APPM 2360 Fall 2024

Project 2

Down The Hatch: Diving Deep into the Mariana Trench

By.

Noah Isakson

Cameron Mars

Jerry Vanim-Botting

1. Introduction

Cradling the Mariana archipelago, the similarly named Mariana Trench is the deepest oceanic trench on the planet. With a maximum depth below sea level that exceeds Mt. Everest's height above it, it is no wonder why this trench has captured the interests of many. The purpose of this report is to utilize linear algebra techniques in reducing the size of the existing large data sets collected by the National Oceanic Atmospheric Administration (NOAA) and to use both of these data sets to perform some basic scientific analysis. To do this, after exploring the raw data, we will be performing an incomplete singular value decomposition to reduce the size of the data while maintaining its integrity.

2. Exploring The Trench

To begin, we have imported the NOAA data on the depths of the trench and surrounding area and processed it into a contour map, Fig 1, to better visualize the data (Appendix A, Section 1). The trench itself, shown as a surface plot and contour map in Fig 1. below, has a maximum depth of approximately 10,930 meters below sea level, located at 11°19’48” N, 142°12’E in an area known as the Challenger Deep. If we define the ocean floor in this area to be at a depth of 6000 meters below sea level, the start of the Hadopelagic oceanic zone, and anything below that to be a part of the trench, we can calculate an average depth of the trench. We do this by taking the mean of all values greater than 6000 in the set to find that the average depth of the trench as we have defined it is approximately 7.2 kilometers (Appendix A, Section 1).

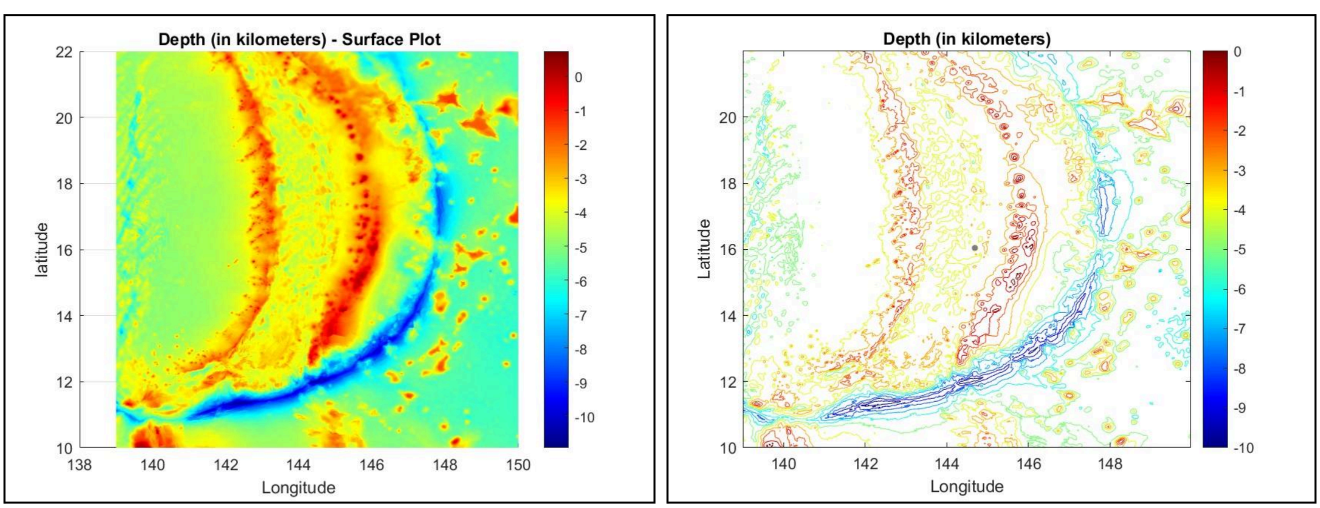

Fig 1. Surface and Contour maps of NOAA Depth Data values

3. Eigenvalues and Eigenvectors

Now that we have extracted some information from the raw NOAA data, we begin our reduction by finding the first eigenvector and eigenvalue of $A^T A$, with A denoting the matrix of depth values. We do this through an iterative process beginning by multiplying $A^T A$ by an initial guess vector of magnitude 1 and dividing the resulting vector by its magnitude to obtain a new guess vector, repeating these steps until the vector stops changing significantly. Our hypothesis as to why this method of finding eigenvectors is effective concerns the essence of matrices and eigenvectors. A matrix can be understood as a linear transformation of the basis vectors of some vector space, and any given vector contained within that space can be written as a linear combination of the basis vectors. It then follows that any given vector in that space, in this case a vector of magnitude 1 with random entries, is transformed by a matrix into a new vector such that it is now an element of the new space. Now, with each iteration of the loop, we "reset" the space, but leave the vector in the same physical place, meaning if we apply the same linear transformation again, the change that each iteration of the vector goes through is progressively less than the transformation it went through in the previous iteration. With enough iterations, the change on the vector by the matrix becomes negligible and it stays in the same physical location after each transformation. This, that is, a vector that is only scaled when a given matrix is applied to it, is the definition of an eigenvector for that given matrix. Performing this method we determined the first eigenvalue to be approximately $3.883x10^{13}$. For visual clarity, we have plotted the values of the associated eigenvector $\vec{V}_1$, presented in Fig. 2 to the right (Appendix A, Section 2).

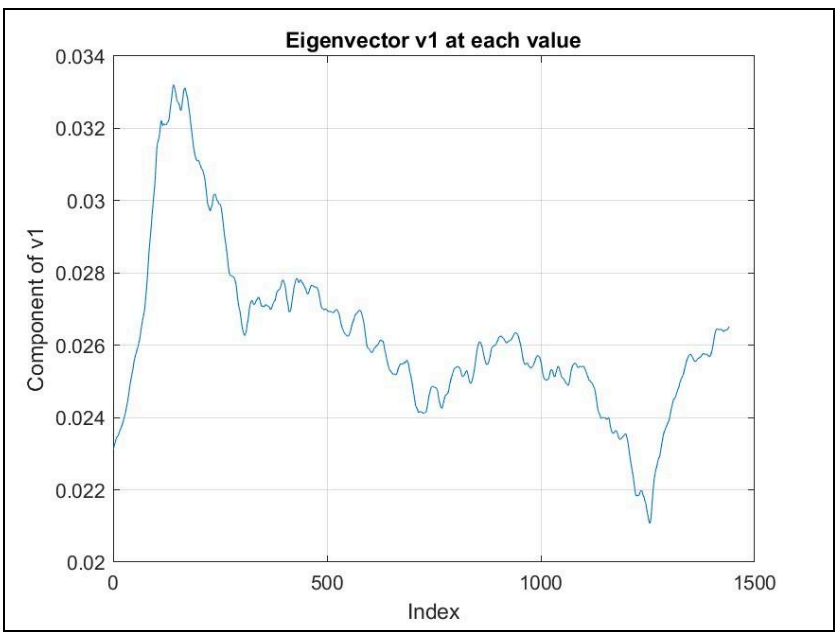

Fig 2. Plot of Eigenvector V_1

Further, we are able to compute the i largest eigenvalue and associated eigenvectors by utilizing a process known as Gram-Schmidt Orthogonalization. This process takes a set of linearly independent vectors, eigenvectors in this case, and turns them into an orthogonal set of vectors that span the same subspace. It does this by taking a random vector $\vec{u}_n$, applies A^TA to it, and then subtracts the component of $\vec{u}_n$ that is equal to the vector projection of that vector onto the eigenvector to get the component of $\vec{u}_n$ that is orthogonal to the eigenvector. It then reassigns $\vec{u}_{n+1}$ to be that new orthogonal vector and process can be repeated. By doing this we are able to compute and store the found eigenvectors as columns in a 1440x50 matrix, V. The eigenvalues found by this process are arranged in Fig. 3 above as a semilog plot (Appendix A, Section 3).

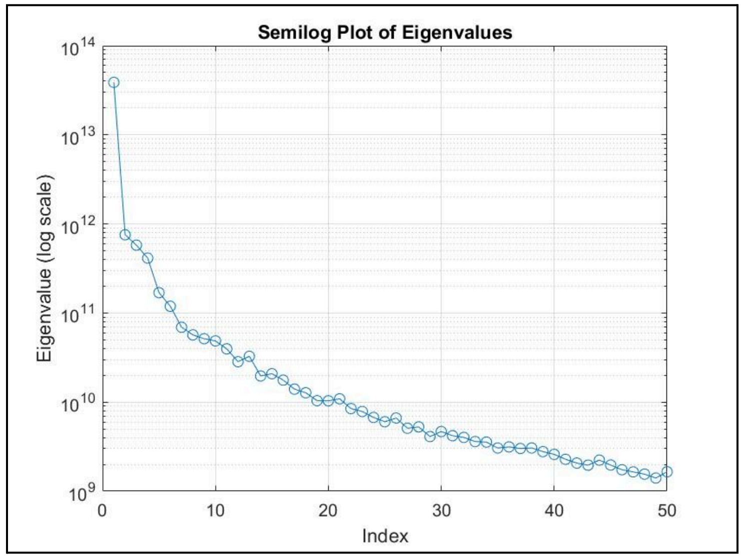

Fig 3. Semilog Plot of Eigenvalues in V

4. Incomplete SVD Decomposition

Finally, with these eigenvalues, we can begin the incomplete single value decomposition to reduce the data to a more manageable size. The square roots of the eigenvalues we found in the previous part become the diagonal elements of matrix Σ, and we further define matrix U such that each column of U is equal to A times the associated column of V, divided by the associated element of Σ. These matrices have been visualized and are shown below in Fig. 4 (Appendix A, Section 4).

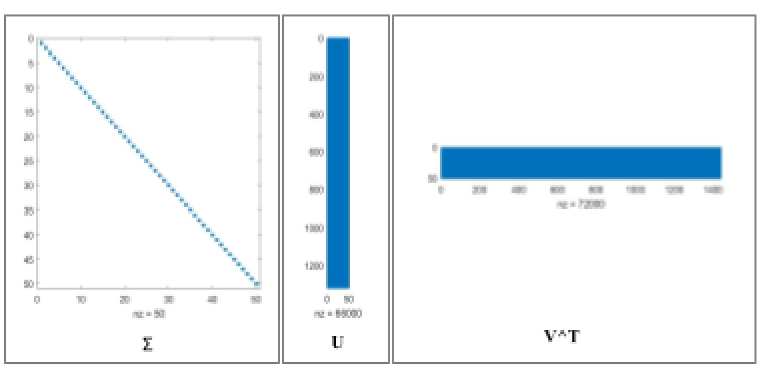

Fig. 4 Visualization of matrices Σ, U, and V^T

Now, with an incomplete decomposition of the original depth matrix A, we have captured all of the relevant details of A while reducing the total number of data points to approximately 7.4% of the original set. Table 1 below details this comparison (Appendix A, Section 4).

	U	Σ	V^T	(U, Σ, V^T) Combined Elements	A
Total Elements	66,000	2,500	72,000	140,500	1,900,800
Total Non-Zero Elements	66,000	50	72,000	138,050	1,900,764

Table 1 - Element Comparison between Raw and ISVD data

To verify the validity of this data, we will compare the results drawn from the matrix product $U\Sigma V^T$ to the corresponding results drawn from the raw data. Fig. 5 shows the new contour map remains largely the same as the previous one (Appendix A - Section 5). The deepest part of the trench, according to this modified data set, is 10,882 meters located at 11° 22 '30"N, 142° 34' 30"E, representing a relatively small change in depth from the raw data of 0.439%. The new average depth of the trench, again defined to be the mean of all depth values greater than 6,000 meters, is 7,734 meters, representing a difference from the raw data average of 0.361% (Appendix A - Section 5).

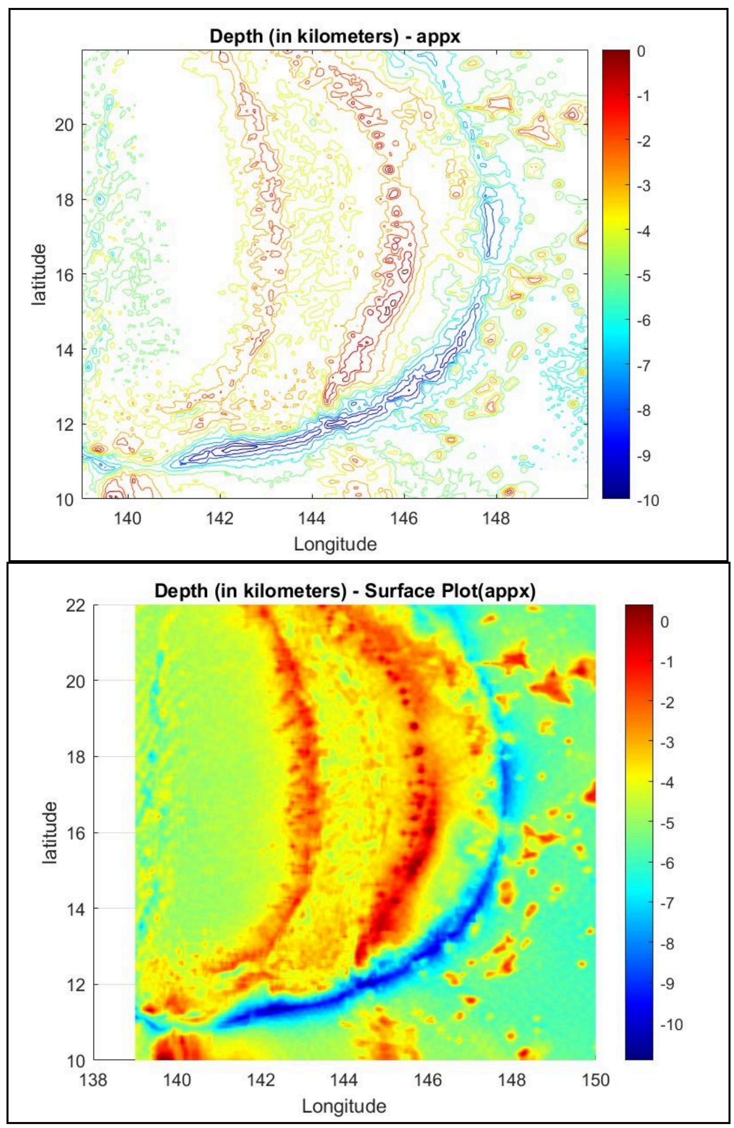

Fig 5. Updated Surface and Contour maps

It is worth noting that when utilizing a fewer number of columns of U and V, in this case 10 columns instead of 50, the contour map generated becomes slightly more legible but less detailed as a result, while the surface plot becomes hazy and harder to read, shown in Fig. 6 below (Appendix A - Section 5).

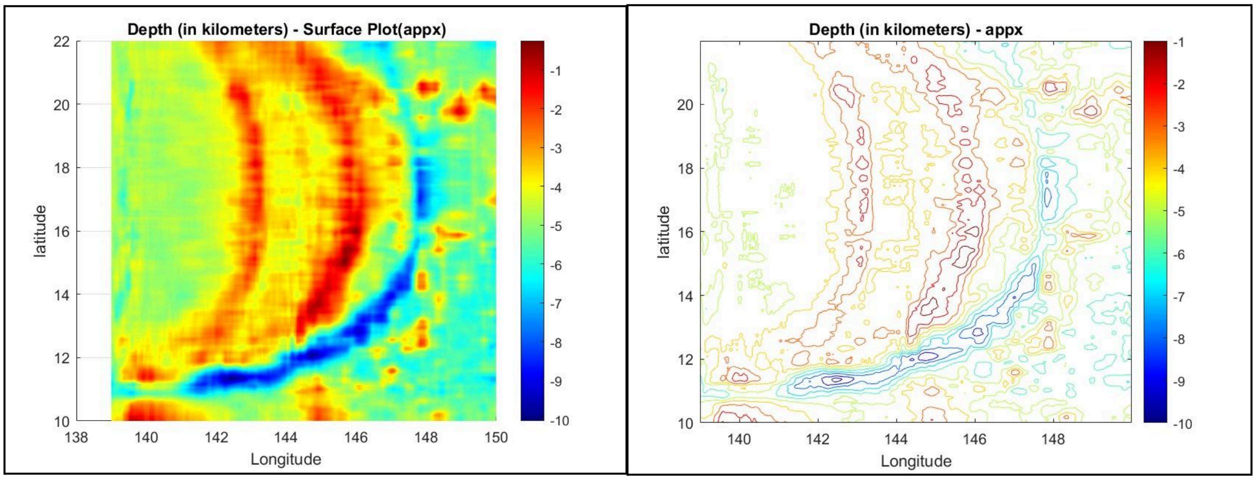

Fig 6. Surface and Contour maps with fewer samples (10)

5. Conclusion

In this report, we initially explored depth data collected by NOAA about the Mariana Trench, calculating both the maximum and average depth of the trench. We then, through an iterative process calculated a single eigenvalue of the depth matrix A, which we then used in conjunction with the Gram-Schmidt Orthogonalization process, computed the 50 largest eigenvalues and associated eigenvectors of A. Finally, using an incomplete single-value decomposition, we were able to reduce the set from an initial number of 1.9 million data points to a much more manageable approximately 140,000 points while maintaining the core integrity of the data.

Appendix A - MATLAB CODE

Section 1:

```
%2.1
A = importdata('mariana_depth (1).csv'); %import data
lon = importdata("mariana_longitude.csv");
lat = importdata("mariana_latitude.csv");

depthkm = A./1000; %convert to km

[latGrid, lonGrid] = meshgrid(unique(lat),unique(lon)); %takes in unique values in lon and lat and turns them into a mesh grid
depthGrid = griddata(lat,lon,depthkm,latGrid,lonGrid); %combine all three matricies into a grid

figure; %for the surface
surf(lonGrid, latGrid, depthGrid); %make a 3D surface
view(2); %view from above
shading interp; %smooth color transitions on the surface
colormap jet; %color map from blue to red based on height
colorbar; %adds color scale to map
xlabel('Longitude');
ylabel('latitude');
title('Depth (in kilometers) - Surface Plot');

figure %for the contour plot
contour(lonGrid, latGrid, depthGrid, -11:1:11); %contours over given interval
clabel(contour(lonGrid, latGrid, depthGrid, -11:1:11), 'manual'); %add labels
colormap jet; %color map from blue to red based on height
colorbar; %adds color scale to map
xlabel('Longitude');
ylabel('Latitude');
title('Depth (in kilometers)');

% Find the minimum depth and its index (returns all occurrences if duplicates)
[minDepth, minIndices] = min(depthkm);

% If there are multiple, pick the first one
minIndex = minIndices(1);

% Get the corresponding latitude and longitude for the minimum depth
minLat = lat(minIndex);
minLon = lon(minIndex);

% Display the result
fprintf('The deepest part of the trench is %.2f km at latitude %.4f and longitude %.4f.\n', minDepth, minLat, minLon);
```

Section 2:

```
%2.2
%q1
A = importdata('mariana_depth (1).csv'); %get A
ATA = A'*A; %find A^TA
n = size(ATA,1); %get the number of rows in the first column of A^TA
u = rand(n,1); %u is a random vector with n rows
u = u./norm(u); %normalize u

for i = 1:10 %about 10 iterations
    u = ATA*u; %apply A^TA to u
    u = u./norm(u); %normalize
end

v1 = u; %v1 is the eigenvector
%upon inspection(compare ATAv1 to v1), lambda = 3.88e13
figure
plot(1:n,v1)
xlabel('1 to n(n = numRows of A)');
ylabel('component of v1');
title('Eigenvector v1 at each value');
grid on;
```

Section 3:

```
%q2
A = importdata('mariana_depth (1).csv'); %get A
ATA = A'*A; %find A^TA
n = size(ATA,1); %get the number of rows in the first column of A^TA
V = zeros(n,50); %matrix of evecs
E = zeros(50,1); %matrix of evals


for i = 1:50
    u1 = rand(n,1); %random unit vector of mag 1
    u1 = u1./norm(u1);

    for k = 1:10 %loop for error reduction(assuming 50 iterations works well since it did in part 1)
        sum = 0;
        u1 = ATA*u1; %apply A^T A to u1
        for j = 1:(i-1)
            sum = sum+(u1'*V(:,j))*V(:,j); %create the orthogonal sum
        end
        u1 = u1-sum; %subtract the sum from u1
        u1 = u1/norm(u1);
    end
    V(:,i) = u1; %reassign v column
    %V1 = ATA*V(:,i); %scaled version of eigenvector
    E(i) = u1'*ATA*u1; %eigenvalue is the ratio between first entry of scaled and unscaled evec
end


semilogy(E, 'o-'); %create the semilogarithmic graph
xlabel('Index');
ylabel('Eigenvalue (log scale)');
title('Semilog Plot of Eigenvalues');
grid on;
```

Section 4:

```
%2.3

%q1
E1 = zeros(50,1); %declare vector of sqrt evals
for i = 1:50
    E1(i) = sqrt(abs(E(i)));
end
E1 = real(E1);
sigma = zeros(50,50);
for i = 1:50 %iterate thru rows of sigma
    for j = 1:50 %iterate thru columns of sigma
        if(i == j)
            sigma(i,j) = E1(i,1);
        end
    end
end

%sigma = real(sigma);
U = zeros(size(A,1),50);
for i = 1:50
    U(:,i) = A*V(:,i)/sigma(i,i);
end

%spy(U) %these are for the end of 2.3.1
%spy(sigma);
%spy(V');

%q2
numel(U) %count total elements
numel(sigma)
numel(V)

numel(A)

nnz(U) %count nonzero elements
nnz(sigma)
nnz(V)

nnz(A)
```

Section 5:

```
%q3
A1 = U*sigma*(V'); %replace A
lon = importdata("mariana_longitude.csv");
lat = importdata("mariana_latitude.csv");

depthkm1 = A1./1000; %convert to km

[latGrid, lonGrid] = meshgrid(unique(lat),unique(lon)); %takes in unique values in lon and lat and turns them into a mesh grid
depthGrid = griddata(lat,lon,depthkm1,latGrid,lonGrid); %combine all three matricies into a grid

figure; %for the surface
surf(lonGrid, latGrid, depthGrid); %make a 3D surface
view(2); %view from above
shading interp; %smooth color transitions on the surface
colormap jet; %color map from blue to red based on height
colorbar; %adds color scale to map
xlabel('Longitude');
ylabel('latitude');
title('Depth (in kilometers) - Surface Plot(appx)');

figure %for the contour plot
contour(lonGrid, latGrid, depthGrid, -11:1:11); %contours over given interval
clabel(contour(lonGrid, latGrid, depthGrid, -11:1:11), 'manual'); %add labels
colormap jet; %color map from blue to red based on height
colorbar; %adds color scale to map
xlabel('Longitude');
ylabel('latitude');
title('Depth (in kilometers) - appx');
```