Computational Linear Algebra (2022-2023)

Coursework 3

(01495449)

Imperial College London

January 12, 2023

1 Answer to question 1a

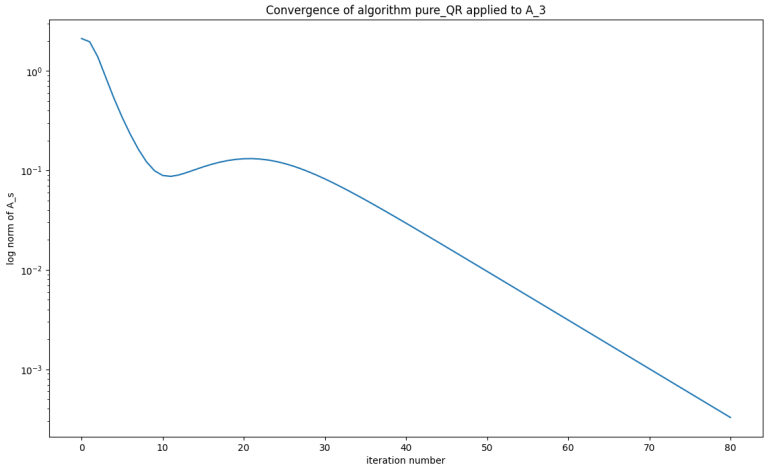

Figure 1: Confirms that the norm of the below diagonal entries of A_k produced by pure_QR tends to 0.

In function q1_a_latter_part I verify that the diagonal entries of A_K are the eigenvalues of A. This is done by performing inverse iteration using a shift for each eigenvalue to recover the corresponding eigenvector.

2 Answer to question 1b

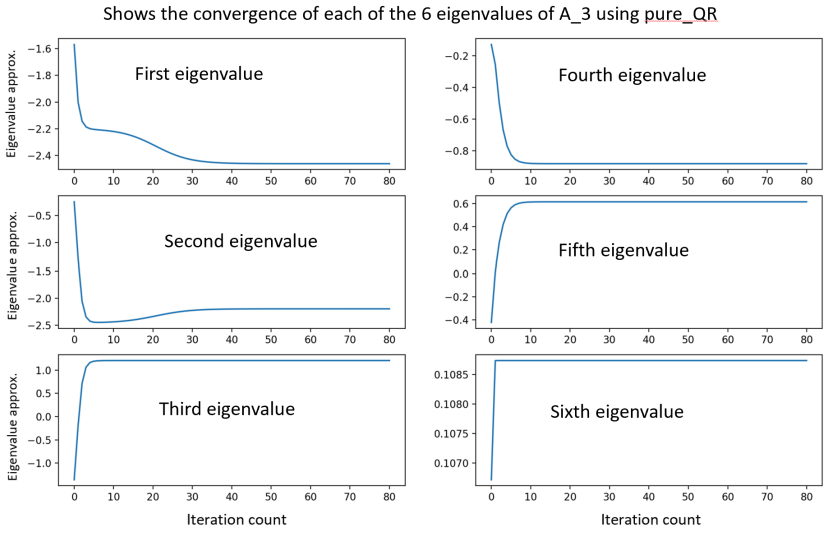

Figure 2: As you can see the eigenvalues appear in the diagonal of A_k from pure_QR function in magnitude going from largest to smallest. The rate of convergence is faster, the smaller the magnitude of the eigenvalue.

3 Answer to question 1c

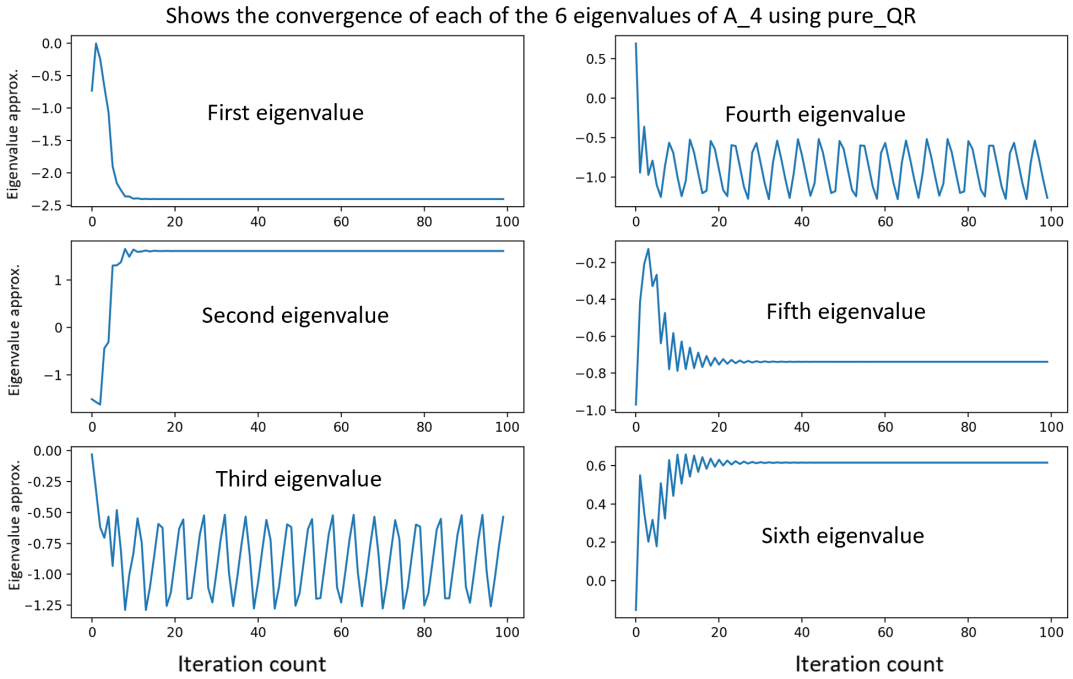

Figure 3: As you can see for matrix A_4, the algorithm pure_QR doesn't converge as two of the diagonal entries of A_k don't converge. These correspond to the two complex eigenvalues of A_4.

pure_QR algorithm will never converge to the Schur decomposition of A_4 because A_4 has complex eigenvalues hence the algorithm converges to a block upper diagonal matrix (with 1 by 1 or 2 by 2 block) rather than an upper triangular matrix. Since the third and fourth diagonal entry of A_k doesn't converge in this case, the third and fourth column contains a 2 by 2 block diagonal, whose eigenvalues are the same as the complex eigenvalues of A_4. This has been confirmed by printing A_k.

4 Answer to question 1d

The test function is called test_pure_QR. This confirms that pure_QR converges for A_4 by checking the eigenvalues from the block diagonals of A_k. I defined my own function for calculating eigenvalues of a 2 by 2 matrix, labelled eigenvalues_calculator.
Since we are given in the question that the pure QR will converge to the form:

$$A' = Q^T A Q = \begin{pmatrix} R_{11} & R_{12} & \dots & R_{1r} \\ 0 & R_{22} & \dots & R_{2r} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & R_{rr} \end{pmatrix} \tag{1}$$

Where the block diagonals are 1 by 1 or 2 by 2. Also:

$$\det(A') = \prod_{i=1}^{r} \det(R_{ii}) \tag{2}$$

From this formula, it is clear that the eigenvalues of A' are the solutions to the characteristic polynomials of the block diagonals of A' as $A' - \lambda I$ preserves block diagonal form.

5 Answer to question 1e

Function q1_e compiles the eigenvalues of A' from pure_QR using a helper function called eigenvalues_calculator. The test for this function is called test_q1_e, which demonstrates that function q1_e works on a matrix of size 6 by 6, 14 by 14 and 21 by 21.

6 Answer to question 1f

We are given (i)$Q_{k-1}R_{k-1} = A_{k-1} - \mu I$ and (ii)$A_k = R_{k-1}Q_{k-1} + \mu I$ for all k.

$$\begin{aligned} A_k &= R_{k-1}Q_{k-1} + \mu I \\ &= Q_{k-1}^*(A_{k-1} - \mu I)Q_{k-1} + \mu I \quad \text{(substituting for } R_{k-1} \text{ after rearranging (i))} \\ &= Q_{k-1}^* A_{k-1} Q_{k-1} - \mu Q_{k-1}^* Q_{k-1} + \mu I \\ &= Q_{k-1}^* A_{k-1} Q_{k-1} - \mu I + \mu I \\ &= Q_{k-1}^* A_{k-1} Q_{k-1} \end{aligned} \tag{3}$$

Since Q is an n by n invertible matrix, we can conclude that A_k is similar to A_{k-1}. Since similar matrices have the same eigenvalues, this procedure preserves the eigenvalues of A.

7 Answer to question 1g

With symmetric matrices, if everything is converging correctly then the last entry of the column ends up converging to the eigenvalue with the smallest magnitude. Then the other values in the last row converge to 0 and by symmetry, the same happens to the other values in the last column. Once this had happened we can drop the last column and row and repeat the algorithm on the matrix left. Hence to check if A_{jj}^k has sufficiently converged to the eigenvalue we can check if the value above, $A_{j-1,j}^k$ has sufficiently converged to 0.

8 Answer to question 1h

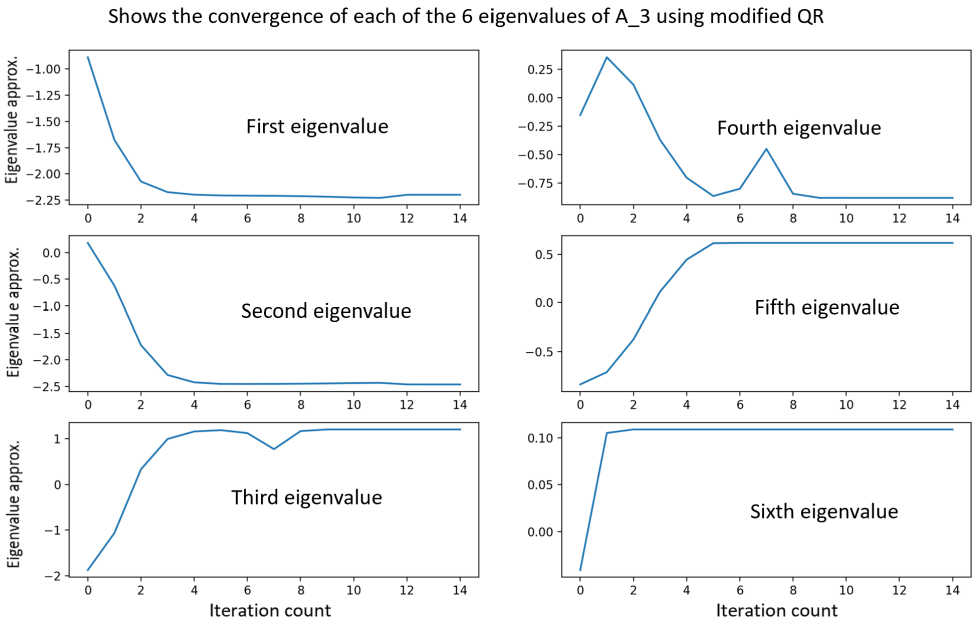

Figure 4: Here we use the modified QR described in question 1f and implement the stopping criteria in question 1g. You can see that the eigenvalues converge much faster than they did for pure_QR. Furthermore, the diagonal entries in the A_k approximation using the modified algorithm don't converge in size order unlike in pure_QR.

The modified QR algorithm has the function q1_h, the function for investigation on $A3$ is called investigate_q1_h_on_A3() and the test function is called test_q1_h.

9 Answer to question 1i

From running function q1_i, we get:
pure QR algorithm took 48.505879640579224 seconds to converge for 100 by 100 matrix
Modified QR algorithm took 3.527719736099243 seconds to converge for 100 by 100 matrix

As you can see the shifting technique used in the modified QR algorithm accelerates the convergence compare to the pure QR algorithm.

10 Answer to question 1j

In the deflation strategy, when we think that the last eigenvalue has converged (the last diagonal value in the matrix) sufficiently, we delete the last column and row of the matrix. We append the converged eigenvalue to a specific list of eigenvalues. Note: we know the last eigenvalue has converged when the values above it and the values to its left in the matrix tend sufficiently close to 0. Now we have a matrix that has had its dimensions reduced. Repeat this procedure on the sub-matrix and at the end, we will end up with a list containing

all the eigenvalues.

This algorithm is faster as we are eliminating the last column and row of the matrix as the procedure takes place. This means that the algorithm computes on smaller matrices than the original matrix for most of the steps, hence leading to fewer elements needing to be processed. This results in it being more computationally efficient.

11 Answer to question 1k

In function q1_k, when I apply the modified QR on $A4$, I got a divide by 0 error in the householder function. This is because $A4$ has two complex eigenvalues, hence the algorithm transforms to a block diagonal matrix (containing blocks of 1 by 1 and 2 by 2). This means that when we check that a diagonal entry has converged by checking that the value above it is sufficiently close to 0. This value above for diagonal in blocks 2 by 2 will never converge to zero, hence the algorithm will fail as we can't get a diagonal matrix.

An alternative to fix this issue is the double-shift QR strategy. If the eigenvalue of the lower diagonal block is complex, we define them as μ_1 and μ_2. Then, these two eigenvalues are used as shifts in consecutive iterations to achieve quadratic convergence. The algorithm is:

$$\begin{aligned} A - \mu_1 I &= Q_1 R_1 \\ A_1 &= R_1 Q_1 + \mu_1 I \\ A_1 - \mu_2 I &= Q_2 R_2 \\ A_2 &= R_2 Q_2 + \mu_2 I \end{aligned} \tag{4}$$

Ordered (first to last) errors for the linear system as the GMRES algorithm runs:

12 Answer to question 2a

The test function test_q2_a forces a false assertion so that the errors can be printed.

For the 20 by 20 matrix	For the 18 by 18 matrix
Error: 4.483474670096159	
Error: 4.418945173950468	Error: 2.3213666758242373
Error: 3.6977284634112517	Error: 2.354219855308156
Error: 3.967140850692315	Error: 3.397528304860553
Error: 3.680922311106461	Error: 3.173375510549389
Error: 3.4430062908238313	Error: 3.171575640477967
Error: 3.8143975462218482	Error: 3.1768990979908
Error: 3.934205687373503	Error: 1.7198983536190173
Error: 3.7590387548572	Error: 1.8146992147659358
Error: 3.264271888224137	Error: 1.5269176499776904
Error: 3.280303482877511	Error: 1.3760388971522617
Error: 3.282819468054206	Error: 1.2025742147103415
Error: 2.9043260880973696	Error: 1.5436651595377067
Error: 1.7668273920134738	Error: 1.6699567687014043
Error: 1.328093216722785	Error: 1.717377729332249
Error: 1.327491139979184	Error: 1.7329763038207362
Error: 2.8814862750500807	Error: 1.8118111016736802
Error: 1.595013182402824	Error: 1.127405153036543
Error: 6.47675499233173	Error: 2.3858347143553285e-14
Error: 4.900329221079513e-12	

Since the errors go to 0, this confirms that the GMRES algorithm converges for these matrices and also that the early stopping works once the error is sufficiently small.

13 Answer to question 2b

We are dealing with a m by m matrix of the form:

$$a_{ij} = \begin{cases} -2 & \text{if } i = j \\ 1 & \text{if } i = j-1 \text{ or } i = j+1 \\ 0 & \text{otherwise.} \end{cases} \tag{5}$$

Assuming that the eigenvectors u of our matrix A are all of the form

$$u_k^{(l)} = \sin(kl\pi/(m+1)), \quad k = 1, \ldots, m \tag{6}$$

with one eigenvector for each value of l = 1,..., m.

If we consider $Au^{(l)} = \lambda u^{(l)}$ and rearrange it to $Au^{(l)} - \lambda u^{(l)} = 0$. Then the j^{th} entry of this equation has the form:

$$\sin(\frac{(j-1)l\pi}{(m+1)}) - (2+\lambda^{(l)})\sin(\frac{jl\pi}{(m+1)}) + \sin(\frac{(j+1)l\pi}{(m+1)}) = 0 \tag{7}$$

Using the formula $\sin\alpha + \sin\beta = 2\sin\left(\frac{\alpha+\beta}{2}\right)\cos\left(\frac{\alpha-\beta}{2}\right)$, the above equation becomes:

$$2\sin(\frac{jl\pi}{(m+1)})\cos(\frac{l\pi}{(m+1)}) - (2+\lambda^{(l)})\sin(\frac{jl\pi}{(m+1)}) = 0 \tag{8}$$

Cancelling the sines and rearranging leads to:

$$\lambda^{(l)} = -2 + 2\cos(\frac{l\pi}{(m+1)}) = -4sin^2(\frac{\pi l}{2(m+1)}) \tag{9}$$

14 Answer to question 2c

The function labelled q2_c implements modified QR algorithm of matrices of various dimensions. It then prints the magnitude difference (error) between the eigenvalues from modified QR and eigenvalues produced by the formula in Answer to q2c. The errors are of the order of machine precision and hence can be classified as zero. This confirms that the eigenvalue formula is correct.
For the 6 by 6 matrix, the error output is 6.748715559041373e-15

For the 14 by 14 matrix, the error output is 4.7208424874789494e-12

For the 25 by 25 matrix, the error output is 2.1633500628132468e-13

For the 50 by 50 matrix, the error output is 4.111725440304322e-12

15 Answer to question 2d

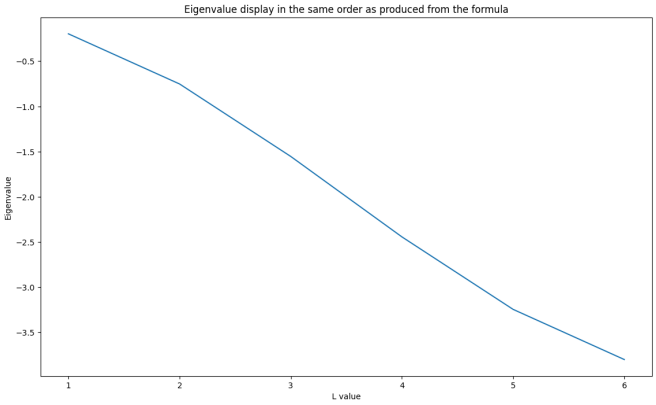

Figure 5: shows the eigenvalue distribution for a 6 by 6 matrix.

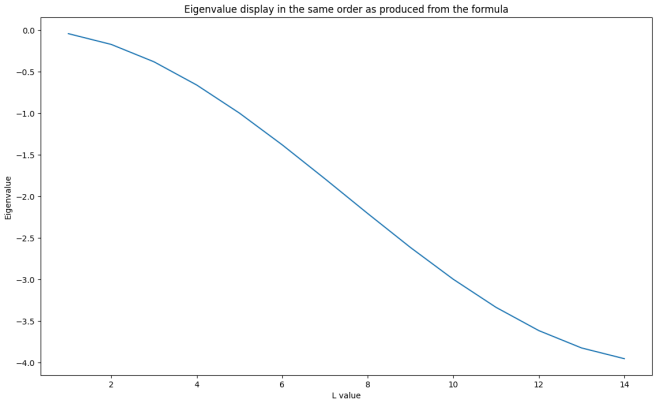

Figure 6: shows the eigenvalue distribution for a 14 by 14 matrix.

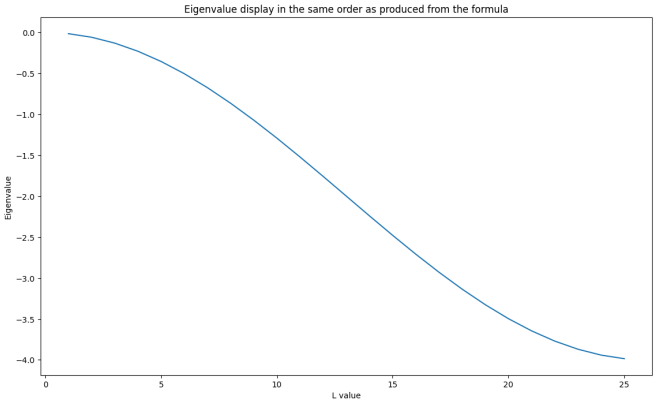

Figure 7: shows the eigenvalue distribution for a 25 by 25 matrix.

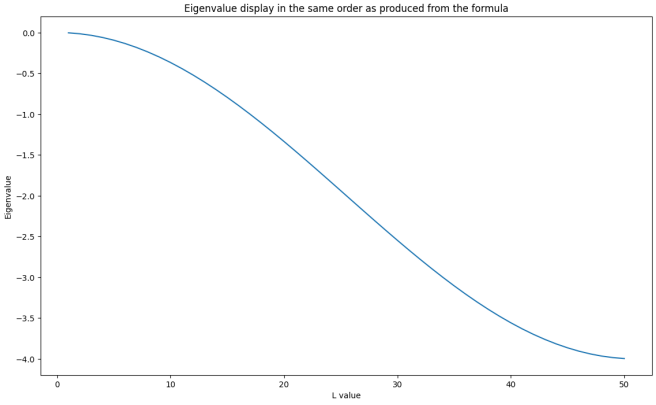

Figure 8: shows the eigenvalue distribution for a 50 by 50 matrix.

The graphs take the shape of $-4sin^2(\frac{\pi l}{2(m+1)})$, which is a sin-squared graph reflected about the x-axis, stretched by a factor of 4 and then stretched by horizontally. As the dimensions increase the distribution converges to $-4sin^2(\frac{\pi l}{2(m+1)})$

We know from lecture notes that the residual in the GMRES algorithm is given by

$$r_n = p(A)b \tag{10}$$

Where $p(z) = 1 - zp'(z)$. Thus the residual is a matrix polynomial p of A applied to b, where p has degree less than n and p(0)=1.

Now GMRES iteration n becomes a polynomial optimisation problem: find p_n such that $p_n(A)b$ is minimised. We get:

$$\frac{\|r_n\|}{\|b\|} \leq \|p_n(A)\| \tag{11}$$

Assuming A is diagonalisable.

$$\|p_n(A)\| \leq k(v)\|p\|_{\Lambda(A)} \tag{12}$$

Where $\Lambda(A)$ is teh set of eigenvalues of A.

$$\|p\|_{\Lambda(A)} = \sup_{x \in \Lambda(A)} \|p(x)\| \tag{13}$$

Hence GMRES will converge faster if V is well-conditioned, and p(x) is small for all $x \in \Lambda(A)$.

$P(0) = 1$, if A has eigenvalues has all eigenvalues clustered in a small number of groups, away from 0. This needs to happen to make the residual estimate small for large m, as we can then find a low-degree polynomial that passes through 1 at x=0, and 0 near each of the clusters.

As m increases, we need a higher degree of polynomial p to approximate the shape and hence more iterations of the GMRES algorithm are required to reduce the residual. This makes the convergence of GMRES slower for larger m.

16 Answer to question 2e

The conditioned number of a matrix has the formula:

$$k(A) = ||A||_2||A^{-1}||_2 \tag{14}$$

where $||.||_2$ is the spectral norm. Using the fact that the spectral norm is equal to the maximum singular value:

$$||A||_2 = \sigma_{max}(A) \tag{15}$$

The singular value of A^{-1} equals one divided by the minimum singular value of A, hence,

$$k(A) = \frac{\sigma_{max}(A)}{\sigma_{min}(A)} \tag{16}$$

Since our A is a tridiagonal symmetric matrix, it is normal. We can decompose $A = Q\Lambda Q^T$, where Q is orthogonal and Λ is diagonal. Hence we can use, $\sigma_i(A) = \sqrt{\lambda_i(A^T A)}$. Using this

$$\sigma_{max}(A) = \sqrt{max(A^T A)} = \sqrt{\lambda_{max}((Q\Lambda Q^T)^T Q\Lambda Q^T)} = \sqrt{\lambda_{max}(Q\Lambda^2 Q^T)} = \sqrt{\lambda_{max}(A)^2} = |\lambda_{max}(A)| \tag{17}$$

Which gives,

$$k(A) = \frac{|\lambda_{max}(A)|}{|\lambda_{min}(A)|} \tag{18}$$

Note: here the numerator is the eigenvalue with the maximum absolute value.
Hence,

$$k(A) = \frac{|-2 + 2\cos(\frac{\pi m}{m+1})|}{|-2 + 2\cos(\frac{\pi}{m+1})|} = \frac{1 - \cos(\frac{\pi m}{m+1})}{1 - \cos(\frac{\pi}{m+1})} \tag{19}$$

Since $cos(x) = -cos(\pi - x)$, we get

$$k(A) = \frac{1 + \cos(\frac{\pi}{m+1})}{1 - \cos(\frac{\pi}{m+1})} \tag{20}$$

We know the Taylor series of cos(x) is $1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \frac{x^6}{6!} + ...$
Lets set $x = \frac{\pi}{m+1}$. Now,

$$k(A) = \frac{1 + \cos(\frac{\pi}{m+1})}{1 - \cos(\frac{\pi}{m+1})} = \frac{2 - \frac{x^2}{2!} + \frac{x^4}{4!} - \frac{x^6}{6!} + ...}{\frac{x^2}{2!} - \frac{x^4}{4!} + \frac{x^6}{6!} - ...} = \frac{2}{\frac{x^2}{2!} - \frac{x^4}{4!} + \frac{x^6}{6!} - ...} - 1 = \tag{21}$$

This is the power series expansion in $\frac{1}{m}$.

This tells us that the condition number explodes as m increases. Hence the GMRES algorithm slows rapidly as we increase m and eventually won't compute in time for a large enough m.
In function q2_e, I did GMRES on a randomly 100 by 100 computed matrix and the algorithm reached the 100th iteration, the errors in the callback started around 6.52 and become around 1 after 25 iterations, it then took an additional 50 iterations for the error to decrease at 0.1. The algorithm seems to decrease error less with each iteration until the very last iteration where it computes the entire system with error-to-machine precision as it has now found the orthogonal basis so can find the exact solution to the least squares problem.
Since GMRES is an accelerating algorithm for errors, GMRES isn't working well because the matrix input isn't well conditioned.

17 Answer to question 2f

The diagonal part of A is a cheap preconditioner as computing the inverse of a diagonal matrix isn't computationally expensive. Since our matrix is heavy on the diagonal, applying this preconditioner will remove this effect leading to faster convergence of GMRES and smaller errors.

18 Answer to question 3a

Below I describe the relationship between the SVD decomposition of A and the eigenvalue decomposition of A^*A. Suppose A has the SVD form $A = UDV^*$. Here the diagonal entries of D are equal to the singular values of A. Then the following two equations clearly hold:

$$A^*A = VD^*U^*UDV^* = V(D^*D)V^* \tag{22}$$

$$AA^* = UDV^*VD^*U^* = U(DD^*)U^* \tag{23}$$

Here the right-hand sides of the equation are the eigenvalue decomposition of the left-hand side. We can now conclude that:
1) The columns of V are eigenvectors of A^*A
2) The columns of U are eigenvectors of AA^*
3) The non-zero elements of D are square roots of the non-zero eigenvalues of A^*A

19 Answer to question 3b

We know $A = UDV^*$, where U (m × m) and V (m × m) are unitary matrices and D is diagonal with non-negative entries.

$$\begin{aligned} H\begin{pmatrix} V & V \\ U & -U \end{pmatrix} &= \begin{pmatrix} 0 & A^* \\ A & 0 \end{pmatrix}\begin{pmatrix} V & V \\ U & -U \end{pmatrix} = \begin{pmatrix} A^*U & -A^*U \\ AV & AV \end{pmatrix} \\ &= \begin{pmatrix} VD & -VD \\ UD & UD \end{pmatrix} = \begin{pmatrix} V & V \\ U & -U \end{pmatrix}\begin{pmatrix} D & 0 \\ 0 & -D \end{pmatrix} \end{aligned} \tag{24}$$

Using rearrangement of $A^* = VDU^*$ to get from the first line of the equation to the second line. We know unitary matrices multiplied by their conjugate is identity.
If we work out the eigenvalue decomposition of H such that, where

$$H = \begin{pmatrix} V & V \\ U & -U \end{pmatrix}\begin{pmatrix} D & 0 \\ 0 & -D \end{pmatrix}\begin{pmatrix} V & V \\ U & -U \end{pmatrix}^{-1} \tag{25}$$

The above should take the form $H = Q\Lambda Q^*$, where Q is unitary and Λ is a diagonal

eigenvalue matrix. We can see that setting the above to this notation results in

$$QQ^* = \begin{pmatrix} V & V \\ U & -U \end{pmatrix} \begin{pmatrix} V & V \\ U & -U \end{pmatrix}^* = \begin{pmatrix} 2I & 0 \\ 0 & 2I \end{pmatrix} \tag{26}$$

Hence we can extract the D, U and V but for the unitary matrices U and V, we need to multiply them by $\frac{1}{\sqrt{2}}$
This algorithm is advantageous as we now don't need to do eigenvalue decomposition on A^*A, which has a large condition number. Hence this method will lead to faster convergence.

20 Answer to question 3c

The function that computes D is q3_c which is tested by test_q3_c.