Analyzing the Mariana Trench

By Vaami Satrasala and Benjamin Jardine APPM 2360 - Project 2 3/31/2026

1 Introduction

The Mariana Trench is a fascinating biosphere, full of wild flora, fauna, and various structures that are of significant interest to researchers. We have been given cleaned data on the dimensions of the trench in terms of latitude, longitude, and depth, and been told to reduce it to an appropriate size for analysis without compromising the described structure of the trench. We also have performed some analysis of the data in the form of contour mapping for the researchers.

To do this, we created an Incomplete Singular Value Decomposition (henceforth referred to as ISVD). The Singular Value Decomposition (SVD) of an mxn matrix allows one to decompose it into 3 smaller matrices of sizes mxm, mxn, and nxn, which can be multiplied together to form the original matrix. An ISVD is an SVD of a matrix wherein only the data associated with the 'k' number of largest eigenvalues is retained, resulting in mxk, kxk, and kxn matrices. The goal of this process is to create an ISVD that removes data that doesn't provide much new information about a dataset, thus compressing the original dataset to a much smaller size without losing key features.

To obtain our results, we used MATLAB to execute and document all the necessary steps of creating an ISVD of the original dataset to resize it. Additionally, we performed contour mapping of the original dataset that we were given and of the resized datasets that we created.

2 Investigating the Mariana Trench

The researchers wanted us first to create a contour map of the original cleaned dataset that they gave us and report some statistics for the data set. We used MATLAB's contour mapping features to create a 3D visualization of the trench's varying depths, as well as a 2D projection of the trench using a color gradient to indicate depth.

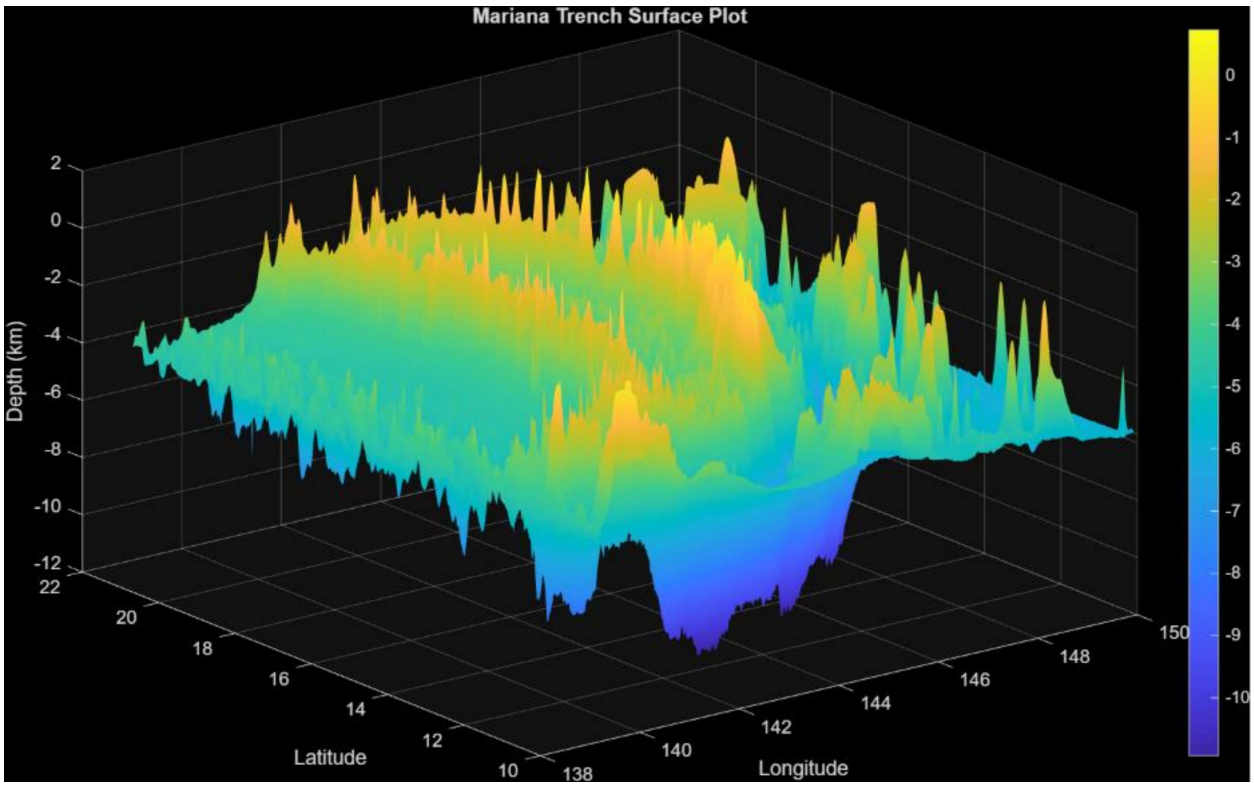

Figure 1 – 3D Contour Plot

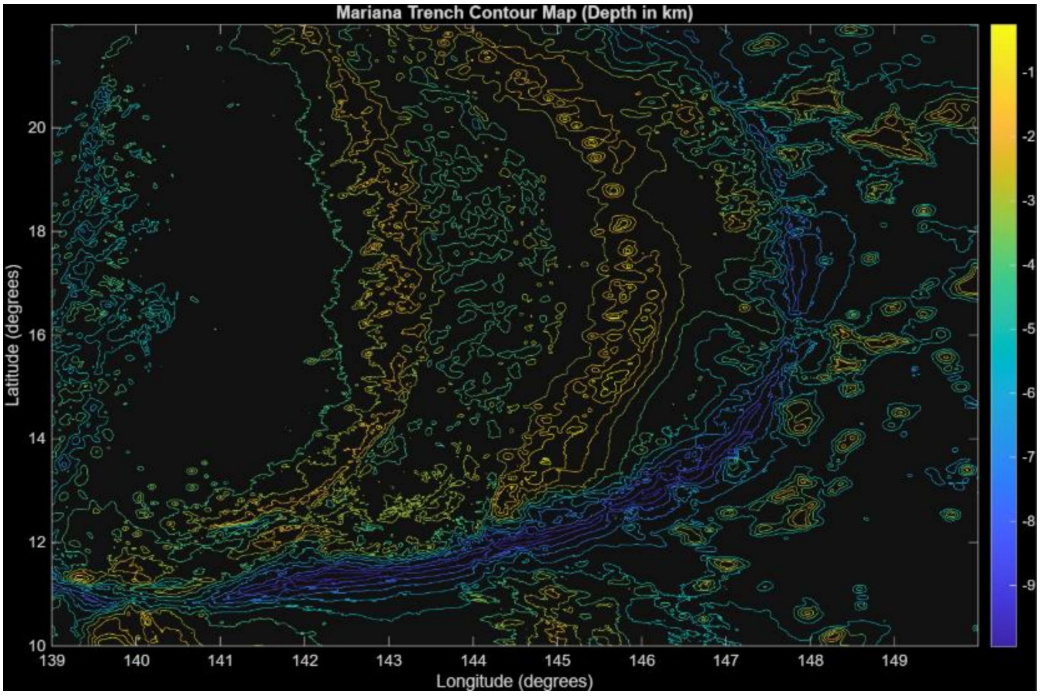

Figure 2 – 2D Contour Plot

The deepest point in the trench lies at a depth of about -10.9300 kilometers at a latitude of 11.33 ° N and a longitude of 142.2 ° E. The trench's average depth below the ocean floor (that is, the average depth below 6 kilometers below the ocean floor) was about -7.2048 kilometers.

In order to achieve an ISVD of the dataset to compress it, we need to, first, find the first 'k' number of largest eigenvalues of the depth matrix from the data we were given, where 'k' depends on how large of a compressed dataset we want to end up with. For the purposes of our analysis, we initially chose a target of 50 eigenvalues. To find these eigenvalues, we chose to use the Power Method.

We first tested finding one eigenvalue and the associated eigenvector of matrix $B = A^T \cdot A$to make sure our method would work. The largest eigenvalue that we found was

$3.8803 \cdot 10^{13}$, which is consistent with the largest eigenvalue that we found using MATLAB's 'eigs' command, so we know that our code works for this purpose. We also plotted the associated eigenvector:

Figure 3 – Eigenvector Plot for Matrix A for the Largest Eigenvalue

We hypothesize that the Power Method for finding the largest eigenvalue and its associated eigenvector works because as we scale the unit vector by $A^T \cdot A$, which is a combination of the components of the basis, it eventually will converge to the largest eigenvector, and then we divide the eigenvector by $A^T \cdot A$ to get the eigenvalue.

Once we determined that our code functioned as intended, we proceeded to find the 50 largest eigenvalues of matrix *B*. To do this, we used Gram-Schmidt Orthogonalization with the Power Method, ensuring that every new eigenvalue that we got

from the power method was the next largest one. We also stored the eigenvectors in matrix V.

To ensure that the first 50 eigenvalues were of decreasing magnitude, we plotted them, using a semilog plot to manage the rapid decrease in magnitude of the values:

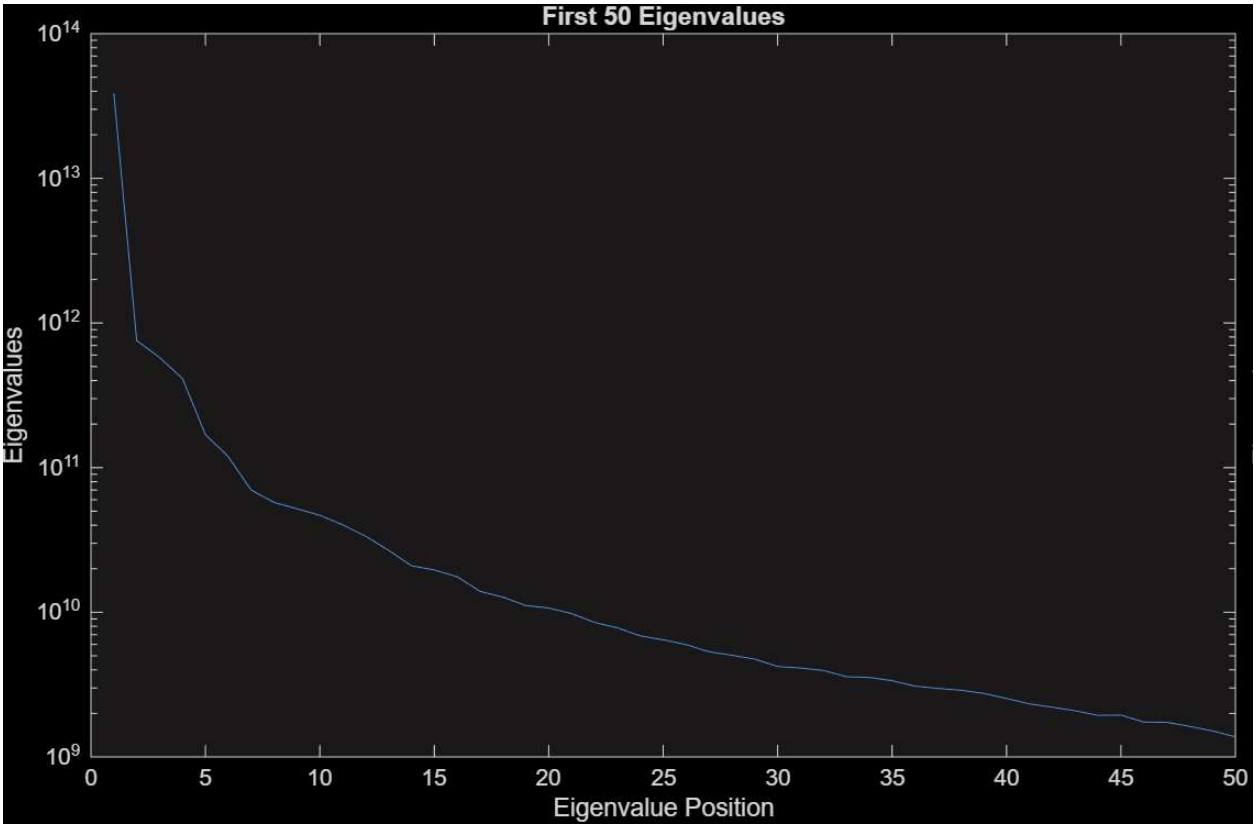

Figure 4 – Semilog Plot of the Largest 50 Eigenvalues Matrix A

To compress the data we were given, we resized the matrices U, V, and Σ that compose the original depth matrix A to only contain data associated with the 50 largest eigenvalues; we essentially 'cut off' any data that was associated with the eigenvalues beyond the 50 largest ones. To visualize, we have provided figures of the resized matrices U, V, and Σ:

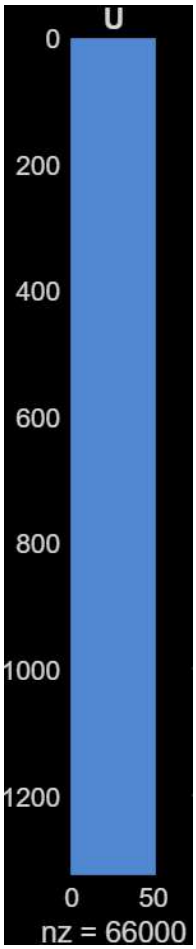

Figure 5 – Matrix U

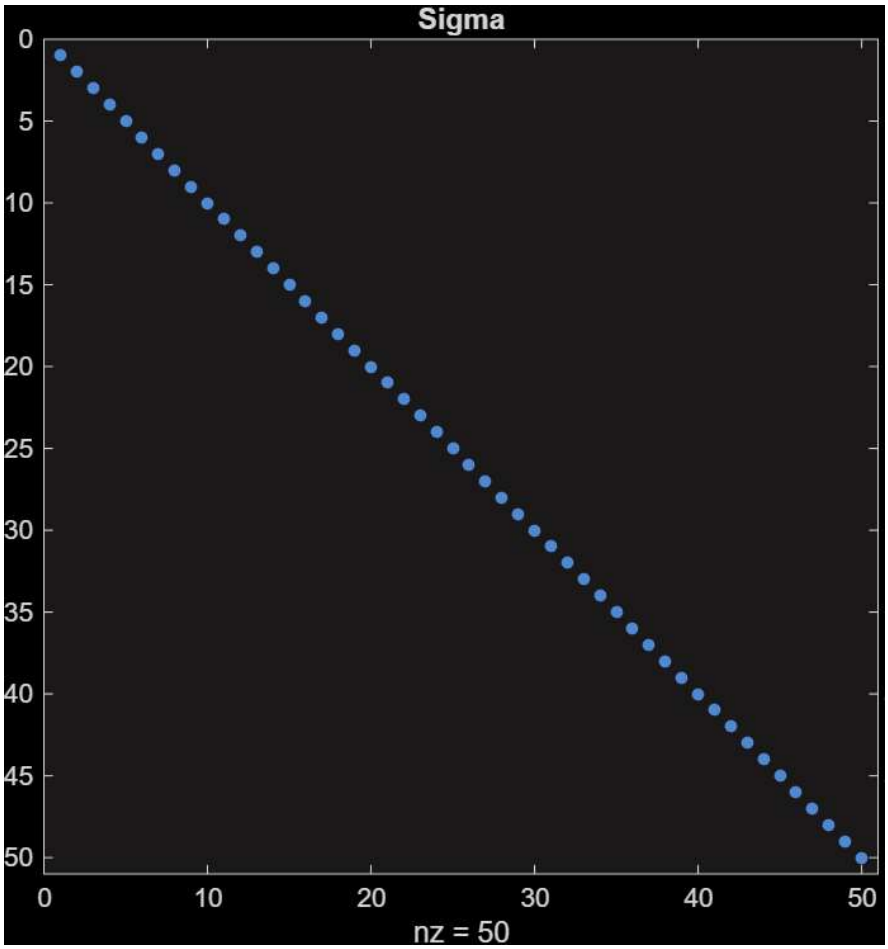

Figure 6 – Matrix Σ

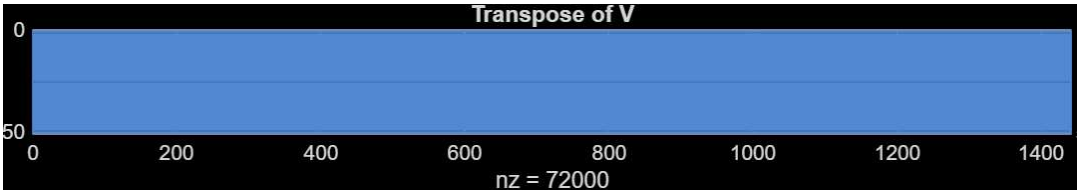

Figure 7 – Matrix V

Quantitatively, the difference in data storage between the original depth matrix and the resized matrix can be calculated. The number of elements in Matrix A vs the ISVD was

1,900,800 vs 140,500, with a difference of 1,760,300 elements (~13.5 times smaller). Regarding solely non-zero elements, the number of elements in Matrix A vs the ISVD was 1,900,764 vs 138,050, with a difference of 1,762,714 elements (~13.8 times smaller). This confirms that we successfully compressed the dataset, such that we only need to store a far smaller quantity of stored values.

To illustrate the benefits of compressing the dataset, we have created new contour maps using the data associated with the 50 largest eigenvalues:

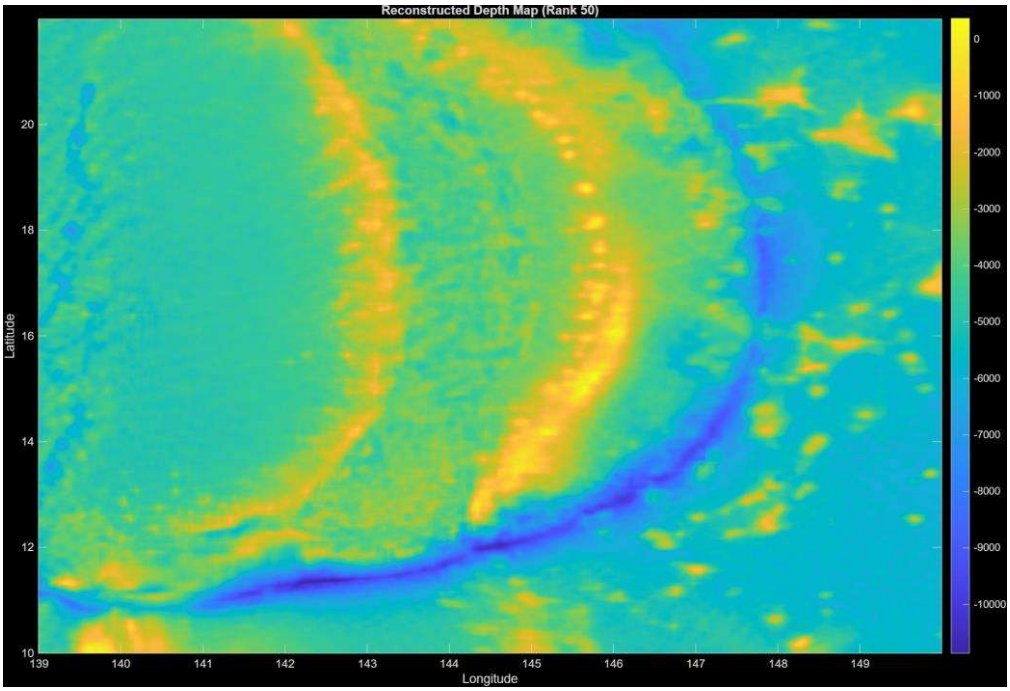

Figure 8 – 3D Contour Plot (Rank 50)

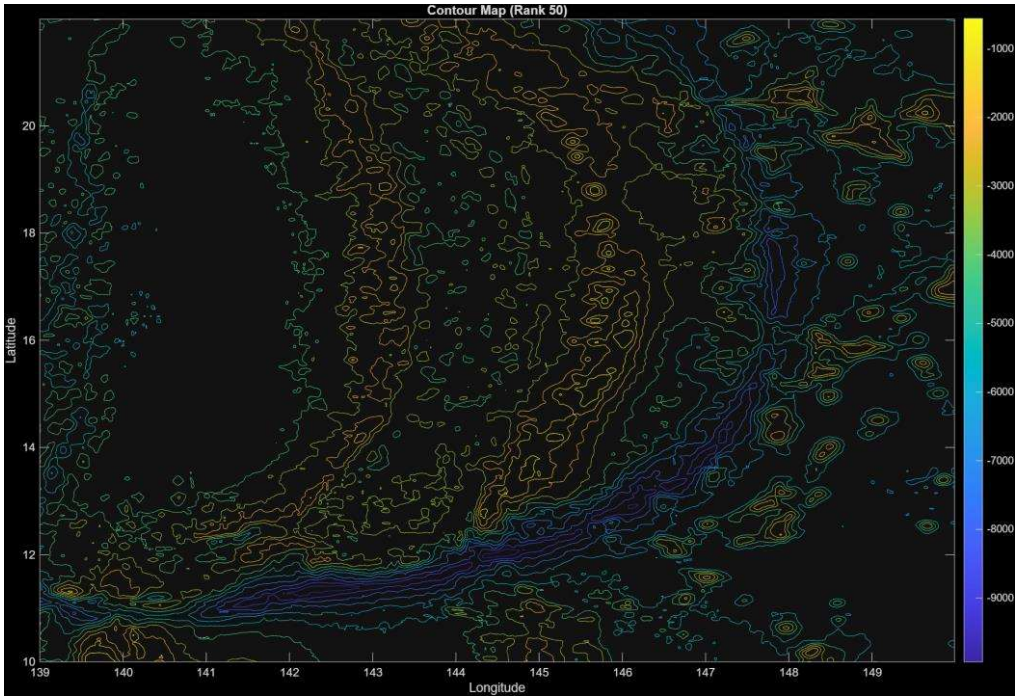

Figure 9 – 2D Contour Plot (Rank 50)

As you can see, when comparing this new contour map and the original contour map for the full dataset, the majority of the features are retained mostly accurately. The broad structure of the map is virtually identical, with the only differences being small changes in the 'wobbliness' of the lines. This implies that we can compress large datasets of topographical data to much smaller sizes without losing much key ideas.

The deepest depth of the reconstructed dataset is -10.8659km, and the mean depth of the new data set is -7.1745. If we compare these to the original values (max depth of -10.9300 and mean depth of -7.2048), we can observe that the values remain similar, lending further credence to the idea that data such as this can be compressed without losing much accuracy.

We also redid our procedure for the 100 largest and 10 largest eigenvalues:

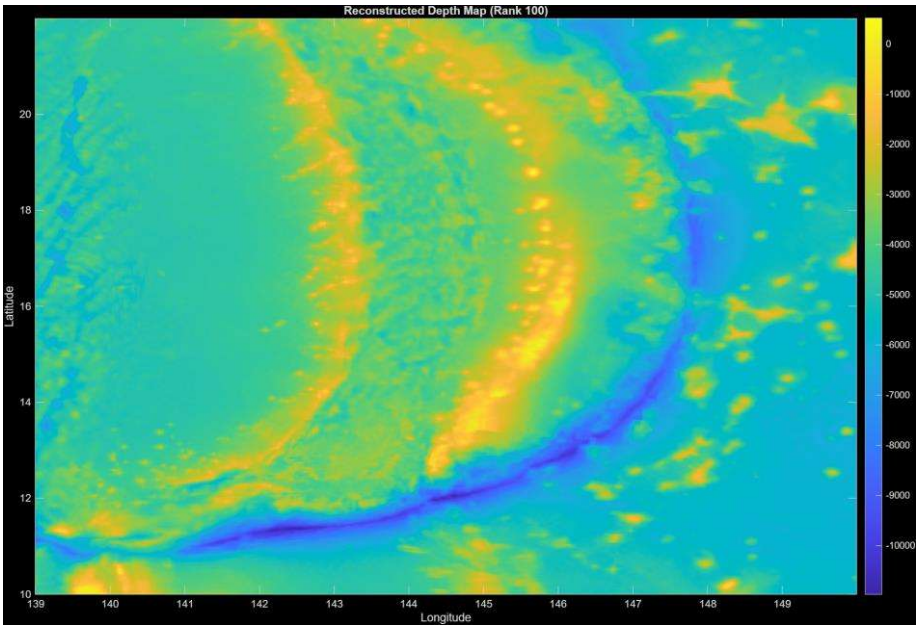

Figure 10 – 3D Contour Plot (Rank 100)

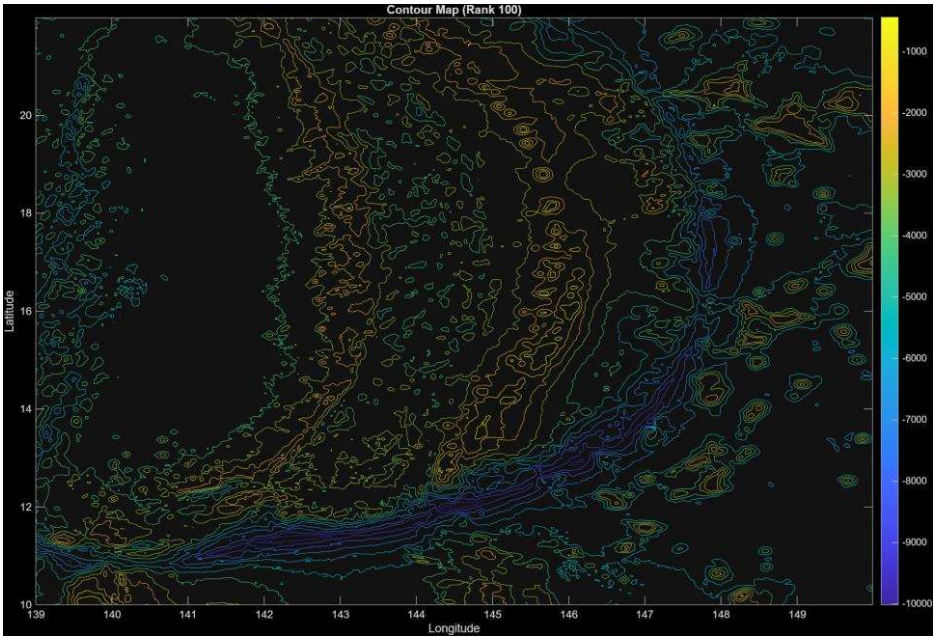

Figure 11 – 2D Contour Plot (Rank 100)

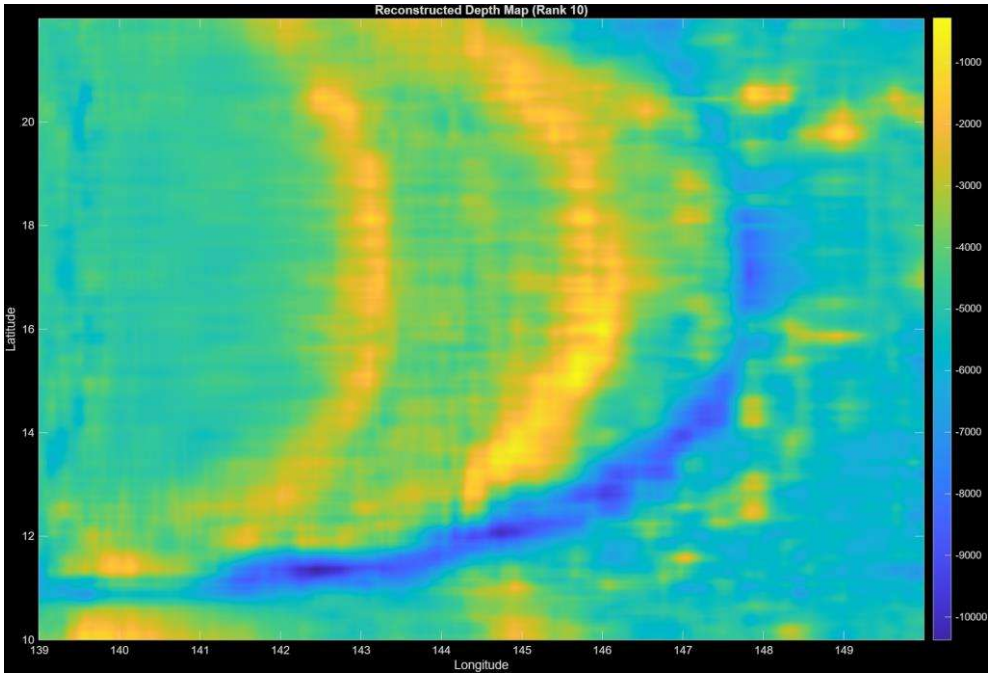

Figure 12 – 3D Contour Plot (Rank 10)

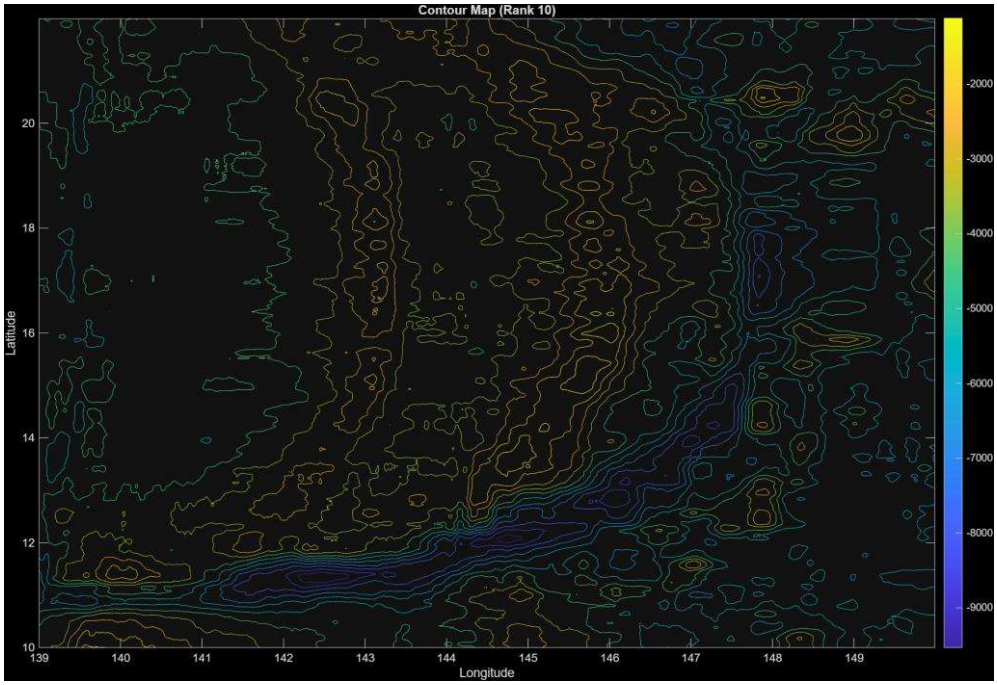

Figure 13 – 2D Contour Plot (Rank 10)

As you can see, the difference between using 50 and 100 values is fairly negligible, however the difference between using 50 and 10 values is profound; 10 values lost much key data. This implies that there are reasonable limits on the level of compression that can be used without compromising the key features of the original dataset. Further investigation on finding a way to quantify and therefore find the tolerable limits of such degradation of data may be warranted. Something interesting to note is that the mean depth of the rank 10 plot is -7.0496 and the deepest depth is -10.3727, while on the other hand, the mean depth of the rank 100 plot is –7.1973, and the deepest being -10.9698. The rank 100 values seem to overshoot the magnitudes of the original values, while rank 10 compression seems to undershoot them.

Conclusion:

We were able to successfully compress the dataset of the topographical data of the Mariana Trench, such that we retained most of the key features of the trench, as well as fairly accurate statistics describing it. This implies that using an ISVD to compress topographical data is a worthwhile measure to decrease the amount of storage and processing power necessary to analyze large datasets.

In this instance, we found that scaling down to using the 50 largest eigenvalues of the depth data was sufficient to not compromise the key features of the trench. However, scaling down to 10 significantly degraded the visual representation. This implies to us that there exists a reasonable threshold of compression that we can feasibly use without

affecting key values. These techniques can be used in streaming, LLM, or file transfer systems where a lot of packets are processed at once, significantly decreasing the size of the data that is being processed while still allowing the user to see close to a complete replication of the original file or idea being presented.

A MATLAB Code

2.1.1, 2.1.2, 2.1.3

```
%References%
%https://www.geeksforgeeks.org/software-engineering/matlab-syntax/%
%https://www.mathworks.com/help/matlab/ref/contour.html%
%https://www.mathworks.com/matlabcentral/answers/514760-create-contour-map-from-
latitude-longitude-and-depth-from-csv-file
%https://www.mathworks.com/help/matlab/ref/for.html
%https://www.geeksforgeeks.org/software-engineering/how-to-iterate-through-each-
element-in-n-dimensional-matrix-in-matlab/
%https://www.mathworks.com/help/matlab/ref/surf.html
%https://www.mathworks.com/help/matlab/ref/double.min.html
%https://www.mathworks.com/matlabcentral/answers/365256-convert-matrix-in-single-
column-row-vector
marianadepth_data = readtable('data\mariana_depth.csv'); % depth csv
marianalat_data = readtable('data\mariana_latitude.csv'); % lat csv
marianalong_data = readtable('data\mariana_longitude.csv'); % long csv
depth = table2array(marianadepth_data) * (1/1000); % meters → km
lat = table2array(marianalat_data); % lat data in degrees
long = table2array(marianalong_data); %long data in degrees
[LONG, LAT] = meshgrid(long, lat); %make a matrix with long as the x coord and lat as
the y coord
DEPTH = depth'; %transpose the depth matrix so it fits with the lat/long meshgrid
figure %Figure 1 (Contour Map)
% Contour map
contour(LONG, LAT, DEPTH, 11) % Contours Corresponding -11 to 11 km
colorbar
xlabel('Longitude (degrees)')
ylabel('Latitude (degrees)')
title('Mariana Trench Contour Map (Depth in km)')
figure %Figure 2, 3d Surface Plot
% 3D surface plot
surf(LONG, LAT, DEPTH)
colorbar
xlabel('Longitude')
ylabel('Latitude')
zlabel('Depth (km)')
title('Mariana Trench Surface Plot')
shading interp
deepestDepth = min(depth(:));
disp(deepestDepth);
meanDepth = mean(DEPTH(DEPTH < -6));
disp(meanDepth)
```

2.2.1

```
% 2.2 Part 2
A = importdata('mariana_depth.csv');
lat = importdata('mariana_latitude.csv');
lon = importdata('mariana_longitude.csv');
AT = transpose(A); %transpose of A
B=AT*A;
V = zeros(1440,50);
lambdas=[];
tol = 0.001;
for i = 1:50;
      u = rand(1440,1);
      u = u/norm(u);
      for n = 1:100;
            u_n = B*u;
            for j = 1:i-1
                  u_n = u_n - (transpose(u_n)*V(:,j))*V(:,j);
            end
            u_n1 = u_n/norm(u_n);
            if norm(u_n1-u)<0.001
                  break
            end
            u=u_n1;
      end
      l = (transpose(u)*B*u)/(transpose(u)*u);
      lambdas = [lambdas, l];
      V(:, i) = u;
end
semilogy(lambdas)
xlabel("Eigenvalue Position")
ylabel("Eigenvalues")
title("First 50 Eigenvalues")
```

2.2.2

```
% 2.2 Part 1
A = importdata('mariana_depth.csv');
lat = importdata('mariana_latitude.csv');
lon = importdata('mariana_longitude.csv');
AT = transpose(A); %transpose of A
B=AT*A;
v = ones(1440,1); %creation of v, a vector of size 1320x1, consisting entirely of
ones
v = v/norm(v); %making v's magnitude = 1
for i = 1:10
      v = (B*v);
      v = v/norm(v);
end
lambda = (transpose(v)*B*v)/(transpose(v)*v); %this is the eigenvalue
disp("lambda =")
disp(lambda)
x = 1:1440; %making a vector x of the same size as v1
plot(x,v,"b") %plotting v (the eigenvector) vs x
```

```
% 2.3 Part 1
%problem 1
A = importdata('data\mariana_depth.csv');
lat = importdata('data\mariana_latitude.csv');
lon = importdata('data\mariana_longitude.csv');
AT = transpose(A); %transpose of A
B=AT*A;
V = zeros(1440,50);
lambdas=[];
tol = 0.001;
for i = 1:50;
      u = rand(1440,1);
      u = u/norm(u);
      for n = 1:100;
            u_n = B*u;
            for j = 1:i-1
                  u_n = u_n - (transpose(u_n)*V(:,j))*V(:,j);
            end
            u_n1 = u_n/norm(u_n);
            if norm(u_n1-u)<0.001
            break
            end
            u=u_n1;
      end
      l = (transpose(u)*B*u)/(transpose(u)*u);
      lambdas = [lambdas, l];
      V(:, i) = u;
end
Sigma = zeros(50,50);
for i = 1:50
      Sigma(i,i) = sqrt(lambdas(1,i));
end
U = [];
for i = 1:50;
      U = [U,((A*V(:,i))/Sigma(i,i))];
end
figure
spy(U)
figure
spy(Sigma)
figure
VT = transpose(V);
spy(VT)
%problem2
disp("Number of Elements in Matrix A")
```

```
numel(A)
disp("Total Elements in the incpmplete SVD decomposition")
totalElements = numel(U) + numel(Sigma) + numel(V)
disp("Difference in the Total Amount of Stored Elements")
spacesaved1 = numel(A) - totalElements
disp("Number of Nonzero Elements in Matrix A")
nnz(A)
disp("Total Nonzero Elements in the incpmplete SVD decomposition")
totalNonZ = nnz(U) + nnz(Sigma) + nnz(V)
disp("Difference in the Amount of Stored Nonzero Elements")
spacesaved2 = nnz(A) - totalNonZ
A_0 = U*Sigma*VT;
figure
imagesc(lon, lat, A_0')
set(gca, 'YDir', 'normal')
colorbar
title('Reconstructed Depth Map (Rank 50)')
xlabel('Longitude')
ylabel('Latitude')
figure
contour(lon, lat, A_0', 11)
colorbar
title('Contour Map (Rank 50)')
xlabel('Longitude')
ylabel('Latitude')
```

2.2.4

```
% 2.3 Part 1 WITH 10
%problem 1
A = importdata('data\mariana_depth.csv');
lat = importdata('data\mariana_latitude.csv');
lon = importdata('data\mariana_longitude.csv');
AT = transpose(A); %transpose of A
B=AT*A;
V = zeros(1440,10);
lambdas=[];
tol = 0.001;
for i = 1:10;
      u = rand(1440,1);
      u = u/norm(u);
      for n = 1:100;
            u_n = B*u;
            for j = 1:i-1
                  u_n = u_n - (transpose(u_n)*V(:,j))*V(:,j);
            end
            u_n1 = u_n/norm(u_n);
            if norm(u_n1-u)<0.001
                  break
            end
            u=u_n1;
      end
      l = (transpose(u)*B*u)/(transpose(u)*u);
      lambdas = [lambdas, l];
      V(:, i) = u;
end
Sigma = zeros(10,10);
for i = 1:10
      Sigma(i,i) = sqrt(lambdas(1,i));
end
U = [];
for i = 1:10;
      U = [U,((A*V(:,i))/Sigma(i,i))];
end
figure
spy(U)
figure
spy(Sigma)
figure
VT = transpose(V);
spy(VT)
%
A_0 = U*Sigma*VT;
```

```
%now map A_0 against lat and long?
figure
imagesc(lon, lat, A_0')
set(gca, 'YDir', 'normal')
colorbar
title('Reconstructed Depth Map (Rank 50)')
xlabel('Longitude')
ylabel('Latitude')
figure
contour(lon, lat, A_0', 11)
colorbar
title('Contour Map (Rank 50)')
xlabel('Longitude')
ylabel('Latitude')



% 2.3 Part 1 WITH 100
%problem 1
A = importdata('data\mariana_depth.csv');
lat = importdata('data\mariana_latitude.csv');
lon = importdata('data\mariana_longitude.csv');
AT = transpose(A); %transpose of A
B=AT*A;
V = zeros(1440,100);
lambdas=[];
tol = 0.001;
for i = 1:100;
      u = rand(1440,1);
      u = u/norm(u);
      for n = 1:100;
            u_n = B*u;
            for j = 1:i-1
                  u_n = u_n - (transpose(u_n)*V(:,j))*V(:,j);
            end
            u_n1 = u_n/norm(u_n);
            if norm(u_n1-u)<0.001
                  break
            end
            u=u_n1;
      end
      l = (transpose(u)*B*u)/(transpose(u)*u);
      lambdas = [lambdas, l];
      V(:, i) = u;
end
Sigma = zeros(100,100);
for i = 1:100
      Sigma(i,i) = sqrt(lambdas(1,i));
```

```
end
U = [];
for i = 1:100;
      U = [U,((A*V(:,i))/Sigma(i,i))];
end
figure
spy(U)
figure
spy(Sigma)
figure
VT = transpose(V);
spy(VT)
%
A_0 = U*Sigma*VT;
%now map A_0 against lat and long?
figure
imagesc(lon, lat, A_0')
set(gca, 'YDir', 'normal')
colorbar
title('Reconstructed Depth Map (Rank 50)')
xlabel('Longitude')
ylabel('Latitude')
figure
contour(lon, lat, A_0', 11)
colorbar
title('Contour Map (Rank 50)')
xlabel('Longitude')
ylabel('Latitude')
```