Numerical analysis: midterm exam

Student: Pratyush Sudhakar
Due: March 20, 2024

Question 1 (8 points):

(a) $\csc(x) - \cot(x)$ for $x \in (-\pi, \pi)$. Using your more accurate expression compute and report the value of this function for $x = 10^{-10}$.

Solution: We will compute the condition number (κ_2) of the function $\csc(x) - \cot(x)$ for $x \in (-\pi, \pi)$. The condition number is given by

$$\kappa_2 = \frac{\|f'(x)\|_2}{\|f(x)\|_2} \tag{1.1}$$

where $f(x) = \csc(x) - \cot(x)$. The derivative of the function is given by

$$f'(x) = -\csc(x)\cot(x) - \csc^2(x) \tag{1.2}$$

The condition number is given by

$$\kappa_2 = \frac{\| -\csc(x)\cot(x) - \csc^2(x)\|_2}{\|\csc(x) - \cot(x)\|_2} \tag{1.3}$$

The function is ill-conditioned when the condition number is large. The condition number is large when the denominator is small. The denominator is small when the function is close to zero. The function is close to zero when x is close to 0. Therefore, the function is ill-conditioned when x is close to 0.

Hence, the accuracy of the function is poor when x is close to 0. To devise an accurate expression for the function,

$$\csc(x) - \cot(x) = \frac{1}{\sin(x)} - \frac{\cos(x)}{\sin(x)} = \frac{1 - \cos(x)}{\sin(x)}$$

We will use the double angle identity to express the function in terms of $\sin(x/2)$ and $\cos(x/2)$. The double angle identity is given by

(a) $\cos(x) = 1 - 2\sin^2(x/2)$
(b) $\sin(x) = 2\sin(x/2)\cos(x/2)$

The function can be expressed as

$$\csc(x) - \cot(x) = \frac{1 - (1 - 2\sin^2(x/2))}{2\sin(x/2)\cos(x/2)} = \frac{2\sin^2(x/2)}{2\sin(x/2)\cos(x/2)} = \frac{\sin(x/2)}{\cos(x/2)} = \tan(\frac{x}{2})$$

Hence, for $\mathbf{x = 10^{-10}}$, the value of the function is $\tan(\frac{\mathbf{10^{-10}}}{\mathbf{2}}) = \mathbf{5 \times 10^{-11}}$.

(b) $\sqrt{x+1} - \sqrt{x}$ for $x > 0$
Solution: The accuracy of the function will be affected by Catastrophic Cancellation when $\sqrt{x+1}$ and $\sqrt{x}$ are close to each other. This occurs when x is large.

$$\lim_{x\to\infty} \sqrt{x+1} - \sqrt{x} = 0$$

We can multiply and divide the function by the conjugate to avoid Catastrophic Cancellation. The conjugate of the function is given by

$$\sqrt{x+1} + \sqrt{x}$$

The function can be expressed as

$$\sqrt{x+1} - \sqrt{x} = \frac{(\sqrt{x+1} - \sqrt{x})(\sqrt{x+1} + \sqrt{x})}{\sqrt{x+1} + \sqrt{x}} = \frac{1}{\sqrt{x+1} + \sqrt{x}}$$

.

Hence, we can use the expression $\frac{1}{\sqrt{x+1}+\sqrt{x}}$ to compute the function when x is large as it does not suffer from Catastrophic Cancellation.

Question 2 (12 points)

(a) Given a symmetric positive definite matrix A and its Cholesky factorization $A = LL^T$ (Where L is lower triangular) prove that

$$\|L\|_2 = \|A\|_2^{1/2}.$$

Solution:

(a) The 2-norm of a symmetric matrix A, denoted as $\|A\|_2$, is the square root of the largest eigenvalue of $A^T A$. Since A is symmetric positive definite, its eigenvalues are real and positive. Therefore, the 2-norm of A is the square root of the largest eigenvalue of A.

$$\|A\|_2 = \sqrt{\lambda_{\max}(A^T A)} = \sqrt{\lambda_{\max}(A^2)} = \sqrt{\lambda_{\max}(A)^2} = \lambda_{\max}(A)$$

(b) The eigenvalues of $L^T L$ are same as the eigenvalues of LL^T because the matrices are simmilar. It can be proved as follows:

$$\text{Let } L^T L = UDU^T \text{ (SVD decomposition) where } U \text{ is orthogonal and } D \text{ is diagonal}$$
$$LL^T = (L^T)^T(L^T) = (LU^T)(LU^T)^T = U(L^T L)U^T = UDU^T$$

(c) The 2-norm of a matrix L is the square root of the largest eigenvalue of $L^T L$. Since L is lower triangular, the eigenvalues of $L^T L$ are same as the eigenvalues of LL^T because the matrices are simmilar. Therefore, the 2-norm of L is the square root of the largest eigenvalue of LL^T.

$$\|L\|_2 = \sqrt{\lambda_{\max}(L^T L)} = \sqrt{\lambda_{\max}(LL^T)} = \sqrt{\lambda_{\max}(A)} = \|A\|_2^{1/2}$$

This completes the proof that the 2-norm of the Cholesky factor L is equal to the square root of the 2-norm of A.

(b) If $A = QR$ is a reduced QR factorization for $A \in \mathbb{R}^{m\times n}$ with $m \geq n$ prove that the singular values of A and R are the same.
Solution: We can write A as

$$A = QR = U\Sigma V^T$$

Therefore, A^TA can be expressed as

$$A^TA = R^TQ^TQR = R^TR \tag{2.1}$$

because $Q^TQ = I$.

We know that the singular values of a matrix are the square roots of the eigenvalues of the matrix A^TA. Therefore, the singular values of A and R are as follows:

$$\sigma(A) = \sqrt{\lambda(A^TA)} = \sqrt{\lambda(R^TR)} = \sigma(R) \quad \text{(From equation (2.1))}$$

Hence, the singular values of A and R are the same.

(c) For any orthogonal projector P prove that $\|x\|_2^2 = \|Px\|_2^2 + \|(I-P)x\|_2^2$. **Solution:** Some properties of an orthogonal projector are as follows:

(a) $P^2 = P$
(b) $P^T = P$

Hence, the orthogonal projector P projects vectors onto a subspace, and $I-P$ projects vectors onto the orthogonal complement of that subspace. We express the squared norms as follows:

$$\begin{aligned}
\|x\|_2^2 &= x^Tx, \\
\|Px\|_2^2 &= (Px)^T(Px) = x^TP^TPx = x^TPx \quad (\text{since } P^T = P \text{ and } P^2 = P), \\
\|(I-P)x\|_2^2 &= ((I-P)x)^T((I-P)x) = x^T(I-P)^T(I-P)x = x^T(I-P)x \quad (\text{since } (I-P)^T = I-P).
\end{aligned}$$

Adding $\|Px\|_2^2$ and $\|(I-P)x\|_2^2$:

$$\begin{aligned}
\|Px\|_2^2 + \|(I-P)x\|_2^2 &= x^TPx + x^T(I-P)x \\
&= x^T(P + I - P)x \\
&= x^TIx \\
&= x^Tx \\
&= \|x\|_2^2.
\end{aligned}$$

Thus, we have proved that for any orthogonal projector P, $\|x\|_2^2 = \|Px\|_2^2 + \|(I-P)x\|_2^2$. This relationship demonstrates that the squared norms of the projections of x onto complementary subspaces sum up to the squared norm of x itself.

Question 3 (10 points)

Assume $A \in \mathbb{R}^{n\times n}$ is non-singular and we have already computed the partially pivoted LU factorization $PA = LU$. We now consider the linear system

$$\begin{bmatrix} A & B \\ C^T & D \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \end{bmatrix}, \tag{3}$$

where $B, C \in \mathbb{R}^{n\times \ell}$, $D \in \mathbb{R}^{\ell\times\ell}$, $b_1 \in \mathbb{R}^n$, and $b_2 \in \mathbb{R}^\ell$. Throughout this problem we will assume that the block matrix in (3) is non-singular and that $\ell < n$ is independent of n. We are also assuming that you are given L, U, and P, so you do not need to take the time to compute them into account in your answers.

(a) Devise a scheme to solve for x in $\mathcal{O}(n^2\ell)$ time. Clearly articulate your scheme and prove that it: (1) always returns the unique solution to the desired linear system and (2) achieves the stated complexity.

Solution: We propose an efficient algorithm to solve the given linear system represented by a block matrix.

Algorithm 1 Solve the linear system with block matrix

```
1: function SolveBlockSystem($A, B, C, D, b_1, b_2, L, U, P$)
2:     $x_2 \leftarrow$ SolveForX2($A, B, C, D, b_1, b_2, L, U, P$)
3:     $x_1 \leftarrow$ SolveForX1($A, B, x_2, b_1, L, U, P$)
4:     return $x_1, x_2$
5: end function
```

Algorithm 2 Function to solve for x_2

```
1: function SolveForX2($A, B, C, D, b_1, b_2, L, U, P$)
2:     $M \leftarrow C^T \cdot$ ApplyInverseAMatrix($B, L, U, P$)
3:     $N \leftarrow b_2 - C^T \cdot$ ApplyInverseAVector($b_1, L, U, P$)
4:     return SolveLinearSystem($D - M, N$)
5: end function
```

Algorithm 3 Function to solve for x_1

```
1: function SolveForX1($A, B, x_2, b_1, L, U, P$)
2:     $b_1' \leftarrow b_1 - B \cdot x_2$
3:     return SolveWithLU($A, b_1', L, U, P$)
4: end function
```

Algorithm 4 Solve a linear system using LU decomposition

```
function SolveLinearSystem($Matrix, Vector$)
    $LU \leftarrow$ DecomposeLU($Matrix$)
    $y \leftarrow$ ForwardSubstitution($LU.L, Vector$)
    return BackwardSubstitution($LU.U, y$)
end function
```

Algorithm 5 Apply the inverse of matrix A to a vector or matrix

```
function ApplyInverseAVector($Vector, L, U, P$)
    $y \leftarrow$ ForwardSubstitution($L, P \cdot Vector$)
    return BackwardSubstitution($U, y$)
end function

function ApplyInverseAMatrix($Matrix, L, U, P$)
    $Columns \leftarrow$ getColumns($Matrix$)
    $ResultMatrix \leftarrow$ emptyMatrix(size($Matrix$))
    for $col$ in $Columns$ do
        $ResultMatrix[:,col] \leftarrow$ ApplyInverseAVector($Matrix[:,col], L, U, P$)
    end for
    return $ResultMatrix$
end function
```

Algorithm Overview: The algorithm consists of two main computational steps:

(a) Solve for x_2 by first applying A^{-1} to both B and b_1, then solving the resultant linear system involving D and x_2.

(b) With x_2 known, solve for x_1 by adjusting b_1 to account for the interaction between x_1 and x_2 via B, then applying A^{-1} to the adjusted b_1.

Time Complexity: The time complexity of our algorithm is primarily determined by the application of A^{-1} to matrices and vectors: The algorithm to solve the given linear system with a block matrix involves several steps, each contributing to the overall time complexity. Here, we detail the time complexity associated with each part of the algorithm.

- **Applying A^{-1} to B and b_1**
  (a) **Applying A^{-1} to a matrix B:** This operation is the most computationally intensive part of the algorithm. For each column of B, which has ℓ columns in total, we solve a linear system using the LU decomposition of A. Since A is an $n \times n$ matrix, each solve operation (involving forward and backward substitution) has a complexity of $\mathcal{O}(n^2)$. Therefore, the total complexity for this step is $\mathcal{O}(n^2\ell)$.
  (b) **Applying A^{-1} to a vector b_1:** Solving a single linear system with the vector b_1 using the LU decomposition also requires forward and backward substitution. Since b_1 is an n-dimensional vector, this step has a complexity of $\mathcal{O}(n^2)$.
- **Solving for x_2**
  (a) After applying A^{-1} to B and b_1, we obtain $C^T A^{-1} B$ and $C^T A^{-1} b_1$, respectively. The computation of $C^T A^{-1} B$ is included in the $\mathcal{O}(n^2\ell)$ complexity from the first step.

(b) The system $(D - C^T A^{-1} B)x_2 = b_2 - C^T A^{-1} b_1$ is then solved. Assuming D is an $\ell \times \ell$ matrix, and since $C^T A^{-1} B$ has already been computed, this step primarily involves solving a linear system of size ℓ, which has a complexity of $\mathcal{O}(\ell^3)$ using direct methods like the LU decomposition. However, this complexity is overshadowed by the $\mathcal{O}(n^2\ell)$ complexity from the earlier step.

- **Solving for** x_1

  (a) With x_2 known, $b_1' = b_1 - Bx_2$ is computed. Multiplying B (an $n \times \ell$ matrix) by x_2 (an ℓ-dimensional vector) has a complexity of $\mathcal{O}(n\ell)$.

  (b) Finally, $Ax_1 = b_1'$ is solved using the LU decomposition of A, which, as mentioned, has a complexity of $\mathcal{O}(n^2)$ for a single solve operation.

Overall Time Complexity

Combining all the steps, the dominant term in the time complexity of the algorithm is $\mathcal{O}(n^2\ell)$, arising from applying A^{-1} to the matrix B. All other steps contribute lesser or equal complexity, ensuring that the overall complexity of solving the linear system with the block matrix is $\mathcal{O}(n^2\ell)$.

Uniqueness of Solution: The algorithm guarantees a unique solution under the assumption that the block matrix and A are non-singular. The uniqueness stems from the invertibility of A, ensuring that each step of the algorithm—whether it involves applying A^{-1} or solving smaller linear systems—produces a unique outcome. This property, combined with the structured approach to solving for x_1 and x_2, ensures that the overall solution to the linear system is unique.

(b) If A is sparse and we further assume that $PA = LU$ has factors L and U with at most a linear number of non-zeros in n can we solve (3) asymptotically faster than $\mathcal{O}(n^2\ell)$? If yes, provide an algorithm that achieves a faster complexity.
Solution: When dealing with sparse matrices and their LU decomposition, the operational complexity for certain key functions in our algorithm changes significantly. This adaptation leverages the sparsity of the matrix A and its LU factors to solve the system more efficiently than $\mathcal{O}(n^2\ell)$.

- **ApplyInverseAVector(**b_1**)**: For a vector $b_1 \in \mathbb{R}^n$, the complexity reduces from $\mathcal{O}(n^2)$ to $\mathcal{O}(n)$, leveraging the sparse forward and backward substitution processes.
- **ApplyInverseAMatrix(**B**)**: Applying A^{-1} to a matrix $B \in \mathbb{R}^{n\times\ell}$ had a complexity of $\mathcal{O}(n^2\ell)$. With A, L, and U being sparse, and assuming a linear number of non-zeros, the complexity reduces to $\mathcal{O}(n\ell)$. This is because each operation within the matrix-vector multiplication now scales linearly with n rather than quadratically.

Given these changes, the overall complexity of solving the block matrix system under the assumption of sparsity in A can be significantly improved. The most computationally intensive part of the process—applying A^{-1} to B—now operates at $\mathcal{O}(n\ell)$, which is a substantial reduction from the dense case.

Moreover, the steps involving the solution of x_2 and subsequently x_1 also benefit from the reduced complexity in applying A^{-1}. While the direct computation involving $D - C^T A^{-1} B$ and solving the smaller system for x_2 might still depend on the properties of D, the predominant operations scale more favorably.

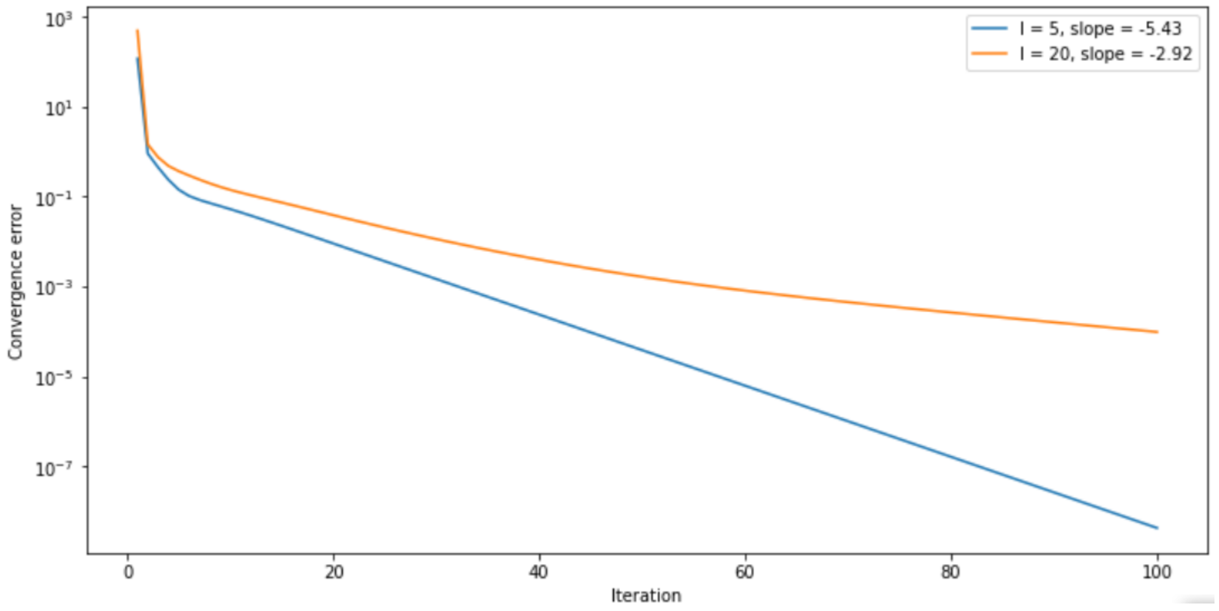

Figure 1: Convergence of $V^{(k)}$ to V_1 as a function of iteration

Hence, the time complexity will be $\mathcal{O}(max(n\ell, \ell^3))$. This will be better than $\mathcal{O}(n^2\ell)$ when $\ell < n$.

Question 4 (8 points)

(a) Let $V_1 \in \mathbb{R}^{n\times\ell}$ represent the eigenvectors associated with λ_1 through λ_ℓ. For $\ell = 5$ and $\ell = 20$ run simultaneous iteration to approximate V_1 and plot the convergence of $V^{(k)}$) to V_1 as a function of iteration. How do you expect the error to behave and do your results match your expectations? Why or why not?

Solution: Given a symmetric matrix A of size 100×100 with eigenvalues $\lambda_i = \frac{1}{i}$, the convergence of a simultaneous iteration for approximating eigenvectors depends critically on the spectral gap. The spectral gap, defined as the difference between successive eigenvalues $\lambda_i - \lambda_{i+1}$, diminishes as i increases. Consequently, the convergence is more rapid for smaller indices due to larger gaps. Specifically, when $\ell = 5$, the subspace iteration converges more swiftly as it encompasses eigenvalues with larger gaps, implying faster eigenvector approximation.

As shown in figure 1, the slope of convergence is steeper for $\ell = 5$ than for $\ell = 20$, indicating faster convergence for the former. This aligns with our expectations, as the spectral gap is larger for smaller indices, leading to faster convergence. The results are consistent with our expectations, demonstrating the influence of the spectral gap on the convergence of simultaneous iteration.

(b) Consider the following two functions:

$$(1) \quad f(x) = \sin(\pi x/2)$$
$$(2) \quad f(x) = 1/(1 + e^{-100(x-.175)}).$$

For the problem in part (a) with $\ell = 5$, would running subspace iteration with these functions help us converge to V_1 in fewer iterations? Please answer the question for both functions and

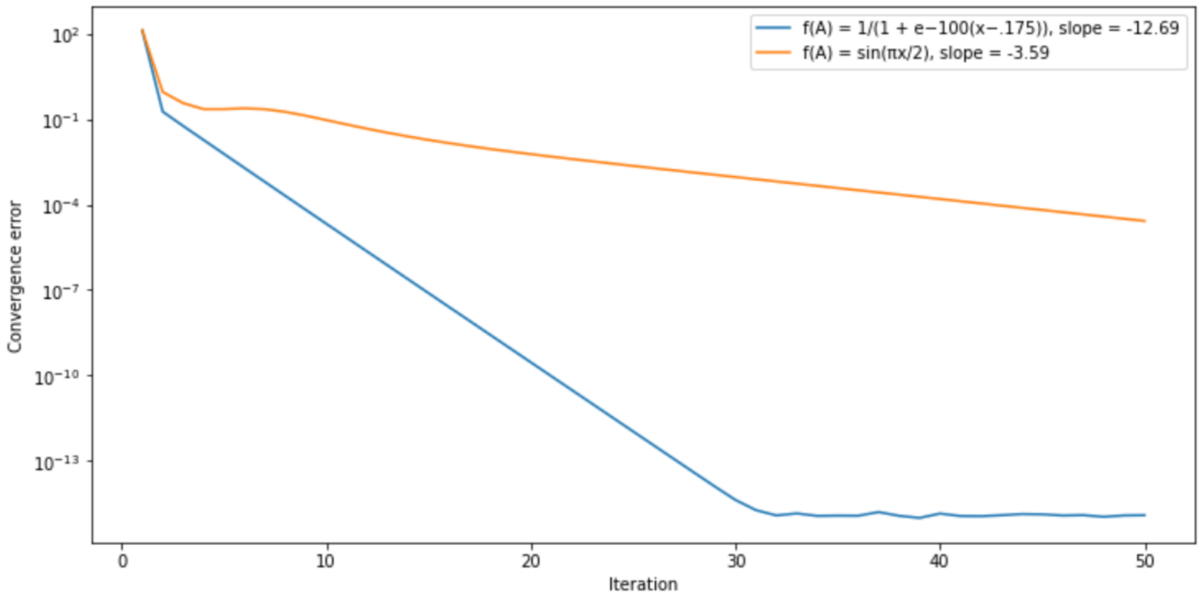

Figure 2: Convergence of $V^{(k)}$ to V_1 as a function of iteration for the two functions

justify your response.

Solution: The convergence of subspace iteration for a transformed matrix $f(A)$ depends on how the transformation function f alters the eigenvalues of A. Given the eigenvalue distribution of A with $\lambda_i = \frac{1}{i}$, we consider the two suggested functions:

$$\begin{aligned} &(1) \quad f(x) = \sin\left(\frac{\pi x}{2}\right) \\ &(2) \quad f(x) = \frac{1}{1 + e^{-100(x-0.175)}}. \end{aligned}$$

For $\ell = 5$, function (1) transforms the eigenvalues into a sinusoidal pattern which has the effect of clustering the transformed eigenvalues $f(\lambda_i)$ closer together, thus reducing the spectral gaps and potentially slowing the convergence for the dominant eigenspace.

Function (2) is a sigmoid function that sharply increases around $x = 0.175$. This transformation will drastically amplify the spectral gap between the eigenvalues less than 0.175 and those greater than 0.175. Since $\lambda_5 = 0.2$, the spectral gap around λ_5 is significantly increased, likely accelerating the convergence of subspace iteration towards the eigenvectors corresponding to the largest transformed eigenvalues $f(\lambda_1)$ through $f(\lambda_5)$.

Therefore, while function (1) may not facilitate faster convergence, function (2) is expected to enhance the separation between the dominant eigenvalues and the rest, thus aiding in more rapid convergence to V_1 for $\ell = 5$.

As shown in figure 2, the convergence of subspace iteration for function (2) is significantly faster than both the original and function (1), aligning with our expectations. This demonstrates the influence of the transformation function on the convergence of subspace iteration, with function (2) accelerating the convergence for $\ell = 5$.

Question 5 (12 points)

Consider $A \in \mathbb{R}^{m\times n}$ with $m < n$.

(a) Prove that the least squares problem

$$\min_x \|Ax - b\|_2^2$$

has an infinite number of solutions for any b.

Solution: Given a matrix $A \in \mathbb{R}^{m\times n}$ with $m < n$, we aim to demonstrate that the least squares problem $\min_x \|Ax - b\|_2^2$ has an infinite number of solutions for any b.

Proof:

(a) **Rank and Dimensions:**
Since A has more columns than rows ($m < n$), it follows that $\operatorname{rank}(A) \le m$. This implies that A cannot span $\mathbb{R}^n$, leading to a non-trivial null space $\mathcal{N}(A)$, meaning there exists at least one non-zero vector $v \in \mathbb{R}^n$ such that $Av = 0$.

(b) **Infinite Solutions:**
For any solution x to the least squares problem, we can add any vector $v \in \mathcal{N}(A)$ to x without affecting the residual norm $\|Ax-b\|_2^2$. This is because $A(x+v) = Ax+Av = Ax$, and the residual norm remains the same.

Conclusion:

Thus, the least squares problem $\min_x \|Ax - b\|_2^2$ admits an infinite number of solutions for any b when $A \in \mathbb{R}^{m\times n}$ with $m < n$, due to the underdetermined nature of the system and the existence of a non-trivial null space for A.

(b) Prove that for any non-singular $M \in \mathbb{R}^{n\times n}$ the regularized problem

$$\min_x \|Ax - b\|_2^2 + \|Mx\|_2^2 \tag{4}$$

has a unique solution.

Solution: Given the regularized problem:

$$\min_x \|Ax - b\|_2^2 + \|Mx\|_2^2,$$

the objective function is:

$$f(x) = (Ax - b)^T(Ax - b) + (Mx)^T(Mx).$$

First Derivative (Gradient): Taking the gradient of $f(x)$ with respect to x, we obtain:

$$\nabla f(x) = 2A^T(Ax - b) + 2M^TMx.$$

Second Derivative (Hessian): The Hessian H of $f(x)$ is:

$$H = \nabla^2 f(x) = 2A^TA + 2M^TM.$$

Strict Convexity: The function $f(x)$ is strictly convex if its Hessian is positive definite. Since A^TA is positive semi-definite and M^TM is positive definite (as M is non-singular), the sum $A^TA + M^TM$ is positive definite.

Horn, Roger A.; Johnson, Charles R. (2013). Matrix Analysis (2nd ed.).

$$x^T(A^TA + M^TM)x = x^TA^TAx + x^TM^TMx > 0, \quad \forall x \neq 0.$$

because $x^TM^TMx > 0$ for $x \neq 0$ and $x^TA^TAx \geq 0$ for all x.

Therefore, $f(x)$ is strictly convex.

Conclusion: Therefore, the objective function $f(x)$ is strictly convex, ensuring the uniqueness of the global minimum. Hence, the given regularized problem has a unique solution.

(c) Provide an algorithm that solves (4) in $\mathcal{O}(n^3)$ time. Be sure to show why your algorithm returns the desired result.
Solution: To solve the regularized least squares problem

$$\min_x \|Ax - b\|_2^2 + \|Mx\|_2^2,$$

we derive an explicit solution and propose an algorithm with computational complexity $\mathcal{O}(n^3)$.
First Derivative (Gradient): Taking the gradient of $f(x)$ with respect to x, we obtain:

$$\nabla f(x) = 2A^T(Ax - b) + 2M^TMx.$$

We can equate the gradient to zero to obtain the solution: Therefore,

$$\begin{aligned} 2A^T(A\hat{x} - b) + 2M^TM\hat{x} = 0 \\ A^TA\hat{x} + M^TM\hat{x} = A^Tb. \end{aligned}$$

The solution is given by the linear system:

$$(A^TA + M^TM)\hat{x} = A^Tb.$$

Algorithm:

(a) Compute A^TA and M^TM. It takes $\mathcal{O}(mn^2)$ and $\mathcal{O}(n^3)$ time, respectively.
(b) Sum A^TA and M^TM to form $(A^TA + M^TM)$. It takes $\mathcal{O}(n^2)$ time.
(c) Use the Cholesky factorization to compute the LU decomposition of $(A^TA + M^TM)$. It takes $\mathcal{O}(n^3)$ time.
(d) Solve the linear system $(A^TA + M^TM)\hat{x} = A^Tb$ using the LU decomposition. It takes $\mathcal{O}(n^2)$ time using forward and backward substitution.

Complexity Analysis: The computational complexity is dominated by matrix multiplication and LU decomposition, both of which have a complexity of $\mathcal{O}(n^3)$. The overall complexity of the algorithm is $\mathcal{O}(n^3)$.

Conclusion: This algorithm efficiently computes the unique solution to the regularized least squares problem using standard linear algebra operations.

(d) If M is upper triangular can we solve (4) asymptotically faster than $\mathcal{O}(n^3)$? If we can, provide an algorithm to do so, prove its complexity, and show why it returns the desired result. **Solution:** The goal is to minimize the regularized least squares objective given by

$$\min_x \|Ax - b\|_2^2 + \|Mx\|_2^2 .$$

Given that M is a non-singular upper triangular matrix, its singular value decomposition (SVD) has some nice properties. Let the SVD of M be $M = XYZ$, where X and Z are orthogonal matrices, and Y is a diagonal matrix whose diagonal values are the singular values of M, which also coincide with the diagonal entries of M since M is upper triangular.

http://arxiv.org/pdf/1202.1490.pdf

If the SVD of A is $A = UEV^T$, where U and V are orthogonal, and E is diagonal, then we can reformulate the normal equation for the regularized problem as

$$(A^TA + M^TM)x = A^Tb$$

which can be written as

$$(VE^2V^T + Y^TY)^{-1}VEU^Tb.$$

We can note that $Y^TY = VY^TYV^T$ because V is an orthogonal matrix and the multiplication by V and V^T on either side does not change the diagonal structure of Y^TY. Therefore, the solution for x can be expressed as

$$x = \left(V(E^2 + Y^TY)V^T\right)^{-1} VEU^Tb,$$

which simplifies to

$$x = V(E^2 + Y^TY)^{-1}EU^Tb.$$

The matrix $E^2 + Y^TY$ is diagonal, and its inversion can be done efficiently. Let E^+ be the matrix such that $E^+_{ii} = \sigma_i/(\sigma_i^2 + M_{ii}^2)$, which represents the updated singular values considering the regularization term. This can be computed in $\mathcal{O}(n)$ time.

The dominant computational factor is the SVD of A, which can be computed in $\mathcal{O}(m^2n)$ time. The matrix multiplications involving orthogonal matrices and diagonal matrices can be done efficiently, and hence, the total computational complexity is dominated by the SVD computation.

$$\begin{aligned} x &= V(E^2 + Y^TY)^{-1}EU^Tb \\ &= V\left(\operatorname{diag}\left(\frac{\sigma_i}{\sigma_i^2 + M_{ii}^2}\right)\right)^{-1} U^Tb \end{aligned}$$

which provides the solution x for the regularized least squares problem.