APPM 2360 - Intro Diff Eq W/Lin Alg Fall 2024

Project 2 - Analyzing The Mariana Trench

Daniel Alemayehu, Eli Grundberg, Samuel Meyn

November 12, 2024

Contents

1 Introduction

We are interested in analyzing and understading the layout of the Mariana Trench. However, we have a large amount of data to work with so we'll need to work to create an approximation of the trench such that we can still have an accurate picture of the trench to better analyze while utilizing less data and memory to do so.

2 Inital Trench Investigations

In order to better understand reducing the trench data we'll take a subset of it and operate on it as normal. Utilizing Matlab's Contour and Surface Plots we can get a good picture of the trench below from our data subset.

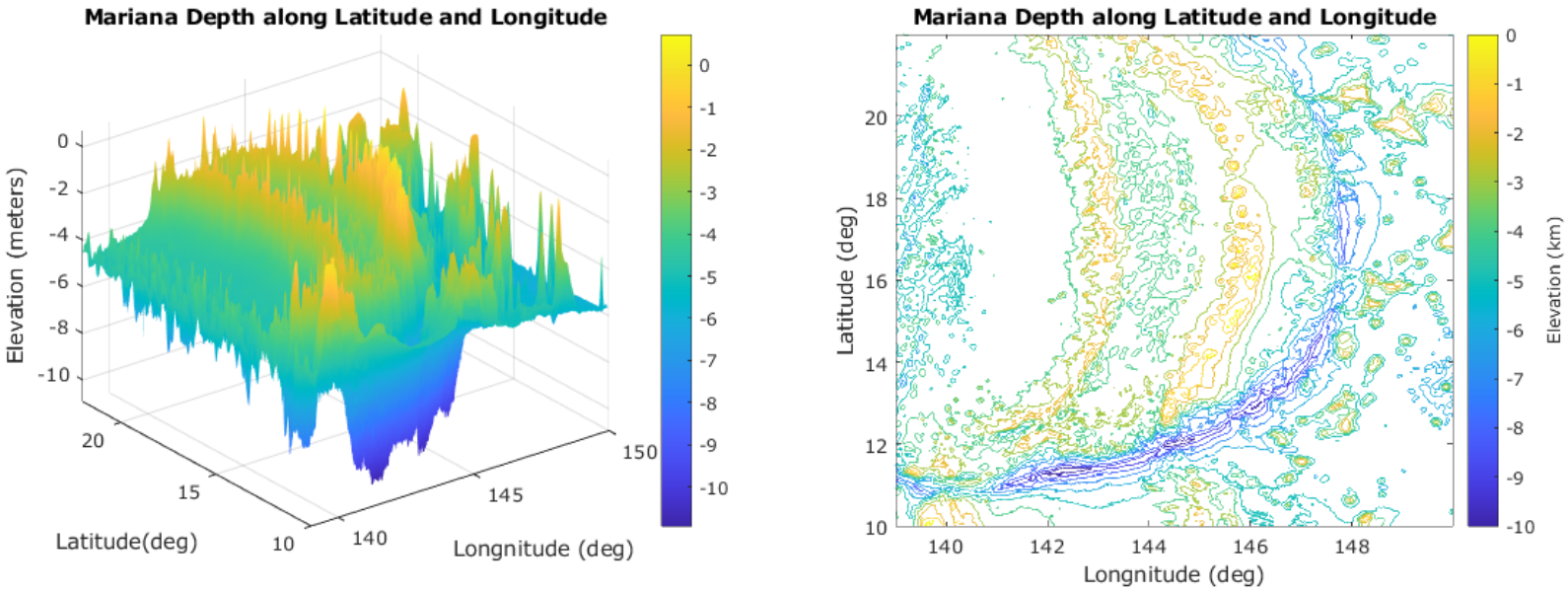


Figure 1: Code in Appendix A

We can also find that the maximum depth of the trench is -10.93 km at lattitude 13.2° longitude 140.3° (calculated in Appendix A). In addition, in our data subset can also find that the average depth of the trench below sea level is given to be -7.2048 km (calculated in

Appendix A).

3 Eigenvalue Computation

We can find an eigenvectors associated with our matrix using an algorithm outlined in Appendix B. The reason this algorithm works can be explained as follows:

The eigenvectors constitute a basis for R^{1440} and $\vec{\mathbf{u_0}}$ is the initial guess vector as a linear combination of the eigenvectors. Thus, we get the result that $A^T * A * V_i = \lambda_i * V_i$. As we solve for u_1, the initial guess vector gets normalized by multiplying it by $A^T * A$ which can be used to find a new vector. As we continue, each of these new vectors gets stored in a matrix and the diagonal stores the eigenvalues associated with the vector. This continues until we get all of the eigenvalues associated with the original matrix. Decomposing the linear combination of the matrix into its parts and finding the eigenvalues. We use this algorithm to find the eigenvector associated with the largest eigenvalue, and then plot the components of the eigenvector as a function of the nth component.

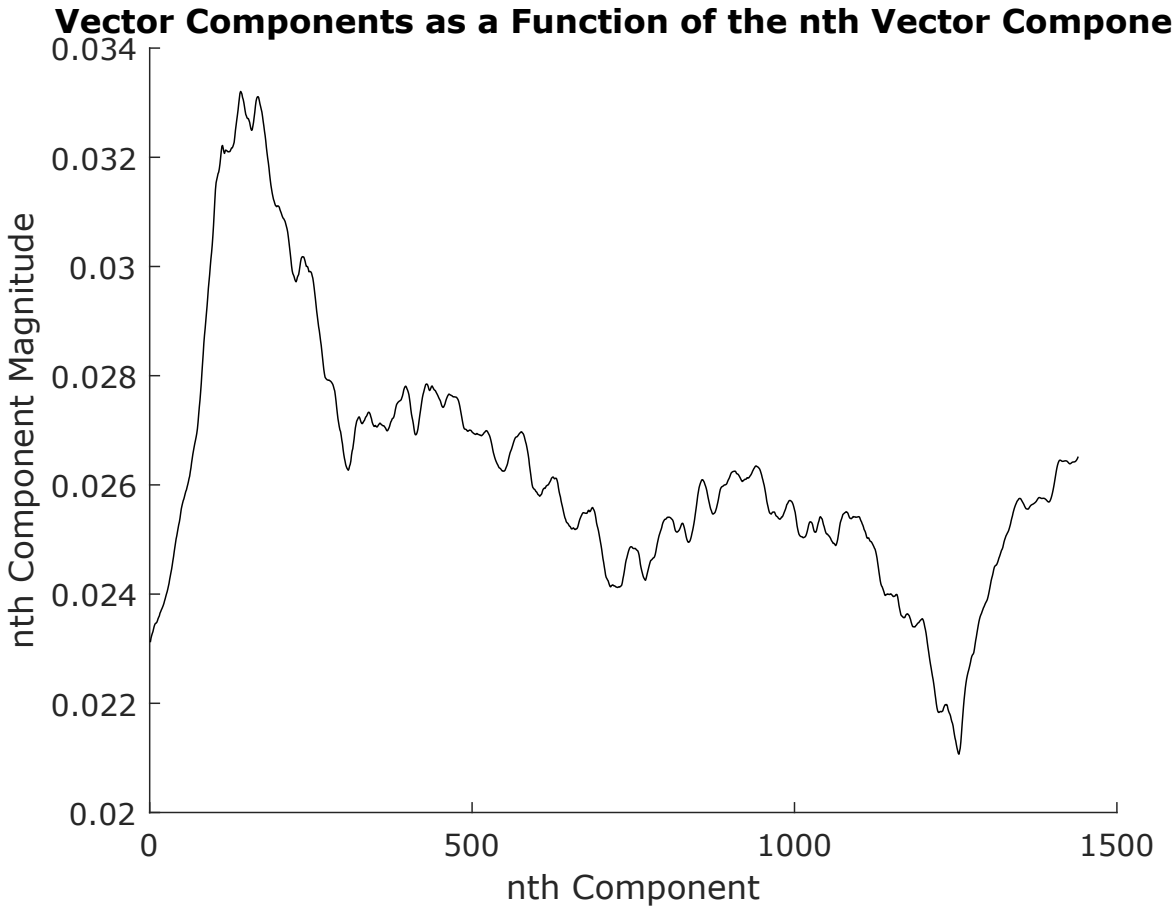


Figure 2: Eigenvector $\mathbf{V_1}$ plot, Eigenvalue: $3.8803 * 10^{13}$, Code in Appendix B

We can also apply *Gram-Schmidt Orthogonalization* to find eigenvectors as well which we'll use for the 50 largest eigenvectors. We plot the 50 largest eigenvalues associated with these eigenvectors:

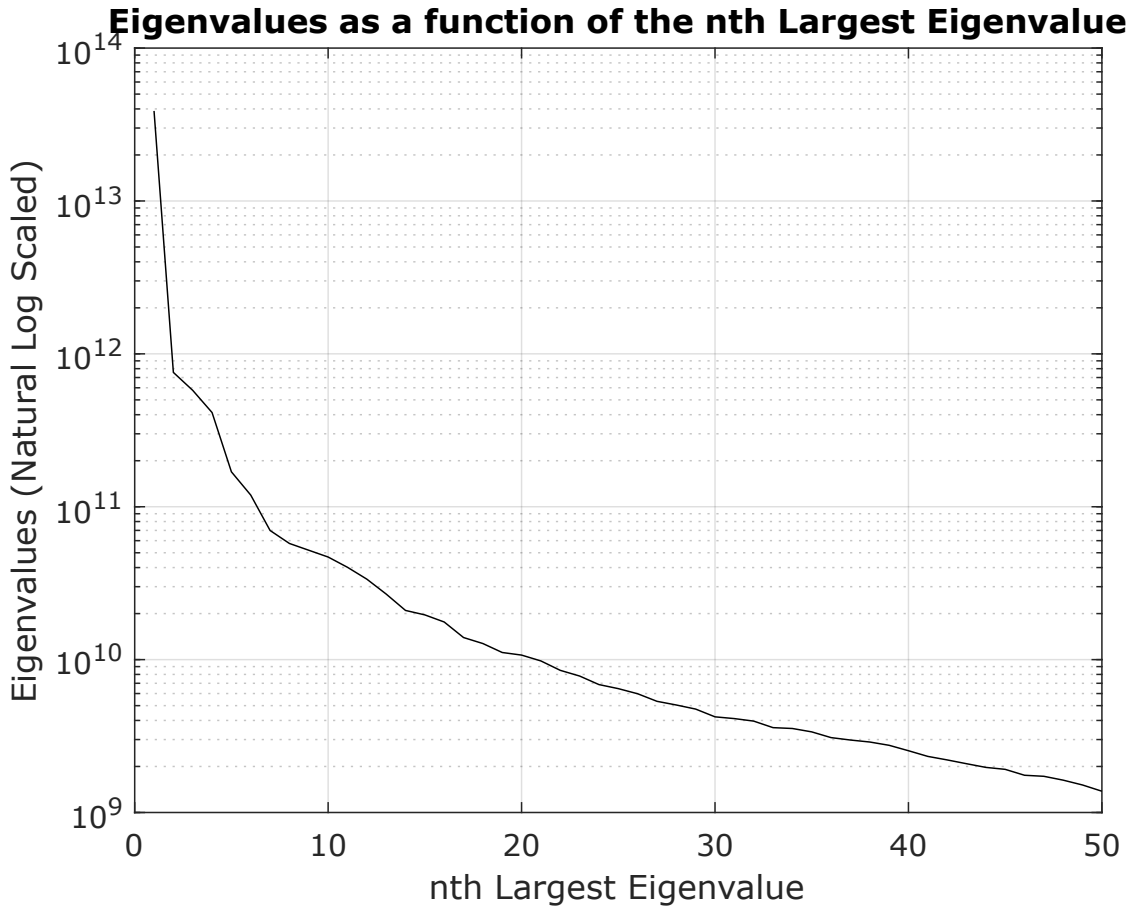
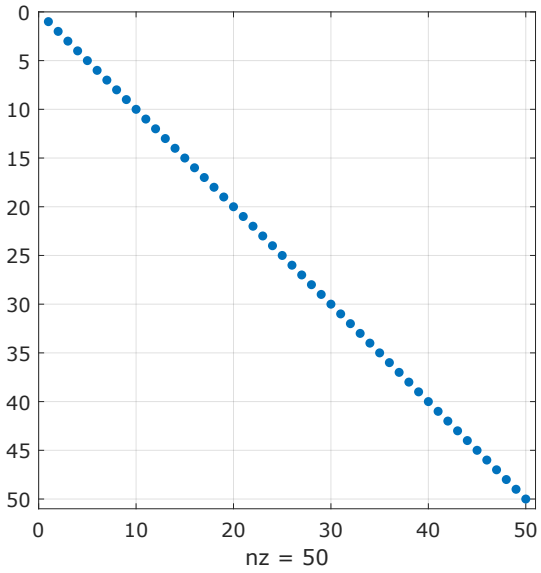


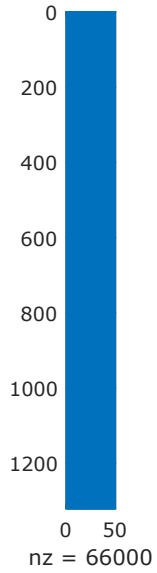
Figure 3: nth largest eigenvalue vs $\ln(E_n)$ plot

4 SVD Incomplete Decomposition

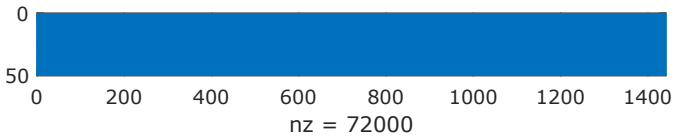
After completing our Incomplete SVD Decomposition (where $\mathbf{U}, \mathbf{V^T}$ have 50 columns) our `spy` output looks like the following:



(a) $\mathbf{\Sigma}$ Matrix



(b) $\mathbf{U}$ Matrix



(c) $\mathbf{V^T}$ Matrix

Figure 4: Code in Appendix C

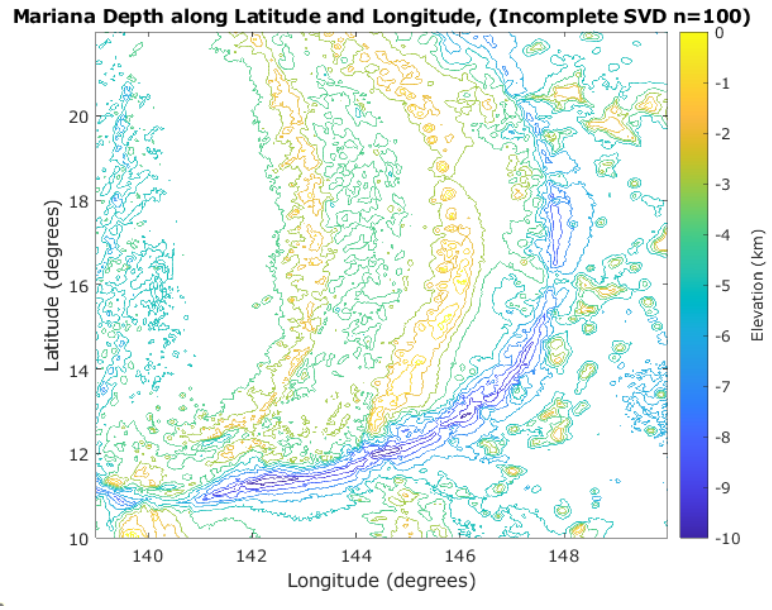
As one method for comparing the efficiency of the Incomplete SVD vs the full matrix of trench data we can compare the total number of entries and the total number of nonzero entries. For our SVD up to the 50th eigenvalue we note the following:

- For the number of total entries, our matrix has 7.39% of the number of total matrix

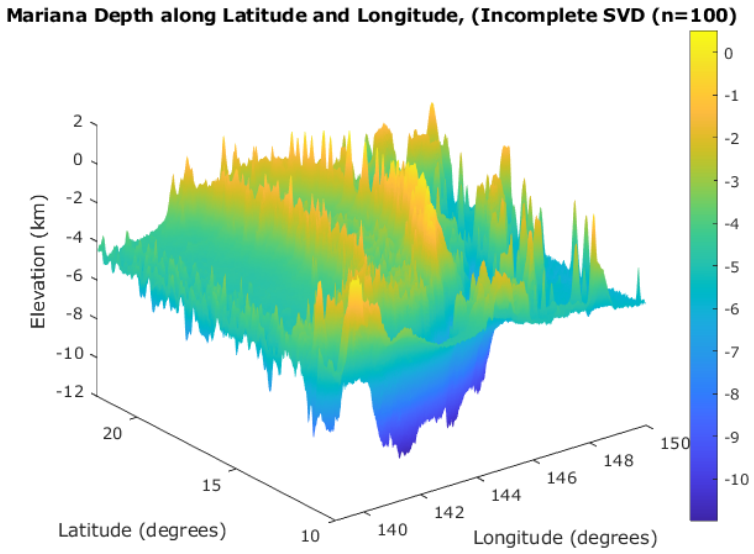
entries as matrix **A**.

- For the number of nonzero entries, our matrix has 7.26% of the number of nonzero entries as matrix **A**.

We've made plots of several SVD decompositions with varying column counts (10, 50, 100) which appear below.



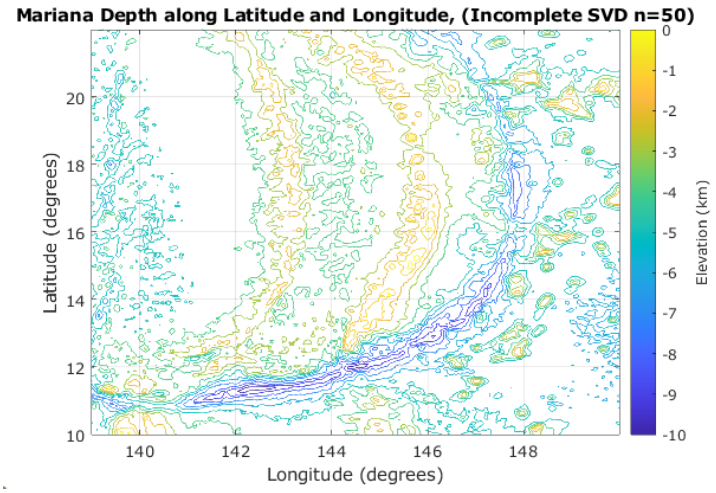
(a) Max Depth at -10.9719 km, Latitude 21.2167° Longitude 140.2167°



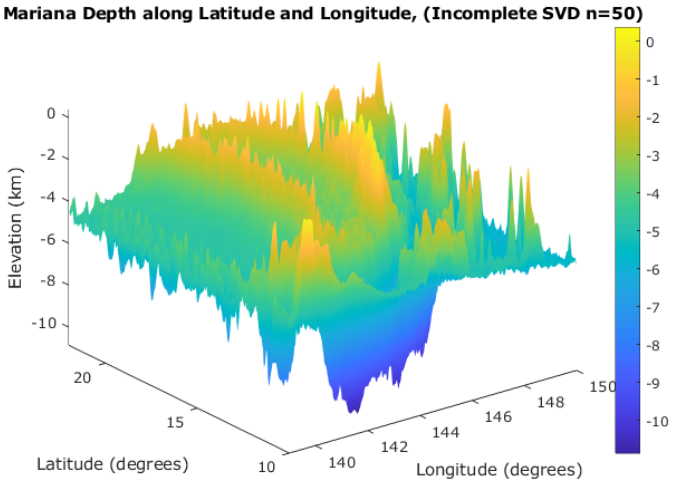
(b) Max Depth at -10.9719 km, Latitude

(c) Mean Depth: -7.1972

Figure 5: Code in Appendix C

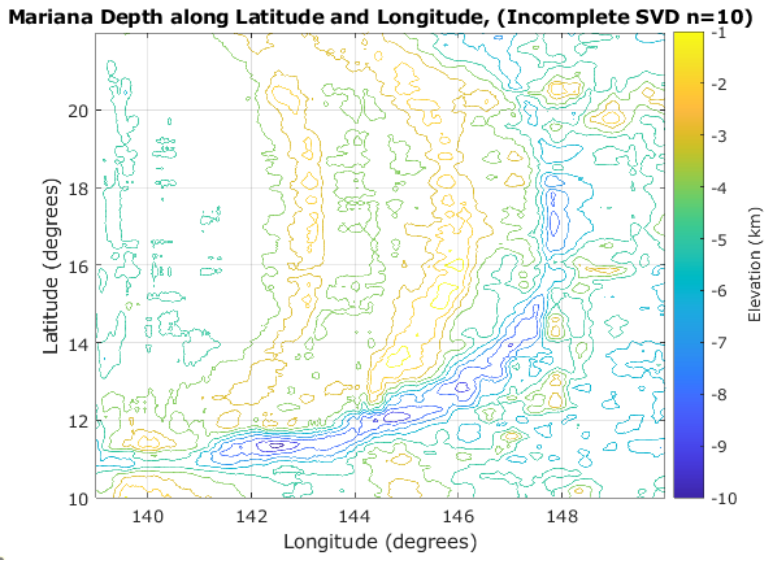


(a) Maximal Depth: -10.8647 km at Latitude 16.5750° Longitude 140.2583°

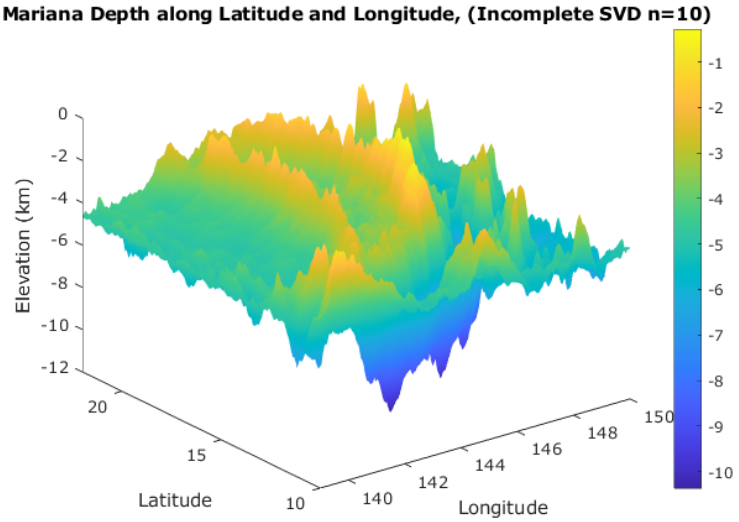


(b) Average Depth Below Sea Level: -7.1742 km

Figure 6: Code in Appendix C



(a) Maximal Depth: -10.3722 km at Latitude 18.4500° Longitude 140.2417°



(b) Average Depth Below Sea Level: -7.0495 km

Figure 7: Code in Appendix C

As we reduce the size of the $\mathbf{\Sigma}, \mathbf{U}, \mathbf{V}$ matricies we notice that our graph appears to "smooth out". Commonly, this is refered to as a lower resolution version of our original data. Additionally, we'll note that the incomplete SVD up to the 10th largest eigenvalue is

far more noticable "low res" than the other two incomplete SVDs we provide which are far closer/more similar to our original data.

5 Conclusion

The Incomplete SVD proves to be an extremely efficient way of maintaining the unique structure of the Mariana trench while being able to perform sophisticated analysis on it. However, depending on whether or not we are interested in the whole trench we may be able to apply the SVD on an even small subset of the data. Another thing to keep in mind is the differences in resolution as we further complete our SVD. What makes the SVD so powerful is that only a small number of rows/columns are needed to get a fairly precise picture.

Appendix A Figure 1 Code

The following code is in MATLAB syntax:

```
clc
clear
close all

%% Part 2.1 - Samuel H. Meyn

%% 2.1.1
%
% Data Plots
%

%Import all data
Z = importdata('mariana_depth.csv');
Y = importdata('mariana_longitude.csv');
X = importdata('mariana_latitude.csv');
Z = transpose(Z);
Zkm = Z .* 1/1000;

figure(1)

Levels = [-11 -10 -9 -8 -7 -6 -5 -4 -3 -2 -1 0 1 2 3 4 5 6 7 8 9 10 11];
contour(Y, X, Zkm, Levels)
title('Mariana Depth along Latitude and Longitude');
xlabel('Longnitude (deg)');
ylabel('Latitude (deg)');
cb = contourcbar("eastoutside");
cb.XLabel.String = "Elevation (km)";

figure(2)

surf(Y,X,Zkm, 'EdgeColor','none')
title('Mariana Depth along Latitude and Longitude');
xlabel('Longnitude (deg)');
ylabel('Latitude(deg)');
zlabel('Elevation (meters)');
colorbar;

%% 2.1.2

% Finding Maximal Depth

%
% Incorrect Code (returns incorrect lat/long
%
```

```
% Min= min(Z, [], 'all');
% for i=1:length(Z)
%     for j=1:width(Z)
%         if Z(i,j) == Min
%             break
%         end
%     end
% end
%
% maxDepthLatitude = j;
% maxDepthLongitude = i;
% maxDepthLongitude;
% maxDepthLatitude;
% Min;
% Z(maxDepthLongitude, maxDepthLatitude);

[Min2, I] = min(Z, [], 'all');
[MinLong, MinLat] = ind2sub([1440 1320], I);
Min2
Z(MinLong, MinLat)
X(MinLat)
Y(MinLong)

%% 2.1.3

% Mean depth sub -6km
meanNum = 0;
N = 0;

for i=1:length(Z)
    for j=1:width(Z)
        if Z(i,j) < -6000
            meanNum = meanNum + Z(i,j);
            N = N + 1;
        end
    end
end

meanTrenchDepth = meanNum/N;
meanTrenchDepth/1000
```

Appendix B Figure 2 and 3 Code

The following code is in MATLAB syntax:

```
clear;
clc;
close all;

A = readmatrix('mariana_depth.csv');
%dataLat = readmatrix('mariana_latitude.csv');
%dataLong = readmatrix('mariana_longitude.csv');


%% Part 1

tic
%Transpose A
A_T = A';
%Create Unit vector
n = (1440);
x = randn(n,1);
x = x / norm(x);

%{
Iterate until the unit vector for n+1 minus the previous unit vector
doesn't change
%}
for i = 1:10

%Initialize the new unit vector as x_new
x_new = A_T*A*x;
x_new = x_new / (norm(x_new));

    %Check x_new - x until the value is small
    if  norm(x_new - x) <= 0
        break;
    end

x = x_new;

end
%Show final eigenvalue and eigenvector
eigen_first = x'*A_T*A*x;
%x_n_vector = sqrt(sum(x^2));
N = length(x);
%plot of the final vector against the length of the matrix from 1 to N
hold on;
plot(1:N,x, 'k-');
```

```
xlabel('nth Component')
ylabel('nth Component Magnitude')
title('Vector Components as a Function of the nth Vector Component')
eigen_first
toc
%% Part 2
tic
%Initializes vectors as matrices of zeros
v_matrix = zeros(length(A_T*A),50);
v_vals50 = zeros(50,1);

%for loop to find eigenvalues and eigenvectors
for i = 1:50

%Initialize a unit vector
u = randn(length(A_T*A),1);
u = u / norm(u);

%checks if
    while true
    u_new_asterisk = A_T*A*u;
        %iterated to compute eigenvalues for each column
        for j = 1:(i-1)
          u_new_asterisk = u_new_asterisk - (u_new_asterisk'*v_matrix(:,j))*
              v_matrix(:,j);
        end

        %Reinitialize unit vector so it can be checked against the previous
        u_new = u_new_asterisk/ norm(u_new_asterisk);

        %Checks eigenvalue for a tolerance less than 0.002
        if norm(u_new - u) < 0.002
            break;
        end
    %Iterates the new unit vector
    u = u_new;

    end
%{
Saves eigen vectors and values to their corresponding 1440 x 50 and 50 x 1
matrices
%}
v_matrix(:,i) = u_new;
v_vals50(i) = (u_new')*(A_T)*(A)*(u_new);

end
%final U_n+1 as v_i
```

```
v_i = u_new;

%eigenvectors as columns of 1440 x 50 matrix
V = v_matrix;

%semilog plot of the 50 eigenvalues computed
figure(2)
semilogy(1:50, trimdata(v_vals50, [50 1]), 'k-');
xlabel('nth Largest Eigenvalue')
ylabel('Eigenvalues (Natural Log Scaled)')
title('Eigenvalues as a function of the nth Largest Eigenvalue')
grid

figure(3)
semilogx(1:50, trimdata(v_vals50, [50 1]), 'k-');
xlabel('nth Largest Eigenvalue (Natural Log Scaled)')
ylabel('Eigenvalues')
title('Eigenvalues as a function of the nth Largest Eigenvalue')
grid

toc
```

Appendix C Figure 4, 5, 6, 7 Code

The following code is in MATLAB syntax:

```
clear;
clc;
close all;
%
% PART 1: Incomplete SVD Decomposition
%
A = readmatrix('mariana_depth.csv');
dataLat = readmatrix('mariana_latitude.csv');
dataLong = readmatrix('mariana_longitude.csv');

A_T = A';

V = zeros(1440, 50);

n = 1440;
u = randn(n,1);
u = u / norm(u);

for i = 1:50
    while 1
        u_new = A_T*A*u;
        summation = 0;
        for j = 1:i
            summation = summation + (u_new'*V(:,j))*V(:,j);
        end
        u_new = u_new - summation;
        u_new = u_new/norm(u_new);
        if norm(u_new - u) < 0.001
            break;
        end
        u = u_new;
    end
    V(:,i) = u_new;
end

sigma = [50 50];
for i = 1:50
    sigma(i,i) = V(:,i)'*A_T*A*V(:,i);
end
sigma = sqrt(sigma);

U = [];
for i = 1:50
    U(:,i) = A*V(:,i)/sigma(i,i);
```

```
end

[Min2, I] = min(U*sigma*V'/1000, [], 'all');
[MinLat, MinLong] = ind2sub([1440 1320], I);
Min2
dataLat(MinLat)
dataLong(MinLong)

meanNum = 0;
N = 0;

TEMP = U*sigma*V'/1000;
for i=1:1320
    for j=1:1440
        if TEMP(i,j) < -6
            meanNum = meanNum + TEMP(i,j);
            N = N + 1;
        end
    end
end

meanTrenchDepth = meanNum/N;
meanTrenchDepth

figure(11)
spy(sigma)
grid
figure(12)
spy(U)
grid
figure(13)
spy(V')
grid

%
% PART 2: SVD Efficacy for 50 columns of U, V
%
SVD_space = numel(sigma) + numel(U) + numel(V);
A_space = numel(A);
space_efficiency = SVD_space/A_space;
space_efficiency

SVD_nzs = nnz(sigma) + nnz(U) + nnz(V);
A_nzs = nnz(A);
nnz_efficiency = SVD_nzs/A_nzs;
nnz_efficiency
```

```
%
% PART 3: Image/Contour Map
%
figure(1)
surf(dataLong,dataLat,((U*sigma*V')/1000)','EdgeColor','none')
colorbar
title('Mariana Depth along Latitude and Longitude, (Incomplete SVD n=50)');
xlabel('Longitude (degrees)');
ylabel('Latitude (degrees)');
zlabel('Elevation (km)');
grid

figure(2)
Levels = [-11 -10 -9 -8 -7 -6 -5 -4 -3 -2 -1 0 1 2 3 4 5 6 7 8 9 10 11];
contour(dataLong,dataLat,(U*sigma*V'/1000)', Levels)
title('Mariana Depth along Latitude and Longitude, (Incomplete SVD n=50)');
xlabel('Longitude (degrees)');
ylabel('Latitude (degrees)');
cb = contourcbar("eastoutside");
cb.XLabel.String = "Elevation (km)";
grid

%
% Part 4: U V column counts?
%

%
% U, V with 10 columns
%
V10 = zeros(1440, 10);

n = 1440;
u = randn(n,1);
u = u / norm(u);

for i = 1:10
    while 1
        u_new = A_T*A*u;
        summation = 0;
        for j = 1:i
            summation = summation + (u_new'*V10(:,j))*V10(:,j);
        end
        u_new = u_new - summation;
        u_new = u_new/norm(u_new);
        if norm(u_new - u) < 0.001
            break;
        end
```

```
        u = u_new;
    end
    V10(:,i) = u_new';
end

sigma10 = [10 10];
for i = 1:10
    sigma10(i,i) = V10(:,i)'*A_T*A*V10(:,i);
end
sigma10 = sqrt(sigma10);

U10 = [];
for i = 1:10
    U10(:,i) = A*V10(:,i)/sigma10(i,i);
end

[Min2, I] = min(U10*sigma10*V10'/1000, [], 'all');
[MinLat, MinLong] = ind2sub([1440 1320], I);
Min2
dataLat(MinLat)
dataLong(MinLong)

meanNum = 0;
N = 0;

TEMP = U10*sigma10*V10'/1000;
for i=1:1320
    for j=1:1440
        if TEMP(i,j) < -6
            meanNum = meanNum + TEMP(i,j);
            N = N + 1;
        end
    end
end

meanTrenchDepth = meanNum/N;
meanTrenchDepth


figure(3)
surf(dataLong,dataLat,((U10*sigma10*V10')/1000)','EdgeColor','none')
colorbar
title('Mariana Depth along Latitude and Longitude, (Incomplete SVD n=10)');
xlabel('Longitude');
ylabel('Latitude');
zlabel('Elevation (km)');
grid
```

```
figure(4)
Levels = [-11 -10 -9 -8 -7 -6 -5 -4 -3 -2 -1 0 1 2 3 4 5 6 7 8 9 10 11];
contour(dataLong,dataLat,(U10*sigma10*V10')/1000)', Levels)
title('Mariana Depth along Latitude and Longitude, (Incomplete SVD n=10)');
xlabel('Longitude (degrees)');
ylabel('Latitude (degrees)');
cb = contourcbar("eastoutside");
cb.XLabel.String = "Elevation (km)";
grid

%
% U, V with 100 columns
%
V100 = zeros(1440, 100);

n = 1440;
u = randn(n,1);
u = u / norm(u);

for i = 1:100
    while 1
        u_new = A_T*A*u;
        summation = 0;
        for j = 1:i
            summation = summation + (u_new'*V100(:,j))*V100(:,j);
        end
        u_new = u_new - summation;
        u_new = u_new/norm(u_new);
        if norm(u_new - u) < 0.001
            break;
        end
        u = u_new;
    end
    V100(:,i) = u_new';
end

sigma100 = [100 100];
for i = 1:100
    sigma100(i,i) = V100(:,i)'*A_T*A*V100(:,i);
end
sigma100 = sqrt(sigma100);

U100 = [];
for i = 1:100
    U100(:,i) = A*V100(:,i)/sigma100(i,i);
end
```

```
[Min2, I] = min(U100*sigma100*V100'/1000, [], 'all');
[MinLat, MinLong] = ind2sub([1440 1320], I);
Min2
dataLat(MinLat)
dataLong(MinLong)

meanNum = 0;
N = 0;

TEMP = U100*sigma100*V100'/1000;
for i=1:1320
    for j=1:1440
        if TEMP(i,j) < -6
            meanNum = meanNum + TEMP(i,j);
            N = N + 1;
        end
    end
end

meanTrenchDepth = meanNum/N;
meanTrenchDepth

figure(5)
surf(dataLong,dataLat,((U100*sigma100*V100')/1000)','EdgeColor','none')
colorbar
title('Mariana Depth along Latitude and Longitude, (Incomplete SVD (n=100)');
xlabel('Longitude (degrees)');
ylabel('Latitude (degrees)');
zlabel('Elevation (km)');
grid

figure(6)
Levels = [-11 -10 -9 -8 -7 -6 -5 -4 -3 -2 -1 0 1 2 3 4 5 6 7 8 9 10 11];
contour(dataLong,dataLat,(U100*sigma100*V100'/1000)', Levels)
title('Mariana Depth along Latitude and Longitude, (Incomplete SVD n=100)');
xlabel('Longitude (degrees)');
ylabel('Latitude (degrees)');
cb = contourcbar("eastoutside");
cb.XLabel.String = "Elevation (km)";
grid
```