**Analyzing the Mariana Trench**

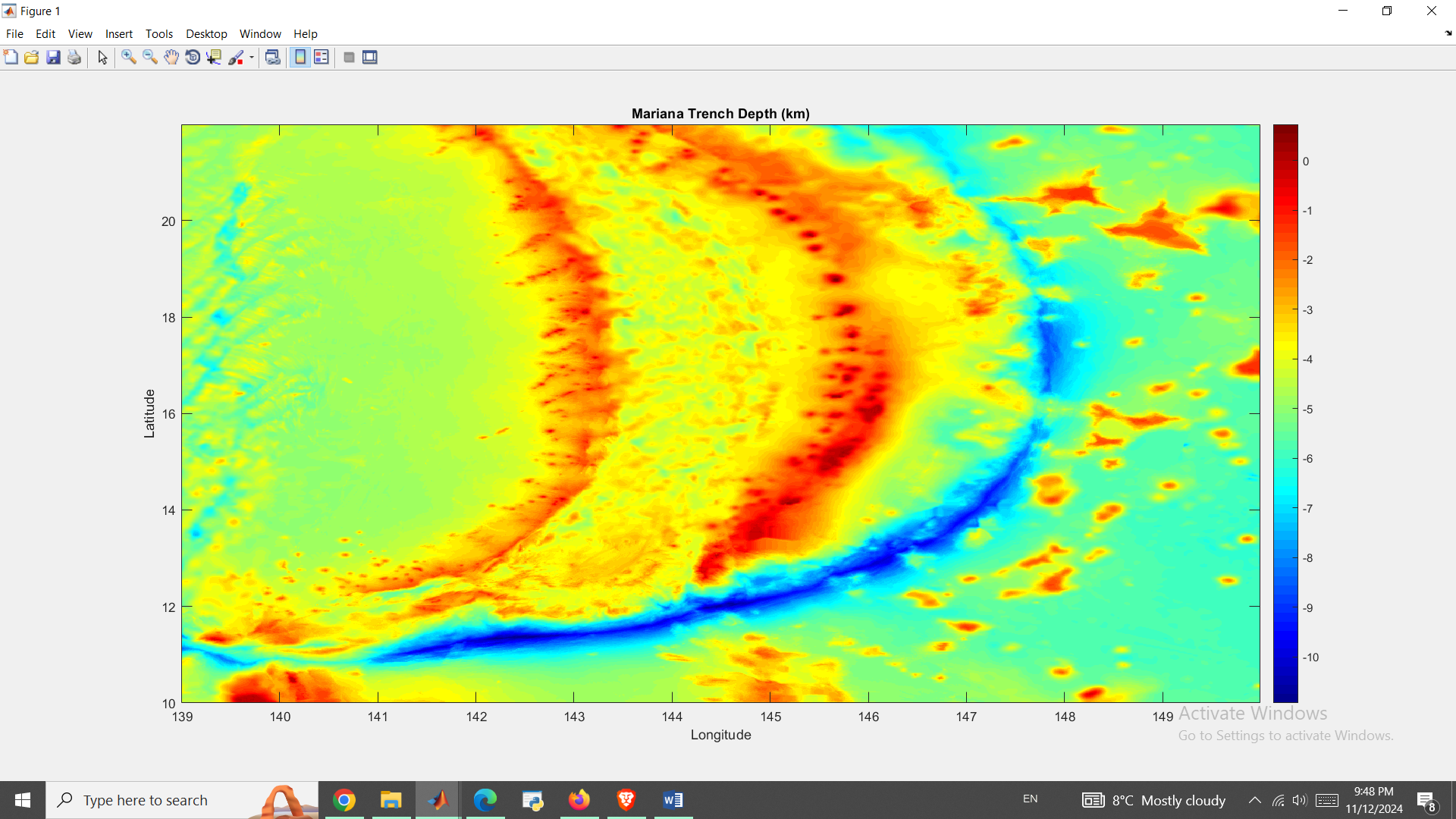
**Introduction**

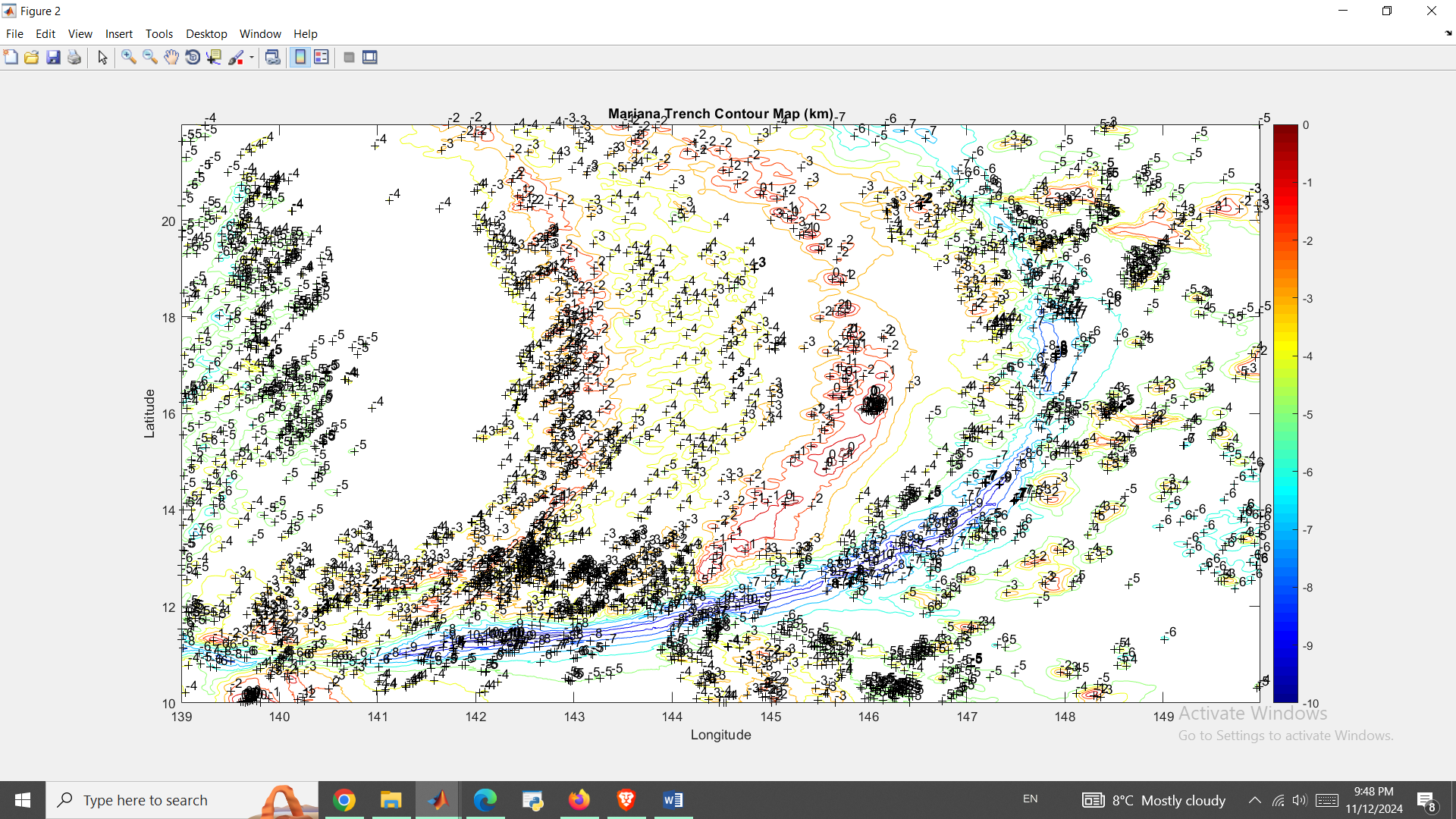
In this report, we examine the “The Mariana Trench “data. The Mariana Trench is the deepest part of the world's oceans which is in the Pacific Ocean between Japan and Papua New Guinea. The United States National Oceanic and Atmospheric Administration (NOAA) has conducted bathymetric investigations of the Mariana Trench and has collected high-resolution data but since data has vast amount of information, it has some challenges in terms of handling and analysis and hence, we want to reduce the size of the dataset while maintaining the overall structure of the trench and to perform scientific analysis using MATLAB. This process will involve many parts such as importing the data, creating visual representations, analyzing the deepest points and employing advanced computational techniques such as eigenvector computation and Singular Value Decomposition (SVD).

**Data import and Visualization**

We start this analysis by importing the data and visualizing it both as an image and a contour map. After importing the depth, latitude and longitude data from CSV files and converting the depth data to kilometers, a meshgrid was created for contour plotting which transposed the depth data to match the grid orientation.

Here are the resulting visualizations:



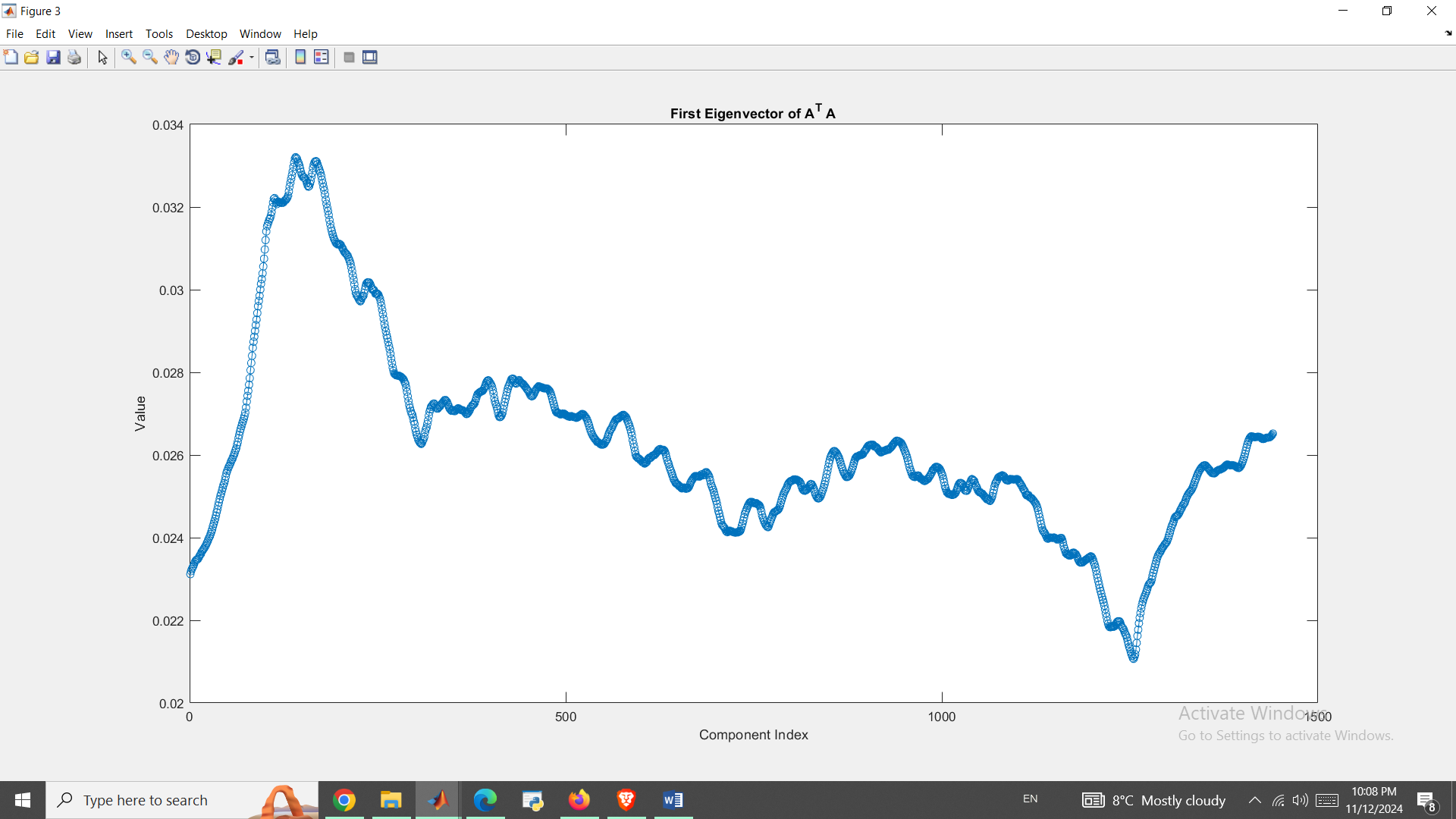


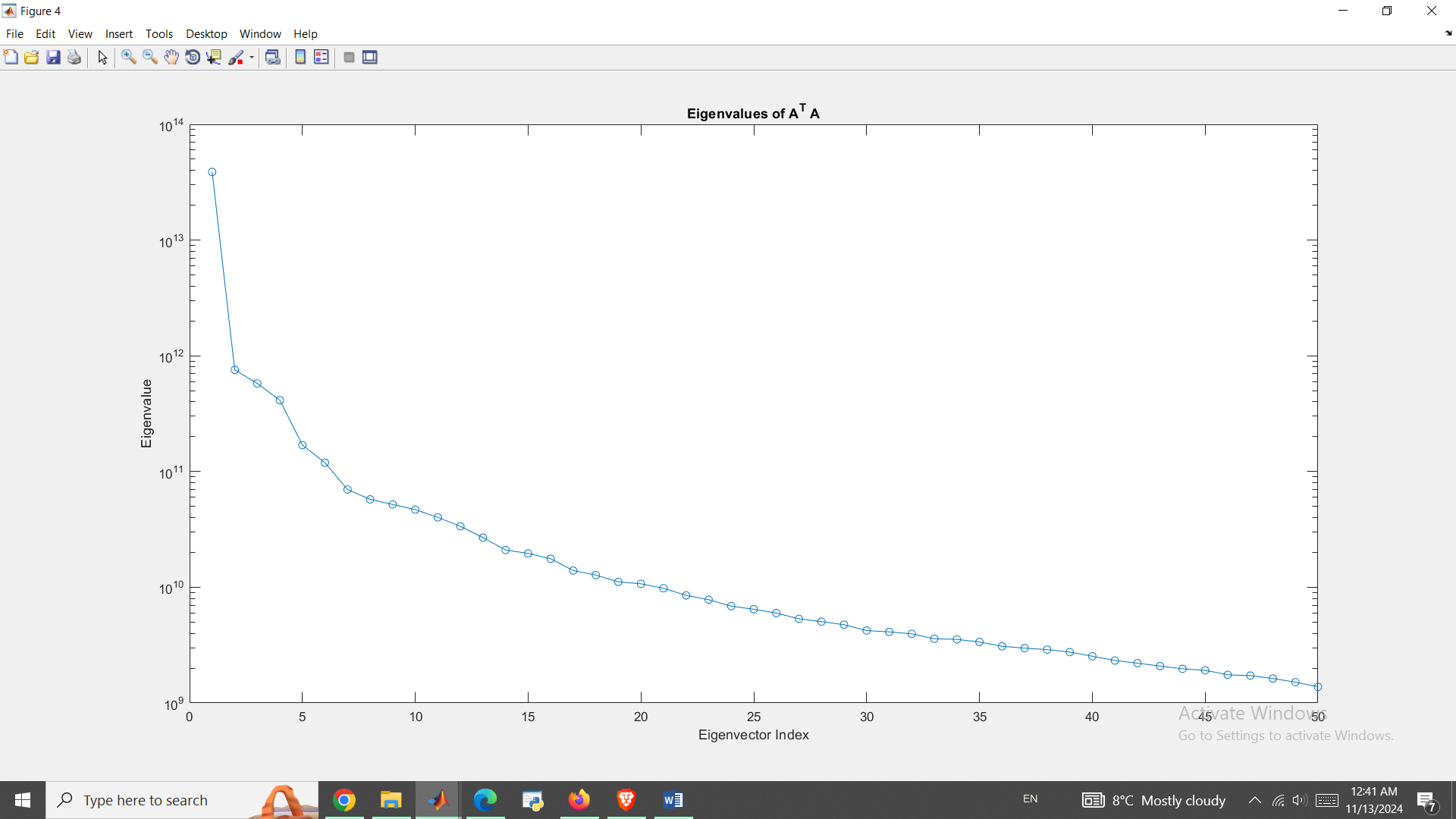
By analyzing the data, we get that the deepest point in the Trench is in depth of 10930.00 meters (or -10.93 km) and is located in Latitude: 13.20 and Longitude: 140.33 (Note that depth means negative sign so depth of 10930 meters is actually -10930 meters)

Defining the ocean floor in this region at a depth of 6 kilometers, we calculated the average depth of the trench below this nominal ocean floor. The mean depth among all points deeper than the ocean floor was determined about -7204.81 meters which shows that the trench has significant depths beyond the nominal ocean floor.

**Eigenvalue and Eigenvector calculation**

In the next step, we developed an algorithm to find the first eigenvector and eigenvalue of ATA, where A denotes the depth matrix of the Mariana Trench. The method involves initializing a random unit vector of magnitude one. and then applying ATA to the vector, then normalize the result by dividing it by its magnitude (to ensure it remains a unit vector) which will update vector for next guess. Process is repeated until the difference between the new and previous vectors is below a set tolerance level.





So, after performing these iterations, we found the first eigenvector and its corresponding eigenvalue.

Note that the plot of the first eigenvector V1 shows the values of its components, indexed from 1 to N.

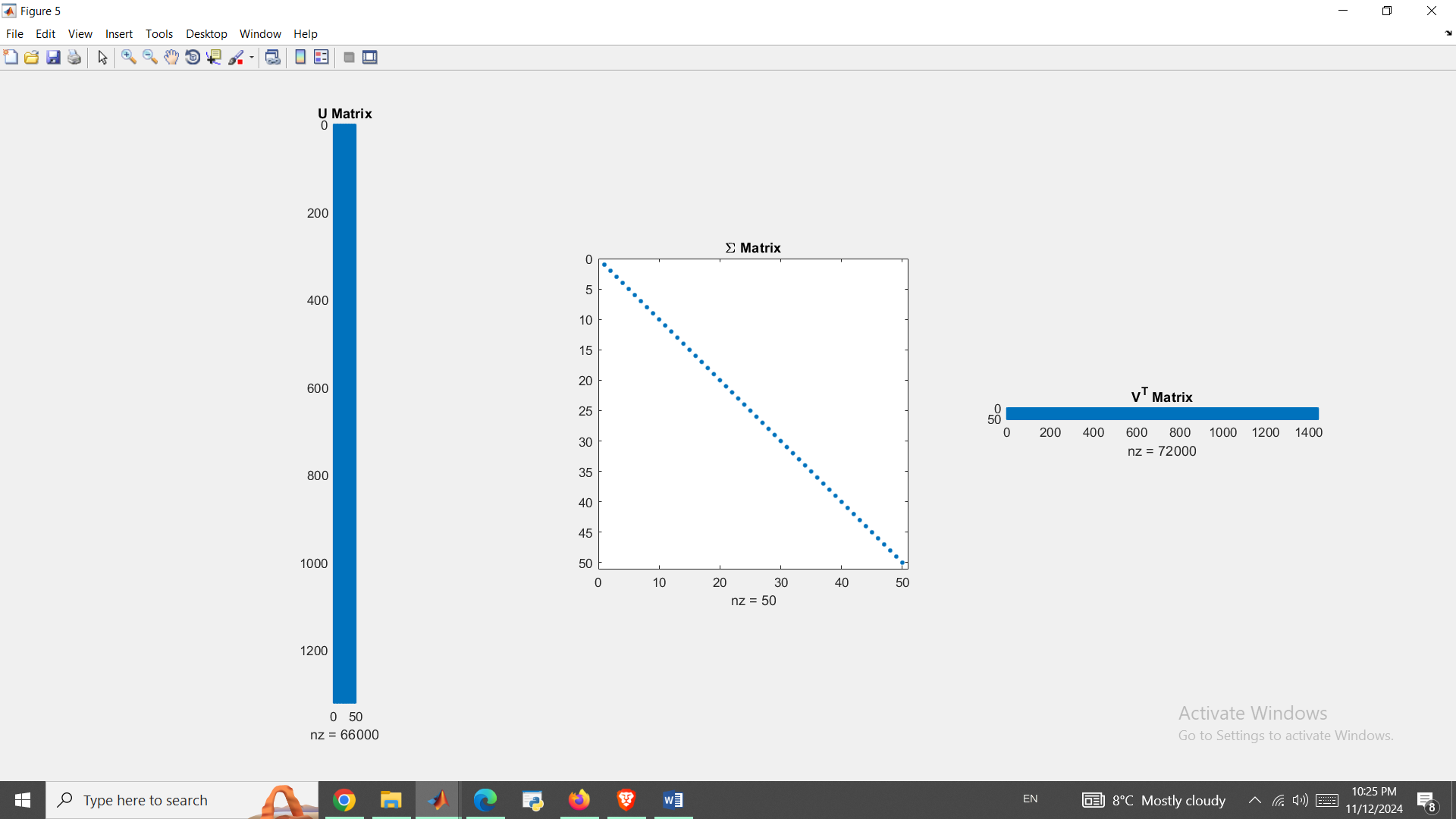
This method of finding the eigenvector works because the eigenvectors of ATA makes a basis of Rn and thus by iteratively applying ATA to a random unit vector, we align the vector with the direction of the highest variance which is the first eigenvector while the repeated normalization make it sure that the vector converges to the true eigenvector.

So, using the Gram-Schmidt orthogonalization process, we computed the 50 largest eigenvalues and their corresponding eigenvectors and we got a semilog plot of the 50 largest eigenvalues which show the magnitude of variance captured by each eigenvector and a matrix V containing the 50 largest eigenvectors as columns.

**Incomplete SVD Decomposition**

In this part, we constructed U, Σ and V matrices for the incomplete Singular Value Decomposition (SVD) of the depth matrix A using the eigenvalues and eigenvectors computed earlier where Σ is a 50×50 diagonal matrix with the square roots of the largest eigenvalues on the diagonal and U is constructed by multiplying A by each column of V and normalizing by the corresponding value in Σ and V Matrix is matrix of the first 50 eigenvectors.

We visualized these matrices using the *spy* command to display their structures.



In order to understand the efficiency of the incomplete SVD, we compared the total number of elements in the U, Σ and V matrices against the original matrix A, where total elements in A was 1,900,800

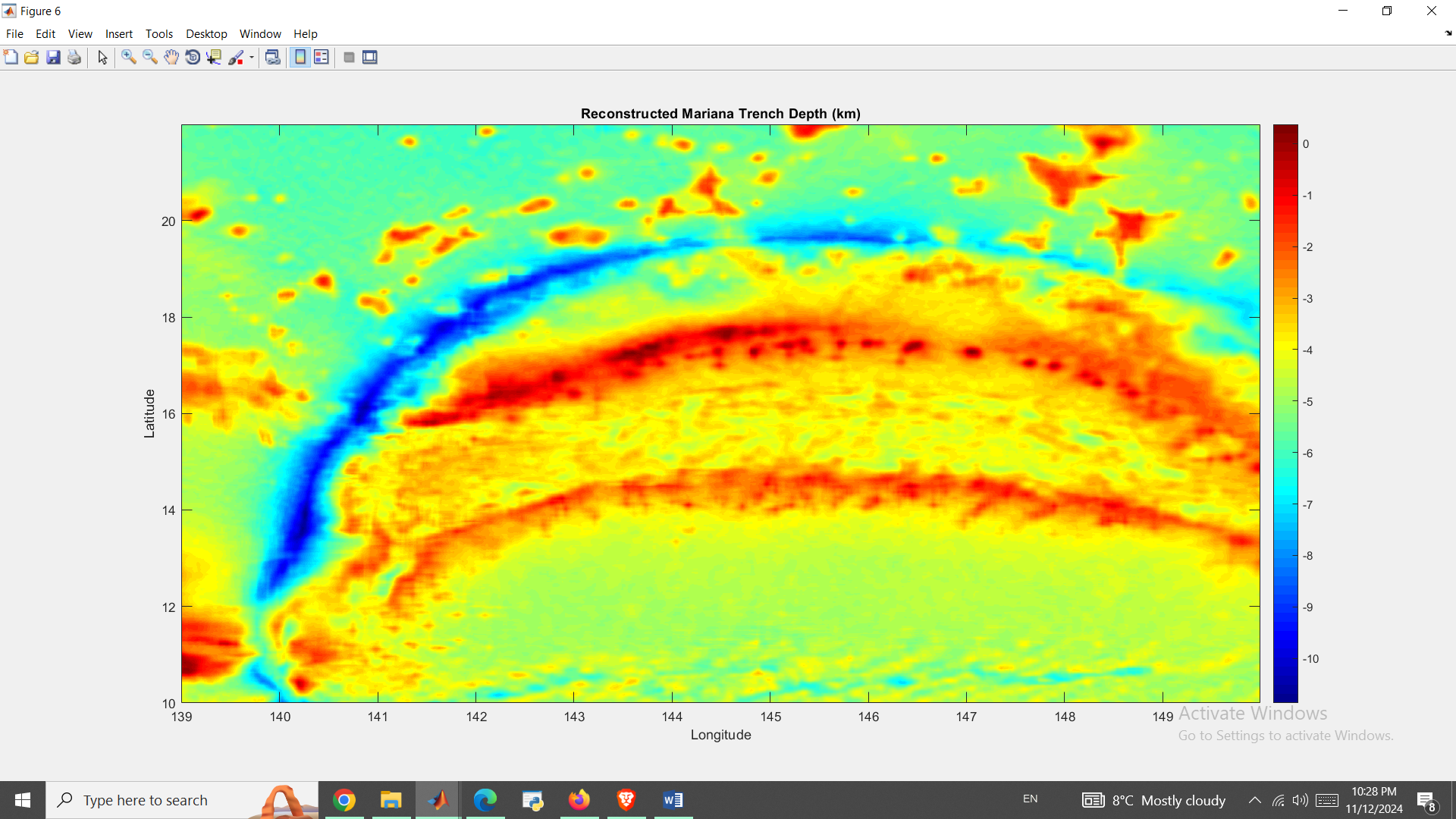
and total elements in U, Σ and V was 140,500.

Similarly, we compared the number of nonzero elements where Nonzero elements in A was 1,900,764 while Nonzero elements in U, Σ and V was 138,050.

This significant reduction in the number of elements demonstrates the efficiency and storage benefits of using SVD.

**Reconstruction of Depth Matrix**

We reconstructed the depth matrix, Aapprox, using the incomplete SVD and visualized it as an image. Here is its graph:



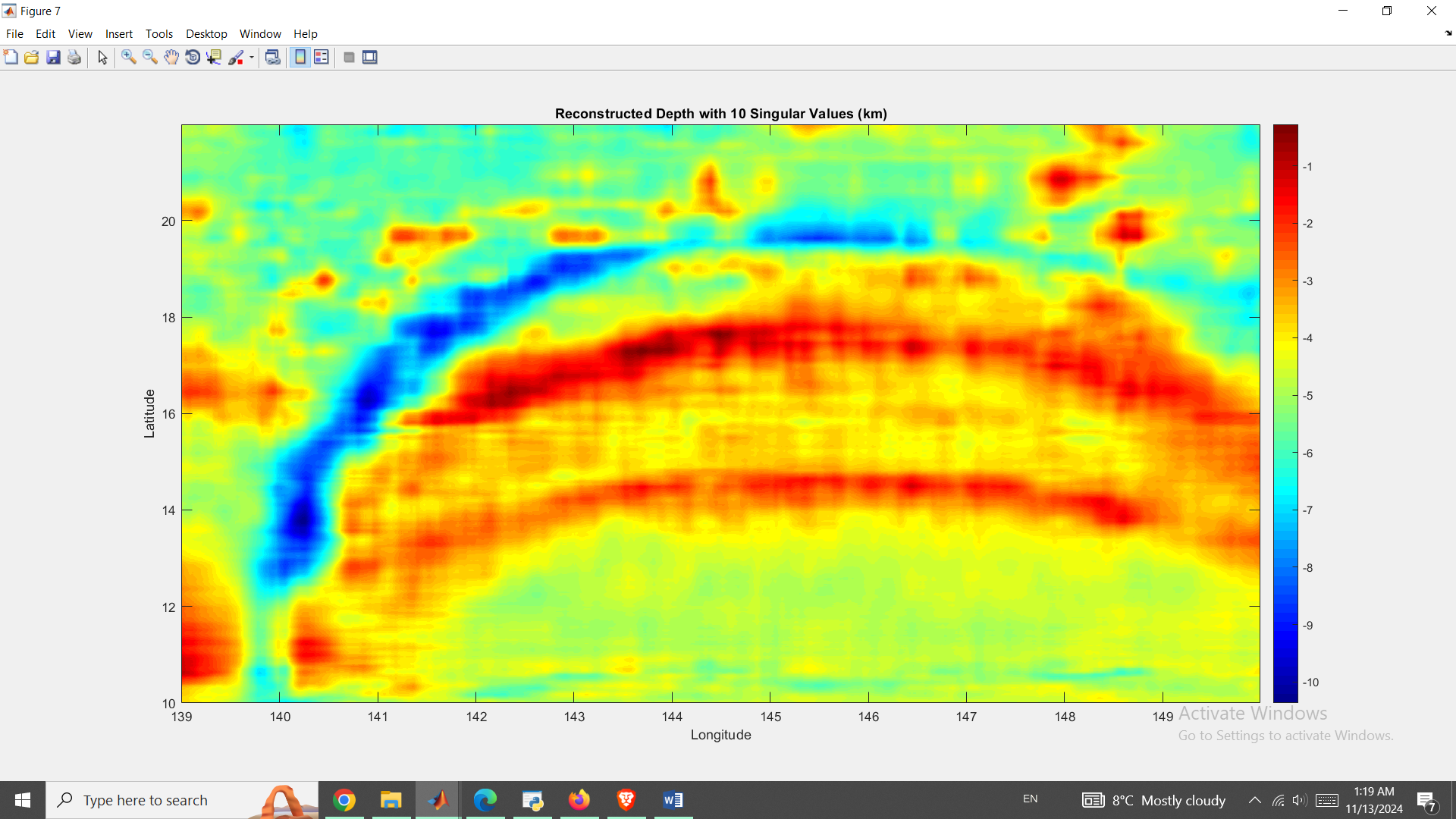
As you can see it is very similar to original depth graph we had earlier which approves our approximation methods.

As for Maximum and average depths in the reconstructed data, the deepest point of reconstructed data is -10,865.01 meters which occurs in Latitude: 13.57 and Longitude: 140.38 that is quite close to what we had for our original data which shows a successful reconstruction.

Also, the Average depth below 6 km of reconstructed data is -4,453.09 meters.

We also experimented reconstruction with using fewer columns like 10 of U and V with a smaller Σ to see how the reconstruction quality holds up. Theoretically, we know that using more singular values results in a closer approximation to the original data while using fewer values reduces storage requirements but may compromise some details.

Result of reconstruction using 10 SV values is shown below:



As you can see, quality is still very high as this depth map is very identical to the original depth dataset we had initially.

So, the essential structure of the trench is preserved but with some loss of detail.

**Conclusion**

In this experiment, Mariana Trench was analyzed using some of computational techniques we learned in this course and MATLAB. We initially showed a visualization of the trench which showed its very deep depths and some of its geographical features through image and contour maps. We located the deepest point in the trench and calculated the average depth below a nominal ocean floor. Then, we computed eigenvectors and eigenvalues to understand the variance and principal components of the depth data and by usage of iterative methods and Gram-Schmidt orthogonalization, we extracted significant eigenvectors which was used in understanding data's dimensionality and reconstruction.

In the last step, we implemented the incomplete Singular Value Decomposition (SVD) of the depth matrix, reducing data size while preserving essential characteristics where by constructing and examining the U, Σ and V matrices, we showed the and accuracy of this data reconstruction.

**APPENDIX**

depth = csvread('mariana\_depth.csv');

latitude = csvread('mariana\_latitude.csv');

longitude = csvread('mariana\_longitude.csv');

% Convert to km

depth\_km = depth / 1000;

longitude = longitude(:);

latitude = latitude(:);

[lonGrid, latGrid] = meshgrid(longitude, latitude);

depth\_km = depth\_km';

figure;

imagesc(longitude, latitude, depth\_km);

colormap('jet');

colorbar;

xlabel('Longitude');

ylabel('Latitude');

title('Mariana Trench Depth Plot');

set(gca, 'YDir', 'normal');

figure;

contour(lonGrid, latGrid, depth\_km, -11:1:11);

clabel(contour(lonGrid, latGrid, depth\_km, -11:1:11));

colormap('jet');

colorbar;

xlabel('Longitude');

ylabel('Latitude');

title('Contour Map of the Mariana Trench');

set(gca, 'YDir', 'normal');

[deepest\_depth, index] = min(depth(:));

[deepest\_lat\_idx, deepest\_lon\_idx] = ind2sub(size(depth), index);

deepest\_latitude = latitude(deepest\_lat\_idx);

deepest\_longitude = longitude(deepest\_lon\_idx);

fprintf('Deepest point is = %.2f meters\n', deepest\_depth);

fprintf('Latitude is = %.2f\n', deepest\_latitude);

fprintf('Longitude is = %.2f\n', deepest\_longitude);

ocean\_floor\_depth = -6000;

ocean\_floor\_depth\_km = -6000/1000;

deeper\_points = depth(depth < ocean\_floor\_depth);

average\_depth = mean(deeper\_points);

fprintf('Average depth below nominal ocean f;loor is = %.2f meters\n', average\_depth);

A = depth;

N = size(A, 2);

u = rand(N, 1);

u = u / norm(u);

tolerance = 1e-6;

difference = inf;

iteration = 0;

while difference > tolerance

iteration = iteration + 1;

u\_new = (A' \* A) \* u;

u\_new = u\_new / norm(u\_new);

difference = norm(u\_new - u);

u = u\_new;

end

V1 = u;

eigenvalue = (u' \* (A' \* A) \* u);

figure;

plot(1:N, V1, '-o');

xlabel('Component Index');

ylabel('Value');

title('First Eigenvector of A^T A');

% Gram-Schmidt

num\_eigenvectors = 50;

V = zeros(N, num\_eigenvectors);

eigenvalues = zeros(num\_eigenvectors, 1);

for i = 1:num\_eigenvectors

u = rand(N, 1);

u = u / norm(u);

difference = inf;

while difference > tolerance

u\_new = (A' \* A) \* u;

for j = 1:i-1

u\_new = u\_new - (u\_new' \* V(:, j)) \* V(:, j);

end

u\_new = u\_new / norm(u\_new);

difference = norm(u\_new - u);

u = u\_new;

end

V(:, i) = u;

eigenvalues(i) = (u' \* (A' \* A) \* u);

end

figure;

semilogy(1:num\_eigenvectors, eigenvalues, '-o');

xlabel('Eigenvector Index');

ylabel('Eigenvalue');

title('Eigenvalues of A^T A (Gram-Schmidt');

Sigma = diag(sqrt(eigenvalues));

U = A \* V;

for i = 1:size(Sigma, 1)

U(:, i) = U(:, i) / Sigma(i, i);

end

VT = V';

figure;

subplot(1, 3, 1);

spy(U);

title('U Matrix');

subplot(1, 3, 2);

spy(Sigma);

title('\Sigma Matrix');

subplot(1, 3, 3);

spy(VT);

title('V^T Matrix');

num\_elements\_A = numel(A);

num\_elements\_U = numel(U);

num\_elements\_Sigma = numel(Sigma);

num\_elements\_V = numel(V);

total\_elements\_SVD = num\_elements\_U + num\_elements\_Sigma + num\_elements\_V;

fprintf('Total elements in A is equal to = %d\n', num\_elements\_A);

fprintf('Total elements in U, Sigma and V is equal to = %d\n', total\_elements\_SVD);

nnz\_A = nnz(A);

nnz\_U = nnz(U);

nnz\_Sigma = nnz(Sigma);

nnz\_V = nnz(V);

total\_nnz\_SVD = nnz\_U + nnz\_Sigma + nnz\_V;

fprintf('Nonzero elements in A is = %d\n', nnz\_A);

fprintf('Nonzero elements in U, Sigma and V = %d\n', total\_nnz\_SVD);

A\_approx = U \* Sigma \* VT;

figure;

imagesc(longitude, latitude, A\_approx / 1000);

colormap('jet');

colorbar;

xlabel('Longitude');

ylabel('Latitude');

title('Reconstructed depth plot of Mariana Trench');

set(gca, 'YDir', 'normal');

[deepest\_depth\_approx, index\_approx] = min(A\_approx(:));

[deepest\_lat\_idx\_approx, deepest\_lon\_idx\_approx] = ind2sub(size(A\_approx), index\_approx);

deepest\_latitude\_approx = latitude(deepest\_lat\_idx\_approx);

deepest\_longitude\_approx = longitude(deepest\_lon\_idx\_approx);

fprintf('Deepest point (reconstructed) is = %.2f meters\n', deepest\_depth\_approx);

fprintf('Latitude is = %.2f\n', deepest\_latitude\_approx);

fprintf('Longitude is = %.2f\n', deepest\_longitude\_approx);

deeper\_points\_approx = A\_approx(A\_approx < -ocean\_floor\_depth);

average\_depth\_approx = mean(deeper\_points\_approx);

fprintf('Average depth below ocean floor in reconstructed form is =: %.2f meters\n', average\_depth\_approx);

k = 10;

U\_k = U(:, 1:k);

Sigma\_k = Sigma(1:k, 1:k);

VT\_k = VT(1:k, :);

A\_approx\_k = U\_k \* Sigma\_k \* VT\_k;

figure;

imagesc(longitude, latitude, A\_approx\_k / 1000);

colormap('jet');

colorbar;

xlabel('Longitude');

ylabel('Latitude');

title('Reconstructed Depth with 10 Singular Values plot');

set(gca, 'YDir', 'normal');