Homework 1 Report

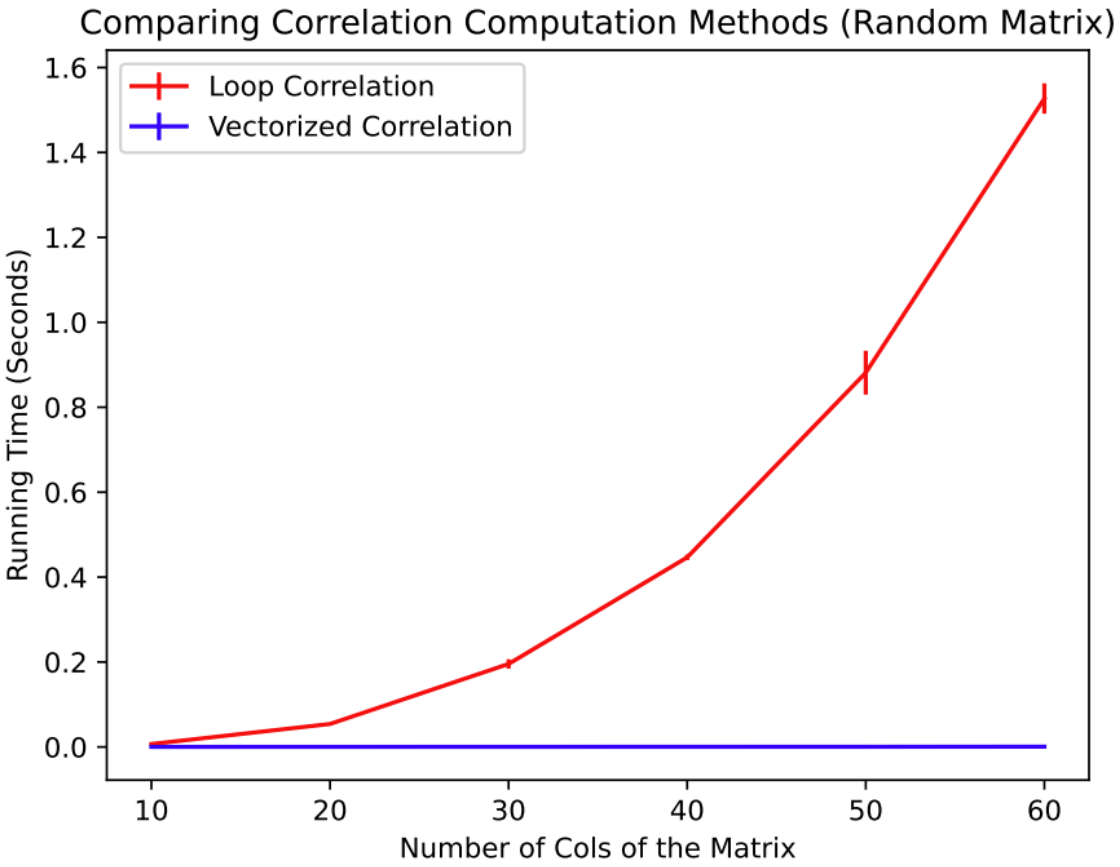
Problem 1: Distance Matrix

Loop-based Version (loop)

- **Input:** Data matrix X with N rows (data points) and D columns (features).
- **Output:** Distance matrix Z with N rows and N columns, where entry Z(i, j) is the Euclidean distance between row i and row j of the data matrix.
- **Steps:**
    1. Start with an empty N × N matrix Z filled with zeros.
    2. For each row i from first to last:
        - For each row j from first to last:
            - Get the data vector for row i (x_i) and row j (x_j)
            - Calculate the distance between x_i and x_j
            - Store the distance in position Z(i, j)
    3. Repeat until all entries are filled.
- **Complexity**: This algorithm checks every pair of rows (N^2), and for each pair goes through all D features. The time cost is N^2 x D.

Vectorized Version (vector)

- **Input:** Data matrix X with N rows and D columns
- **Output:** Distance matrix Z with N rows and N columns
- **Steps:**
    1. Compute the squared length of every row in X. This gives a vector of size N
    2. Compute all pairwise dot products by multiplying X with its transpose. This gives an N × N matrix
    3. For every pair (i, j), get the squared distance using: Z(i,j)^2 = length(i) + length(j) – 2 × dot(i,j)
    4. Take the square root of every entry to form the final distance matrix Z
- **Complexity**: The algorithm uses matrix multiplication and broadcasting. The work is still on the time cost of N^2 × D.



The plot shows how runtime changes as the number of rows N increases while the number of columns D is fixed. The loop version becomes much slower as N grows, while the vectorized version stays very fast. Both methods have complexity O(N^2 * D), but the vectorized implementation is far more efficient in practice because it uses optimized matrix operations.

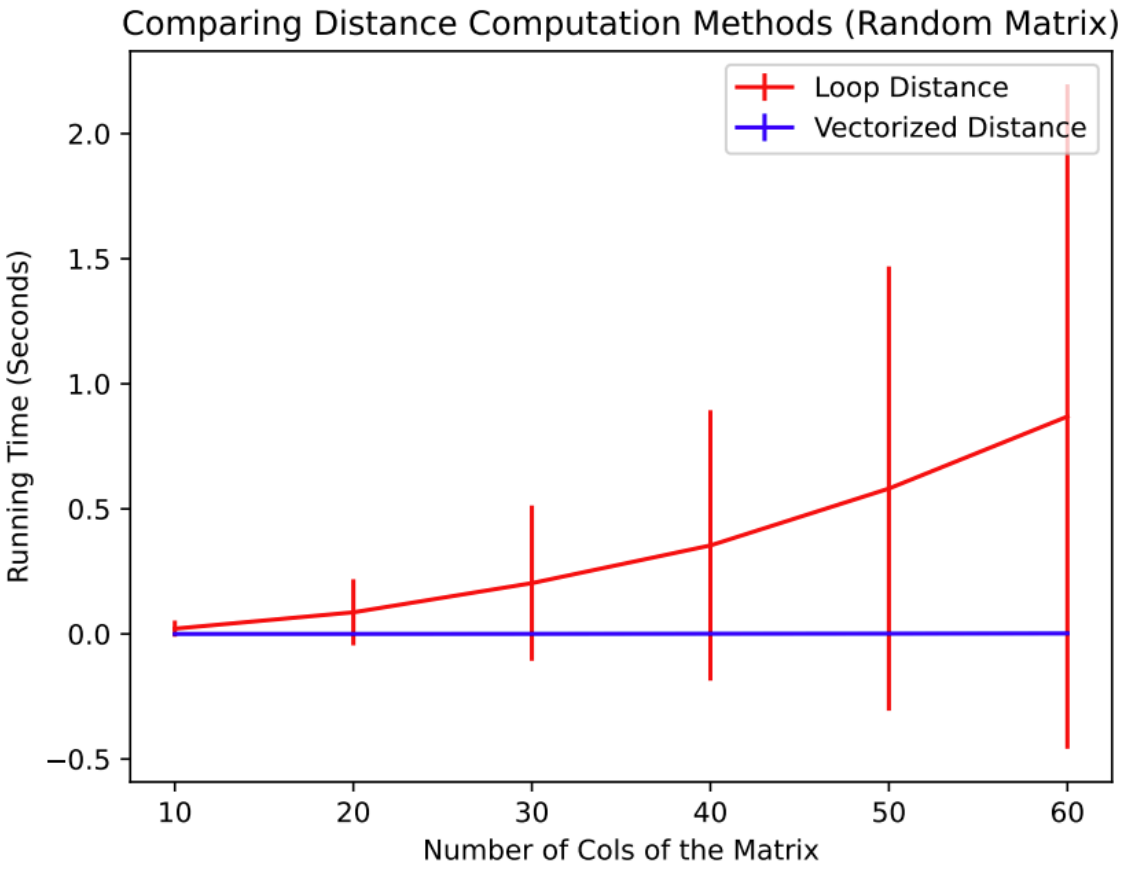
Problem 2: Correlation Matrix

Loop-based Version (loop)

- **Input:** Data matrix X with N rows (data points) and D columns (features).
- **Output:** Correlation matrix R with D rows and D columns, where entry R(i,j) measures the linear relationship between feature i and feature j.
- **Steps:**
    1. Compute the mean of each column of X (vector of length D).
    2. For each feature i from first to last:
        - For each feature j from first to last:
            - Go through all N rows.
            - Subtract the column means: (X(n,i) – mean(i)) and (X(n,j) – mean(j)).
            - Multiply these deviations for each row and add them up.
            - Divide the sum by (N – 1) to get the covariance S(i,j).
            - Divide S(i,j) by the product of the standard deviations of features i and j to get R(i,j).
    3. Store R(i,j) in the correlation matrix.
- **Complexity**: Requires two nested loops over D × D feature pairs, and an inner loop over N rows. The time cost is D^2 × N.

Vectorized Version (vector)

- **Input:** Same data matrix X (N × D).
- **Output:** Correlation matrix R (D × D).
- **Steps:**
    1. Subtract the mean of each column from X to create a centered matrix (N × D).
    2. Multiply the transpose of the centered matrix with the centered matrix, then divide by (N – 1). This produces the full covariance matrix S (D × D).
    3. Extract the standard deviations from the diagonal of S.
    4. Form a denominator matrix by taking the outer product of the standard deviation vector with itself.
    5. Divide S element-by-element by this denominator matrix to obtain R.
- **Complexity**: The time cost is still D^2 × N, but the computation is much faster in practice because it uses optimized matrix operations instead of explicit loops.



The plot shows the runtime of loop and vectorized correlation computations on random matrices. As the number of rows N increases with D fixed, the loop version slows down quickly, while the vectorized version stays nearly flat. Both methods have complexity O(D^2 * N), but the vectorized approach runs much faster in practice due to optimized operations.

Problem 3:

Datasets

- Iris: N = 150, D = 4
- Breast Cancer: N = 569, D = 30
- Digits: N = 1797, D = 64

For each dataset, the computation time was recorded using two methods:

- loop implementation (with nested loops)
- vector implementation (using vectorization)

Distance Matrix Computation Times

Dataset	loop (s)	vector (s)	SpeedUp
Iris	0.0947	0.0002	554.6
Breast Cancer	1.3710	0.0034	398.6
Digits	13.7171	0.0429	319.5

Correlation Matrix Computation Times

Dataset	loop (s)	vector (s)	SpeedUp
Iris	0.0009	0.0001	16.6
Breast Cancer	0.1788	0.0002	1020.0
Digits	2.6354	0.0006	4430.4

Experiments Outcome

The timing results show that vectorized implementations are much faster than loop loop-based versions across all three problems. The control variable is the dataset size (N and D, the number of rows and features). As datasets get larger, the performance gap increases significantly. For example, in the Digits dataset, the vector distance computation ran in under 0.0429 seconds, compared to over 13.7171 seconds for the loop version. The same pattern holds for correlation matrices. This shows that vectorized methods are far more efficient in practice, even if the theoretical complexity, such as both being N^2 x D, is the same.