```
%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);

%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
%ATA*v1 (for when we want to inspect)
%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;

%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;

%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)

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

depthkm1 = A1./5000; %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');
```