1. **Least $C$-norm solution.** Let $A \in \mathbb{R}^{m \times n}$ be full-rank and fat, and $C \in \mathbb{R}^{k \times n}$ be full-rank and tall.

   (a) Find the solution to
   
   $Ax = b$
   
   that minimizes
   
   $\|Cx\|$.  

   (b) Find the solution to $Ax = b$ that minimizes $\|C(x - x_0)\|$. 

   (c) Show that
   
   $\|Cx - d\|^2 - \|C(x - (CTC)^{-1}CTd)\|^2$
   
   does not depend on $x$. 

   (d) Show that the solution to $Ax = b$ that minimizes $\|Cx - d\|$ is
   
   $x^* = (CTC)^{-1}CTd + (CTC)^{-1}A^T(A(CTC)^{-1}A^T)^{-1}(b - A(CTC)^{-1}CTd)$. 

2. **Eigenvalues.** Suppose that $A$ has $\lambda_1, \lambda_2, \ldots, \lambda_n$ as its eigenvalues.

   (a) Show that $\det(A) = \lambda_1 \cdot \lambda_2 \cdots \lambda_n$. 

   (b) Show that the eigenvalues of $A^T$ are $\lambda_1, \lambda_2, \ldots, \lambda_n$, that is, $A$ and $A^T$ have the same set of eigenvalues. 

   (c) Show that the eigenvalues of $A^k$ are $\lambda_1^k, \lambda_2^k, \ldots, \lambda_n^k$ for $k = 1, 2, \ldots$. 

   (d) Show that $A$ is invertible if and only if it does not have a zero eigenvalue. 

   (e) Suppose that $A$ is invertible. Show that the eigenvalues of $A^{-1}$ are $\lambda_1^{-1}, \lambda_2^{-1}, \ldots, \lambda_n^{-1}$. 

   (f) Show that $A$ and $T^{-1}AT$ have the same set of eigenvalues, that is, eigenvalues are invariant under a similarity transformation $A \mapsto T^{-1}AT$. 

3. **Trace.** We define the *trace* of $A \in \mathbb{R}^{n \times n}$ as

   $\text{tr}(A) = A_{11} + A_{22} + \cdots + A_{nn}$, 

   that is, the sum of its diagonal entries.

   (a) Suppose that $A$ has $\lambda_1, \lambda_2, \ldots, \lambda_n$ as its eigenvalues. Show that

   $\text{tr}(A) = \lambda_1 + \lambda_2 + \cdots + \lambda_n$. 

1
(b) Show that
\[ \text{tr}(A^k) = \lambda_1^k + \lambda_2^k + \cdots + \lambda_n^k \]
for \( k = 1, 2, \ldots \).


(a) Show that \( e^{A+B} = e^A e^B \) if \( A \) and \( B \) commute, namely, \( AB = BA \).

(b) Show that \( A \) and \( e^{tA} \) commute for every \( t \in \mathbb{R} \).

(c) Suppose that \( \lambda_1, \lambda_2, \ldots, \lambda_n \) are eigenvalues of \( A \). Show that the eigenvalues of \( e^A \) are \( e^{\lambda_1}, e^{\lambda_2}, \ldots, e^{\lambda_n} \).

(d) Show that \( \det(e^A) = e^{\text{tr}(A)} \).

(e) Show that
\[ \frac{d}{dt} e^{tA} = A e^{tA} = e^{tA} A. \]

(f) Show that
\[ e^A = \lim_{k \to \infty} \left( I + \frac{A}{k} \right)^k. \]

5. Square root of a matrix. Let \( A \in \mathbb{R}^{n \times n} \) be diagonalizable with eigenvalue decomposition \( T \Lambda T^{-1} \). We say that a matrix \( B \) is a square root of \( A \) if \( B^2 = A \). Show that \( B = T \Lambda^{1/2} T^{-1} \) is a square root of \( A \), where \( \Lambda^{1/2} \) is a diagonal matrix with entries \( \gamma_1, \gamma_2, \ldots, \gamma_n \) such that \( \gamma_i^2 = \lambda_i \).
Programming Assignment

Write down your code as clearly as possible and add suitable comments.

1. **Orthogonal matching pursuit.** Consider a linear equation $Ax = b$, where $A \in \mathbb{R}^{m \times n}$ is full-rank and fat. Suppose that we are interested in finding the sparsest solution, namely, the solution $x$ that has the fewest number of nonzero entries. In class, we discussed that the uniquely sparsest solution with sparsity $k = \|x\|_0$ exists if $2k \leq m$ and every tuple of $m$ columns of $A$ is linearly independent.

Let $a_1, a_2, \ldots, a_n$ be the columns of $A$, which will be referred to collectively as the dictionary. The orthogonal matching pursuit (OMP) algorithm aims to find a sparse linear combination of dictionary vectors that matches the target $b$. In the first step, the algorithm finds the dictionary vector $a_{j_1}$ that is the most aligned with $b$, that is,

$$j_1 = \arg\max_i \frac{|\langle a_i, b \rangle|}{\|a_i\|}.$$  

The algorithm then computes the best approximation $b_1$ of the target vector $b$ using the dictionary vector $a_{j_1}$, and the corresponding residual $r_1 = b - b_1$. In the second step, the algorithm finds the dictionary vector $a_{j_2}$ that is the most aligned with the residual $r_1 = b - b_1$, that is,

$$j_2 = \arg\max_i \frac{|\langle a_i, r_1 \rangle|}{\|a_i\|}.$$ 

The algorithm then computes the best approximation $b_2$ of $b$ using the dictionary vectors $a_{j_1}$ and $a_{j_2}$, and the corresponding residual $r_2 = b - b_2$. In each subsequent step, the algorithm finds the dictionary vector that is the most aligned with the residual from the previous step and finds the best approximation by the least squares method and the corresponding residual. The algorithm continues until the approximation is sufficiently close to the target vector $b$. Note that the same dictionary vector is never selected again and that the algorithm takes at most $m$ steps to finish.

We now describe the algorithm more formally. The algorithm takes as input the target vector $b \in \mathbb{R}^m$ and a dictionary matrix $A \in \mathbb{R}^{m \times n}$ and returns an index set $\mathcal{J}$ of chosen dictionary vectors and the sparse linear representation of $b$ in terms of those vectors.

1. Initialize with the residual $r_0 = b$, the index set $\mathcal{J}_0 = \emptyset$, and the iteration counter $t = 1$.
2. For each step $t$, find the dictionary vector that is the most aligned with $r_{t-1}$ as

$$j_t = \arg\max_i \frac{|\langle a_i, r_{t-1} \rangle|}{\|a_i\|}.$$ 

When there is a tie, choose the smallest index.
3. Augment the index set as

$$\mathcal{J}_t = \mathcal{J}_{t-1} \cup \{j_t\}.$$ 

Augment the matrix composed of the chosen dictionary vectors as

$$\tilde{A}_t = [\tilde{A}_{t-1} \ a_{j_t}].$$
(4) Approximate $b$ using the dictionary vectors by the least squares method as

$$\tilde{x}_t^* = \arg \min_{\tilde{x}} \| b - \tilde{A}\tilde{x} \|$$

and compute the best approximation as

$$b_t = \tilde{A}_t \tilde{x}_t^*.$$

(5) Compute the residual as

$$r_t = b - b_t.$$  

(6) If $t = m$ or $\|r_t\|$ is sufficiently small, stop and output $J_t$ and $\tilde{x}_t^*$. Otherwise, increment $t$ and return to step 2.

In this problem, we implement the OMP algorithm when $A$ is randomly generated and examine when the algorithm finds the unique sparsest solution.

(a) Write a Julia function $\text{OMP}(b, \text{thresh}, A)$ that takes as input an $m$-dimensional observation vector $b$, a stopping threshold $\text{thresh}$, an $m \times n$ matrix $A$, and uses the OMP algorithm to return a sparse $n$-dimensional vector $x$ such that $b-Ax$ has the Euclidean-norm less than $\text{thresh}$ (i.e., you can stop the iterations once the residual has the Euclidean norm less than $\text{thresh}$). Each step of OMP solves a least squares problem. How would you make this step more efficient?

(b) Let $m = 50$, and $n = 1000$, and $k_{\max} = 25$. For each $k \in \{1, 2, \ldots, k_{\max}\}$, repeat the following steps one hundred times.

(i) Generate a random matrix $A \in \mathbb{R}^{m \times n}$ with i.i.d. entries drawn from $N(0, 1)$.
(ii) Generate a sparse vector $x \in \mathbb{R}^n$ with random support of cardinality $k$ and non-zero entries drawn as uniform random variables in the range $[-10, -1] \cup [1, 10]$.
(iii) Compute the observation vector $b = Ax$ and use $b$ and $A$ as inputs to the $\text{OMP}$ function from part (b) to compute an estimate $\hat{x}$ of $x$. Use a threshold of 0.01.
(iv) Compute the relative error

$$E_{\text{rel}} = \frac{\|x - \hat{x}\|}{\|x\|}$$

and the support distance

$$d_{\text{support}}(x, \hat{x}) = \frac{\max\{|\text{supp}(x)|, |\text{supp}(\hat{x})|\} - |\text{supp}(x) \cap \text{supp}(\hat{x})|}{\max\{|\text{supp}(x)|, |\text{supp}(\hat{x})|\}}.$$  

For each $k$, compute the average relative error and the average distance between supports over the 100 runs. The average distance between supports will be referred to as the probability of error in support. Plot the probability of error in support and the average relative error vs. $k$, respectively.