**Tyler Yamori-Little**

**4/26/21**

# Geodynamics Final Report

**Abstract:**

The data provided includes the fingerprint simulations of 1m of Greenland's and Antarctica's ice sheets melting. The latitudes and longitudes and corresponding rates at 100 different sites are also known. The melting of the ice sheets on global sea-level change is determined through the fingerprint values at each location. The equation Ax = y represents this phenomenon where A is a 100x2 matrix of fingerprint rates, and y is a 100x1 matrix of the sea level rates at each site. Thus, the calculated variable x is a 2x1 matrix containing a value for Antarctica's impact and a value for Greenland's impact. Because of extraneous error values, I used a weighted matrix to find more accurate values. With some linear algebra, the final equation to solve is (ATWA)-1 * ATWy.

**Procedure**:

Step 1: Import data
Step 2: Create a cell for the tide sites
Step 3: Plot an individual tide site
Step 4: Get rid of NaN values
Step 5: Use least squares regression to find rates for the y matrix and error for the W matrix
Step 6: Plot the cleaned-up data with a confidence interval
Step 7: Correlate the latitudes and longitudes of the sites with the fingerprints
Step 8: Solve the matrix equation with and without weighting

**Step 1:**

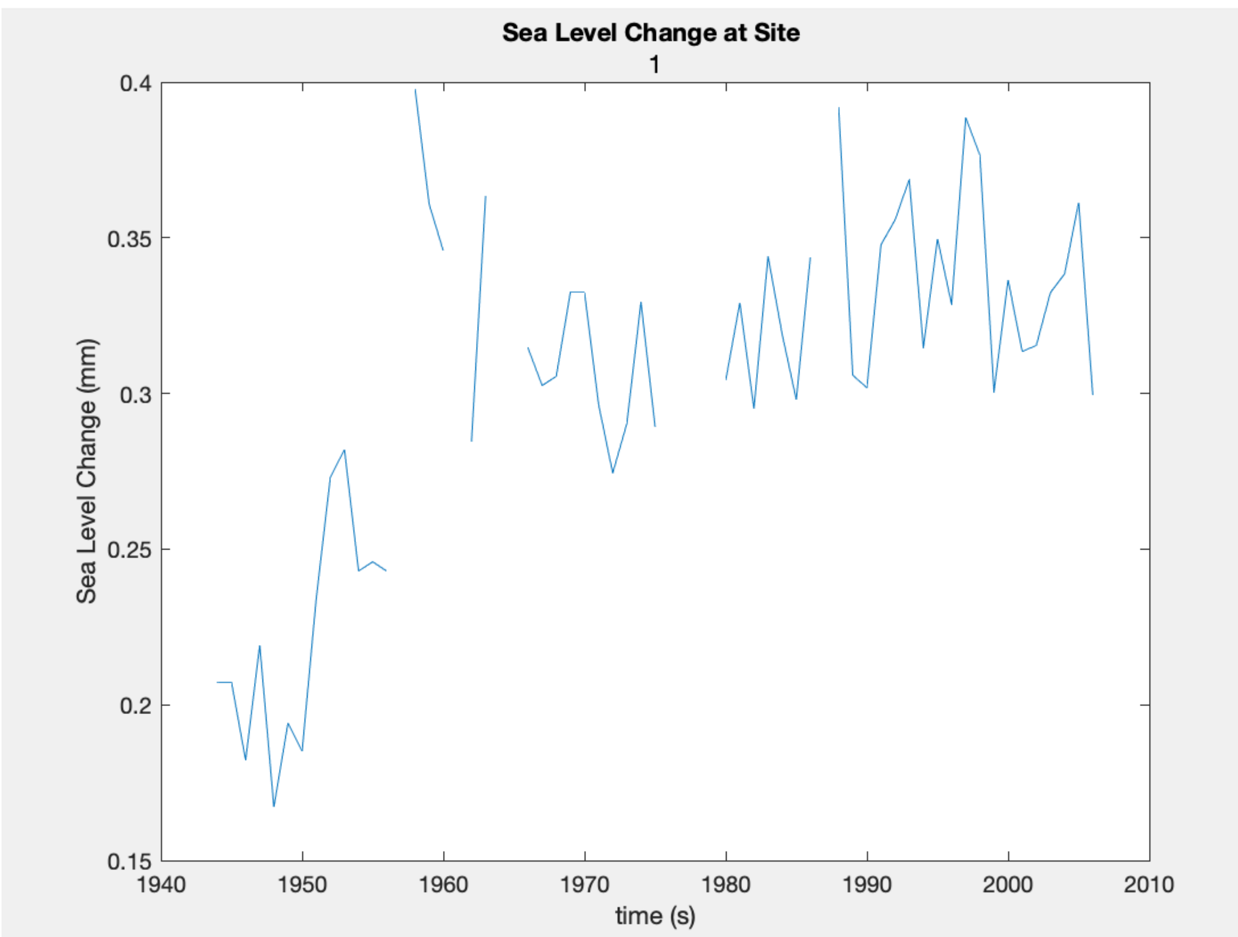
I imported the data using Matlab's built-in import function. These values came as matrices.

**Step 2:**

For future plotting, I concatenated the tide data with the time matrix. Since there already exist two variables (time and rates), I created a cell array to store the 100 individual sites using a for loop iteration.

**Step 3:**

To ensure that the cell array worked properly, I plotted the yearly sea level rate of the first site.




The plot has missing data values and outliers.

**Step 4:**

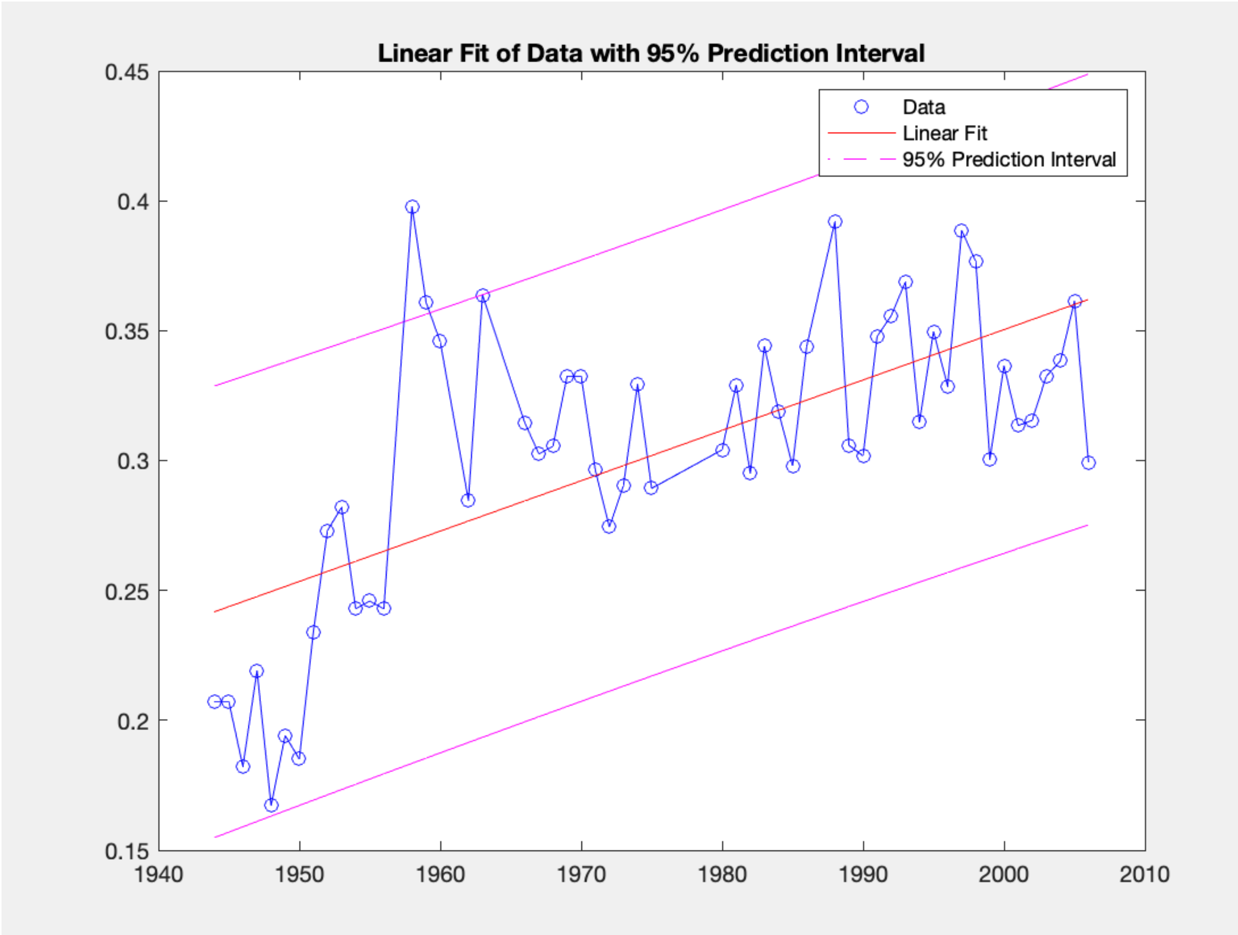
To regress the rates at each site with least squares regression, Matlab has a built-in regression function, polyfit. However, this function does not work with NaN values. To address this, I iterated through each rate and index the values at which the tide site measured NaN. With this indexing, I can remove both those NaN values and the times associated with them. I then re-concatenated these values to be used for regression.

**Step 5:**

Matlab's polyfit function outputs a 2x1 matrix of the regression, Ax + B, and a 1x1 structure contains values necessary for error calculations. The structure contains R, a 2x2 covariance matrix, the degrees of freedom, and the norm of the residuals. I calculated sigma using these values. I can then form the weighted matrix by creating the row echelon form of the identity matrix containing the 100 values of 1/sigma^2.

**Step 6:**

With the error values, I calculated a confidence interval for the data. For example, two standard deviations would result in an approximately 95% confidence interval. I can also determine a line of best fit through the Ax+B values obtained from polyfit.




**Step 7:**

For each site, the latitude and longitudes have to match with that of the fingerprints. I used Matlab's find function, which points towards the index containing the equation assigned to the function. The equation I used to correlate the latitudes and longitudes was the absolute minimum difference between the data sets. Once indexed, I can then concatenate the values into the A matrix used to solve the matrix equation to determine the coefficients for the Antarctica and Greenland ice sheet fingerprints.

**Step 8:**

The coefficients alpha and beta for each fingerprint (x = [alpha; beta]) can be solved using the matrix equation (ATWA)-1 * ATWy. I calculated the equation using both the weighted matrix as well as without it. The weighted solutions were slightly greater for Antarctica and slightly lesser for Greenland. The weighted alpha coefficient (Antarctica's fingerprint) is 8.83*10-4. The weighted beta coefficient (Greenland's fingerprint) is 5.58*10-4. Antarctica is slightly more influential to global sea-level rise.

**Conclusion:**

It appears that Antarctica is slightly more influential to global sea-level rise. This result is likely due to its size and the circumpolar current around it, leading to faster dispersion of melted ice.

# Appendix:

```
%Tyler Yamori-Little
%Geodynamics Final Project
%4/26/21


%%
%import data
Antarctica_fingerprint = importdata('Antarctica_fingerprint.dat');
FP_lat = importdata('FP_lat.dat');
FP_lon = importdata('FP_lon.dat');
Greenland_fingerprint = importdata('Greenland_fingerprint.dat');
lats = importdata('lats.dat');
lons = importdata('lons.dat');
tidedata = importdata('tidedata.dat');
time = importdata('time.dat');

%%
%create a cell array that houses each tide site to a single row
TideSite = cell(100,1);                %declare a cell array
for i = 1:1:100                        %interate through it
 TideSite{i} = [time(1,:);tidedata(i,:)];   %create a matrix that has time vs data
end


%%
%plot an individual tide guage to make sure it works
figure(1);
plot(TideSite{1,1}(1,:),TideSite{1,1}(2,:))
title('Sea Level Change at Site ', 1);
xlabel('time (s)');
ylabel('Sea Level Change (mm)');


%%
%editing the data in preparation of least squares fitting
EditedTideSite = cell(100,1);                  %declare a cell for future editing
RowTideSite = zeros(1,200);                    %declare a matrix for future editing
Row2TideSite = zeros(1,200);                   %declare a matrix for future editing
for j = 1:1:100
   RowTideSite = TideSite{j,1}(1,:);           %Set the edited row to the iterated site (time values)
   Row2TideSite = TideSite{j,1}(2,:);          %Set the edited row to the iterated site (water level values)
   isNaN = isnan(Row2TideSite);                %find data points that are NaN
   RowTideSite(isNaN) = [];                    %get rid of those data points
   Row2TideSite(isNaN) = [];                   %get rid of those data points
 EditedTideSite{j,1} = [RowTideSite;Row2TideSite];  %Concatenate the changed data into a new cell array
end

%%
%Determine the rate at each site

TideSiteRate = cell(100,1);                                          %declare a cell for iteration
TideSiteError = cell(100,1);                                         %declare a cell for iteration
weighted = zeros(100,1);
for l = 1:1:100
   [TideSiteRate{l},TideSiteError{l}] = polyfit(EditedTideSite{l,1}(1,:),EditedTideSite{l,1}(2,:),1);      %rates determined by the Matlab
linear squares regression function
   if size(TideSiteError{l}.R) == 2                                                  %also creates a structure to determine error
    C = (inv(TideSiteError{l}.R)*(inv(TideSiteError{l}.R))')*(TideSiteError{l}.normr)^2/(TideSiteError{l}.df); %calculate the covariance
matrix
    sigma = sqrt(C(1,1));      %calculate sigma
    weighted(l,:) = 1/(sigma^2); %create a weighted matrix
   else
      weighted(l,:) = 0;      %some values for sigma are incalculable because the covariance matrix is 1x2
```

```
   end

end
W = eye(100);             %create an identity matrix
for i = 1:1:100
W(i,i) = weighted(i,:);     %the weighted matrix is equal to the weights in reduced row echelon form
end
%%
%Use the calculated rate and error to find a 95% level of confidence for
%the first site

[y_fit,delta] = polyval(TideSiteRate{1},EditedTideSite{1,1}(1,:),TideSiteError{1});              %calculate delta off of the error function
using polyval
figure(2);
plot(EditedTideSite{1,1}(1,:),EditedTideSite{1,1}(2,:),'b')                              %plot the edited data
hold on
plot(EditedTideSite{1,1}(1,:),y_fit,'r-')                                            %plot the linear regression
plot(EditedTideSite{1,1}(1,:),y_fit+2*delta,'m--',EditedTideSite{1,1}(1,:),y_fit-2*delta,'m--')     %plot the bounds of interval of
confidence
title('Linear Fit of Data with 95% Prediction Interval')
legend('Data','Linear Fit','95% Prediction Interval')


%%
%create the A matrix by correlating lats and lons of the fingerprint to
%the tide sites
AntarcticaIndex = zeros(100,1); %declare a matrix for editing
GreenlandIndex = zeros(100,1);  %declare a matrix for editing
for i = 1:1:100
     indexlat = find(abs(FP_lat(:,1)-lats(i)) == min( abs(FP_lat(:,1)-lats(i)))); %correlate latitudes through finding the absolute
difference between the site matrix and lats matrix
     indexlons = find(abs(FP_lon(:,1)-lons(i)) == min( abs(FP_lon(:,1)-lons(i))));%correlate longitudes through finding the absolute
difference between the site matrix and lons matrix
     AntarcticaIndex(i,1) = Antarctica_fingerprint(indexlat,indexlons); %Build the fingerprint matrix for antarctica at the sites
    GreenlandIndex(i,1) = Greenland_fingerprint(indexlat,indexlons);    %Build the fingerprint matrix for greenland at the sites
end


%%
%Prepare to solve the matrix equation (Transpose(A)WA)^-1 * transpose(A)Wy
A = [AntarcticaIndex, GreenlandIndex]; %concatenate the A matrix using the fingerprint matrices

y = zeros(100,1);                  %declare a matrix for the rates
for i=1:1:100
   y(i) = TideSiteRate{i}(1);        %pull out values from the cell and put into a matrix for y

end

%%
%solve with weight and without it


solution = inv(transpose(A)*A)*transpose(A)*y;
alpha = solution(1,:);
beta = solution(2,:);

solution_weighted = inv(transpose(A)*W*A)*transpose(A)*W*y;
alpha_weighted = solution_weighted(1,:);
beta_weigted  = solution_weighted(2,:);
```