3\pi}{2})^2 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & \frac{5\pi}{3} & 0 & (\frac{5\pi}{3})^2 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & \frac{5\pi}{3} & 0 & (\frac{5\pi}{3})^2 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & \frac{7\pi}{4} & 0 & (\frac{7\pi}{4})^2 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & \frac{7\pi}{4} & 0 & (\frac{7\pi}{4})^2
\end{pmatrix}
\begin{pmatrix} a_{0,0} \\ b_{0,0} \\ a_{1,0} \\ b_{1,0} \\ a_{2,0} \\ b_{2,0} \\ a_{0,1} \\ b_{0,1} \\ a_{1,1} \\ b_{1,1} \\ a_{2,1} \\ b_{2,1} \end{pmatrix}
=
\begin{pmatrix}
0 \\ 2 \\ \frac{\sqrt{2}}{2} \\ \frac{3\sqrt{2}}{2} \\ 1 \\ 1 \\ \frac{\sqrt{2}}{2} \\ \frac{-\sqrt{2}}{2} \\ 0 \\ -2 \\ -1 \\ -1 \\ \frac{-\sqrt{3}}{2} \\ \frac{-\sqrt{3}}{2} + 1 \\ \frac{-\sqrt{2}}{2} \\ \frac{\sqrt{2}}{2}
\end{pmatrix}$$

$Bx = d$:

$$\begin{pmatrix}
1 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & -2\pi & 0 & -(2\pi)^2 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & -2\pi & 0 & -(2\pi)^2 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & -4\pi & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & -4\pi \\
1 & 0 & \pi & 0 & \pi^2 & 0 & -1 & 0 & -\pi & 0 & -\pi^2 & 0 \\
0 & 1 & 0 & \pi & 0 & \pi^2 & 0 & -1 & 0 & -\pi & 0 & -\pi^2 \\
0 & 0 & 1 & 0 & 2\pi & 0 & 0 & 0 & -1 & 0 & -2\pi & 0 \\
0 & 0 & 0 & 1 & 0 & 2\pi & 0 & 0 & 0 & -1 & 0 & -2\pi
\end{pmatrix}
\begin{pmatrix} a_{0,0} \\ b_{0,0} \\ a_{1,0} \\ b_{1,0} \\ a_{2,0} \\ b_{2,0} \\ a_{0,1} \\ b_{0,1} \\ a_{1,1} \\ b_{1,1} \\ a_{2,1} \\ b_{2,1} \end{pmatrix}
=
\begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

To solve this, consider the QR factorisation of B^T and proceed as was done in the beginning of the question.

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

where Q is an orthogonal matrix, $y_1 \in \mathbb{R}^8$ and $y_2 \in \mathbb{R}^4$.

The problem can now be rewritten as:

$$\min_{\{y_2 \in \mathbb{R}^4\}} ||A_1 y_1 + A_2 y_2 - b||^2, \; Cy_1 = d, \; C = R^T_{[:,1:8]}$$

where $A_1 = (AQ)_{[:,1:8]}$ and $A_2 = (AQ)_{[:,9:12]}$.

Now that we know C, we can work out the unique value of y_1 that satisfies the constraint by using the $householder - solve$ function from $exercises3$.

This now leaves us with y_2 and the following problem (with y_1 now known):

$$\min_{\{y_2 \in \mathbb{R}^4\}} ||A_2 y_2 - (b - A_1 y_1)||^2$$

which is in fact a least squares problem with no constraints and can be solved using QR factorisation as per the method in question two. Once y_2 is obtained using the least squares method, we can concatenate y_1 and y_2 and then left multiply by Q to obtain x, as required. The code of the solution to this least squares problem is in the $q4.py$ file.

The coefficients x of the polynomials of degree 2 in the order as they appear on the previous page are:

[2.91417919e-01, -1.99912115e-02, 6.97160477e-01, 9.17040549e-03, -2.21913072e-01, -2.91903073e-03, 4.67180638e, 3.76281455e-02, -2.09148143, -2.75112165e-02, 2.21913072e-01, 2.91903073e-03].

Below are plots of the polynomials for the two components of $\hat{C}(\theta)$ and the resulting curve, $\hat{C}$ alongside C:

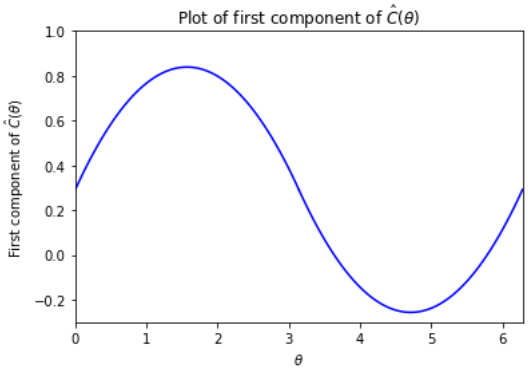

Figure 23: Polynomial of first component of $\hat{C}(\theta)$

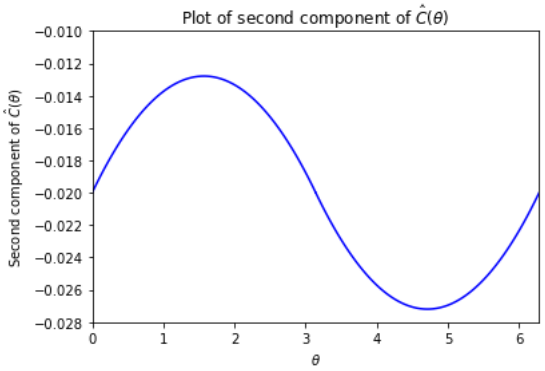

Figure 24: Polynomial of second component of $\hat{C}(\theta)$

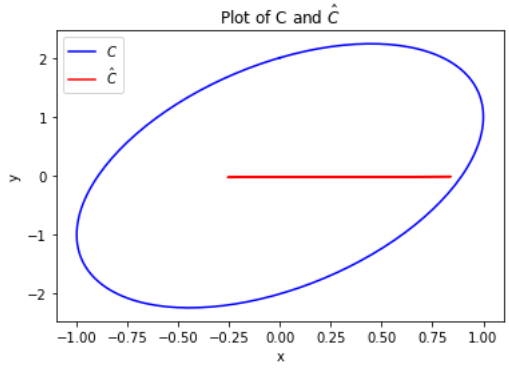

Figure 25: $\hat{C}(\theta)$ and $C(\theta)$

From the plots above, it appears that both components of $\hat{C}(\theta)$ do respect the boundary conditions, with no kinks manifesting themselves at $\theta = \pi$ and $\hat{C}(2\pi) = \hat{C}(0)$ appearing to be obeyed for both components.

It can also be seen that the polynomial in the first component appears to be a scaled version of the polynomial in the second component, which explains the straight red line. The test for this solution is *test*4 and it can be run in the *testq*4.*py* file. The test requires that the constraints and boundary conditions of the problem are satisfied.

What the plot of both curves tells us is that to form a good approximation, we require a polynomial of degree greater than 2 which is what was used in the plot. Subsequent investigations could therefore be carried out for higher p values.