Final Project

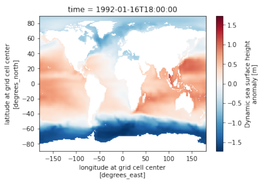
In [16]:
```
#computation imports
import xarray as xr
import numpy as np

#visualization imports
import matplotlib.pyplot as plt
from matplotlib.gridspec import GridSpec
import ipywidgets as widgets
import cartopy.crs as ccrs
%matplotlib inline

#stop complaining
import warnings
warnings.filterwarnings('ignore')
```

Open Raw Data

In [2]:
```
ssh = xr.open_mfdataset('~/Desktop/FALL 2023/Honors Thesis/data/ssh/SEA_SURFACE_HEIGHT_mon_mean_*')
```

In [3]:
```
ssh['SSH'].isel(time = 0).plot()
```

Out[3]:
```
<matplotlib.collections.QuadMesh at 0x7fdb96a8eee0>
```



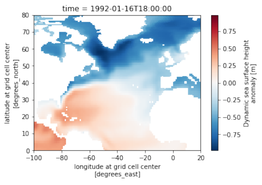
Trim Spatial Scope to North Atlantic

In [4]:
```
def spatial_trim(ds, extent = [0.,80.,-85.,20.]):

    ds_trim = ds.sel(latitude = slice(str(extent[0]), str(extent[1]))).sel(longitude = slice(str(extent[2]),str(extent[3])))

    return ds_trim
```

In [5]:
```
ssh_atl = spatial_trim(ssh)
```

In [6]:
```
ssh_atl['SSH'].isel(time = 0).plot()
```

Out[6]:
```
<matplotlib.collections.QuadMesh at 0x7fdb50032250>
```



Detrend Spatial Data with Respect to Time

In [7]:
```
def detrend_linear2(da, dim):
    """ linear detrend DataArray along the axis dim """
    params = da.polyfit(dim=dim, deg=1)
    fit = xr.polyval(da[dim], params.polyfit_coefficients)
    da = da-fit
    return da, params.polyfit_coefficients[0,:,:]
```

In [8]:
```
det, params = detrend_linear2(ssh_atl['SSH'], 'time')
```

Prep Data for SVD

In [9]:
```
ssh_ad = det.groupby('time.year').mean('time')
```

In [10]:
```
y = ssh_ad.stack(ll=("latitude","longitude"))
w = np.nan_to_num(y)
```

Do SVD

In [11]:
```
U, s, Vt = np.linalg.svd(w, full_matrices = False)
```

Convert SVD output to Xarray DataSet for Plotting

In [12]:
```
maps = Vt.reshape(26,160,240)  #reshape stacked lat and lon to mappable format
index = np.where(maps == 0.0)  #find places where spatial value is zero
maps[index] = np.nan           #replace these places with nan to make continents white

#setting up coordinates for dataset
lat_ = ssh_ad['latitude']
lon_ = ssh_ad['longitude']
comp = np.arange(0,26)
y = ssh_ad['year']
```

In [13]:
```
EOF = xr.Dataset(
        data_vars=dict(
            spatial = (['nth_comp', 'lat', 'lon'], maps),
            temporal = (['year','nth_comp'], U),
            sv = (['nth_comp'], s),
        ),
        coords=dict(
            nth_comp = comp,
            lat = lat_.values,
            lon = lon_.values,
            year = y,
        ),
        attrs=dict(description='EOF ATL'),
    )
```

Reconstructing origional map from SVD

In [14]:
```
total = 0
for i in range(0,26):
    total += EOF['spatial'].isel(nth_comp = i) * EOF['sv'].isel(nth_comp = i) * EOF['temporal'].isel(nth_comp = i)
```

Plotting Routine

In [15]:
```
#interactive plotting

def update_plot(time, n, intensity):

    a = n[0] #a is the first svd plotted, updated by slider output
    z = n[1] #z is the last SVD component plotted, updated by slider output

    i_time = time #i_time is the time step to be plotted, updated by time slider output 'time'

    #initializing sum of SVD componets (ssum = spatial, tsum = temporal)
    nth_ssum = 0
    nth_tsum = 0

    #generating map of cumulative SVD components
    for i in range(a,z+1):
        nth_ssum += EOF['spatial'].isel(nth_comp = i) * EOF['sv'].isel(nth_comp = i) * EOF['temporal'].isel(nth_comp = i)
        nth_tsum += EOF['sv'].isel(nth_comp = i) * EOF['temporal'].isel(nth_comp = i)

    #initalize figure
    fig = plt.figure(figsize = [8.5,3.5], dpi = 200)
    fig.suptitle('SSH SVD Components')


    #set up coastline projection
    usemap_proj = ccrs.PlateCarree()

    #set layout
    gs = GridSpec(2, 3, width_ratios=[3.25, 1, 1], height_ratios = [3,1.25])
    ax1 = fig.add_subplot(gs[0:,0], projection=usemap_proj)    #all rows but only first column for first subplot (large map)
    ax2 = fig.add_subplot(gs[0, 1], projection=usemap_proj)    #2nd column, 1st row
    ax3 = fig.add_subplot(gs[0, 2], projection=usemap_proj)    #3rd column, 1st row
    ax4 = fig.add_subplot(gs[1, 1:])                           #2rd row, 2-3 columns

    fig.tight_layout(h_pad = 3, w_pad = 3) #spacing between subplots

    #large subplot
    cont = ax1.contourf(np.linspace(-99.75, 19.75, 240),np.linspace(0.25, 79.75, 160), nth_ssum.sel(year = i_time), levels = 50, cmap = 'RdBu', vmin =  -0.1 + (100./intensity),
    ax1.set_extent([-99.75, 19.75, 0.25, 79.75], crs = usemap_proj)
    ax1.set_xticks(ticks = [-100, -60, -20, 20])
    ax1.set_xticklabels(labels = [-100, -60, -20, 20], fontsize = 6)
    ax1.set_xlabel('longitude', fontsize = 6)
    ax1.set_yticks(ticks = [0, 40, 80])
    ax1.set_yticklabels(labels = [0, 40, 80], fontsize = 6)
    ax1.set_ylabel('latitude', fontsize = 6)
    ax1.coastlines(linewidth = 0.3)
    ax1.set_title(str(a) + ' - ' + str(z) + ' PC Components')

    #small subplot (total SVD components)
    tot = ax2.contourf(np.linspace(-99.75, 19.75, 240),np.linspace(0.25, 79.75, 160), total.sel(year = i_time), levels = 50, cmap = 'RdBu', vmin = -0.1, vmax = 0.1)
    ax2.set_xticks(ticks = [-100, -60, -20, 20])
    ax2.set_xticklabels(labels = [-100, -60, -20, 20], fontsize = 6)
    ax2.set_xlabel('longitude', fontsize = 6)
    ax2.set_yticks(ticks = [0, 40, 80])
    ax2.set_yticklabels(labels = [0, 40, 80], fontsize = 6)
    ax2.set_ylabel('latitude', fontsize = 6)
    ax2.set_title('Total')
    ax2.coastlines(linewidth = 0.1)

    #small subplot (residual (All SVD - selected SVD))
    ax3.contourf(np.linspace(-99.75, 19.75, 240),np.linspace(0.25, 79.75, 160), total.sel(year = i_time) - nth_ssum.sel(year = i_time), levels = 50, cmap = 'RdBu',vmin = -0.1, v
    ax3.set_title('Residual')
    ax3.set_xticks(ticks = [-100, -60, -20, 20])
    ax3.set_xticklabels(labels = [-100, -60, -20, 20], fontsize = 6)
    ax3.set_xlabel('longitude', fontsize = 6)
    ax3.set_yticks(ticks = [0, 40, 80])
    ax3.set_yticklabels(labels = [0, 40, 80], fontsize = 6)
    ax3.set_ylabel('latitude', fontsize = 6)
    ax3.coastlines(linewidth = 0.1)

    #time series subplot
    ax4.plot(EOF.year, nth_tsum, color = '#398CBF')
    ax4.plot(EOF.year.sel(year = i_time), nth_tsum.sel(year = i_time), 'o', color = '#BF3F3F')
    ax4.grid(axis = 'y', alpha = 0.3)
    ax4.set_yticks(ticks = [-4, 0, 4])
    ax4.set_yticklabels(labels = [-4, 0, 4], fontsize = 6)
    ax4.set_xticks(ticks = [1992, 2000, 2008, 2018])
    ax4.set_xticklabels(labels = [1992, 2000, 2008, 2018], fontsize = 6)
    ax4.set_xlabel('year', fontsize = 6)
    ax4.set_title('time series')


    #main colorbar
    cbar = fig.colorbar(cont, ax=ax1,  aspect = 20, shrink = 0.9)
    cbar.ax.tick_params(labelsize = 4)
    cbar.set_label('ssh (m)', fontsize = 6)

    #total & residual color bar
    cbar2 = fig.colorbar(tot, ax = [ax2, ax3],location = 'bottom', orientation = 'horizontal', aspect = 30, pad = 0.2)
    cbar2.ax.tick_params(labelsize = 4)
    cbar2.set_label(label = 'ssh (m)', size = 6)



#interactive part
w = widgets.interact(
    update_plot,
    time = widgets.IntSlider(min = 1992, max = 2017, step = 1, value = 1992, description = 'Year'),            #call update plot function
    intensity = widgets.IntSlider(min = 100, max = 400, step = 10, value = 100, description = 'Intensity %'),  #seting time slider
    n = widgets.IntRangeSlider(min = 0, max = 24, step = 1, value = [0,1], description = 'PC Range'),          #setting nth SVD component slider
    single = widgets.Checkbox(value = False, description = 'single PC'),                                       #setting sinlge check box
)
```

```
interactive(children=(IntSlider(value=1992, description='Year', max=2017, min=1992), IntRangeSlider(value=(0, …
```

In [ ]:
```

```