ATMO 615 Midterm Project

Jinjun Liu

November 11, 2022

1 Introduction

This project is aimed to use a statistical model to predict the wind stress anomalies by sea surface temperature anomalies (SSTA). The relationship can be represented as $\tau_s = CT$, where τ_s and T are state vectors of wind stress anomolies and SSTA, respectively, and C is a constant coefficient (a matrix). Our goal is to derive C using Singular Value Decomposition (SVD) analysis on a training dataset, and apply the relationship on a testing dataset.

This project is going to:

1. Formulate a statistical atmosphere model for predicting surface wind stress anomalies for given SST anomalies;
2. Validate the simulated surface wind stress anomalies against observations;
3. Perform sensitivity tests to see how the results depend on the number of SVDs used to compute the anomalies;
4. Discuss whether the results make physical sense and why;

2 Dataset and Method

Surface temperature (T_s), zonal wind stress (u), meridional wind stress (v), and a land mask dataset are given. T_s, u, v are three-dimensional, with two space dimensions and one time dimension. Datasets are divided into two parts: data in 1948-1999 are used as training dataset, and data in 2000-2016 are used as testing dataset.

Steps to perform the SVD analysis:

1. Normalize all the anomaly fields by dividing each of variables by its own standard deviation;
2. Form a state vector for SSTA (T), and another one for zonal and meridional wind stress anomaly (τ_s). Note that τ_s contains both wind stress component and has larger dimension than T;
3. Form a normalized covariance matrix, $A = \tau_s T^{'}$, whose dimension should be $M \times N$, where M is the length for τ_s and N is length for T;
4. Apply SVD on A by calling Matlab's SVD function, which results in $A = USV^{'}$, S is a diagonal matrix with diagonal elements representing singular values - explained squared covariance in each SVD, U contains all wind stress singular vectors and V contains all SST singular vectors, which are self-orthogonal, i.e., $VV^{'} = I$ and $UU^{'} = I$;
5. Project T onto each SST SVD mode, V_i, gives i-th PC time series, $b_i =< V_i^{'}T > / < V_i^{'}Vi >$; Project τ_s onto each wind stress SVD mode, U_i, gives i-th PC time series, $d_i =< U_i^{'}\tau_s > / < U_i^{'}U_{>}$; These pairs of time series should be highly correlated

3 Results

3.1 SVD analysis

Following the above steps, the Python script to perform SVD analysis is below. The first four modes are plotted.

[ ]:
```
import numpy.linalg as la
import xarray as xr
import numpy as np

def normalize_data(x):
    nt, ny, nx = x.shape
    x = x.reshape(nt//12, 12, ny, nx)
    # seasonal mean
    x_mean = x.mean(axis=0, keepdims=True)
    # remove seasonal cycle
    x = x - x_mean
    # reshape time dimension
    x = x.reshape(nt, ny, nx)
    # normalize by standard deviation
    x = x / x.std()
    return x
# read in the data
dir = "./NECP_monthly_mean_data/"
taux_ds = xr.open_dataset(dir + "uflx.sfc.mon.mean.tropics.nc", use_cftime=True)
tauy_ds = xr.open_dataset(dir + "vflx.sfc.mon.mean.tropics.nc", use_cftime=True)
sst_ds = xr.open_dataset(dir + "skt.sfc.mon.mean.tropics.nc", use_cftime=True)
grid_ds = xr.open_dataset(dir + "lsmask.tropics.nc", use_cftime=True)
# get the data
taux_da, tauy_da, sst_da, grid_da = taux_ds.uflx, tauy_ds.vflx, sst_ds.skt,␣
 ↪grid_ds.lsmask
# reverse the latitude dimension to make it increasing
taux_da = taux_da.reindex(lat=taux_da.lat[::-1])
tauy_da = tauy_da.reindex(lat=tauy_da.lat[::-1])
sst_da = sst_da.reindex(lat=sst_da.lat[::-1])
grid_da = grid_da.reindex(lat=grid_da.lat[::-1])
# select data from 1948 to 1999 for training,
# and in a target region (30N-30S, 100E-60W)
lat_min, lat_max = -30, 30
lon_min, lon_max = 100, 300
taux = taux_da.sel(time=slice("1948", "1999"), lon=slice(lon_min, lon_max))
tauy = tauy_da.sel(time=slice("1948", "1999"), lon=slice(lon_min, lon_max))
sst = sst_da.sel(time=slice("1948", "1999"), lon=slice(lon_min, lon_max))
grid = grid_da.sel(lon=slice(lon_min, lon_max))
lat, lon = taux.lat.values, taux.lon.values
time = sst.time.values
# get the data as numpy arrays
```

```
taux, tauy, sst, grid = taux.values, tauy.values, sst.values, grid.values
grid = grid[0,:,:]
# get dimensions
nt, ny, nx = taux.shape
# standard deviation
taux_std = taux.std()
tauy_std = tauy.std()
sst_std = sst.std()
# normalize the data
taux_anom = normalize_data(taux)
tauy_anom = normalize_data(tauy)
sst_anom = normalize_data(sst)
# mask out the land
taux_anom[:, grid == -1] = 0
tauy_anom[:, grid == -1] = 0
sst_anom[:, grid == -1] = 0
# concatenate taux and tauy
tau = np.concatenate((taux_anom, tauy_anom), axis=1)
# reshape vars to space x time
tau = tau.reshape(nt, 2*ny*nx).T
sst = sst_anom.reshape(nt, ny*nx).T
# form matrix A
A = 1/nt * tau @ sst.T
# apply SVD on A
U, S, V = la.svd(A, full_matrices=True)
```

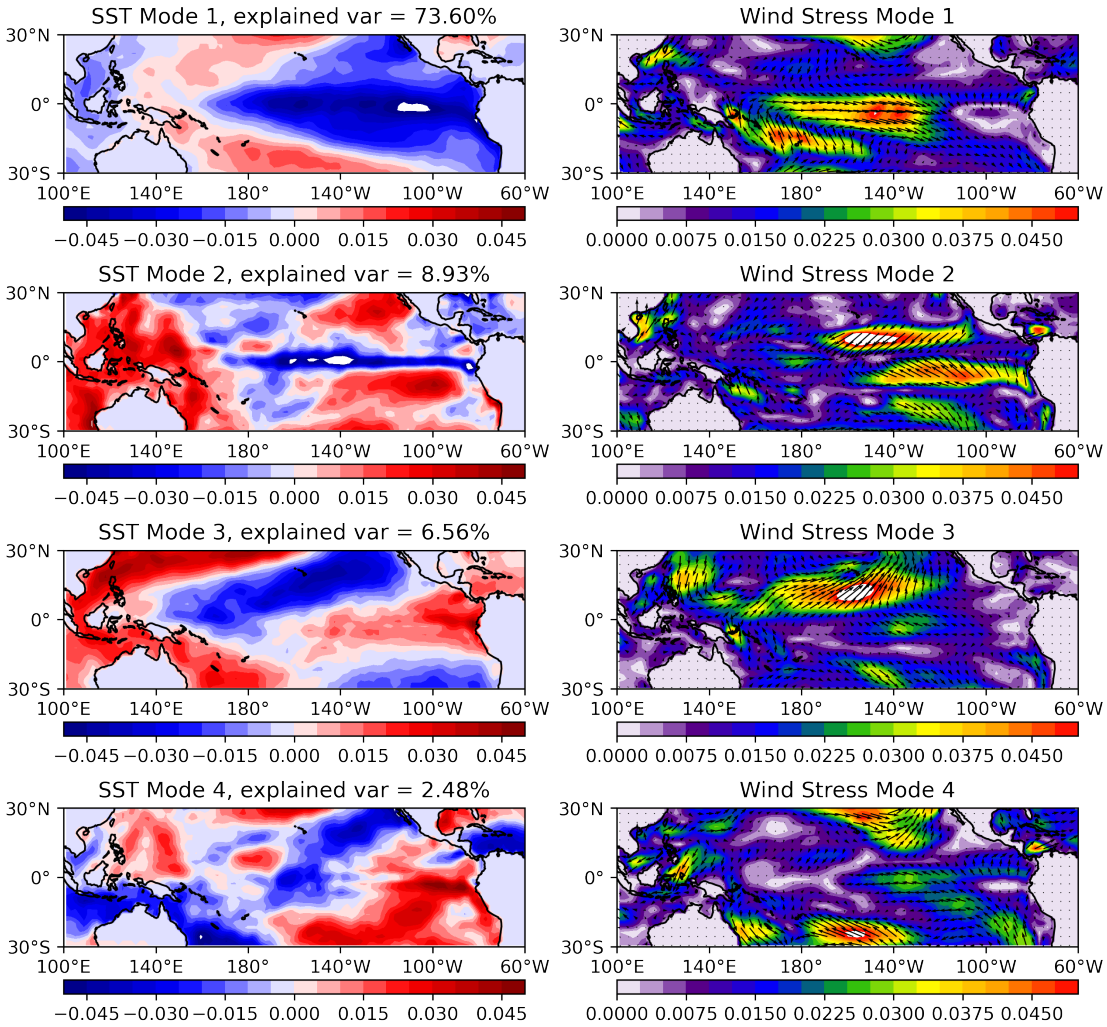
[ ]:
```
print("Shape of U:", U.shape)
print("Shape of S:", S.shape)
print("Shape of V:", V.shape)
n_modes = 10
# get the explained variance
var = (S**2) / (S**2).sum()
# get the first n_modes modes
tau_modes = U[:, :n_modes]
sst_modes = V[:n_modes, :]
sigma_modes = S[:n_modes]
var_modes = var[:n_modes]
print(f"First {n_modes} sigma values:", sigma_modes)
print(f"First {n_modes} explained variance:", var_modes)
# get the spatial patterns
tau_modes = tau_modes.reshape(2*ny, nx, n_modes)
taux_modes = tau_modes[:ny, :, :]
tauy_modes = tau_modes[ny:, :, :]
sst_modes = sst_modes.reshape(n_modes, ny, nx)
```

```
Shape of U: (6848, 6848)
Shape of S: (3424,)
```

```
Shape of V: (3424, 3424)
First 10 sigma values: [343.34018  119.610916 102.532234  62.98886   51.755688
42.648064
  39.13009   35.601448  29.313498  27.253736]
First 10 explained variance: [0.7359658  0.08932026 0.06563405 0.02477056
0.0167234  0.01135552
 0.00955939 0.00791305 0.00536468 0.00463725]
```

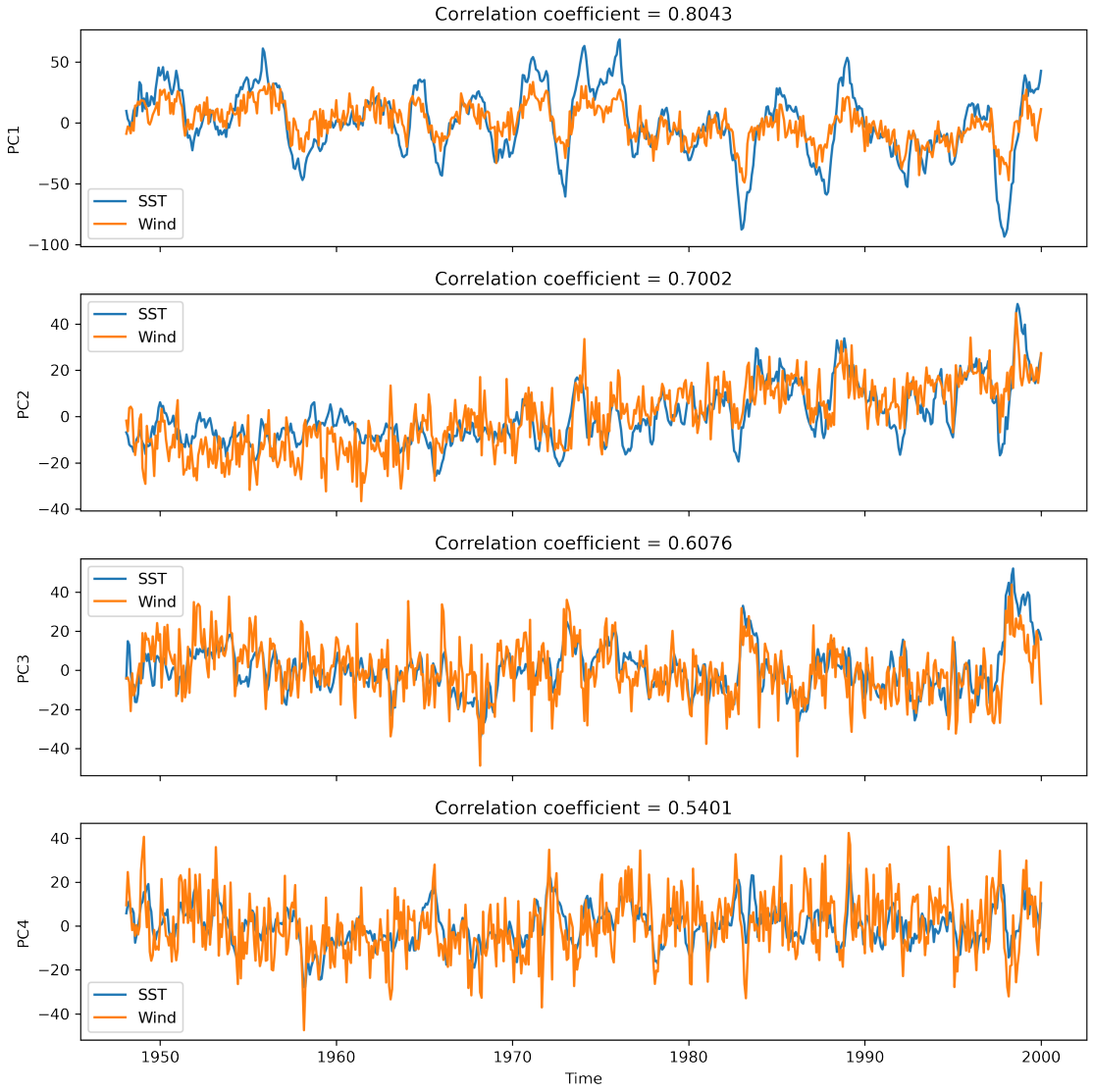
[ ]:
```python
# plot the spatial patterns
import matplotlib.pyplot as plt
import cartopy.crs as ccrs
import cartopy.feature as cfeature
from cartopy.mpl.ticker import LongitudeFormatter, LatitudeFormatter
import cmaps
from mpl_toolkits.axes_grid1.inset_locator import inset_axes

plot_modes = 4 # number of modes to plot
projection = ccrs.PlateCarree(central_longitude=180)
fig, axes = plt.subplots(plot_modes, 2, figsize=(10, 10),
            subplot_kw={'projection': projection}, dpi=300)
for i in range(plot_modes):
    ax = axes[i, 0]
    # set extent
    ax.set_extent([lon_min, lon_max, lat_min, lat_max], crs=ccrs.PlateCarree())
    ax.set_title("SST Mode " + str(i+1) + f", explained var = {var_modes[i]*100:
 ↪.2f}%")
    levels = np.linspace(-0.05, 0.05, 21)
    p1 = ax.contourf(lon, lat, sst_modes[i, :,:], transform=ccrs.PlateCarree(),
                     levels=levels, cmap=cmaps.BlWhRe)
    # lat and lon ticks
    lon_formatter = LongitudeFormatter(zero_direction_label=True)
    lat_formatter = LatitudeFormatter()
    ax.set_xticks(np.arange(lon_min, lon_max+1, 40), crs=ccrs.PlateCarree())
    ax.set_yticks(np.arange(lat_min, lat_max+1, 30), crs=ccrs.PlateCarree())
    ax.xaxis.set_major_formatter(lon_formatter)
    ax.yaxis.set_major_formatter(lat_formatter)
    # add colorbar below the plot
    axins = inset_axes(ax, width="100%", height="10%", loc='lower center',␣
 ↪borderpad=-2.6)
    fig.colorbar(p1, cax=axins, orientation="horizontal")
    ax.coastlines()
    ax = axes[i, 1]
    # set extent
    ax.set_extent([lon_min, lon_max, lat_min, lat_max], crs=ccrs.PlateCarree())
    tau_mag = np.sqrt(taux_modes[:,:,i]**2 + tauy_modes[:,:,i]**2)
    ax.set_title("Wind Stress Mode " + str(i+1))
    levels = np.linspace(0, 0.05, 21)
```

```
    p2 = ax.contourf(lon, lat, tau_mag, transform=ccrs.PlateCarree(),
                levels=levels, cmap=cmaps.WhViBlGrYeOrRe)
    # add wind arrows
    ax.quiver(lon[::2], lat[::2], taux_modes[::2, ::2, i], tauy_modes[::2, ::2,␣
 ↪i],
                transform=ccrs.PlateCarree(), scale=1, color='k')
    # lat and lon ticks
    lon_formatter = LongitudeFormatter(zero_direction_label=True)
    lat_formatter = LatitudeFormatter()
    ax.set_xticks(np.arange(lon_min, lon_max+1, 40), crs=ccrs.PlateCarree())
    ax.set_yticks(np.arange(lat_min, lat_max+1, 30), crs=ccrs.PlateCarree())
    ax.xaxis.set_major_formatter(lon_formatter)
    ax.yaxis.set_major_formatter(lat_formatter)
    axins = inset_axes(ax, width="100%", height="10%", loc='lower center',␣
 ↪borderpad=-2.6)
    fig.colorbar(p2, cax=axins, orientation="horizontal")
    ax.coastlines()
plt.show()
```



[ ]:
```python
# time series of each SVD mode
sst_ts = V[:n_modes, :] @ sst
tau_ts = U[:, :n_modes].T @ tau
# cftime to float
time_fl = [t.year + t.month/12 for t in time]
# plot the time series
fig, axes = plt.subplots(plot_modes, 1, figsize=(10, 10), dpi=300, sharex=True)
for i in range(plot_modes):
    ax = axes[i]
    ax.plot(time_fl, sst_ts[i, :], label=f"SST")
    ax.plot(time_fl, tau_ts[i, :], label=f"Wind")
    # correlation coefficient
    corr = np.corrcoef(sst_ts[i, :], tau_ts[i, :])[0, 1]
    ax.set_title(f"Correlation coefficient = {corr:.4f}")
    if i == plot_modes - 1:
```

```
        ax.set_xlabel("Time")
    ax.set_ylabel(f"PC{i+1}")
    ax.legend()
fig.tight_layout()
plt.show()
```



3.2 Build Statistical Atmosphere Model

The steps to construct a statistical atmosphere model are: 1. Take any SSTA state vector at any given time from the second period, T, normalize it by the standard deviation, and then project it onto a leading SST SVD mode, V_i, to obtain a time series b_i, $b_i =< V_i'T > / < V_i'V_i >$; 2. Take b_i as the corresponding time series for the i-th wind stress SVD, u_i, and multiply b_i to u_i; b_iu_i represents the wind stress variation driven by the i-th SST SVD, v_i; 3. Add all b_iu_i for first L (say

$L = 6$) leading SDVs, $\sum^{L} b_i u_i$, and multiply it by the standard deviation of the wind stress; we obtain wind stress anomalies driven by the SST anomaly at that time; 4. Repeat the above for all time points in the second period, we obtain a wind stress anomaly driven by SSTA during the entire second period; 5. Validate the skill of this statistical atmosphere model against observations by computing correlations between modeled and observed wind stresses at each grid point.

Python script to construct a statistical atmosphere model is below:

[ ]:
```
taux_test = taux_da.sel(time=slice('2000', '2016'), lon=slice(lon_min, lon_max))
tauy_test = tauy_da.sel(time=slice('2000', '2016'), lon=slice(lon_min, lon_max))
sst_test = sst_da.sel(time=slice('2000', '2016'), lon=slice(lon_min, lon_max))
# convert to numpy array
taux_test = taux_test.values
tauy_test = tauy_test.values
sst_test = sst_test.values
# shape
nt, ny, nx = taux_test.shape
# standard deviation
taux_test_std = taux_test.std()
tauy_test_std = tauy_test.std()
sst_test_std = sst_test.std()
# normalize the data
taux_test_anom = normalize_data(taux_test) * taux_test_std
tauy_test_anom = normalize_data(tauy_test) * tauy_test_std
sst_test_anom = normalize_data(sst_test)
# mask out land
taux_test_anom[:, grid==-1] = 0
tauy_test_anom[:, grid==-1] = 0
sst_test_anom[:, grid==-1] = 0
# reshape to space x time
sst_test = sst_test_anom.reshape((nt, ny*nx)).T
# choose n_modes in [1, 2, 4, 6, 8, 10]
corrs = np.zeros((6, ny, nx))
n_modes_list = [1, 2, 4, 6, 8, 10]
for m, n_modes in enumerate(n_modes_list):
    # predict wind stress from SST
    tau_ts = V[:n_modes, :] @ sst_test # shape: (n_modes, nt)
    tau_pred = np.sum(tau_ts[:,:,np.newaxis] * U[:,np.newaxis,:n_modes].T,␣
 ↪axis=0) # shape: (nt,2*ny*nx)
    taux_pred = tau_pred[:,:ny*nx] * taux_std # shape: (nt, ny*nx)
    tauy_pred = tau_pred[:,ny*nx:] * tauy_std # shape: (nt, ny*nx)
    # compute the correlation coefficient between the predicted and observed␣
 ↪wind stress
    taux_test_anom = taux_test_anom.reshape((nt, ny*nx))
    tauy_test_anom = tauy_test_anom.reshape((nt, ny*nx))
    tau_mag_test = np.sqrt(taux_test_anom**2 + tauy_test_anom**2)
    tau_mag_pred = np.sqrt(taux_pred**2 + tauy_pred**2)
    corr = np.zeros((ny, nx))
```

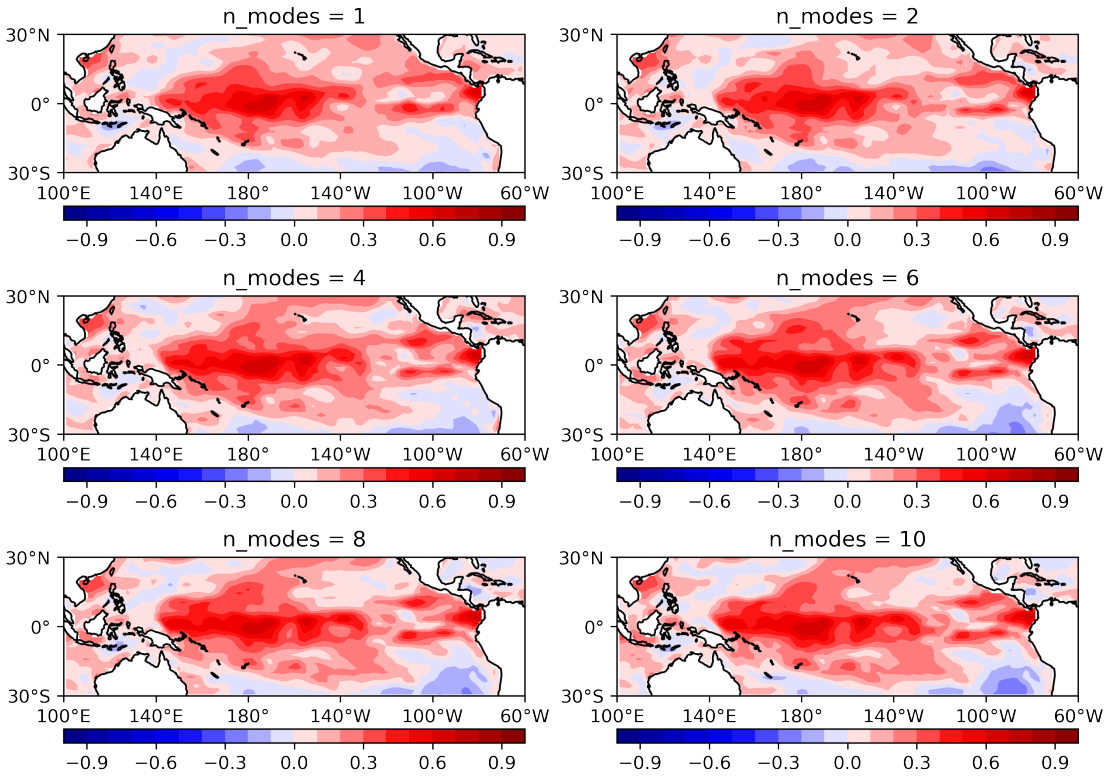
```
    for i in range(ny):
        for j in range(nx):
            corr[i, j] = np.corrcoef(tau_mag_test[:,i*nx+j], tau_mag_pred[:
↪,i*nx+j])[0, 1]
    corr = corr.reshape((ny, nx))
    corrs[m] = corr
```

```
c:\Users\liujj\miniconda3\envs\TC\lib\site-
packages\numpy\lib\function_base.py:2559: RuntimeWarning: invalid value
encountered in true_divide
  c /= stddev[:, None]
c:\Users\liujj\miniconda3\envs\TC\lib\site-
packages\numpy\lib\function_base.py:2560: RuntimeWarning: invalid value
encountered in true_divide
  c /= stddev[None, :]
```

3.3 Plot the correlation between modeled and observed wind stress

The correlation between modeled and observed wind stress is plotted below:

[ ]:
```
# plot the correlation coefficient
projection = ccrs.PlateCarree(central_longitude=180)
fig, axes = plt.subplots(3, 2, figsize=(10, 7.5), dpi=300,␣
↪subplot_kw={'projection': projection})
levels = np.linspace(-1, 1, 21)
for i in range(6):
    ax = axes[i//2, i%2]
    p = ax.contourf(lon, lat, corrs[i], transform=ccrs.PlateCarree(),␣
↪levels=levels, cmap=cmaps.BlWhRe)
    ax.set_title("n_modes = " + str(n_modes_list[i]))
    ax.coastlines()
    ax.set_xticks(np.arange(lon_min, lon_max+1, 40), crs=ccrs.PlateCarree())
    ax.set_yticks(np.arange(lat_min, lat_max+1, 30), crs=ccrs.PlateCarree())
    ax.xaxis.set_major_formatter(lon_formatter)
    ax.yaxis.set_major_formatter(lat_formatter)
    axins = inset_axes(ax, width="100%", height="10%", loc='lower center',␣
↪borderpad=-2.6)
    fig.colorbar(p, cax=axins, orientation="horizontal")
# fig.tight_layout()
plt.show()
```



4 Analysis

4.1 Sensitivity test

We change the number of SVDs used to compute the anomalies and plot the correlation between modeled and observed wind stress. The results are shown above. In general, the correlation is not sensitive to the number of SVDs used. This is because the first few SVDs are more important than the later ones. The first SVD can already explain 73.6% of the total variance, and the first four SVDs can explain 91.6% of the total variance. Therefore, increasing the number of SVDs only improves the correlation a little.

4.2 Physical interpretation

This statistical model makes physical sense. The first mode of SST is ENSO, which drives the Walker Circulation. The second mode of the SST is the meridional change (seasonal cycle?) of SST, which can influence the trade winds. Therefore, using SSTA to predict wind stress makes physical sense.

The correlation between modeled and observed wind stress anomalies is high in most regions, especially near the equator. On the other hand, the correlation is low in the North-West Pacific and South-East Pacific. This is because the Walker circulation pattern is near the equator. Trade winds are also strong near the equator. However, the winds are more complicated in the North-East

Pacific region where the monsoon plays an important role, making it difficult to predict winds only based on SSTA. In the South-East Pacific, the model also performs not well.