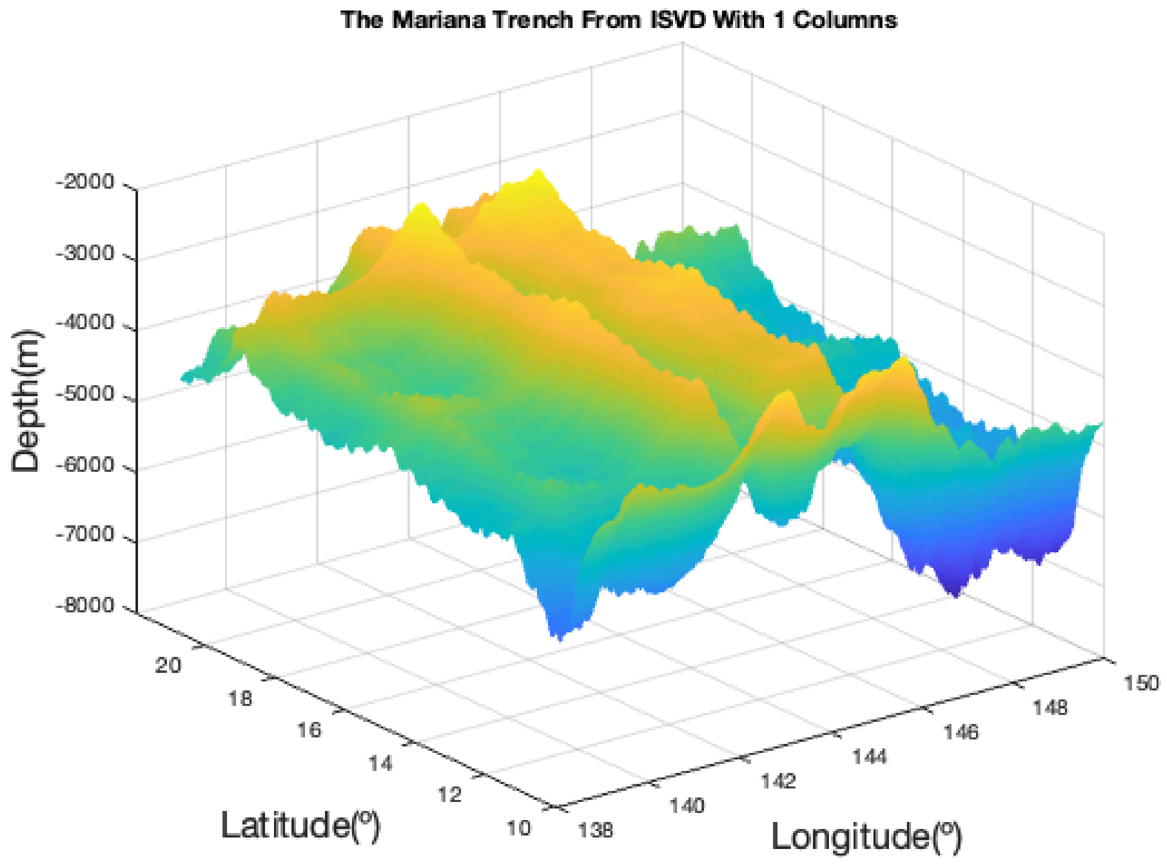
```
A = importdata('mariana_depth.csv');
lon = importdata('mariana_longitude.csv');
lat = importdata('mariana_latitude.csv');



%2.2.2
wantedEigenSize = 10;
fprintf('OUTPUT: \n');
numOfEignens =
 [1,10,50,100,500];%[1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,22,24,26,28,30,32,
compressedRatings=zeros(1,size(numOfEignens,2));
accurateRatings=zeros(1,size(numOfEignens,2));
temp=zeros(1,size(numOfEignens,2));
for i = 1:size(numOfEignens,2)
    [accurateRating,compressedRating] = main(numOfEignens(i),A,lat,lon);
    fprintf('For ISVD With %i Columns: compression rating %f , accuracy rating
 %f (lower is better)\n',numOfEignens(i),compressedRating,accurateRating);
    compressedRatings(1,i)=compressedRating;
    accurateRatings(1,i)=accurateRating;
end

figure(100000);
plot(numOfEignens,compressedRatings,'red');
title(sprintf('Compression Rating for Different Number of Columns'));
xlabel('Number of Columns','FontSize',16);
ylabel('Compression Rating at the Given Number of Columns','FontSize',16);

figure(100001);
plot(numOfEignens,accurateRatings,'blue');
title(sprintf('Accuracy Rating for Different Number of Columns'));
xlabel('Number of Columns','FontSize',16);
ylabel('Accuracy Rating at the Given Number of Columns','FontSize',16);

function [accurateRating,compressedRating] = main(wantedEigenSize,A,lat,lon)
    A_t=transpose(A);
    B= A_t*A;

    [V,eigenValues]= betterEigen(B,wantedEigenSize);

    if(wantedEigenSize==50)
        figure(2);
        semilogy(1:wantedEigenSize,eigenValues,'blue');%plot of
        title(sprintf('Semilog Plot Of Eigenvalues'));
        xlabel('Column','FontSize',16);
        ylabel('Eigenvalue at the Given Column','FontSize',16);
    end
    %2.3.1
    sigma = zeros(wantedEigenSize,wantedEigenSize);
    U = zeros(size(A,1),wantedEigenSize);

    for i = 1:wantedEigenSize
```

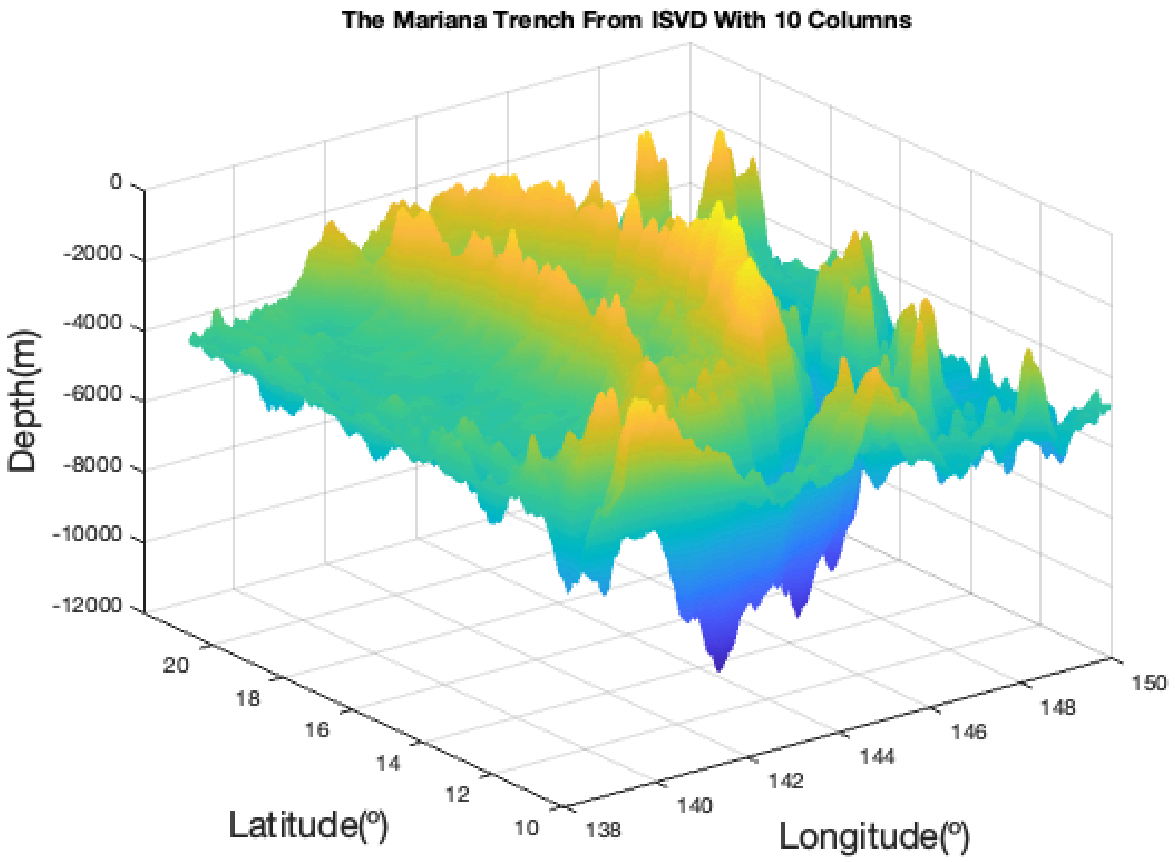
```
        sigma(i,i)=sqrt(eigenValues(1,i));
        U(:,i) = ( A*V(:,i) ) / sigma(i,i) ;
    end

    %2.3.3
    figure(wantedEigenSize)
    compressedA = U*sigma*transpose(V);
    grid = meshgrid(lat,lon);
    mesh(lon,lat,compressedA',compressedA');
    %s = pcolor(lon,lat,compressedA');
    %set(s, 'EdgeColor', 'none');
    xlabel('Longitude(°)','FontSize',16);
    ylabel('Latitude(°)','FontSize',16);
    zlabel('Depth(m)','FontSize',16);
    title(sprintf('The Mariana Trench From ISVD With %i
 Columns',wantedEigenSize));
    accurateRating = accuracyRating(A,compressedA);
    compressedRating = compressionRating(A, wantedEigenSize);

    %fprintf('compression rating : %f (lower is better), for an eigenspace of
 dimension: %i \n',accuracyRating(A,compressedA), wantedEigenSize);
    count=0; %used to count the number of points below 6km
    currentTotal=0; %used to keep tract of the sum of the depths below 6km
    for i = 1:size(compressedA,1)
       for j = 1:size(compressedA,2)
          if(compressedA(i,j)<-6000) %if the point is deeper than 6km, then
 update our variables that keep track
              count = count+1;
              currentTotal = currentTotal+compressedA(i,j);
          end
       end
    end
    averageDepthUnder6km = currentTotal/count; %arithmetic mean
    fprintf('Average Depth of the Trench For ISVD With %i Columns : %f (km)
 \n',wantedEigenSize,averageDepthUnder6km/1000);%Average Depth of the trench:
 -7204.636665

end

function total = compressionRating(A, numEigens)
    numInA = size(A,1)*size(A,2);
    total= (numEigens*(size(A,1)+numEigens+size(A,2)) )/numInA;
end

function total = accuracyRating(A,compressedA)
    total=0;
    for i = 1:size(A,1)
       for j = 1:size(A,2)
          total = total+ abs( A(i,j)-compressedA(i,j) );
       end
    end
    total=total/100000000;
end
```

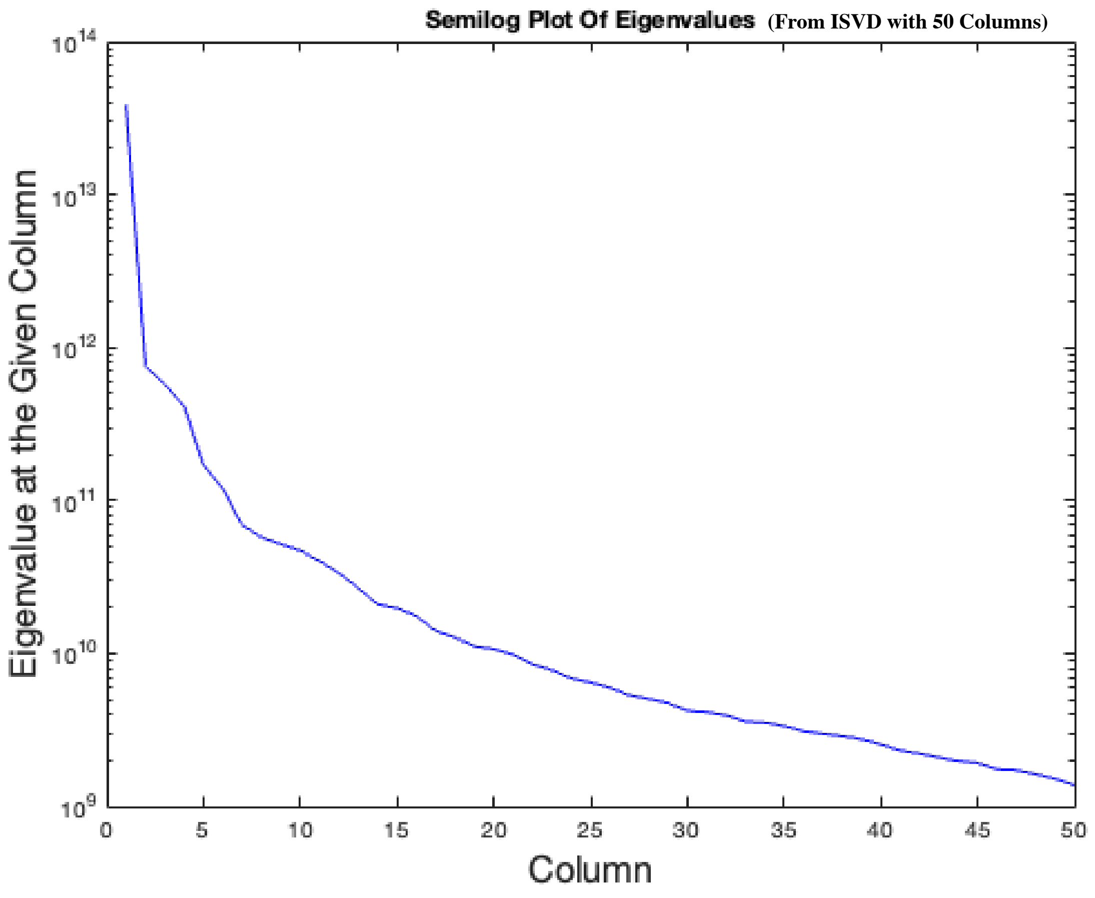
```
function [matrixOfVectors,matrixOfValues] = betterEigen(matrix,numOfEigens)
    %this method uses the second given algorithm to find the first
    %[numOfEigens] amt of eigenvectors adn their associated values.
    matrixOfVectors = zeros(sizeColVect(matrix),numOfEigens);
    matrixOfValues = zeros(1,numOfEigens); %row vectors
    for i = 1:numOfEigens
        u = randomUnitColVector2(sizeColVect(matrix)); %i


        u1star = matrix*u;%ii

        summationResult=0;
        for j = 1:(i-1)
            summationResult=summationResult
+(transpose(u1star)*matrixOfVectors(:,j))*matrixOfVectors(:,j);
        end

        u1=u1star-summationResult;%iii
        u1=unitVect(u1);%iv

        whileCount = 0;
        smallNumber = 1e-3;
        while(mag(u1-u)>smallNumber)
            whileCount=whileCount+1;
            u=u1;
            u1star = matrix*u;%ii

            summationResult=0;
            for j = 1:(i-1)
                summationResult=summationResult
+(transpose(u1star)*matrixOfVectors(:,j))*matrixOfVectors(:,j);
            end

            u1=u1star-summationResult;%iii
            u1=unitVect(u1);%iv
        end
        matrixOfVectors(:,i)=u1;

        %process of getting associated eigenvalue
        scaledV1 = matrix*u1;
        matrixOfValues(1,i) = scaledV1(1,1)/u1(1,1);


        %fprintf('whileCounter: %i \n',whileCount);%

    end
end

function sizeVect = sizeColVect(colVect)
    b=size(colVect);
    sizeVect=b(1);
end

function vect = randomUnitColVector1(size,randomUpperBound)
```

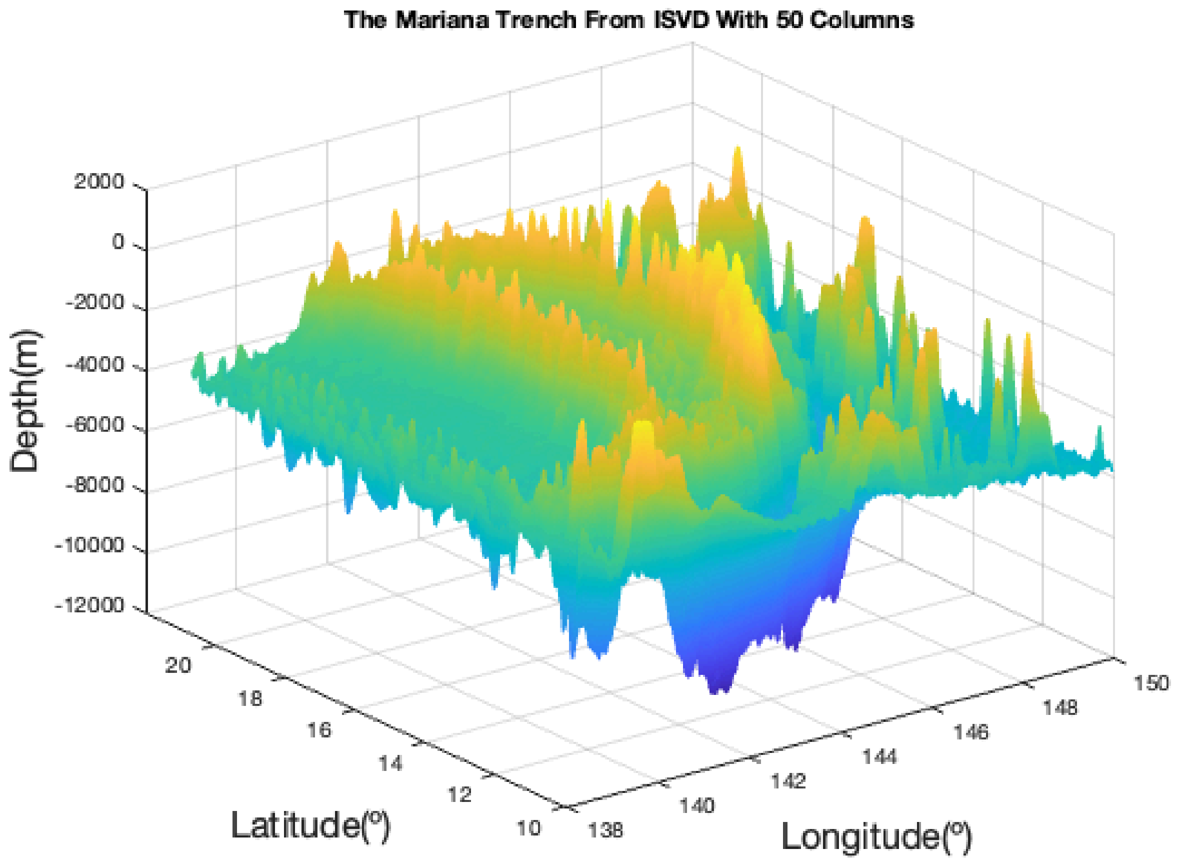
```
    vect = zeros(size,1);
    for k = 1:size
          vect(k,1)=randi(randomUpperBound,1);%first num in randi is random
 int generator
    end
    vect=unitVect(vect);
end

function vect = randomUnitColVector2(size)
    vect=randomUnitColVector1(size,10);
end

function unitVector = unitVect(array)
    unitVector = array/mag(array);
end

function magnitude = mag(array)
    magnitude = 0;
    for k = 1:sizeColVect(array)
      magnitude=magnitude+(array(k,1))^2;
    end
    magnitude= sqrt(magnitude);
end
```

```
OUTPUT:
Average Depth of the Trench For ISVD With 1 Columns : -6.353018 (km)
For ISVD With 1 Columns: compression rating 0.001453 , accuracy rating
 15.437696 (lower is better)
Average Depth of the Trench For ISVD With 10 Columns : -7.048793 (km)
For ISVD With 10 Columns: compression rating 0.014573 , accuracy rating
 5.946841 (lower is better)
Average Depth of the Trench For ISVD With 50 Columns : -7.174009 (km)
For ISVD With 50 Columns: compression rating 0.073916 , accuracy rating
 1.869230 (lower is better)
Average Depth of the Trench For ISVD With 100 Columns : -7.196642 (km)
For ISVD With 100 Columns: compression rating 0.150463 , accuracy rating
 0.837695 (lower is better)
Average Depth of the Trench For ISVD With 500 Columns : -7.203606 (km)
For ISVD With 500 Columns: compression rating 0.857534 , accuracy rating
 0.134671 (lower is better)
```

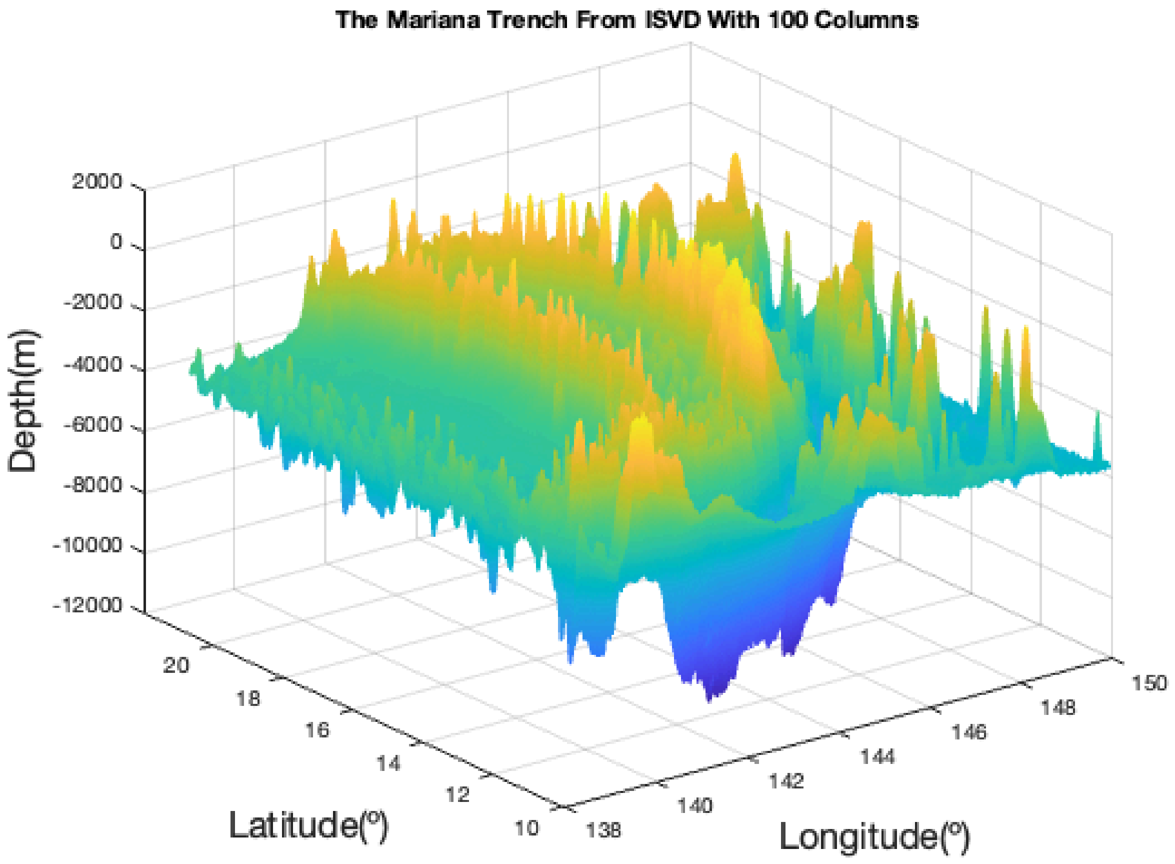

The Mariana Trench From ISVD With 1 Columns
Depth(m)
-2000
-3000
-4000
-5000
-6000
-7000
-8000
Latitude(º)
20
18
16
14
12
10
Longitude(º)
138
140
142
144
146
148
150

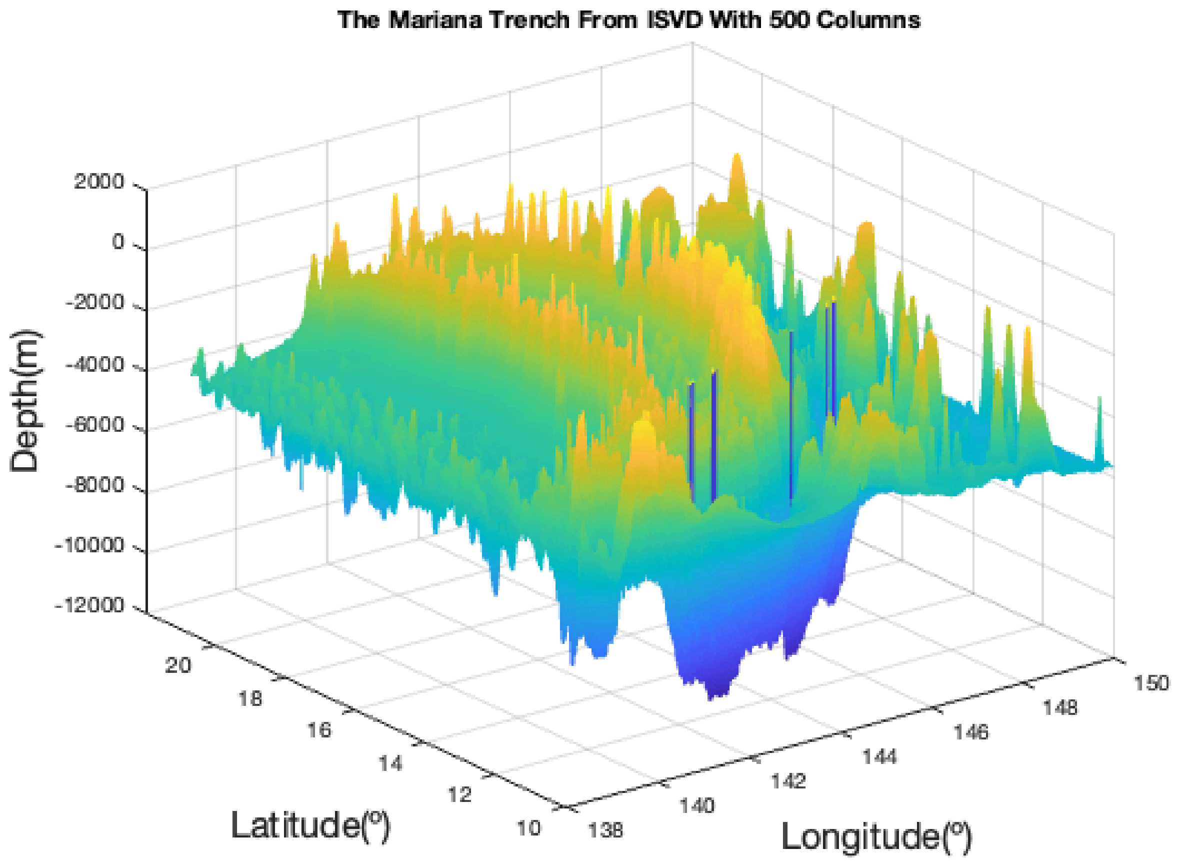

The Mariana Trench From ISVD With 10 Columns
Depth(m)
0
-2000
-4000
-6000
-8000
-10000
-12000
Latitude(º)
20
18
16
14
12
10
Longitude(º)
138
140
142
144
146
148
150

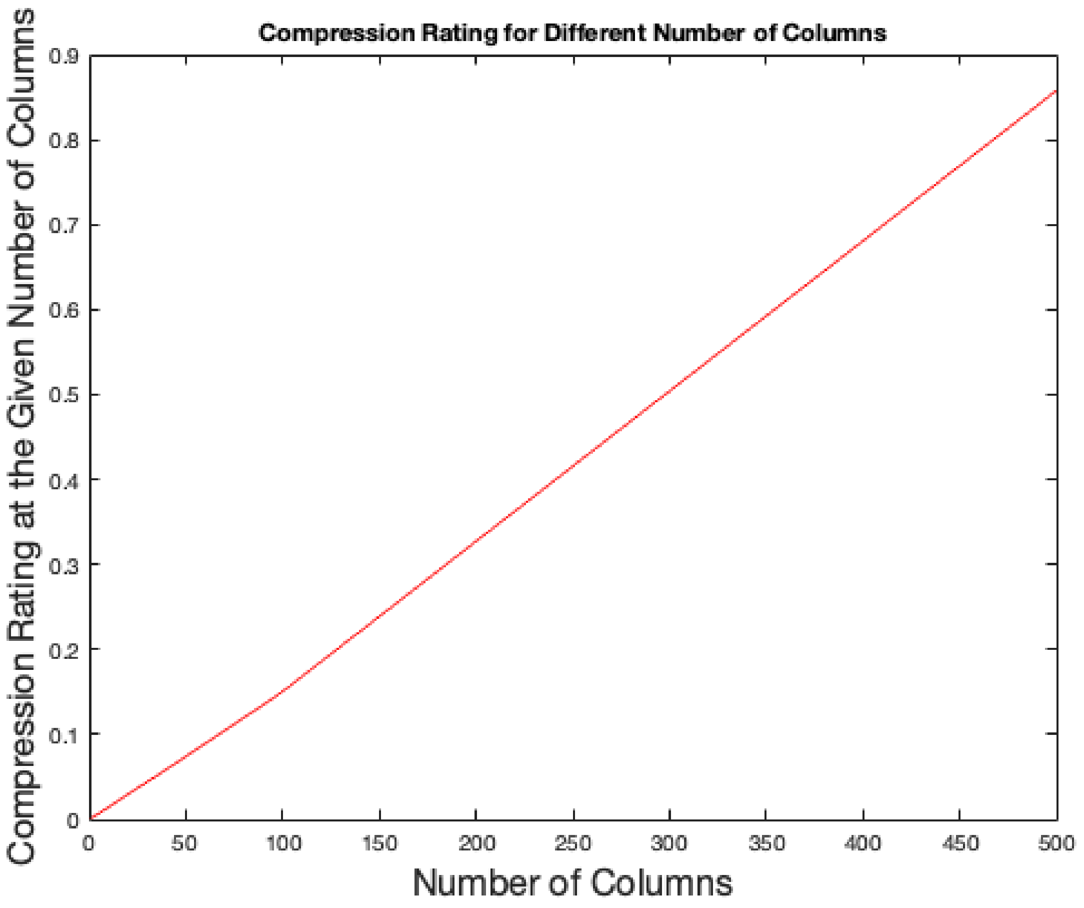

Semilog Plot Of Eigenvalues (From ISVD with 50 Columns)
Eigenvalue at the Given Column
Column
10^{14}
10^{13}
10^{12}
10^{11}
10^{10}
10^{9}
0
5
10
15
20
25
30
35
40
45
50

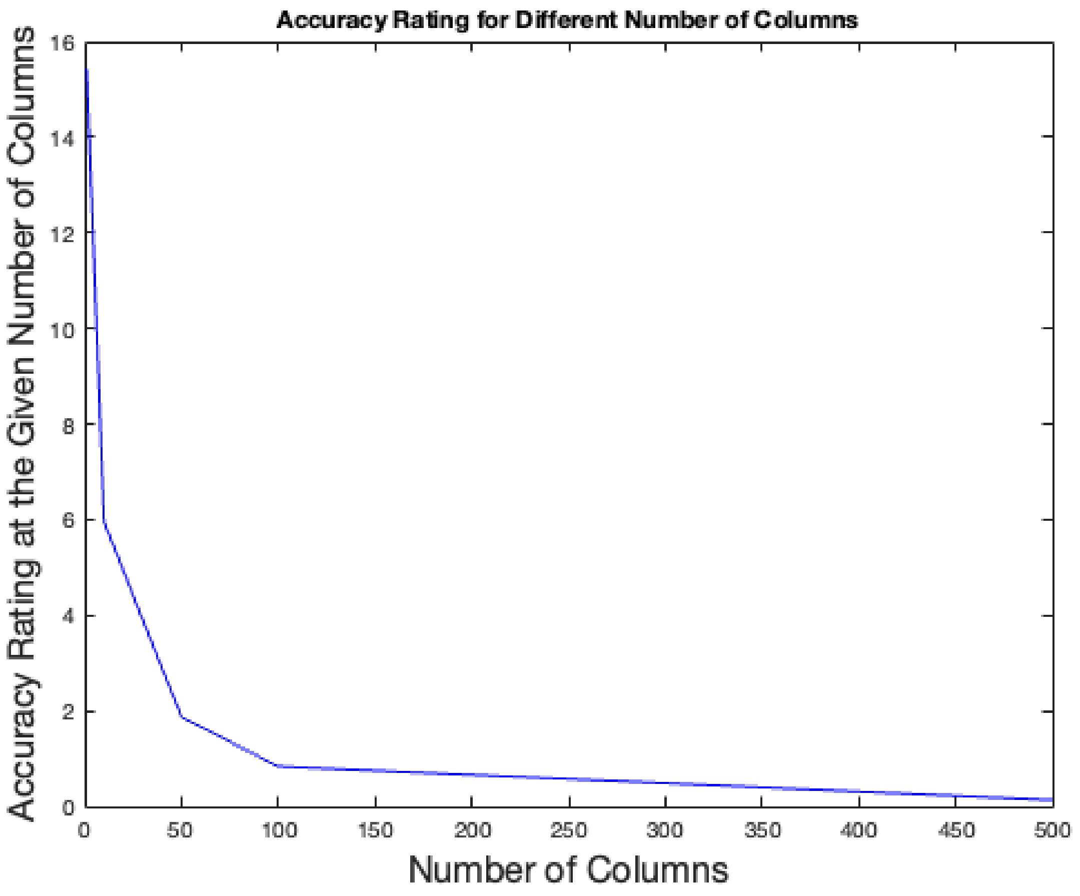


The Mariana Trench From ISVD With 50 Columns
Depth(m)
2000
0
-2000
-4000
-6000
-8000
-10000
-12000
Latitude(º)
20
18
16
14
12
10
Longitude(º)
138
140
142
144
146
148
150


The Mariana Trench From ISVD With 100 Columns
Depth(m)
2000
0
-2000
-4000
-6000
-8000
-10000
-12000
Latitude(º)
20
18
16
14
12
10
Longitude(º)
138
140
142
144
146
148
150


The Mariana Trench From ISVD With 500 Columns
Depth(m)
2000
0
-2000
-4000
-6000
-8000
-10000
-12000
Latitude(º)
20
18
16
14
12
10
Longitude(º)
138
140
142
144
146
148
150


Compression Rating for Different Number of Columns
Compression Rating at the Given Number of Columns
Number of Columns
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
0
50
100
150
200
250
300
350
400
450
500



Published with MATLAB® R2021b