AMATH 563 Spring 2018 Homework 2
Multi-scale decomposition

Tun Sheng Tan

Abstract. This assignment focuses on analyzing large spatial-temporal dataset with wavelet transform. We are able to extract meaningful features from sea surface temperature and differentiate the El Nino and La Nina phenomena.

1. Introduction.

This assignment explores the idea of multi-resolution analysis and implementation of randomized algorithm. In section 2, We introduce the motivation for multi-resolution analysis and randomized singular-value decomposition. In section 3, we discuss the implementation of randomized SVD. In section 4, we perform multi-resolution analysis on the sea surface temperature data to extract useful information about El Nino (La Nina) phenomenon.

2. Theoretical Background [2].

2.1. Wavelet Transform.

Fourier transform (FT) decomposes a signal (time) into the reciprocal space (frequency). Formally, FT is defined to be:

$$F[\omega] = \int f(t)e^{-i\omega t}dt \tag{2.1}$$

FT is a very important transform in physical science because of its linear property and simple natural interpretation of the reciprocal space. In addition, fast Fourier transform (FFT) algorithm makes discrete Fourier transform the choice of transform in modern computing. Localized functions in one space is transformed to a delocalized functions in another space via FT. This property makes it not suitable for analyzing transient behavior.

Gabor introduced an alternative kernel for FT. By performing FT on successive segment of a signal with a window function g (known as windowing), one can localize FT in time and frequency domain. This transform is know aw Windowed Fourier transform or Gabor transform or short time Fourier transform. The mathematical definition of the transform is

$$G[\tau,\omega] = \int g(t-\tau)f(t)e^{-i\omega t}dt \tag{2.2}$$

Wavelet is a generalization of windowed Fourier transform. Instead of decomposing a function into the Fourier basis, wavelet transform is a decomposition of a function into a square integrable orthonormal basis generated by translation and dilation, known as the mother wavelet.

Definition 2.1. ***Wavelet***

If the function ψ satisfies:

$$\int_{-\infty}^{\infty} dt\|\psi(t)\|dt < \infty \tag{2.3}$$

$$\int_{-\infty}^{\infty} dt\|\psi(t)\|^2 dt = 1 \qquad \text{(Square integrable)} \tag{2.4}$$

$$\int_{-\infty}^{\infty} dt\ \psi(t)dt = 0 \qquad \text{(Zero mean)} \tag{2.5}$$

Then, ψ is a wavelet.

Definition 2.2. ***Mother Wavelet***
Suppose that ψ is a wavelet. Then, the function $\psi_{a,b}$ defined below

$$\psi_{a,b} = \frac{1}{\sqrt{2}}\psi(\frac{t-b}{a}) \tag{2.6}$$

is a mother wavelet.

Definition 2.3. ***Wavelet Transform***
Suppose that ψ is a wavelet with Fourier transform $\hat{\psi}$and f is a function of t with reciprocal space ω. The following integral

$$W_\psi[f](a,b) = \int_{-\infty}^{\infty} f(t)\psi^*_{a,b}(t)dt \tag{2.7}$$

is a wavelet transform if

$$C_\psi = \int_{-\infty}^{\infty} \frac{\hat{\psi}(\omega)}{\omega} d\omega < \infty \tag{2.8}$$

The inverse wavelet transform is defined to be

$$W_\psi^{-1}(t) = \frac{1}{C_\psi}\int\int \frac{dadb}{a^2} W_\psi[f](a,b)\psi_{a,b}(t) \tag{2.9}$$

The key difference between Wavelet transform and Gabor transform is shown in Figure 1. Gabor transform localizes the signal in the same size as the localization in frequency space while wavelet transform has a wider localization in time for longer wavelength (but still of the same area).

To complete our discussion, we should look at how we can generate other wavelet. We have already seen can generate other wavelet from a single wavelet, called mother wavelet. Mother wavelet can be scaled with a and translated with b. In addition, we can also construct new wavelet by using the following theorem.

Theorem 2.4. *If ψ is a wavelet and ϕ is bounded, then $\psi \cdot \phi$ is a wavelet.*

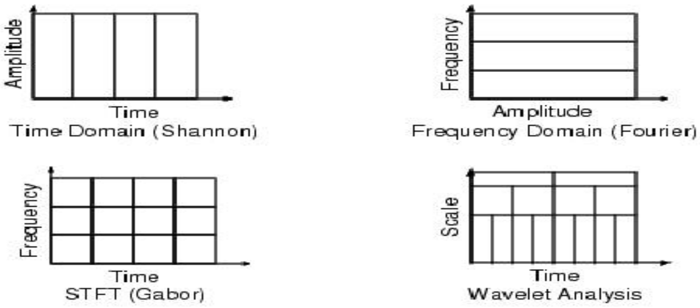

Figure 1. *Visualization of Fourier, Gabor, Shanon and Wavelet transform*

Algorithm 2.1 Randomized SVD

Find orthogonal Q such that $A \approx QQ^T A$
Take a subspace of A: $B = Q^T A$
SVD on subspace: $B = S\Sigma V^T$
Project back to original space: $U = QS$
return $A \approx U\Sigma V^T$

2.2. Randomized SVD. Singular value decomposition (SVD) is a very useful dimensionality reduction technique. While eigen-decomposition of a matrix is not guaranteed for a non-symmetric matrix, SVD of a matrix always exist. SVD of a $m \times n$ matrix A is given by

$$A = U\Sigma V^T \tag{2.10}$$

where U is the $m \times r$ orthogonal left singular matrix, Σ is the $r \times r$ diagonal singular values and V^T is the $r \times n$ orthogonal right singular matrix.

When dealing with big data, the matrix A will grow exponentially with the number of points. In most big data application, the matrix A is not able to be loaded on to the memory of a computer and SVD can be extremely time-consuming to compute. In 2009, Halko et al. proposed a new technique to approximate matrix decompositions by performing SVD on a smaller subspace of original matrix. This method is called randomized SVD (rSVD). The main challenge of the algorithm shown in Algorithm 2.1 to do that is finding a suitable matrix Q to project the matrix A to a subspace. The proposed method is to construct a random matrix Ω and perform a QR decomposition on the random projection of A, $Y = A\Omega = QR$. Special structured random matrices such as sub-sampled random Fourier transform and sub-sampled randomized Hadamard transform are proposed to improve the approximation by driving down the spectrum of Y.

3. Algorithm Implementation and Development.

3.1. Finding Q. To find the matrix Q, we will implement the simple random normal matrix method as shown in Algorithm 3.1

3.2. CWT Wavelet Selection. The choice of wavelet for CWT is chosen based on the reconstruction ability of the wavelet. We noticed that Morlet wavelet is not able to reconstruct the data well. We chose the derivative of Gaussian (DOG) wavelet.

Algorithm 3.1 Finding Orthogonal projection matrix

1: Input: $m \times n$ matrix A
2: Construct $n \times p$ random normal matrix X
3: Perform projection $Y = AX$
4: QR decompose $Y = QR$
5: **return** Q

3.3. Future State Prediction. To predict the future with the modes that we have extracted, we turn to dynamic mode decomposition (DMD). DMD is useful in this case because it decomposes a signal into functions with Fourier time dependence. Unlike neural network, DMD has the ability to extrapolate. We use the Matlab code written by Bingni Brunton.

4. Computational Results. According to the National Centers For Environmental Information, El Nino is a phenomenon in the Pacific Ocean characterized by a five consecutive 3-month running mean of sea surface temperature [4]. The threshold for El Nino (La Nina) is $\pm 0.5°C$. from the mean. The average period for El Nino is five years but it varies from 2 to 7 years.

The raw data for sea surface temperature is a $360 \times 180 \times 1455$ matrix. Even though we can fit the matrix onto the computer, it is not possible to perform regular SVD with a 4GB ram. Thus, we use randomized SVD to reduce the dimensionality to rank 50. The eigen-spectrum does not fall off exponentially to show a clear truncation point and "50" is chosen based on a visual comparison with the ground truth. (Note that any rank greater than 100 takes a very long time to compute even with rSVD). The first 5 eigen-maps obtained from the right singular matrix are shown in Figure 2, with rescaled amplitude for a better visual comparison. The first dominant mode (a) corresponds to the global average of the surface temperature. Second (b), third (c) and forth modes (d) correspond to the seasonal changes of the earth. The fifth mode (e) contributes to the El Nino effect with a clear fluctuation signature at the equator. The interesting mode (e) has aperiodic behavior which is related to the El Nino effect. The oscillations of these modes are shown in Figure 3 and they support our previous observations. In addition, given that the fifth mode contributes to El Nino, we looked at the next 5 modes as show in Figure 4, and found that they also contributes to the El Nino.

Next, we perform continuous wavelet transform(CWT) to extract meaningful features on different time and space scales. First, we look at the points on El Nino effect region and perform CWT for every point at different time scales. We obtain scalogram like the one shown in Figure 5 for each point. It is clear that the high frequency modes (inverse of the scaling power) is the seasonal mode of the hemisphere., alternating between hot and cold for every year. Interesting features for El Nino and La Nina are the modes with periods of 2 to 8 years (low frequencies). And, by keeping only the that frequency range, we are able to extract the El Nino and La Nina modes as shown in Figure 6.

Next, we perform a spatial CWT for every time step by flattening the data. By doing so, we are sacrificing the vertical correlation between neighboring patch for speed. At small spatial scales, we are able to capture the temperature variations near the coast as shown in Figure 7. At large spatial scales, we can see that the mode describe the latitudinal variation

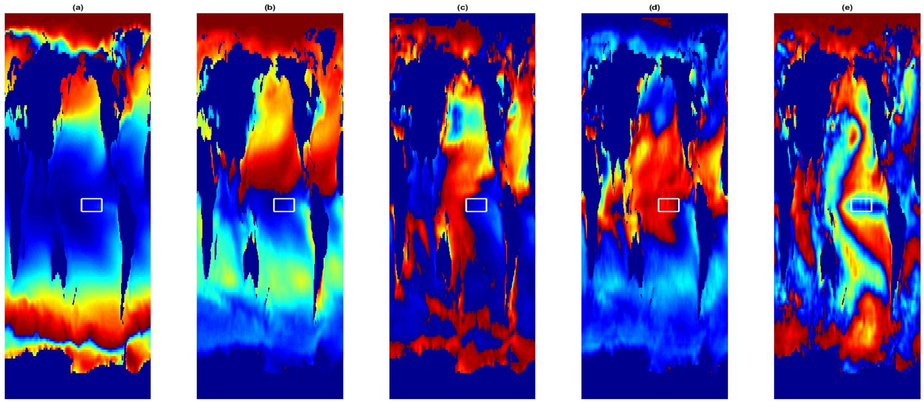

Figure 2. *First five eigen-map for sea surface temperature. For the purpose of comparison, the amplitudes are rescaled such that (blue to red) represents a range from -1 to 1. The average temperature in white box is used to identify anomaly for El Nino and La Nina.*

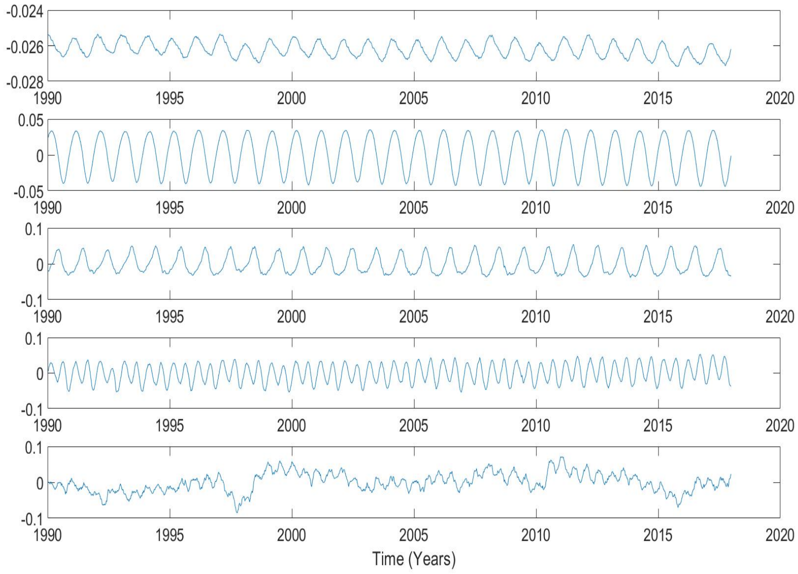

Figure 3. *The dynamic for the first 5 eigen-map shown in Figure 2 (from top to bottom).*

in temperatures. And, the spatial length scale for El Nino is between the two extremes.

Finally, we chose the area within the white rectangle shown in Figure 2 to determine the anomaly which is close the to the definition defined by the National Oceanic and Atmospheric Administration (NOAA). We average the temperature within that region and look at the anomaly in temperature. We use the temporal modes that we found as our predictors. Then,

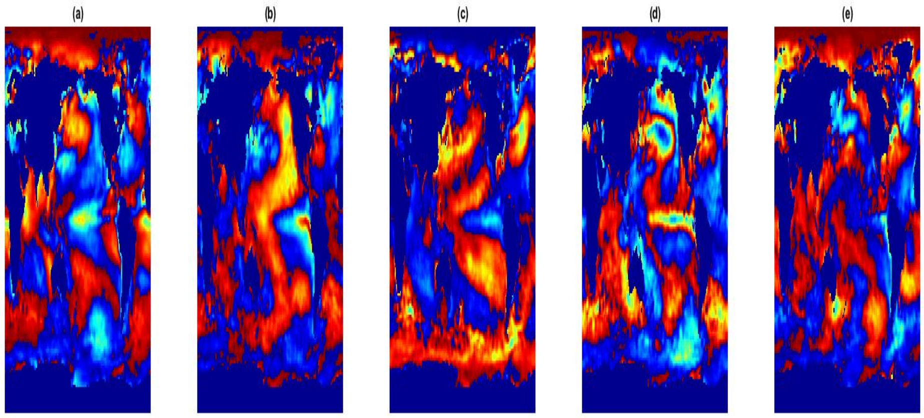

Figure 4. *Eigen-map from the 6th to 10th modes for sea surface temperature. For the purpose of comparison, the amplitudes are rescaled such that (blue to red) represents a range from -1 to 1.*

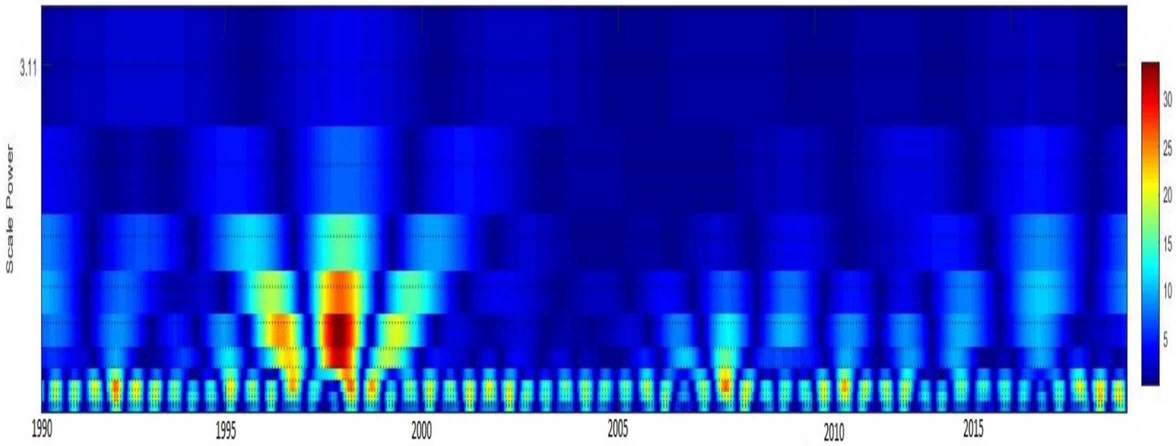

Figure 5. *Scalogram for a point on the El Nino effect region.*

we apply dynamic mode decomposition (DMD) on the modes in the region in the white box as described earlier to perform future state predictions. We use the data from 1990 to 2014 as training set and try to predict from 2014 to 2018 (and compare it with actual data). Whenever the temperature raises above the threshold, we will know if its El Nino or La Nina. The result of the prediction in shown in Figure 8. The problem with this method is that we need a certain amount of training data and prediction window is limited to a fraction of the fitted data since the modes form a singular matrix.

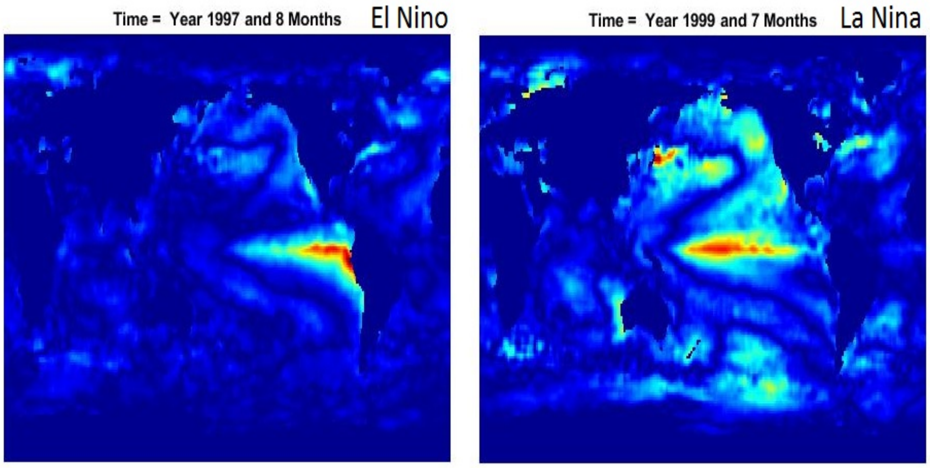

Figure 6. *Absolute value of El Nino temporal modes and La Nina temporal mode (with period of 2-8 years) from CWT.*

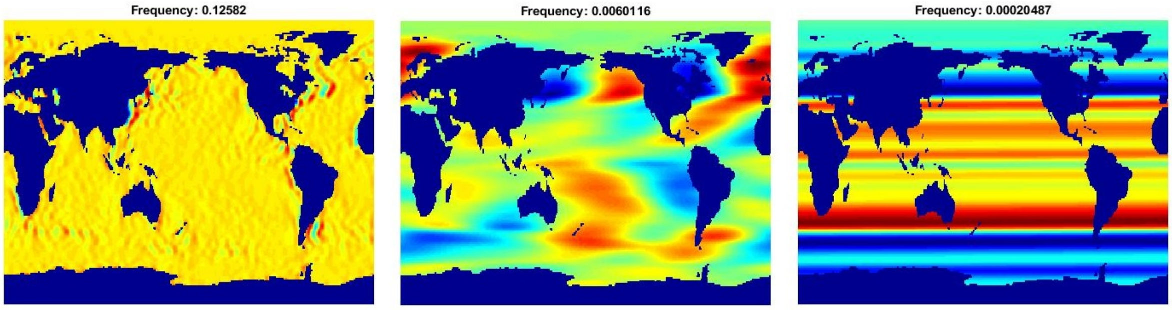

Figure 7. *Spatial modes from CWT at different spatial sampling frequencies (high to mid to low).*

5. Summary and Conclusions. Using randomized algorithm, we are able to perform svd on this large dataset. Consequently, we are able to extract meaningful spatial and temporal modes from the data using continuous wavelet transform. We can classify the modes that contributes to global oscillations, seasonal variations and El Nino,La Nina phenomena at the equatorial. The future state prediction is quite limited in the window prediction size because our choice of method for prediction with DMD is plague with ill-conditioned matrix for our predictors.

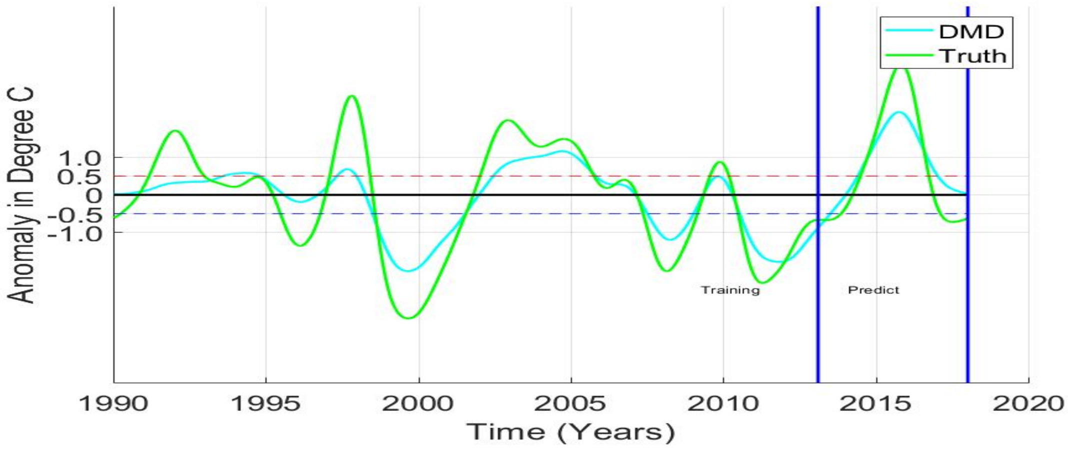

Figure 8. *Prediction for El Nino and La Nina with DMD. The dashed lines (red and blue) represents the threshold for El Nino and La Nina respectively.*

Appendix A. Important Matlab functions used and brief explanation [3].

1. **cwtft2** - return the 2D continuous wavelet transform using Fast Fourier transform algorithm
2. **cwtft** - return the 1D continuous wavelet transform using Fast Fourier transform algorithm
3. **qr** - return the QR decomposition of input matrix

Appendix B. MATLAB codes.

B.0.1. Main Script.

1. Main Script

```
% AMATH 563 Homework 3 Spring 2018
% Tun Sheng Tan

clear all; close all; clc;

% Preparation + Dimensionality reduction
[sst,mask,N,L]=loadData();
A = reshape(sst, [N, L]);
[U,S,V] = rsvd2(A, 100);  % Perform Randomized SVD with standardized
    stuff
m = 360; n = 180;


Q = U*S*V';                    % Reconstruct low rank
P = reshape(Q, [m n L]);
visualInspect(P,sst,mask,2,0.001,false,true); % Visual Inspection

clear A;
%%
```

```
% 1. Dimensionality reduction and spatio-temporal modes
maskN = mask;
maskN(maskN == 0) = nan;
modes = {1,2,3,4,5};
% modes = {6,7,8,9,10};
% eigenMap(U,modes,maskN);                          % Eigen-Maps
eigenTrend(V,modes,mask);                         % Eigen-Trends
elNinoModes = {5,6,7,8,9,10};
% reconModes(U,S,V,elNinoModes,mask,100,0.01,true); % Selective Recon

%%
% 2. Multi-resolution analysis to extract features on different time
%    and space scales

% Plot temporal MRA modes modes
% 1Months, 3 Months, 6 Months, 1 Year, 2 Years
% scales = [4,13,26,52, 104];
% wave = 'dog';
% clear temporalModes;
% temporalModes=temporalMRA2(Q,wave,scales,false,mask,100,0.01);

% Plot spatial MRA modes modes
% scales = [3,5,10,15,20];wave = 'morl';
% spatialModes=spatialMRA(P,wave,scales,true,mask,100,0.01);
%% Temporal CWTFT
t = 1:1:L;
t = 1990+t/52;
ma = movmean(Q(sub2ind([m,n], 278,77+19),:), 200);
% ma = smoothdata(Q(sub2ind(size(K), 278,77+19),:),'movmean',13);
sig = {Q(sub2ind([m,n], 278,77+19),:), 1/52};
cwtstruct=cwtft(sig,'wavelet','dog','plot');
colormap jet;

scales = cwtstruct.scales;
freq = cwtstruct.frequencies;
% freq = 1./(scales);
% contour(t,freq,abs(cwtstruct.cfs));
% xlabel('Years'); ylabel('Pseudo-frequency');

line([1,1],[-6,3],'Color','white')
line([3,3],[-6,3],'Color','white')
line([4,4],[-6,3],'Color','white')
line([5,5],[-6,3],'Color','white')
line([6,6],[-6,3],'Color','white')
line([7,7],[-6,3],'Color','white')
line([8,8],[-6,3],'Color','white')
line([9,9],[-6,3],'Color','white')
line([10,10],[-6,3],'Color','white')
line([11,11],[-6,3],'Color','white')
line([12,12],[-6,3],'Color','white')
line([13,13],[-6,3],'Color','white')
line([14,14],[-6,3],'Color','white')
line([15,15],[-6,3],'Color','white')
line([16,16],[-6,3],'Color','white')
```

```
line([17,17],[-6,3],'Color','white')
line([18,18],[-6,3],'Color','white')
line([19,19],[-6,3],'Color','white')
line([20,20],[-6,3],'Color','white')
text(1, 3,'El','Color','white')
text(4, 3,'El','Color','white')
text(7, 3,'El','Color','white')
text(12, 3,'El','Color','white')
text(14, 3,'El','Color','white')
text(16, 3,'El','Color','white')
text(19, 3,'El','Color','white')
text(5, 3,'La','Color','white')
text(8, 3,'La','Color','white')
text(10, 3,'La','Color','white')
text(17, 3,'La','Color','white')

line([1991 1991], [0 25], 'Linewidth',2)
line([1993 1993], [0 25], 'Linewidth',2)
line([1994 1994], [0 25], 'Linewidth',2)
line([1995 1995], [0 25], 'Linewidth',2)
line([1996 1996], [0 25], 'Linewidth',2)
line([1997 1997], [0 25], 'Linewidth',2)
line([1998 1998], [0 25], 'Linewidth',2)
line([1999 1999], [0 25], 'Linewidth',2)
line([2000 2000], [0 25], 'Linewidth',2)
line([2001 2001], [0 25], 'Linewidth',2)
line([2002 2002], [0 25], 'Linewidth',2)
line([2003 2003], [0 25], 'Linewidth',2)
line([2004 2004], [0 25], 'Linewidth',2)
line([2005 2005], [0 25], 'Linewidth',2)
line([2006 2006], [0 25], 'Linewidth',2)
line([2007 2007], [0 25], 'Linewidth',2)
line([2008 2008], [0 25], 'Linewidth',2)
line([2009 2009], [0 25], 'Linewidth',2)
line([2010 2010], [0 25], 'Linewidth',2)
text(1991, 5,'El','Color','black')
text(1994, 5,'El','Color','black')
text(1997, 5,'El','Color','black')
text(2002, 5,'El','Color','black')
text(2004, 5,'El','Color','black')
text(2006, 5,'El','Color','black')
text(2009, 5,'El','Color','black')
text(1995, 3,'La','Color','black')
text(1998, 3,'La','Color','black')
text(2000, 3,'La','Color','black')
text(2007, 3,'La','Color','black')

% i = 9;
% j = 14;
% % Frequency Selection 19
% freq = cwtstruct.frequencies;
% cwtstruct.cfs(1:i-1,:) = 0;
% cwtstruct.cfs(j+1:19,:) = 0;
%
```

```
% % Reconstruction
% xrec = icwtft(cwtstruct);
% clear cwtstruct;
%
% % Check reconstruction map
% figure;
% subplot(2,1,1)
% plot(t,xrec)
% grid on
% title("Sampling size: "+1/freq(i)+" and " + 1/freq(j) +" Years");
%
% subplot(2,1,2)
% ma = movmean(xrec,52);
% plot(t,xrec)
% line([ 1990 2010], [-0.5, -0.5]);
% line([ 1990 2010], [0.5, 0.5]);
% grid on
%
% clear i j;

%% Run Full Temporal CWT
temporary = zeros([m n L]);
temporalModes = cell(N,1);
% for i = 210:300 %201:280 187:232
%     for j = 70:100 %77:107 87:94
for i = 1:m %201:280 187:232
   for j = 1:n %77:107 87:94
     cwtstruct=cwtft({Q(sub2ind([m,n], i, j),:), 1/52},'wavelet','dog');
     temporalModes{sub2ind([m,n], i, j)} = cwtstruct;
     cwtstruct.cfs(1:5,:) = 0;
     cwtstruct.cfs(14:19,:) = 0;
     temporary(i,j,:) = icwtft(cwtstruct);
   end
end
%% Temporal Movie
figure;
tempcfs = zeros([m n L]);
for i = 1:m
    for j = 1:n
        cfs = temporalModes{sub2ind([m,n], i, j)}.cfs;
        tempcfs(i,j,:) = abs(sum(cfs(9:14,:)));
    end
end
for i = 480:500

%     subplot(1,2,1)
%     t = ((temporary(:,:,i)).*mask).';
%     imagesc(t)
%     colorbar
%     colormap jet
%     title("Time =  Year "+ (1990+floor(i/52)) + " and " ...
%      + round(12*mod(i,52)/52)+" Months");


```

```
%      subplot(1,2,2)
    imagesc((tempcfs(:,:,i).*mask)');
    colormap jet
    colorbar
    axis off tight
  title("Time =  Year "+ (1990+floor(i/52)) + " and " ...
    + round(12*mod(i,52)/52)+" Months");
    pause(0.001)
end


%%
h = figure;
% axis off tight manual
% filename = 'testAnimated.gif';
for i = 1:10
    K = reshape(temporalModes(:,:,i,4), [m*n 1]);
    K = reshape(K, [m n]);
    imagesc((K.*mask)');
    colormap jet
    colorbar
    title("Time =  Year "+ (1990+floor(i/52)) + " and " ...
            + round(12*mod(i,52)/52)+" Months");
%      frame = getframe(h);
%      im = frame2im(frame);
%      [imind, cm] = rgb2ind(im,256);
    % Write to the GIF File
%      if i == 1
%        imwrite(imind,cm,filename,'gif', 'Loopcount',inf);
%      else
%        imwrite(imind,cm,filename,'gif','WriteMode','append');
%      end
    pause(0.001)
end

%% Spatial Mode Inspection (Low, Mid, High)
clear xrec;
cwtstruct=cwtft(Q(:,60),'wavelet','dog');
freq = cwtstruct.frequencies;

% Frequency Selection
% cwtstruct.cfs(freq > 0.006 | freq < 0.0002,:) = 0;
cwtstruct.cfs(1:24,:) = 0;
cwtstruct.cfs(26:end,:) = 0;

% t = 1:1:(m*n);
% T1 = 0; T2 = m*n;
% F1 = 0.002; F2 = 0.004;
% spatialModes = abs(cwtstruct.cfs);

% cfs = cwtstruct.cfs; freq = cwtstruct.frequencies;
% imagesc(abs(cfs))
% axis tight;
% xlabel('Seconds'); ylabel('Hz');
```

```
% title('CWT with Mexican Hat Wavelet');


% Reconstruction
xrec = icwtft(cwtstruct);
clear cwtstruct;


% Check reconstruction map
figure;
K = reshape(xrec, [m n]).*mask;
% subplot(1,2,1);
imagesc(K');
axis off
colormap jet
title("Frequency: "+freq(25))

% cwtft(xrec,'wavelet','dog','plot');
% cwtft(Q(:,50),'wavelet','mexh','plot');

%% Spatial Mode Movie
clear spatialModes;
clear recon xrec;
spatialModes = cell(L,1);
xrec = zeros(N,L);
for i = 1:L
    cwtstruct=cwtft(Q(:,i),'wavelet','mexh');
    spatialModes{i} = cwtstruct;
    xrec(:,i) = icwtft(cwtstruct).';
    if (mod(i,500) == 0)
        i
    end
end
for i = 1:L
    cwtstruct = spatialModes{i};
    cwtstruct.cfs(20:30,:) = 0;
    xrec(:,i) = icwtft(cwtstruct).';
    if (mod(i,200) == 0)
        i
    end
end

figure;
set(gcf, 'Position', [500, 500, 500, 500])
for i = 1:500
    subplot(1,2,1)
    imagesc((reshape(xrec(:,i),[m,n]).*mask)');
    colormap jet
    axis off tight
    colorbar
    title("Time =  Year "+ (1990+floor(i/52)) + " and " ...
              + round(12*mod(i,52)/52)+" Months");
    line([190,190],[87,93],'Color','white','Linewidth',2)
    line([240,240],[87,93],'Color','white','Linewidth',2)
```

```
    line([190,240],[87,87],'Color','white','Linewidth',2)
    line([190,240],[93,93],'Color','white','Linewidth',2)
    subplot(1,2,2)
    cfs = spatialModes{i}.cfs;
    imagesc((reshape(abs(sum(cfs(8:15,:))),[m,n]).*mask)');
    colormap jet
    colorbar
    axis off tight
    line([190,190],[87,93],'Color','white','Linewidth',2)
    line([240,240],[87,93],'Color','white','Linewidth',2)
    line([190,240],[87,87],'Color','white','Linewidth',2)
    line([190,240],[93,93],'Color','white','Linewidth',2)
    title(indicator((1990+floor(i/52))));
    pause(0.001)
end


%%
% 3. Extract El Nino mode from the system to build a predictor
%    for transient phenomenon.

%% Anomaly Calculation
t = 1:1:L;
t = 1990+t/52;
meaninspace = zeros([L 1]);
figure;
for i = 1:L
   meaninspace(i)  = mean2(P(190:240,87:93,i));
end
ma = smooth(meaninspace,52);
yp = smooth(meaninspace-ma,13);
yn = smooth(meaninspace-ma,13);
yp(yp < 0) = 0;
yn(yn > 0) = 0;

b1 = bar(t,yp);
b1.FaceColor = 'red';
hold on
b2 = bar(t,yn);
b2.FaceColor = 'blue';

line([t(1),t(L)],[0.5,0.5],'LineStyle','--','Color','red')
line([t(1),t(L)],[0,0],'Linewidth',2,'Color','black')
line([t(1),t(L)],[-0.5,-0.5],'LineStyle','--','Color','blue')
ylabel("Anomaly in Degrees C")
xlabel("Year")
set(gca, 'FontSize', 20);
grid on

%% Prepare datafor SVM
t = 1:1:L;
t = 1990+t/52;
meaninspace = zeros([L 1]);
% tempcfs = zeros([m n L]);
```

```
% for i = 1:m
%     for j = 1:n
%         cfs = temporalModes{sub2ind([m,n], i, j)}.cfs;
%         tempcfs(i,j,:) = (sum(cfs(9:14,:)));
%     end
% end
for i = 1:L
    meaninspace(i)  = mean2(tempcfs(190:240,87:93,i));
end
% ma = smooth(meaninspace,52);
yp = smooth(meaninspace,1);
yn = smooth(meaninspace,1);
yp(yp < 0) = 0;
yn(yn > 0) = 0;

b1 = bar(t,yp);
b1.FaceColor = 'red';
hold on
b2 = bar(t,yn);
b2.FaceColor = 'blue';

line([t(1),t(L)],[10,10],'LineStyle','--','Color','red')
line([t(1),t(L)],[0,0],'Linewidth',2,'Color','black')
line([t(1),t(L)],[-10,-10],'LineStyle','--','Color','blue')
ylabel("Anomaly in Degrees C")
xlabel("Year")
set(gca, 'FontSize', 20);
grid on

labelsP = ones([L 1]); % Labels for El Nino
labelsN = ones([L 1]); % Labels for La Nina
labelsP(yp < 10) = 0;
labelsN(yn > -10) = 0;
%%
L = 1455;
Lpredict = L;
Ltrain = L;
% p = tempcfs(190:240,70:90,1:Ltrain);
p =  tempcfs(:,:,1:Ltrain);
% p = p./std2(p);
[mp np Ltrain] = size(p);
[Phi, mu, lambda, diagS, x0] = DMD(reshape(p, [mp*np Ltrain]));
[Xhat z0] = DMD_recon(Phi, lambda, x0, Lpredict);
Xhat = reshape(Xhat, [mp np Lpredict]);

%%
ma = zeros([Lpredict 1]);
for i = 1:Lpredict
    ma(i)  = mean2(real(Xhat(190:240,70:90,i)));
end

ta = 1:1:Lpredict;
ta = 1990+ta/52;
figure;
```

```
hold on
plot(ta, ma,'Linewidth',2,'Color','cyan')
plot(t(1:Lpredict), meaninspace(1:Lpredict),'Linewidth',2,'Color','green')
line([t(1),t(L)],[10,10],'LineStyle','--','Color','red')
line([t(1),t(L)],[0,0],'Linewidth',2,'Color','black')
line([t(1),t(L)],[-10,-10],'LineStyle','--','Color','blue')
line([t(1200),t(1200)], [-100,100],'Linewidth',2,'Color','blue')
line([t(Ltrain),t(Ltrain)], [-100,100],'Linewidth',2,'Color','blue')
text(t(1250), -50, 'Predict');
text(t(1000), -50, 'Training');
xlabel('Time (Years)')
legend('DMD','Truth')
hold off
grid on
```

B.1. Helper Functions.

1. rSVD written by Antoine Liutkus

```
function [U,S,V] = rsvd2(A,K)

% random SVD
% Extremely fast computation of the truncated Singular Value Decomposition,
% using randomized algorithms as described in Halko et al. 'finding
% structure with randomness
%
% usage :
%
%  input:
%  * A : matrix whose SVD we want
%  * K : number of components to keep
%
%  output:
%  * U,S,V : classical output as the builtin svd matlab function

% Antoine Liutkus  (c) Inria 2014

[M,N] = size(A);
P = min(2*K,N);
X = randn(N,P);
Y = A*X;
W1 = orth(Y);
B = W1'*A;
[W2,S,V] = svd(B,'econ');
U = W1*W2;
K=min(K,size(U,2));
U = U(:,1:K);
S = S(1:K,1:K);
V=V(:,1:K);
```

2. DMD written by Bingni Brunton [1]

```
function [Phi, mu, lambda, diagS, x0] = DMD(Xraw, varargin)
% function [Phi mu lambda diagS x0] = DMD(Xraw, varargin)
```

```
% computes the Dynamic Mode Decomposition of data matrix Xraw
%
% INPUTS:
%
% rows of Xraw are assumed to be measurements
% columns Xraw are assumed to be time points, sampled at equal dt's
%
% optional parameters: {'parameter_name', [default_value]}
%   {'dt', [1]}
%   {'r', [1e32]}          truncate svd basis to first r features
%   {'scale_modes', 1}     0 or 1, scale dmd modes by singular values (
%  energy)
%   {'nstacks', 1}         number of stacks of the raw data

%
% OUTPUTS:
%
% Phi, the modes
% mu, the fourier spectrum of modes (mu = log(lambda)/dt)
% lambda, the DMD spectrum of modes
% diagS, the singular values of the data matrix
% x0, the initial condition vector corresponding to Phi
%
%
% BWB, Nov 2013
% mods: return diag(S), BWB Dec 2013
%       added corrected truncation of modes, BWB Jan 2014
%       added scaled modes, BWB Mar 2014
%       added stacking to get augmented data matrix, BWB Mar 2014
%       added option to use optimal singular value hard threshold (SVHT,
%            see Gavish & Donoho 2013), BWB Jun 2014

%% input parsing
p = inputParser;

% required inputs
p.addRequired('Xraw', @isnumeric);

% parameter value iputs
p.addParameter('dt', 1, @(x)isnumeric(x) && x>0);
p.addParameter('r', 1e32, @(x)isnumeric(x) && x>0);
p.addParameter('use_optimal_SVHT', 0, @isnumeric);
p.addParameter('scale_modes', 1, @isnumeric);
p.addParameter('nstacks', 1, @(x)isnumeric(x) && x>0);

% now parse the inputs
p.parse(Xraw, varargin{:});
inputs = p.Results;

%% stacking the data matrix
if inputs.nstacks > 1,
    Xaug = [];
    for st = 1:inputs.nstacks,
        Xaug = [Xaug; Xraw(:, st:end-inputs.nstacks+st)]; %#ok<AGROW>
```

```
    end;

    X = Xaug(:, 1:end-1);
    Y = Xaug(:, 2:end);
else
    X = Xraw(:, 1:end-1);
    Y = Xraw(:, 2:end);
end;

%% DMD
[U, S, V] = svd(X, 'econ');
diagS = diag(S);

% if we want to use optimal singular value hard threshold, compute the
% truncation order r
if inputs.use_optimal_SVHT > 0,
    beta = size(X,1)/size(X,2); if beta > 1, beta = 1/beta; end;
    omega = optimal_SVHT_coef(beta,0) * median(diagS);
    r = sum(diagS > omega);
else
    r = inputs.r;
end;

if r >= size(U,2),
    % no truncation
    Atilde = U'*Y*V/S;

    if inputs.scale_modes == 0,
        [W, D] = eig(Atilde);
    else % scaling modes
        Ahat = S^(-1/2) * Atilde * S^(1/2);
        [What, D] = eig(Ahat);
        W = S^(1/2) * What;
    end;

    Phi = Y*V/S*W;
else
    % truncate modes
    U_r = U(:, 1:r);
    S_r = S(1:r, 1:r);
    V_r = V(:, 1:r);

    Atilde = U_r' * Y * V_r / S_r;

    if inputs.scale_modes == 0,
        [W_r, D] = eig(Atilde);
    else % scaling modes
        Ahat = (S_r^(-1/2)) * Atilde * (S_r^(1/2));
        [What, D] = eig(Ahat);
        W_r = S_r^(1/2) * What;
    end;

    Phi = Y*V_r/S_r*W_r;
end;
```

```

lambda = diag(D);
mu = log(lambda)/inputs.dt;
x0 = X(:,1);
```

```
function [Xhat z0] = DMD_recon(Phi, lambda, x0, M, varargin)
% function [Xhat z0] = DMD_recon(Phi, lambda, x0, M, varargin)
% reconstructs the data matrix X from its Dynamic Mode Decomposition,
% where [Phi mu lambda diagS x0] = DMD(X)
%
% INPUTS:
%
% Phi, lambda, x0 are outputs from DMD.m
% M is the number of time points desired in the reconstruction
%
% optional parameters: {'parameter_name', [default_value]}
%   {'keep_modes', []}, if empty, use all modes in reconstruction of Xhat
%                       if not empty, use only modes indexed here in the
%                       reconstruction of Xhat
%
%
% OUTPUTS:
%
% Xhat, the reconstructed data matrix, will be n x M where
%       n is the length of x0
%
%       NOTE -- Xhat may come out with non-zero imaginary components; if
%       your original data matrix X is strictly real valued, then you need
%       to use real(Xhat)
%
%
% BWB, Apr 2014

%% input parsing
p = inputParser;

% required inputs
p.addRequired('Phi', @isnumeric);
p.addRequired('lambda', @isnumeric);
p.addRequired('x0', @isnumeric);
p.addRequired('M', @(x)isnumeric(x) && x>0);

% parameter value iputs
p.addParamValue('keep_modes', [], @isnumeric);

% now parse the inputs
p.parse(Phi, lambda, x0, M, varargin{:});
inputs = p.Results;

%% if we're only using a subset of the modes in the reconstruction
if ~isempty(inputs.keep_modes),
    Phi = Phi(:, inputs.keep_modes);
    lambda = lambda(inputs.keep_modes);
```

```
end;

%% reconstruct
z0 = Phi\x0; % initial conditions
% z0 = pinv(Phi)*x0;
Z = zeros(length(z0), M);
for tt = 1:M,
    Z(:, tt) = z0 .* lambda.^tt;
end;

Xhat = Phi * Z;
```

3. Load Data

```
function [sst,mask,N,L]=loadData()
    sst = ncread('sst.wkmean.1990-present.nc','sst');
    mask = ncread('lsmask.nc','mask');

%      maskN = mask;
    mask(mask == 0) = nan;
    N = 360*180;
    L = 1400;

    L = 1455;
end
```

4. Eigen-map and eigen-trend

```
function eigenMap(U,modes,mask)
    % Plot eigen maps

    figure();
    titles = {'(a)','(b)','(c)','(d)','(e)'};
    % modes = {6,7,8,9,10};
    % modes = {11,12,13,14,15};
    for i = 1:5
        % Eigen-waves
        subplot(1,5,i);

        % Rescale for better visual
        K = U(:,modes{i});
        a = min(K);
        b = max(K);
        for j = 1:64800
            if K(j) > 0
                K(j) = 2*K(j)/b -1;
            elseif K(j) < 0
                K(j) = -2*K(j)/a + 1;
            end
        end
        u = reshape( K , [360, 180]);
        u = u.*mask;
        imagesc(u(:,:)');
        colormap 'jet';
        caxis([-1 1]);
        axis off;
```

```
        title(titles{i});
        line([190,190],[87,93],'Color','white','Linewidth',2)
        line([240,240],[87,93],'Color','white','Linewidth',2)
        line([190,240],[87,87],'Color','white','Linewidth',2)
        line([190,240],[93,93],'Color','white','Linewidth',2)
    end

end
```

```
function eigenTrend(V,modes,mask)
    timescale = 1990+(1:1455)/52;
    figure();
    for i = 1:5
        % Eigen-trends
        subplot(5,1,i);
        plot(timescale,V(:,modes{i}));
        set(gca, 'FontSize', 20);
    end
    xlabel("Time (Years)")
end
```

5. Visualization Script

```
function visualInspect(P,sst,mask,N,pauseN,elnino,colorbarOn)
    figure();
    for j = 1:N
        P_mask(:,:,j) = P(:,:,j).*mask;
        if elnino
            imagesc(P_mask(120:290,60:120,j)')
        else
           imagesc(P_mask(:,:,j)')
        end
        axis off
        colormap jet
        if colorbarOn
            colorbar
        end

        line([190,190],[87,93],'Color','white','Linewidth',2)
        line([240,240],[87,93],'Color','white','Linewidth',2)
        line([190,240],[87,87],'Color','white','Linewidth',2)
        line([190,240],[93,93],'Color','white','Linewidth',2)
        title("Truth: t = Year "+ (1990+round(j/52)) );
        pause(pauseN)
        %      subplot(2,1,2);
        %      sst_mask(:,:,j) = sst(:,:,j).*mask;
        %      imagesc(sst_mask(:,:,j)')
    end
end
```

REFERENCES

[1] B. Brunton, *Dmd-neuro*. https://github.com/bwbrunton/dmd-neuro, 2016.

[2] N. Kutz, *Data-Driven Modeling & Scientific Computation*, Oxford, 2013.
[3] MATLAB and S. Toolbox, 9.3.0.713579 (R2017b).
[4] N. Oceanic and A. Administration, *Equatorial pacific sea surface temperatures*. https://www.ncdc.noaa.gov/teleconnections/enso/indicators/sst/, 2018.