1. Suppose that \(a_1, \ldots, a_k\) are orthonormal \(n\)-vectors and \(\beta_1, \ldots, \beta_k\) are scalars. Assume \(x = \sum_{i=1}^k \beta_i a_i\). Express \(\|x\|\) in terms of \(\beta = (\beta_1, \ldots, \beta_k)\).

2. Consider the list of \(n\) \(n\)-vectors, \(a_1, \ldots, a_n\) with,

\[ a_k = \sum_{i=1}^k e_i, \quad k = 1, \ldots, n. \]

(a) Describe what happens when you run the Gram-Schmidt algorithm on this list of vectors. I.e., determine what \(q_1, \ldots, q_n\) are.

(b) Is \(a_1, \ldots, a_n\) a basis for \(\mathbb{R}^n\)?

(c) Implement the Gram-Schmidt algorithm in Julia. Run your code on \(a_1, \ldots, a_n\) for \(n = 10\) and \(n = 100\). Does the output differ if you use a different order for \(a_1, \ldots, a_n\)? That is, if you run the algorithm on a non-trivial permutation of \(a_1, \ldots, a_n\), do you get a different result? Explain.

3. Let \(A\) and \(B\) be two \(m \times n\) matrices. Under each of the assumptions below, determine whether \(A = B\) must always hold, or whether \(A = B\) holds only sometimes. Explain/prove your answer.

(a) Suppose \(Ax = Bx\) holds for all \(n\)-vectors \(x\).

(b) Suppose \(Ax = Bx\) for some nonzero \(n\)-vector \(x\).

4. Take any matrix \(A \in \mathbb{R}^{m \times n}\). Show that \(A^T A\) has the same null space as \(A\).

5. An \(n \times n\) matrix \(A\) is called skew-symmetric if \(A^T = -A\).

(a) Find all \(2 \times 2\) skew-symmetric matrices.

(b) Explain why the diagonal entries of a skew-symmetric matrix must be zero.

(c) Show that for a skew-symmetric matrix \(A\), and any \(n\)-vector \(x\), \((Ax) \perp x\). This means that \(Ax\) and \(x\) are orthogonal.

(d) Now suppose \(A\) is a matrix for which \((Ax) \perp x\) for any \(n\)-vector \(x\). Show that \(A\) must be skew-symmetric.

(e) Create a Julia function that creates a random skew-symmetric matrix of order \(n\). Use it to empirically check that \((Ax) \perp x\) by generating 10,000 random matrices of order \(n = 5\) and 10,000 random vectors.

6. For this problem we consider several linear functions of a monochrome image with \(N \times N\) pixels. We represent the image as a \(N^2\)-vector with ordering based on columns of the image (column-major). Each of the operations or transformations below defines a function \(y = f(x)\) where the \(N^2\)-vector \(x\) represents the original image, and the \(N^2\)-vector \(y\) represents the resulting transformed image. For each of these operations, define the \(N^2 \times N^2\) matrix \(A\) such that \(f(x) = Ax\). Try it in Julia on the image \(\text{image} = [(i+j)^2 \text{ for } i \text{ in } 1:10, j \text{ in } 1:10]\). Present your results via \text{heatmap( ,yflip = true)}.

(a) Turn the original image upside-down.

(b) Rotate the original image clockwise 90°.

(c) Translate the image up by 2 pixels and to the right by 2 pixels. In the translated image, assign the value 0 to the pixels in the first 2 columns and last 2 rows.

(d) Set each pixel value to be the average of the neighbours of the pixel in the original image (there are several alternative meanings to “neighbours” - choose one meaning and explain the meaning that you use).
7. Consider a function \( f : [-1, 1] \to \mathbb{R} \). We are interested in estimating the definite integral 
\[ \alpha = \int_{-1}^{1} f(x) \, dx \] based on the value of \( f \) at some points \( t_1, \ldots, t_n \). The standard method for estimating \( \alpha \) is to form a weighted sum of the values \( f(t_i) \):
\[ \hat{\alpha} = w_1 f(t_1) + \ldots + w_n f(t_n). \]
Here the estimate \( \hat{\alpha} \) approximates \( \alpha \). This method is called quadrature. There are many quadrature methods (i.e. choices of points \( t_i \) and weights \( w_i \)).

(a) A typical requirement of the quadrature is that the approximation be exact (i.e. \( \hat{\alpha} = \alpha \)) when \( f \) is any polynomial up to degree \( d \), where \( d \) is given. In this case we say that the quadrature method has order \( d \). Express this condition as a set of linear equations on the weights \( Aw = b \), assuming the points \( t_1, \ldots, t_n \) are given.

(b) Show that the following quadrature methods have order 1, 2 and 3 respectively:
   (i) Trapezoid rule: \( n = 2 \), \( t_1 = -1 \), \( t_2 = 1 \), \( w_1 = w_2 = 1/2 \).
   (ii) Simpson’s rule: \( n = 3 \), \( t_1 = -1 \), \( t_2 = 0 \), \( t_3 = 1 \), \( w_1 = 1/3 \), \( w_2 = 4/3 \), \( w_3 = 1/3 \).
   (iii) Simpson’s 3/8 rule: \( n = 4 \), \( t_1 = -1 \), \( t_2 = -1/3 \), \( t_3 = 1/3 \), \( t_4 = 1 \), \( w_1 = 1/4 \), \( w_2 = 3/4 \), \( w_3 = 3/4 \), \( w_4 = 1/4 \).

(c) Implement these for the function \( f(x) = \sin(x)/x \) and compare their performance to the actual value of \( \alpha \).

(d) Use your answer to (a) to find a rule of a higher order that outperforms the rules above for \( f(x) = \sin(x)/x \). You choose the \( t_i \) values as you wish. Demonstrate your method outperforms the other rules.

8. Let \( a \) and \( b \) be \( n \)-vectors. The inner product is symmetric, i.e. \( a^T b = b^T a \). The outer product of the two vectors is generally not symmetric. What are the conditions on \( a \) and \( b \) under which \( ab^T = ba^T \)? You can assume that all the entries of \( a \) and \( b \) are nonzero. The conclusion you come to will hold even when some entries of \( a \) and \( b \) are zero.

9. The sum of the diagonal entries of a square matrix is called the trace and denoted by \( \text{tr}(A) \).

(a) Suppose \( A \) and \( B \) are \( m \times n \) matrices. Show that,
\[ \text{tr}(A^T B) = \sum_{i=1}^{m} \sum_{j=1}^{n} A_{ij} B_{ij}. \]

(b) The number \( \text{tr}(A^T B) \) is sometimes referred to as the inner product of the matrices \( A \) and \( B \). Show that \( \text{tr}(A^T B) = \text{tr}(B^T A) \).

(c) Show that \( \text{tr}(A^T A) = ||A||^2 \).

(d) Show that \( \text{tr}(A^T B) = \text{tr}(B A^T) \), even though in general \( A^T B \) and \( B A^T \) can have different dimensions, and even when they have the same dimensions, they need not be equal.

10. Suppose the \( n \times k \) matrix \( A \) has QR factorization \( A = QR \). We define the \( n \times i \) matrices,
\[ A_i = [a_1 \cdots a_i], \quad Q_i = [q_1 \cdots q_i], \]
for \( i = 1, \ldots, k \). Define the \( i \times i \) matrix \( R_i \) as the sub matrix of \( R \) containing its first \( i \) rows and columns, for \( i = 1, \ldots, k \). Using index range notation, we have,
\[ A_i = A_{1:i,n,1:i}, \quad Q_i = A_{1:i,n,1:i}, \quad R_i = R_{1:i,1:i}. \]
Show (prove) that \( A_i = Q_i R_i \) is the QR factorization of \( A_i \). This means that when you compute the QR factorization of \( A \), you are also computing the QR factorization of all sub matrices \( A_1, \ldots, A_k \). Demonstrate this in Julia on example matrices for \( n = 5 \).